NATIONAL TECHNICAL UNIVERSITY OF UKRAINE

“IGOR SIKORSKY KYIV POLYTECHNIC INSTITUTE”

Faculty of Informatics and Computer Engineering

Department of Computer Engineering

Admitted to the defense:

Acting Department Head

Mykhailo NOVOTARSKY
(signature)

«___»_____2025___p.

Graduation project

for obtaining a bachelor's degree

in the educational and professional program

“Software Engineering for Computer Systems”

specialty 121 "Software Engineering"

on the topic: Mobile Application for Supporting the Religious Practices of Muslims

The work was performed by: 4 year student, IO-17 group
(group code)

Darwish Jamil Mohammad Jamil _____
(Full Name) (signature)

Project supervisor Lecturer Yurii Berdnyk _____
(position, academic degree, academic status, surname and initials) (signature)

Advisor (normative control) Associate Professor of CED, Ph.D.,
Valerii Pavlov _____
(section name) (position, academic degree, academic status, surname and initials) (signature)

Reviewer Senior lecturer Volodymyr Shymkovych _____
(position, academic degree, academic status, surname and initials) (signature)

I certify that in this graduation project there are no borrowings from the works of other authors without proper references.
Student _____
(signature)

Kyiv – 2025

NATIONAL TECHNICAL UNIVERSITY OF UKRAINE

“IGOR SIKORSKY KYIV POLYTECHNIC INSTITUTE”

Faculty of Informatics and Computer Engineering

Department of Computer Engineering

Level of higher education - first (bachelor's)

Educational and professional program

«Software Engineering of Computer Systems»

Specialty - 121 «Software Engineering»

Admitted to the defense:

Acting Department Head

Mykhailo NOVOTARSKY

(signature)

«___»_____2025___p.

TASK

for a student's bachelor's degree project

Darwish Jamil Mohammad Jamil

(Full Name)

1. The topic of the project "Mobile Application for Supporting the Religious Practices of Muslims", the supervisor Volodymyr Shymkovych, senior lecturer of System Programming and Specialized Computer Systems Department, approved by the order of the university № 34/25-ci dated «16» May 2025.

2. Deadline for submission of the completed project by the student «16» June 2025.

3. Output data to the the project user interface, database and basic functions of the mobile application to meet the religious needs of Muslims.

4. The content of the explanatory note (the list of issues to be developed):

Introduction, Analysis of Existing Solutions, Requirements Specification, System Architecture, User Interface and User Experience, Implementation, Testing, Conclusion, References

5. List of graphic material (with exact designation of obligatory drawings):

Application Architecture, Class diagram, Sequence diagram

6. Consultants of project sections

Section	Surname, initials and position of the consultant	Signature, date	
		issued the task	accepted the task
Normative control	Valeriy Pavlov, associate professor		

7. Date of issue of the task

Calendar plan

№	Naming the stages of the diploma project	Timeframe for completing the phases	**Notes**
1	Approval of the subject of the project	01.02.2025 –05.02.2025	
2	Study and analysis of the task	05.02.2025 –25.02.2025	
3	Development of architecture and general system structures	26.02.2025 –10.03.2025	
4	Development of separate structures subsystems	11.03.2025 –25.03.2025	
5	Implementation of the web application	26.03.2025 –05.04.2025	
6	Writing an explanatory note	06.04.2025 –20.04.2025	
7	Project testing and adjustments	21.04.2025 – 5.04.2025	
8	Preliminary protection	05.05.2025	
9	Protection	20.05.2025	

Graduate student Darwish Jamil Mohammad Jamil

Project supervisor Yurii Berdnyk

Annotation

The present paper is devoted to the development and implementation of a cross-platform mobile application for the Muslim audience, which provides support for daily spiritual life in accordance with the requirements of the Islamic tradition. The primary objective of this study is to develop a tool that can accurately calculate the time for the five obligatory prayers, with consideration for the user's geographical location, the selected madhab, and the chosen calculation method. Moreover, the application incorporates modules that facilitate the determination of the direction of the Qiblah through the utilisation of the device's sensors. It also encompasses the integration of the Islamic calendar, access to the text of the Qur'an and hadith in multiple languages, a digital tasbih counter, a notification system, and motivational daily reminders.

The development was executed utilising Apache Cordova, a software that guaranteed cross-platform compatibility (Android and iOS) and extensive coverage of the target demographic. The interface is based on HTML5, CSS3, and JavaScript, with the use of modern UI frameworks and libraries to support multilingualism, adaptability, and accessibility. It is imperative to note that particular attention is paid to security and privacy. All personal data is stored locally, encryption and validation mechanisms for religious content integrity are in place, and compliance with international data protection standards is ensured.

The project is notable for its recognition of the religious diversity of the Islamic world, a feature that facilitates user choice according to their legal school and local traditions. Adaptive algorithms have been implemented to ensure correct operation in regions where atypical conditions prevail.

Keywords: MOBILE APPLICATION, ISLAM, GEOLOCATION, RELIGIOUS CALENDAR, CROSS-PLATFORM, SPIRITUAL PRACTICE

Анотація

Дана робота присвячена розробці та реалізації кросплатформного мобільного застосунку для мусульманської аудиторії, який забезпечує підтримку щоденного духовного життя відповідно до вимог ісламської традиції. Основна увага приділена створенню інструменту для точного розрахунку часу п’яти обов’язкових молитов з урахуванням геолокації користувача, обраного мазхабу та обраного методу розрахунку. Окрім цього, застосунок включає модулі для визначення напрямку Кібли за допомогою сенсорів пристрою, інтеграцію ісламського календаря, доступ до тексту Корану та хадісів різними мовами, цифровий лічильник тасбіху, систему сповіщень та мотиваційні щоденні нагадування.

Розробка здійснювалась із використанням Apache Cordova, що дозволило забезпечити кросплатформну сумісність (Android та iOS) і широке охоплення цільової аудиторії. Інтерфейс реалізовано на основі HTML5, CSS3 та JavaScript із залученням сучасних UI-фреймворків і бібліотек для підтримки багатомовності, адаптивності та доступності. Особливу увагу приділено безпеці та приватності: усі персональні дані зберігаються локально, впроваджено механізми шифрування та валідації цілісності релігійного контенту, а також забезпечено відповідність міжнародним стандартам захисту даних.

Проєкт враховує релігійну різноманітність ісламського світу, дозволяючи користувачам обирати параметри відповідно до своєї правової школи та локальних традицій. Реалізовано адаптивні алгоритми, що забезпечують коректність роботи навіть у регіонах із нетиповими астрономічними умовами.

Ключові слова: МОБІЛЬНИЙ ЗАСТОСУНОК, ІСЛАМ, ГЕОЛОКАЦІЯ, РЕЛІГІЙНИЙ КАЛЕНДАР, КРОСПЛАТФОРМНІСТЬ, ДУХОВНА ПРАКТИКА.

Information	Format	Value	Name	Number of sheets	copy number	Addition
1			Project documentation			
2						
3	A4	**ІАЛЦ.467200.002 ТЗ**	Mobile Application for	4		
4			Supporting the Religious			
5			Practices of Muslims			
6			Technical Project notice			
7	A4	**ІАЛЦ.467200.003 ПЗ**	Mobile Application for	94		
8			Supporting the Religious			
9			Practices of Muslims			
10			Explanatory note			
11	A1	**ІАЛЦ.467200.004 Д1**	Mobile Application for	1		
12			Supporting the Religious			
13			Practices of Muslims			
14			Application Architecture			
15	A1	**ІАЛЦ.467200.005 Д2**	Mobile Application for	1		
16			Supporting the Religious			
17			Practices of Muslims			
18			Class diagram			
19	A1	**ІАЛЦ.467200.006 Д3**	Mobile Application for	1		
20			Supporting the Religious			
21			Practices of Muslims			
22			Sequence diagram			
23	A1	**ІАЛЦ.467200.006 Д4**	Mobile Application for	24		
24			Supporting the Religious			
25			Practices of Muslims			
26			Program Codes			
27						
28						

CHG	Sheet	№ document	Sign.	Date

ІАЛЦ.467200.001 ОА

Developed	Jamil D.	Mobile Application for Supporting the Religious Practices of Muslims **Album description**	Type: — / Sheet: 1 / Sheets: 1
Checked	Berdnyk Y.		
Reviewer	Shymkovych V.		**FIOT "KPI" IO-17**
Norm.control	Pavlov V.		
Approved	Novotarsky M.		

CONTENTS

1. NAME AND AREA OF APPLICATION

Development name: islamLife

Application Area: This app is a cross-platform mobile application for Muslims, offering prayer times, Qibla direction, Quran and Hadith access, Islamic calendar, digital tasbih, and notifications. It serves users of all ages seeking a convenient and culturally adapted tool for daily religious practice.

2. REASONS FOR DEVELOPMENT

Most existing Islamic mobile apps are limited in scope, lacking comprehensive features, multilingual support, or reliable content verification. Few address regional diversity, accessibility, or privacy needs. This project was developed to provide an all-in-one, user-friendly, and secure platform that fully supports the daily religious needs of Muslims.

3. PURPOSE OF THE DEVELOPMENT

The primary goal of islamLife is to allow users to:

- Provide accurate daily prayer times with location-based adjustments
- Show Qibla direction using device sensors
- Offer full access to the Quran and Hadiths in multiple languages
- Display Islamic calendar with important dates and events
- Include a digital tasbih counter for dhikr
- Send personalized notifications for prayers and religious events
- Enable offline access to core features
- Support multilingual interface (Arabic, English, Ukrainian, Turkish, etc.)

The purpose also includes building a secure, scalable, and responsive interface.

4. THE SOURCES OF DEVELOPMENT

The project was developed using:

- Framework/Platform: Apache Cordova
- Localization: i18next
- Plugins: cordova-plugin-geolocation, cordova-plugin-device-orientation, cordova-plugin-local-notifications
- APIs Integrated: AlQuran Cloud API, Prayer Times API, Hadith API
- Frontend: HTML5, CSS3, JavaScript

5. TECHNICAL REQUIREMENTS

5.1 Requirements for the product to be developed

- Accurate calculation of prayer times based on user location
- Qibla direction display using device sensors
- Full access to Quran and Hadith texts with multilingual support
- Islamic calendar with key dates and event reminders
- Digital tasbih counter with session saving
- Personalized notifications for prayers and important events
- Offline access to core features
- Multilingual user interface (Arabic, English, Ukrainian, Turkish, etc.)
- Data privacy and local storage of sensitive information
- Simple, intuitive, and accessible UI adaptable for various devices
- Compatibility with Android and iOS platforms

5.2 Software requirements

- Node.js
- Apache Cordova
- Android Studio
- Modern web browser (Chrome, Firefox, Edge, Safari)
- Operating system: Windows 10/11, macOS, or Linux

5.3 Requirements for the hardware

Minimum:

- Processor: Intel i3 or equivalent
- RAM: 4 GB
- Storage: 2 GB free space

Recommended:

- Processor: Intel i5 or higher
- RAM: 8 GB
- SSD storage

6. DESIGNING STAGES

№	Naming the stages of the diploma project	Timeframe for completing the phases
1	Approval of the subject of the project	01.02.2025 –05.02.2025
2	Study and analysis of the task	05.02.2025 –25.02.2025
3	Development of architecture and general system structures	26.02.2025 –10.03.2025
4	Development of separate structures subsystems	11.03.2025 –25.03.2025
5	Implementation of the web application	26.03.2025 –05.04.2025
6	Writing an explanatory note	06.04.2025 –20.04.2025
7	Project testing and adjustments	21.04.2025 – 5.04.2025
8	Preliminary protection	05.05.2025
9	Protection	20.05.2025

Explanatory note
to the diploma project

on the topic: Mobile Application for Supporting the Religious Practices of Muslims

Kyiv – 2025

CONTENTS

LIST OF TERMS AND ABBREVIATIONS

UI – The visual and interactive part of an application for the user

UX – The overall experience and satisfaction of the user when interacting with the application

API – A set of routines and protocols for interacting with software components

SDK – A collection of software tools for developing applications

GPS – A system for determining geographic location

CRUD – The main operations for working with data

CLI – A method of interacting with software using text commands

JSON – A lightweight data-interchange format

GDPR – The European Union regulation for data protection and privacy

APK – The file format for distributing and installing apps on Android devices

HTML – The markup language for creating web pages

CSS – A style sheet language for describing the presentation of web pages

JS – A programming language used in web development

INTRODUCTION

In today's digital world, mobile applications have become an integral part of everyday life for people across the globe. Their active use is evident not only in communication, education, commerce, entertainment, and healthcare, but also in religious life. The significance of mobile technologies for religious purposes lies in their ability to provide spiritual connection for believers, especially when they are far from traditional places of worship.

The Muslim community, whose population continues to grow and, according to international organizations, now exceeds 1.8 billion people [1], actively embraces digital solutions to support and organize their spiritual lives.

Mobile applications assist users in accurately determining prayer times based on their geolocation, identifying the direction of the Qibla using a compass or GPS, reading and listening to the Holy Quran, studying hadiths, making dua, planning Hajj and Umrah, locating halal food establishments, finding routes to nearby mosques, participating in online lessons and conferences, connecting with fellow believers, and acquiring religious knowledge in an interactive format [2].

The growing popularity of such digital solutions can be attributed to several factors: the globalization of society, increased migration of Muslims to various parts of the world, the widespread integration of internet technologies and mobile devices into daily life, and the need to preserve religious identity in multicultural settings. Mobile applications enable Muslims to maintain their spiritual practices regardless of location, save time and resources, and strengthen their connection to the religious community.

In particular, these applications play a key role in promoting religious education among young people, who spend a significant portion of their time in the digital environment. They serve as an effective tool in combating misinformation about Islam and contribute to shaping and maintaining a positive image of Muslim culture on a global scale.

Significant attention is also given to integrating these apps with portable devices that are almost constantly with users – such as smartwatches, fitness bands, and voice assistants – as well as ensuring offline access to essential features.

Despite the wide range of available solutions, most mobile applications offers limited functionality, focusing only on specific aspects of religious practice. There is a noticeable lack of universal platforms that integrate ritual, educational, and social components within a single environment. Additionally, many applications do not adequately account for the cultural and linguistic diversity of different regions, offer limited options for service personalization, or fall short in ensuring robust protection of users' personal data. The verification of religious content – crucial for the spiritual well-being of Muslims – also remains an area that demands particular attention.

Therefore, a key task is the development and enhancement of mobile solutions for Muslims that offer comprehensive support for religious practice while meeting modern technological standards and user expectations.

CHAPTER 1. OVERVIEW OF MODERN MOBILE APPLICATIONS

In the context of rapid digitalization, mobile applications have become a vital tool for supporting both the daily and spiritual lives of individuals – particularly within the Muslim community. Following the analysis of key trends and the significance of mobile technologies in the religious sphere, as outlined in the introduction, it is appropriate to move on to a detailed examination of the current state and development features of mobile applications designed to meet the religious needs of Muslims. This section will explore the classification of these services, analyze their functional capabilities and technological solutions, and highlight the main challenges currently faced by both users and developers in this evolving industry [5].

Amid globalization, urbanization, and active migration, millions of Muslims worldwide face limited access to traditional religious institutions – such as mosques, imams, and faith communities [10]. This challenge is particularly common among Muslim youth studying or working in countries with predominantly secular or Christian populations. For many, mobile applications often serve as the sole means to stay connected with their religious traditions: to perform daily prayers on time, observe fasting, study the Quran, use the Islamic calendar, or seek guidance on ethics and Sharia [11].

Although the market for Islamic mobile services is rapidly evolving, it remains fragmented. Many applications offer individual features such as access to the Quran, Qibla direction, prayer time reminders, or an Islamic calendar. However, most are limited to one or two functions and do not provide users with a comprehensive digital environment to fully support their religious life [5]. This highlights the need for a universal application that integrates the key aspects of a Muslim's religious experience – ritual, educational, informational, social, communicative, and motivational – within a single, unified interface.

A significant challenge in developing such applications is ensuring the accuracy and religious authenticity of the information they provide. In an open digital environment, there is a risk of spreading misinformation or radicalized interpretations of Islamic teachings [12]. Therefore, mobile applications should be developed in close collaboration with qualified theologians, Sharia consultants, and under the supervision or endorsement of official religious institutions [13]. This approach not only enhances user trust but also protects users from disinformation and potential manipulation.

Another important task is the localization and cultural adaptation of the service. The Muslim Ummah is highly diverse, encompassing various ethnic, linguistic, and cultural groups [11]. For instance, Muslims in Ukraine, Turkey, Indonesia, Egypt, and Nigeria differ in daily traditions, language preferences, and certain religious practices. Therefore, a modern application should offer a multilingual interface, allow users to select their madhhab, support regional calendars, and display holidays according to local customs.

Equally important is adopting a personalized and inclusive approach in designing such digital products. The application should be user-friendly for individuals with varying levels of religious knowledge – ranging from newly converted Muslims to lifelong practitioners [12]. Additionally, it must address the needs of people with disabilities, women, children, and the elderly. To achieve this, features such as text-to-speech narration, adjustable font sizes, voice control, and a simple, intuitive interface should be incorporated.

Information security is another critical aspect that deserves special attention. Religious applications often handle sensitive data, such as users' geolocation, daily activities, requests to imams, or participation in religious discussions. In countries with high levels of censorship or political tension, this information could be exploited against users. Therefore, modern applications must ensure robust data protection by employing encryption, local data storage, support for anonymous modes, and preferably open-source code that allows the community to verify its security and compliance with standards [13].

Thus, the development and deployment of mobile applications for the religious life of Muslims demand a comprehensive approach that integrates technological innovation, cultural sensitivity, and ethical responsibility. Such an approach will not only guarantee user convenience and security but also support the preservation and growth of religious identity in an increasingly globalized world.

1.1 Classification of mobile applications

Based on the typology established through recent research and a review of popular mobile services for Muslims, several main types of applications actively used in daily religious practice can be identified [4]. One key category includes mobile applications designed for Hajj and Umrah. These services offer users step-by-step guidance for performing pilgrimage rituals, interactive maps, GPS navigation to holy sites, and timely reminders for important actions. They often incorporate recommendations from religious authorities, travel advice, and safety information – examples include the Hajj Navigator (fig. 1.1) and SmartHajj applications [4].

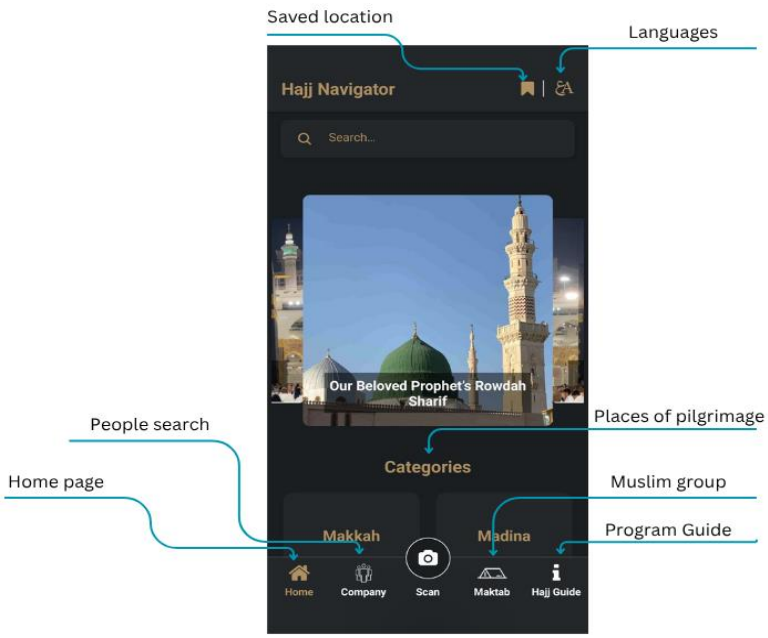


Figure 1.1 – User interface of the Hajj Navigator mobile application

The second common category includes applications for determining prayer times and the direction of the Qibla. These apps accurately calculate the five daily prayer times based on the user's location, display the Qibla direction using a compass or GPS, and incorporate the Islamic calendar. Examples include Muslim Pro, Athan, and Al-Moazin Lite, which offer customizable prayer time calculation methods according to the traditions of various Islamic centers [12].

Applications for reading and studying the Quran and hadith also play a vital role in spiritual life. They provide access to the complete text of the Holy Quran in Arabic, translations in dozens of languages, audio recitations by renowned reciters, and search functions for words, verses, or surahs. Notable examples include Quran Majeed, Ayat, and Islam 360 (fig. 1.2). Many of these apps also integrate tafsir (interpretations) and commentary to enhance users' understanding [11].

Figure 1.2 – User interface of the Islam 360 mobile application

Services offering collections of dua, azkar, and tasbih also attract a significant audience. These applications provide prayer texts for various life situations, morning and evening azkar, and electronic counters for tracking repetitions of prayer formulas. Apps like MyDuaa (Hisnul Muslim) and Muslim Assistant often include audio versions of duas, translations, and features to monitor the user's prayer practice [14].

A substantial portion of the mobile Islamic services market is dedicated to educational and motivational platforms. These include interactive lessons, videos, audio lectures, podcasts, and quizzes that help both children and adults deepen their knowledge of Islamic sciences. Many applications incorporate gamification elements to encourage user engagement through achievement systems and points. Examples include Muslim Central, OnePath Network, and Niyyah [12].

A distinct niche is occupied by applications offering information services and social features. These platforms enable users to locate mosques, halal food establishments, and Islamic centers; engage with religious communities; participate in charitable initiatives; and even submit online applications for zakat or qurban payments. In many countries, services providing online consultations with imams, muftis, or sheikhs are becoming increasingly popular [13].

Current trends in the Islamic mobile app market show a growing preference for hybrid solutions that combine multiple functionalities within a single product. For example, platforms like Muslim Pro and WeMuslim offer comprehensive tools for religious life – from prayer time notifications and Quran reading to social network integration and charity services – allowing users to meet a wide range of spiritual and social needs in a globalized digital environment [14].

Depending on their purpose, these services can be classified into key categories: Hajj and Umrah, prayer time and Qibla direction determination, Quran and hadith reading, dua and azkar, educational and informational platforms, and hybrid solutions.

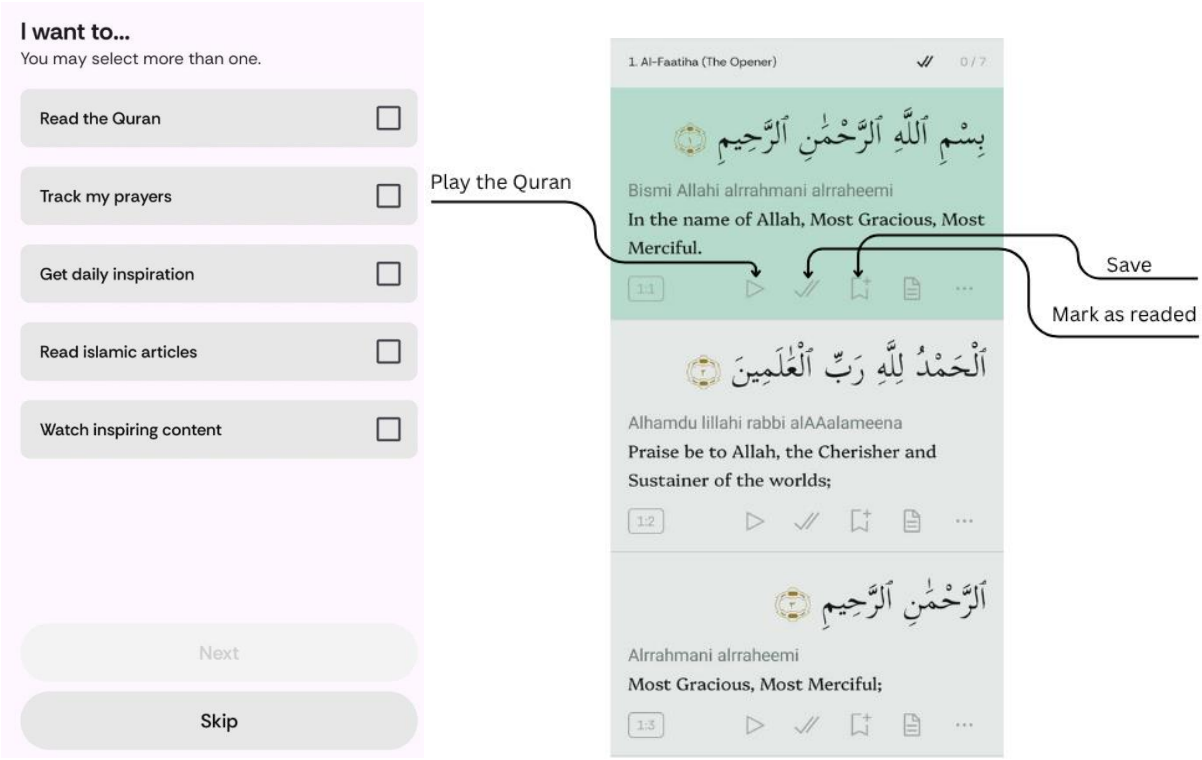
1.1.1 Hajj and Umrah

This category includes applications specifically designed to meet the needs of pilgrims during Hajj and Umrah. As noted in previous studies [5], over 100 Hajj- and Umrah-related apps on Google Play were analyzed. The core functionalities of these services include turn-by-turn navigation along pilgrimage routes, GPS maps of holy sites, step-by-step instructions for performing rituals, ritual schedules, and timely notifications with useful advice. Navigation relies on global positioning technologies (GPS, GNSS) with multi-frequency positioning to enhance accuracy in dense urban environments. Offline navigation is supported by caching map data in MBTiles format using the Mapbox SDK. Notifications are managed through Firebase Cloud Messaging (FCM), enabling the delivery of both regular and priority messages. Personal data protection is ensured via the OAuth 2.0 protocol, while integration with external services – such as security and medical care – is implemented through RESTful APIs. Typical features include interactive routes to Masjid al-Haram and Masjid an-Nabawi, prayer and ritual reminders, and informational sections covering the legal norms of Hajj and Umrah.

1.1.2 Prayer and Qibla Direction

Applications of this type, including Muslim Pro and Athan, provide accurate prayer times, Qibla direction, and the Islamic calendar. They utilize geolocation services – such as the Fused Location Provider on Android and Core Location on iOS – along with digital compasses, accelerometers, and magnetometers for precise spatial orientation.

Prayer time calculations are based on standards like ISNA, MWL, and Umm Al-Qura, implemented using programming languages such as Java, Kotlin, or Swift. Their interfaces are optimized for wearable devices and integrated with device calendars via system APIs.



A B

Figure 1.3 - User interface of the Muslim Pro mobile application: (A) the start page after downloading the application; (B) the page with Quran materials

1.1.3 Quran and Hadith

Quran reading apps like Quran Majeed, Ayat, and Islam 360 enable users to read the full text of the Holy Quran, switch between multiple languages, and listen to audio recitations. Text data is stored locally using databases such as SQLite or Room. Advanced search algorithms, including Trie data structures and Full-Text Search, are employed to facilitate fast and accurate text queries.

In addition to text, these applications stream audio using standard technologies such as HLS, with media players like ExoPlayer on Android and AVFoundation on iOS. All data is securely protected using AES-256 encryption. Multilingual support is implemented through built-in localization modules available in most modern development platforms.

1.1.4 Dua, azkar and tasbih

These applications offer users structured collections of supplications (dua) and azkar, featuring audio playback, repetition counters, and performance tracking. The supplications are typically organized into lists or categories (e.g., morning and evening azkar, duas for special occasions) and stored in built-in databases such as SQLite or Room, or as local structured files in JSON or XML formats. Offline access is supported through caching mechanisms, ensuring uninterrupted use even without an internet connection.

Audio playback of prayers is implemented using media frameworks such as ExoPlayer on Android and AVFoundation on iOS, supporting both local playback and streaming of audio files via protocols like HLS and DASH. These applications often offer features such as playback speed adjustment, interactive control of playback position, and buffering to ensure smooth and uninterrupted audio streaming.

To count prayer repetitions, electronic counters use Gesture Recognizer APIs – such as TapGestureRecognizer and LongPressGestureRecognizer – providing tactile or visual feedback. User interactions are tracked via touch or click events and stored as session records including timestamps and repetition counts. Some applications enhance the experience with animations or sound cues to improve user engagement.

Local data storage is implemented using file systems such as SharedPreferences (Android) and NSUserDefaults (iOS), as well as built-in databases like SQLite and CoreData.

These store user session histories, prayer progress, and application settings. For cloud storage, services like Firebase Firestore or Realtime Database are used to synchronize statistics across devices and enable data recovery in case of device loss or replacement. Data transfer between the client and cloud services is secured using HTTPS with TLS encryption.

Notifications and reminders are implemented using the native mechanisms of operating systems: AlarmManager on Android for scheduling periodic or one-time alerts, and UNUserNotificationCenter on iOS to manage notification scheduling, categories, and user interactions. Additionally, applications integrate push notifications via Firebase Cloud Messaging (FCM) to enable flexible delivery of service messages and data updates.

More advanced solutions employ a hybrid client-server architecture, utilizing microservices dedicated to specific functions such as statistics generation, data storage processing, and notification management. User activity analytics are gathered through integrated platforms like Google Analytics for Firebase, enabling developers to monitor feature popularity, interface usability, and prayer usage frequency.

Thus, dua and azkar applications integrate local and cloud technologies with multi-tiered data storage, multimedia frameworks, and notification systems, ensuring functionality, reliability, and flexibility to effectively meet users' religious needs.

1.1.5 Educational and information services

After examining applications centered on rituals and prayers, it is important to highlight educational and informational services. These play a crucial role in spreading religious knowledge and supporting the spiritual growth of users across various age groups.

A key feature of these applications is streaming video to broadcast lectures, sermons, Quran lessons, and online courses in religious studies. To enable this, APIs from platforms like YouTube and Vimeo are used, allowing video embedding within the app interface, playlist management, content access control, and user activity analytics. This approach significantly reduces the load on the application's own infrastructure, as video processing and delivery are handled by robust platforms with extensive server networks and scalable resources.

Gamification is employed to boost user motivation and engagement in religious education and practice. Technically, it is implemented through integration with services like Google Play Games on Android and Game Center on iOS. These platforms provide APIs for creating achievement systems, leaderboards, and progress tracking, enabling users to monitor their spiritual growth, participation in religious initiatives, and completion of educational courses. Additionally, gamification can be built on proprietary server infrastructure by storing user achievements, levels, and rewards in centralized databases.

Online testing and surveys within these applications often rely on REST APIs to exchange data between the client side – typically a mobile or web app built on modern frameworks – and the server responsible for evaluating answers and storing results. For scenarios requiring real-time synchronous interaction, such as shared Quran readings, webinar participation, or live quizzes, WebSockets are employed to enable two-way, real-time data exchange. This approach fosters a more interactive and engaging user experience.

To collect data on user behavior and evaluate content effectiveness, platforms integrate analytics services such as Google Analytics for Firebase and AppMetrica.

These tools enable tracking of key metrics, including session counts, user engagement duration, and the popularity of specific sections or features. This data allows developers and administrators to make informed improvements to the application. To ensure fast and reliable delivery of multimedia content – such as images, audio, and video – platforms utilize content delivery networks (CDNs) like Cloudflare, Amazon CloudFront, or Akamai. CDNs help reduce loading delays, enhance performance, and provide resilience during peak usage periods, such as the holy month of Ramadan or the Hajj pilgrimage.

The architecture of such applications typically employs modern data storage and security solutions.

The backend is commonly hosted on cloud platforms like Google Cloud Platform, AWS, or Microsoft Azure, enabling flexible resource scaling and ensuring high service availability. User authentication and authorization are managed using OAuth 2.0 and JWT tokens, while data transmission is secured with SSL/TLS protocols to prevent interception. Databases such as PostgreSQL, MongoDB, or MySQL reliably store user information, including religious preferences, learning progress, and test results.

1.1.6 Hybrid platforms

Modern comprehensive applications, such as Muslim Pro and WeMuslim, are built using a microservice architecture with REST or GraphQL APIs. User data is stored in cloud storage solutions like AWS S3 and Firebase, while sensitive information is processed through edge computing to enhance privacy and performance. Continuous integration and continuous deployment (CI/CD) pipelines automate updates, ensuring rapid delivery of new features and security patches.

Figure 1.4 – WeMuslim mobile app user interface

Thus, mobile applications for Muslim audiences integrate a broad spectrum of modern technologies – such as GPS, HLS streaming, encryption, REST APIs, and gamification – ensuring reliability, security, and multifunctionality to effectively support religious life.

In recent years, technological approaches such as Progressive Web Apps (PWA) have gained popularity in religious mobile services, offering users an almost native experience directly through browsers, along with offline access, push notifications, and automatic updates. Blockchain technology is increasingly used to verify the authenticity of religious content, enhancing user trust and preventing substitution or manipulation. Personalization of features and content is also advancing through machine learning algorithms that consider individual habits, religious activities, and preferences.

Furthermore, the integration of Islamic applications with IoT devices – such as smartwatches, bracelets, and other gadgets – is becoming more relevant, enabling prayer reminders, spiritual activity tracking, and automation of certain religious practices within a dynamic digital environment.

To objectively assess the effectiveness of mobile applications, quantitative metrics such as daily active users (DAU), monthly active users (MAU), average session duration, retention rate, and conversion rates from free to paid versions are utilized. Analyzing these indicators enables developers and service administrators to identify the most popular features, uncover user experience bottlenecks, and make informed decisions about future product development.

1.1.7 Comparative analysis of popular mobile applications

To objectively assess the quality and popularity of mobile applications for the Muslim audience, a comparative analysis based on key quantitative indicators is advisable. For example, as of June 2024, the Muslim Pro application has over 100 million downloads on Google Play, a user rating of 4.7/5, and an installation size of 44 MB. The Athan application boasts more than 10 million downloads, a rating of 4.6/5, and a size of 38 MB.

The MyDuaa and Muslim Assistant applications show somewhat lower but stable demand, each with over 5 million downloads and strong user ratings of 4.8/5 and 4.7/5, respectively, reflecting a high level of user trust.

The comparative analysis highlights key advantages such as broad functionality, multilingual support, regular updates, and positive user feedback. However, some applications also have drawbacks, including the presence of advertising in free versions and the requirement of an Internet connection to access full functionality [6].

Table 1.1 – Technologies and technical approaches used in different types of mobile applications

Application type	**Technologies, architectural solutions and technical details**
Hajj and Umrah	– GPS/GNSS for high-precision navigation using multi-frequency positioning; – Map tile caching for offline cartography; – Push notifications on FCM with priority messaging support; – OAuth 2.0 for profile data protection; – RESTful API for integration with medical/safety services during pilgrimage.
Prayer and Qibla	– geolocation via fused location provider / Core Location for precise location; – magnetometer and accelerometer for calculating device orientation; – prayer time calculation algorithms: implementation of ISNA, MWL, Umm Al-Qura standards in the form of custom classes; – interface adapted for wearable devices; – integration with the system calendar via Intent API.
Quran and Hadith	– local databases (SQLite, Room persistence library) for saving texts and bookmarks; – text indexing via Trie-trees or FTS; – streaming audio with HLS; – media decoding via ExoPlayer or AVFoundation; – data encryption (AES-256) for protecting text/audio files; – multilingual localization module via i18n/built-in resources.

End of table 1.1

Dua, azkar and tasbih	– counters based on gesture recognizer API for tracking touches; – offline access via preloaded JSON/XML files; – local notifications via AlarmManager / UNUse– counters based on gesture recognizer API for tracking touches; – offline access via preloaded JSON/XML files; – local notifications via AlarmManager; – simple sound support via system media players with low-latency mode; – saving statistics on local or cloud storage.rNotificationCenter; – simple sound support via system media players with low-latency mode; – saving statistics on local or cloud storage.
Educational platforms	– streaming services; – gamification via custom achievements engine or integration with Google Play Games Services / Game Center; – online tests with real-time results on Node.js/PHP backend + WebSocket data exchange; – analytics systems: Google Analytics for Firebase, AppMetrica; – CDN for delivering large amounts of video/audio content.
Information services	Information services – Google Places API, OpenStreetMap Nominatim for working with locations; – chats/forums via WebSocket or Firebase Realtime DB; – OAuth 2.0 / OpenID Connect for single sign-on; – implementation of multi-level map layers via Leaflet / Mapbox GL; – notification systems via pub/sub models.
Hybrid platforms	– microservice architecture using REST API / GraphQL API for flexible interaction building; – cloud storage for user data and media; – data synchronization between devices via Cloud Firestore / AWS DynamoDB; – Edge computing for processing sensitive data locally; – CI/CD for application updates.

1.2 Problems and limitations

Despite significant progress in the development of mobile applications designed to meet the religious needs of Muslim users, current solutions remain far from ideal. Existing platforms face numerous technical, functional, and organizational challenges that considerably impact user experience, data security, and service accessibility [15]. Key issues include insufficient functionality and complexity, limited localization options, inadequate compliance with modern security standards, lack of guaranteed content authenticity, low levels of inclusivity, and suboptimal technical performance.

Many mobile applications suffer from limited functionality, typically focusing on individual tasks such as determining prayer times, reading the Quran, or displaying azkars. This lack of an integrated approach compels users to install multiple separate apps to fulfill different religious needs, thereby diminishing overall usability [16]. Another significant challenge is the low level of localization: many applications fail to account for regional linguistic nuances, variations in prayer time calculation methods, or local religious customs. This oversight creates difficulties for users across diverse countries and cultural backgrounds, reducing the software's versatility and relevance [17].

Protecting users' personal information and ensuring data privacy remain critical challenges for developers of religious applications. Research indicates that many apps incorporate third-party trackers, lack up-to-date encryption protocols, and do not offer two-factor authentication. These shortcomings increase the risk of data breaches and undermine user trust [15, 18].

Therefore, guaranteeing security and adherence to information protection standards remains one of the industry's foremost challenges.

Another significant issue for many religious applications is their lack of compliance with contemporary international and local data protection laws, such as the GDPR in EU countries.

Insufficient transparency regarding the collection, storage, and processing of personal data, along with the absence of clear mechanisms for data deletion upon user request, undermines user trust and exposes developers and platform owners to potential legal risks.

The issue of content reliability stems from the absence of systematic verification of information sources. An analysis of popular Islamic lifestyle applications revealed a lack of mechanisms for guaranteed verification and limited opportunities for collaboration with authoritative religious organizations [16].

Outdated interface design and poor user experience have been highlighted in a usability study of Islamic learning applications, especially affecting children and elderly users. These findings point to the need for improved navigation, interface adaptation, and better management of advertisements [19]. Furthermore, many applications fail to comply with WCAG 2.1 accessibility standards, lacking features such as high-contrast modes, adaptive fonts, voice control, and text alternatives. This significantly restricts accessibility for users with disabilities [17].

Technical limitations continue to pose significant challenges. Many applications require a constant Internet connection and lack support for offline-first approaches and caching mechanisms (such as IndexedDB or SQLite), which is particularly problematic in regions with unstable digital infrastructure [17]. Additionally, insufficient optimization for low-end devices results in excessive memory usage, higher energy consumption, and overall reduced performance.

At the architectural level, many applications suffer from a monolithic design, lacking modularity and the implementation of design patterns such as dependency injection or observer. Furthermore, the absence of edge computing capabilities complicates scalability, maintainability, and the enhancement of user privacy [16].

Finally, insufficient integration with wearable devices (WearOS, watchOS) and voice assistants restricts mobility and usability, limiting the full potential of modern mobile solutions [16].

Table 1.2 – Functionalities of the main categories of mobile applications for Muslims.

Function category	**Main features**	**Application examples**	**Advantages**	**Disadvantages**
Prayer and Qibla	Prayer time calculation, reminder, Qibla compass	Muslim Pro, Athan	High accuracy, support for many calculation methods	Advertising in free versions, battery consumption
Quran and Hadith	Reading, audio, translations, search	Quran Majeed, Ayat, Islam 360	Availability of multilingual content, listening option	Not all apps have authentic sources and certification
Dua and azkar	Prayer libraries, audio, tasbih counter	MyDuaa, Muslim Assistant	Convenience for daily practice, integration of counters	Lack of prayer personalization, sometimes outdated interface
Hajj and Umrah	Interactive maps, routes, tips	Hajj Navigator, SmartHajj	Pilgrimage support, GPS navigation	Limited number of language versions, internet required
Educational and motivational	Video, audio, podcasts, tests	Muslim Central, Niyyah	Expanding knowledge, convenient access	Limited number of language versions, internet required

End of table 1.2

Information services	Locations of mosques, halal establishments	WeMuslim	Travel convenience, time saving	Data needs frequent updating

1.3 Development prospects

Analysis of existing solutions demonstrates that the market for mobile applications targeting the Muslim audience holds significant potential for further development. A key direction is the creation of universal platforms that integrate ritual, educational, social, and informational services. These platforms should consider the cultural and linguistic characteristics of different regions and offer personalized content tailored to the individual needs of users.

The implementation of artificial intelligence to offer personalized recommendations on religious practices, adapt educational materials to the user's proficiency level, and automatically generate optimized routes during Hajj holds great promise. Integration with wearable devices, voice assistants, and augmented reality technologies can significantly enhance the usability and accessibility of applications.

Furthermore, employing advanced encryption methods, two-factor authentication, and verifying the authenticity of religious content through partnerships with official religious institutions should become standard practices in future solutions.

A promising direction involves implementing hybrid architectures based on microservice principles, utilizing RESTful and GraphQL APIs to enable flexible and efficient data exchange between platform modules and third-party services. Leveraging containerization technologies such as Docker and orchestration tools like Kubernetes can simplify system scaling, automate update deployments through CI/CD pipelines (e.g., GitLab CI, Jenkins), and enhance stability under heavy user loads.

It is anticipated that edge computing solutions will be deployed for local processing of sensitive data – such as geolocation and prayer activity – thereby reducing latency, optimizing network traffic costs, and enhancing user privacy. Communication modules employing protocols like MQTT or WebSockets will enable more efficient real-time operation of chats and community interactions. To bolster security, modern authentication standards (OAuth 2.0, OpenID Connect), robust data encryption methods (AES-256, TLS 1.3), and multi-level access control policies (RBAC) will be integrated. Emphasis should also be placed on developing multi-platform applications using frameworks like Flutter or React Native, which accelerate product releases and ensure a consistent user experience across diverse devices. Integration with augmented reality services (ARCore, ARKit) will open new possibilities for visualizing rituals and educational content. In analytics, the adoption of big data systems for processing and personalizing behavioral models is expected to grow, alongside on-device machine learning techniques that generate recommendations locally without transmitting sensitive data to servers.

An important trend is the implementation of WCAG 2.1 standards throughout all stages of development, ensuring accessible interfaces for users with special needs. To achieve optimal performance, the use of Content Delivery Networks (CDNs) for distributing heavy media content, combined with lazy loading techniques, helps to reduce device resource consumption and improve loading times.

Conclusion on Chapter 1

The first section analyzed modern mobile applications designed for the Muslim audience. Existing solutions were systematized, and the main categories of applications were identified, including services for Hajj and Umrah, prayer time and Qibla direction, Quran and Hadith reading, dua and azkar, educational platforms, and hybrid solutions.

For each category, key functionalities were examined, along with the technologies employed – such as GPS, encryption, streaming services, and local databases – and features related to architecture and data protection were analyzed.

Particular attention was given to the challenges facing the market, including fragmented functionality, insufficient localization for the cultural and linguistic specifics of different regions, non-compliance with modern security and privacy standards, and the lack of effective verification of religious content. It was also identified that most applications need enhanced offline capabilities, better optimization for low-performance devices, and improved integration with wearable gadgets and voice assistants.

Based on the analysis, promising directions for the development of mobile solutions targeting the Muslim community have been identified. These include the adoption of microservice architecture, leveraging artificial intelligence for personalized experiences, utilizing edge computing, integrating augmented reality technologies, and developing cross-platform applications.

Additionally, implementing accessibility standards in accordance with WCAG 2.1 is essential. The analysis underscores the importance of creating universal platforms that combine functionalities across various aspects of religious life while meeting contemporary demands for security, inclusiveness, and cultural adaptation.

Thus, the first section established an analytical and theoretical foundation for the development and implementation of a comprehensive mobile application designed to meet both the current technological standards and the spiritual needs of the Muslim community.

CHAPTER 2. CHOOSING ARCHITECTURE AND TOOLS FOR MOBILE APPLICATION DEVELOPMENT

This section explores the fundamental requirements for a mobile application's operation, compares various implementation approaches, justifies the selection of an architectural solution, and identifies the key technologies needed for development. Additionally, it analyzes the characteristics of handling religious content and the particularities of integrating Islamic functionalities within the digital ecosystem.

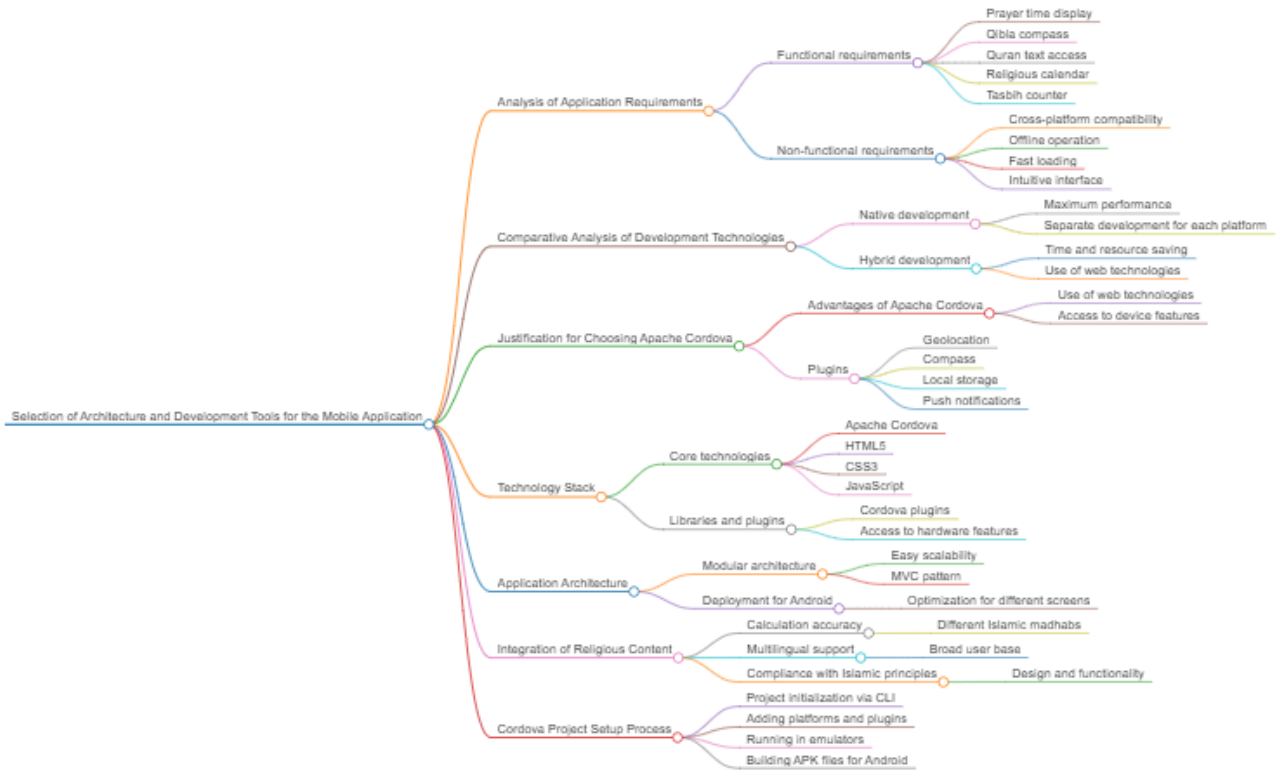


Figure 2.1 – Choosing architecture and tools for mobile application development

2.1 Analysis of mobile application requirements

Requirements analysis is a crucial stage in software development planning, especially for mobile applications designed for religious purposes, where accuracy, usability, and adherence to spiritual principles are paramount. Such an application should equip Muslim users with essential tools to support their daily spiritual practices – prayers, Quran reading, reminders, and other acts of worship. The informed selection of technologies and architectural solutions relies on a thorough analysis of both core functional requirements and non-functional constraints. Therefore, it is essential to focus on key aspects, including functional requirements that specify what the application must accomplish, and non-functional requirements that address overall system qualities such as performance, usability, and compatibility.

A detailed discussion of both functional and non-functional requirements is provided in the following sub-sections.

2.1.1 Functional requirements

Functional requirements form the foundation of software development, defining how the application should behave in response to user interactions. For an Islamic mobile application designed to serve as a personal digital assistant in daily religious life, these requirements specify the essential features that determine the app's core usefulness, usability, and overall value to its users.

One of the fundamental functions of the application is to accurately determine and notify the user of the five obligatory daily prayer times: Fajr, Dhuhr, Asr, Maghrib, and Isha. Prayer time calculations should be performed automatically, based on the user's geographical location obtained via GPS or other geolocation methods.

Therefore, supporting various Islamic schools of thought and prayer time calculation methods – such as Umm al-Qura, Muslim World League, and the Egyptian General Authority – is essential.

Additionally, the application should allow users to independently select and modify these calculation parameters within the settings.

An essential prerequisite for performing prayer is accurately determining the Qibla direction. This functionality is implemented by utilizing the mobile device's sensors – specifically, the magnetometer and accelerometer – accessed through appropriate interfaces such as Cordova plugins. The system must deliver real-time, precise directional information, compensating for potential magnetic interference and mitigating errors caused by external environmental factors.

Access to the Quranic text is another core feature that must be implemented with a high standard of usability and content completeness. Users should be able to read the entire Quran, organized by surahs and verses, with support for multiple translations including Arabic, English, Turkish, Ukrainian, and more. The interface should offer intuitive navigation, as well as features such as text search, zooming, and customizable display styles to enhance the reading experience.

Another key component is the integration of an Islamic calendar that enables users to track important religious dates such as Ramadan, Eid al-Fitr, Eid al-Adha, and Mawlid. This calendar must be based on the Hijri lunar system and support synchronization or conversion between Hijri and Gregorian dates. Additionally, it should account for the user's local time zone and provide options for personalizing events and reminders.

The digital tasbih (subha) counter function enables users to record the number of recitations of specific phrases or prayers during meditation or worship. It should feature an intuitive interface for easy counting, allow users to reset counts, and save historical data for tracking progress over time.

Special attention should be given to the daily spiritual inspiration feature, which presents users with a selected hadith or dhikr each day. This content can be stored locally within the app or dynamically updated via an Internet connection to provide fresh material.

This functionality serves to boost the user's spiritual motivation and fosters a continuous sense of support in their daily religious practice.

The final, yet equally important, feature is the notification system. It should support both local and push notifications to inform users about prayer times, upcoming significant dates or events, and reminders to complete or continue tasbih sessions. A flexible settings interface must enable users to customize which events trigger notifications and select preferred times for receiving reminders.

Thus, the functional requirements for an Islamic mobile application create a comprehensive framework aimed at fulfilling both the spiritual and practical needs of its target users. Successfully implementing these requirements ensures the application's high quality, widespread demand, user trust, and adherence to the religious context.

2.1.2 Non-functional requirements

Non-functional requirements, unlike functional ones, do not specify particular behaviors or features of the application but instead define its overall qualities – those attributes that influence the user's experience and interaction with the software product. For a mobile application designed for a Muslim audience, these non-functional characteristics are just as critical as the core functionalities. They ensure convenience, reliability, security, and accessibility, which are especially vital for an app serving as a daily spiritual companion [23].

One of the primary requirements is ensuring cross-platform compatibility, meaning the application must support the most widely used mobile operating systems – Android and iOS.

This approach maximizes audience reach by eliminating technical barriers to accessing religious content. To achieve this, a hybrid development framework based on Apache Cordova is employed, allowing the creation of mobile applications using a single codebase built with JavaScript, HTML, and CSS.

This solution significantly streamlines development, updating, and maintenance processes, reducing both time and resource expenditures.

Particular attention is given to the application's autonomy, which is crucial in environments with limited or unstable Internet connectivity. Core features – such as displaying prayer times, accessing Quranic verses, using the tasbih counter, or reading daily hadiths – must remain accessible offline. This is achieved by utilizing local storage solutions, including embedded JSON files or on-device databases, ensuring uninterrupted access to essential spiritual resources.

Ensuring high application performance is critical. The interface must respond swiftly to user actions, maintaining minimal latency even on devices with limited processing capabilities. To achieve this, several technical strategies are employed, including JavaScript code optimization, minification and compression of styles and scripts, effective data caching, and prudent management of device resources. This is particularly important when utilizing sensors like the compass or GPS, where minimizing power consumption is essential to preserving device autonomy and providing a seamless user experience.

The user interface must be accessible and intuitive, catering to a diverse audience that ranges from tech-savvy individuals to first-time users. The design should be clean and functional, incorporating Islamic visual motifs and using highly readable fonts – especially for Arabic script. Clear navigation structures and localized prompts are essential to provide cultural relevance and foster a sense of familiarity and comfort [27].

Multilingual support is a crucial feature that broadens the application's accessibility to Muslims from diverse linguistic backgrounds. The interface should be available in key languages such as Arabic, English, Ukrainian, Turkish, Urdu, and others, tailored to the target user base. Localization must encompass not only navigation and interface elements but also all content, including prayers, texts, and notifications.

The application should enable dynamic language switching without requiring a restart, enhancing user convenience and engagement [27].

Security and privacy concerns are paramount in today's digital landscape. The application must ensure that personal user data is never transmitted to external servers without explicit consent. Sensitive information – such as tasbeeh settings, selected languages, and recently recited verses – should be stored locally and safeguarded against loss or unauthorized access. Furthermore, the app should request only the minimal system permissions necessary for its core functionalities, avoiding unnecessary access that could compromise user privacy.

Additionally, to enhance security, the application employs encryption for local storage of users' personal and religious data – such as settings, reading history, and personal bookmarks. Modern symmetric encryption algorithms are used to protect this information, ensuring that even if the device falls into unauthorized hands, the data remains inaccessible [24].

Regarding data collection, the application follows a strict data minimization policy – no personal information is transmitted to third-party servers without the user's explicit, informed consent. All features involving data collection or transmission, such as push notifications, are implemented with full transparency and offer users the option to opt out at any time. This approach aligns with current international standards on personal data protection, particularly GDPR requirements, which is crucial for applications targeting a global audience [24].

Additionally, data integrity validation mechanisms are implemented to safeguard critical religious content (e.g., Quranic verses, hadiths) against accidental or malicious corruption. These mechanisms utilize checksums or digital signatures to verify the authenticity and integrity of the information during storage and updates.

Finally, reliability and fault tolerance are crucial attributes of the application. In the event of an unexpected shutdown or abrupt closure, all essential user data – such as the last Surah read, Tasbeeh progress, and personal settings – must be preserved. This is ensured through an automatic, periodic save mechanism that operates seamlessly in the background without requiring user intervention.

Finally, reliability and fault tolerance are essential qualities of the application. To safeguard user data in the event of an unexpected shutdown or abrupt termination, all critical information – such as the last Surah read, Tasbeeh progress, and personalized settings – must be preserved. This is typically achieved through an automatic, periodic background save mechanism that operates transparently without interrupting the user experience.

2.2 Comparative analysis of development technologies

At the design stage of a mobile application, selecting the appropriate technological approach – whether native, hybrid, or cross-platform – is of critical importance. Each method offers distinct advantages and disadvantages, which must be carefully weighed against the target audience, budget limitations, development timeline, and specific requirements for functionality and performance. This subsection presents a comparative analysis of the two primary approaches – native and hybrid – in the context of developing an Islamic mobile application. Key considerations include the need for extensive functionality, reliable offline support, integration with hardware sensors (e.g., GPS, compass), and a multilingual user interface.

Native development entails creating separate applications for each target platform using platform-specific programming languages and development environments. For Android, this typically involves Java or Kotlin within the Android Studio environment, while for iOS, Swift or Objective-C is used in conjunction with Xcode.

The primary advantage of native development lies in maximum performance and full utilization of device capabilities. Native applications demonstrate high stability, seamless access to all operating system APIs, and optimal resource efficiency. This enables the implementation of complex animations, smooth and responsive user interfaces, effective memory and battery usage, as well as direct integration with hardware modules – such as the compass, GPS, or accelerometer – through native system APIs [20].

However, the native development approach also presents a number of significant drawbacks, particularly when cross-platform support is required. Applications must be developed, tested, and maintained separately for each operating system, which substantially increases the demand for resources – both in terms of time and personnel. Moreover, native development necessitates specialized expertise in platform-specific technologies, such as Kotlin for Android and Swift for iOS. This complexity can hinder team collaboration and raise overall development costs. Consequently, the native approach may be less appropriate in scenarios involving limited budgets, tight deadlines, or the need for rapid deployment across multiple platforms.

In contrast, hybrid development relies on building mobile applications using standard web technologies – such as HTML5, CSS3, and JavaScript – which are encapsulated within a native container to enable access to device-level functionality. Popular frameworks supporting this approach include Apache Cordova, Ionic, and PhoneGap. In a hybrid model, the application functions primarily as a web page rendered inside a WebView component, which is embedded in a native wrapper compatible with both Android and iOS platforms. This architecture allows for faster development cycles and simplified maintenance, as a single codebase can serve multiple platforms simultaneously.

The primary advantage of the hybrid development approach lies in its ability to utilize a single, unified codebase that operates seamlessly across multiple mobile operating systems.

This significantly reduces both development time and costs, streamlines the process of updating and maintaining the application, and facilitates rapid project scaling. Access to native device functionality is achieved through Cordova plugins, which serve as intermediaries between JavaScript code and the platform-specific APIs. As a result, hybrid applications are capable of supporting essential features such as geolocation, compass access, local storage, and push notifications, thereby offering a user experience that closely approximates that of native applications.

Among the notable limitations of the hybrid approach is the potentially lower performance compared to native applications, particularly in scenarios involving intensive data processing or complex graphical rendering. This is primarily due to the reliance on WebView components and the abstraction layers required to bridge web code with native device capabilities. However, in the case of applications with primarily informational functionality – such as religious apps centered on text content, notifications, and simple interactions – these performance drawbacks are generally negligible. Moreover, skilled developers can apply a variety of optimization techniques (e.g., code minification, lazy loading, and plugin management) to ensure that the end-user experience remains smooth and responsive, often indistinguishable from that of a native application.

Considering the above, we can conclude that the hybrid approach is more appropriate for implementing the task. It allows us to simultaneously ensure cross-platform compatibility, reduce development costs, support functions related to hardware sensors, and create a scalable and flexible application [21, 22].

Table 2.1 – Comparison of native and hybrid approaches

Criterion	Native development	Hybrid development
Technologies	Java/Kotlin (Android), Swift/Obj-C (iOS)	HTML5, CSS3, JavaScript + Cordova/Ionic

End of table 2.1

Criterion	Native development	Hybrid development
Productivity	High (maximum resource efficiency)	Moderate (may decrease with complex tasks)
Accessing device features	Full, via system APIs	Through Cordova plugins
Code reuse	Full, via system APIs	Yes, a single code base
Development speed	Low (longer due to individual projects)	High (universal development)
Development cost	High (more resources)	Lower (less time and human costs)
Support and updates	Complicated	Simplified
Testing	Separately for each platform	Unified testing environment
Multilingual support	Complete, but more difficult to implement	Simplified through web technologies
Offline support	Yes	Yes, through local storage
Working with hardware sensors	Directly	Through plugins (may be limited)

2.3 Development platform

In the process of selecting a technological platform for implementing a hybrid mobile application, particular attention was given to several critical factors. These include the level of support for device hardware functions (such as GPS, camera, compass, and biometric sensors), the availability and maturity of development tools, the activity and size of the developer community, the stability of the platform over time, and the potential for future project scaling. A well-balanced consideration of these aspects ensures not only the technical feasibility of the application but also its sustainability, maintainability, and ability to adapt to evolving user needs and technological trends.

In the contemporary landscape of mobile development, several hybrid and cross-platform frameworks are widely utilized, including Apache Cordova, Ionic, React Native, and Flutter.

Each of these platforms offers distinct advantages and limitations that influence their suitability for various types of applications.

Apache Cordova is known for its simplicity, long-standing presence in the industry, and broad plugin support. However, its popularity has declined in recent years due to performance constraints and the challenges of implementing complex and responsive user interfaces, which are increasingly expected by modern users.

Ionic, which can be based on Cordova or the newer Capacitor engine, provides a rich UI component library and tools for building visually appealing applications. While it offers improved interface design capabilities, it still relies on WebView-based rendering, which may impact performance and limit direct access to native device APIs.

React Native stands out by delivering near-native performance and offering a vibrant and active developer community, along with a robust ecosystem of pre-built components. Nevertheless, it requires in-depth knowledge of JavaScript, along with familiarity with native development concepts, which may present a steep learning curve. Integration with existing web infrastructures may also pose additional challenges.

Flutter, developed by Google, has experienced rapid adoption due to its high performance, modern development tools, and the use of a single programming language (Dart) across platforms. It allows for the creation of highly responsive UIs with a consistent look and feel. However, plugin support for niche hardware features is still developing, and the use of Dart, while powerful, can represent a barrier to entry for developers with a background primarily in web technologies.

Given the specific requirements of the project – namely, multilingual support, offline functionality, and integration with hardware sensors, all within a framework of moderate user interface complexity – Apache Cordova emerged as the most appropriate technological choice.

This decision is primarily justified by Cordova's high compatibility with standard web technologies (HTML, CSS, JavaScript), its ease of deployment across platforms, and the extensive library of plugins available for accessing native device features. Moreover, the low entry threshold for developers and the minimal overhead in terms of infrastructure and learning curve further reinforce its suitability.

While more modern alternatives such as Flutter and React Native offer superior performance and advanced UI capabilities, they were deemed excessively complex and resource-intensive for the goals of an informational religious application. Their advanced features and higher development demands were considered redundant in this context. Consequently, Cordova was selected as it offers an optimal balance between development speed, functional coverage, and team resource efficiency for the 2024–2025 planning horizon.

Table 2.2 – Comparison of hybrid development platforms.

Criterion	Apache Cordova	Ionic	React Native
Plugin support	Wide selection of Cordova plugins	Uses Cordova /Capacitor plugins	Through Native Modules or third-party libraries
Working with hardware APIs	Due to plugins, sometimes refinements are needed	Capacitor provides modern APIs on top of Cordova	Native access via JS bridges
Stability	Due to plugins, sometimes refinements are needed	Stable, active development	High, but there may be failures during updates
Community activity	Less active than before	Active, especially with Capacitor	Very active, large global community
Community activity	CLI, Android Studio / Xcode	Native CLI, Visual Studio Code, etc.	Expo, React Dev Tools, CLI
Ease of integration	Integrates well with frameworks (Angular, Vue)	Designed for Angular, Vue/React support	Experience with JavaScript and native development required

End of table 2.2

Scalability	May require optimization for complex UIs	Better scaling thanks to components	Experience with JavaScript and native development required
Target audience	Web application developers	Better scaling thanks to components	Performance-oriented JS developers

Apache Cordova is an open-source framework that enables the development of mobile applications using standard web technologies such as HTML, CSS, and JavaScript. The platform functions by wrapping a web application within a native container, thereby allowing it to interact with the underlying device hardware through a set of specialized JavaScript APIs. This approach facilitates the deployment of a single codebase across multiple mobile operating systems, primarily Android and iOS, eliminating the need for platform-specific development. As a result, Cordova provides a cost-effective and efficient solution for creating cross-platform mobile applications, particularly for projects where performance-critical features are limited and rapid development is a priority.

One of the key advantages of Apache Cordova is its extensive ecosystem of plugins that significantly expand the framework's capabilities by providing access to various hardware features of mobile devices. These plugins enable developers to integrate functionalities such as compass, geolocation, camera, file storage, and motion sensors, as well as implement support for push notifications and local databases.

This modular approach allows the application to achieve a high level of functional completeness while retaining the inherent benefits of a cross-platform architecture (fig. 2.2).

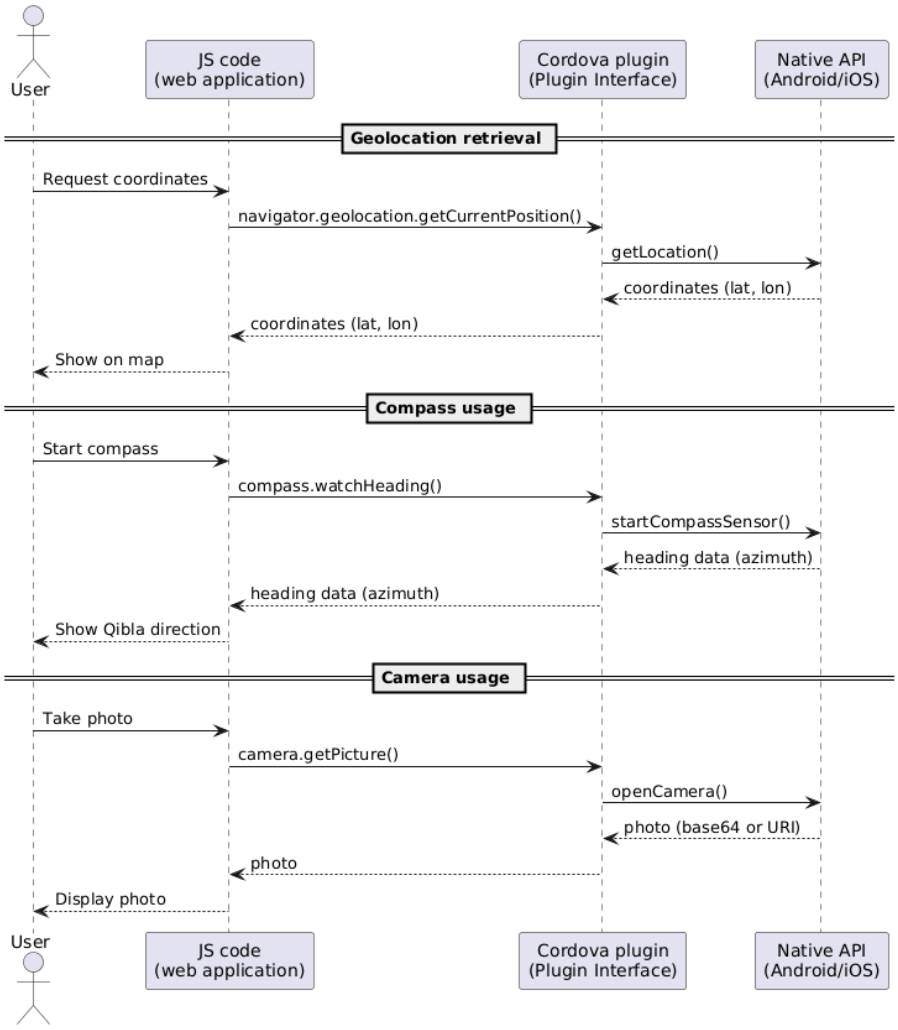


Figure 2.2 – Interaction of JavaScript code with Cordova plugins and native APIs

In addition, a significant advantage of choosing Apache Cordova lies in its maturity and adherence to industry standards. The platform benefits from a large and active community of users, which facilitates rapid resolution of technical issues, abundant availability of educational resources, and consistent ecosystem updates. Moreover, being an open-source project, Cordova integrates seamlessly with popular frameworks such as Angular and Vue.js, thereby enabling the development of sophisticated user interfaces and the reuse of well-established libraries, further enhancing development efficiency and application complexity.

In the context of developing a mobile application with a religious focus – where the primary functionality does not demand intensive graphics processing but requires extensive integration with hardware features, such as determining the Qibla direction or displaying prayer times based on the user's geolocation – Apache Cordova's capabilities are more than adequate. Utilizing this platform ensures broad compatibility across the majority of modern mobile devices, supports application autonomy, and meets essential non-functional requirements, including fast loading times, efficient resource optimization, and an intuitive user interface.

Table 2.3 – Evaluation of the Apache Cordova platform by key criteria

Criterion	**Description / Benefits of Cordova**
Hardware feature support	Through plugins — geolocation, compass, camera, push, storage, motion sensors
Development tools	HTML5, CSS3, JavaScript, CLI support, Angular/Vue.js compatibility
Community activity	Open source software, documentation, lots of training materials, regular updates
Scalability	Easy integration with other frameworks, reuse of libraries
Compatibility and cross-platform	Support for Android and iOS without codebase duplication
Non-functional requirements	High download speed, autonomy, resource optimization, user-friendly interface
Graphics performance	May be inferior to native platforms, but not essential for informational/religious software

However, despite the clear advantages of hybrid development, certain risks and limitations may impact the application's future operation and evolution. A major concern is the gradual decline in community activity surrounding the Apache Cordova platform, which has been noticeable in recent years. This slowdown could result in delayed support for new operating system features – particularly on iOS and Android – and may cause compatibility issues with updated mobile OS versions or restrict access to the latest hardware capabilities.

Another limitation lies in the indirect access to native device capabilities through Cordova plugins. When new sensor types, APIs, or specialized features emerge, there can be delays or challenges in implementing support, which constrains product innovation. Furthermore, the performance of hybrid applications – particularly when handling complex graphics or numerous animations – may fall short compared to native solutions.

Minimizing these risks involves regularly updating the Cordova plugins, closely monitoring official Cordova releases, and preparing contingency plans – such as transitioning to more modern frameworks like Flutter or React Native if requirements change significantly or Cordova support ends. Additionally, designing the application architecture with maximum modularity and loose coupling between components is crucial, as it facilitates potential migration to alternative platforms with minimal time and resource investment.

2.4 Development technology stack

In developing a cross-platform mobile application aimed at fulfilling religious functions and daily use by the Muslim community, a critical design phase involves defining the technology stack. Given the specific functional requirements, user expectations, and mobile device constraints, the selected technologies must not only enable core interface and business logic implementation but also satisfy heightened demands for accessibility, reliability, and performance.

One of the primary requirements was cross-platform compatibility, as the target audience includes users of both Android and iOS devices. Utilizing a single codebase for multiple operating systems significantly reduces development, testing, and maintenance costs. Therefore, a solution based on Apache Cordova was chosen, enabling the creation of applications with standard web technologies – HTML5, CSS3, and JavaScript – while integrating them seamlessly into the native mobile environment.

This approach facilitates effective adaptation of the interface to various screen sizes, which is crucial for implementing an intuitive design and displaying structured religious content, such as Quranic verses, hadiths, and prayer times.

HTML5, as a markup language, establishes the logical structure of the user interface. Its support for semantic elements enhances both usability and the scalability of the codebase. CSS3 handles styling and responsive design, enabling visual compliance with Islamic traditions through restrained color palettes and the use of characteristic Arabic typography, while ensuring easy navigation even on small screens. JavaScript serves as the core tool for implementing application logic, including event handling, page navigation, local data processing, offline mode functionality, and interaction with external APIs.

Apache Cordova acts as a bridge between the web application and the device's native capabilities. Through its plugin system, developers gain access to hardware components such as the GPS module, magnetometer, internal storage, notifications, calendar, and even system settings. This enables the implementation of all essential features for a religious application – such as determining the Qibla direction, calculating prayer times based on the user's geolocation, locally storing tasbih counts or selected surahs, and maintaining data confidentiality by avoiding the transfer of sensitive information to external servers.

In addition to core web technologies, the technology stack incorporates various third-party libraries to enhance functionality and simplify development. For date handling, libraries like Moment.js or its modern alternatives enable precise calculation of religious dates based on the Islamic calendar. To support a multilingual interface, i18next is integrated, providing seamless language switching without page reloads. Mobile-adapted UI frameworks are also employed to implement components that support gesture controls, dark mode, and animations, aligning with contemporary UX design standards.

At the implementation stage, the choice of developer tools is crucial. Visual Studio Code serves as the primary environment for coding and debugging, offering extensive support for Cordova projects through extensions, seamless Git integration, and the ability to debug applications directly on physical devices or emulators. For testing, the Android Emulator is used to simulate device environments, while services like BrowserStack enable functional verification on a wide range of real devices. This setup helps detect and resolve potential performance and compatibility issues early in the development process.

2.4.1 Plugin system

An integral component of an efficient cross-platform mobile application built with Apache Cordova is its plugin system, which grants standardized JavaScript access to a device's hardware resources without requiring native platform code. This approach greatly simplifies development and maintains a unified codebase – crucial when working with limited resources or needing to rapidly scale functionality.

For the implementation of the geolocation component – essential for accurate prayer time calculations – the application utilizes the Cordova Geolocation plugin. This plugin provides the user's current coordinates, enabling the app to adapt prayer schedules according to the selected calculation method (e.g., Umm al-Qura or ISNA) based on precise geographic location. Similarly, to determine the Qibla direction, data from the device's built-in magnetometer is accessed through a dedicated plugin, which processes orientation information. This enables the app to display a reliable compass pointing towards Mecca, even offline – a common use case for the application (fig. 2.3).

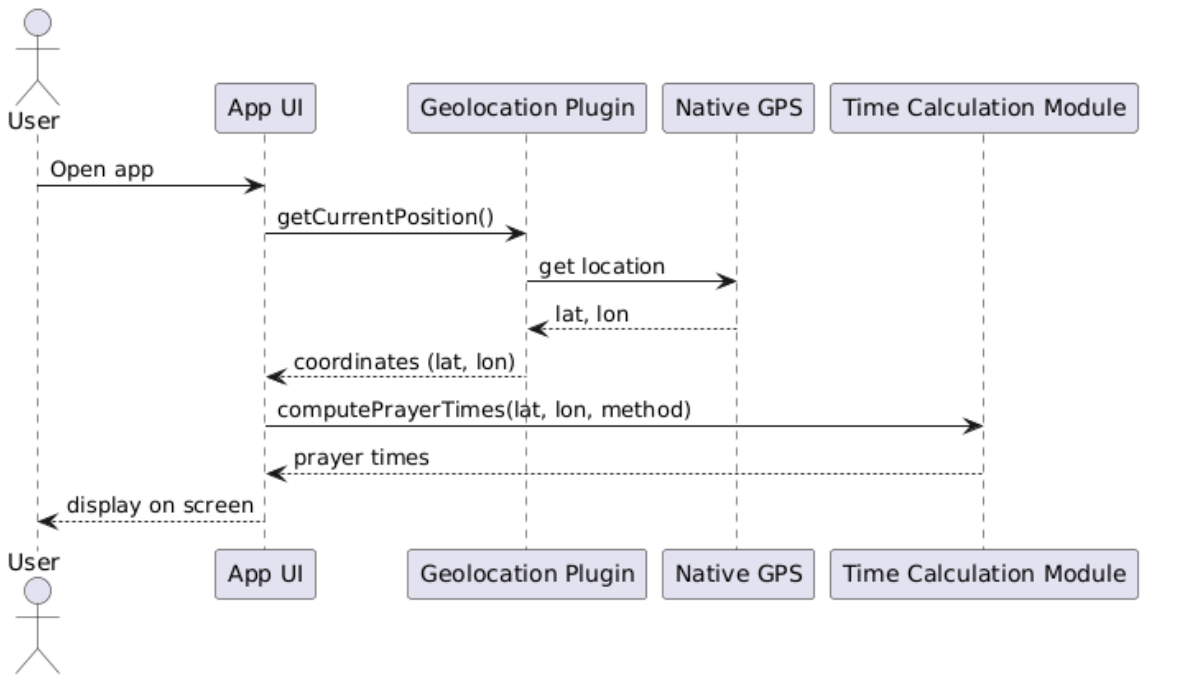


Figure 2.3 – Prayer time calculation sequence diagram

Of particular importance for a mobile application designed for daily use is robust offline functionality. To preserve user data – such as selected surahs, tasbih progress, and language preferences – local storage mechanisms are employed. It is advisable to utilize a file system plugin or integrate libraries that offer an abstraction layer over various browser storage options. This approach not only ensures the persistence of personalized data across sessions but also minimizes the risk of losing context during unexpected interruptions.

The implementation of a notification system, a vital feature for a religious application, warrants special attention. Sending local or push notifications – such as reminders for upcoming prayer times, festive events, or personal alerts – is managed through specialized plugins that interact directly with the device's calendar and handle notification scheduling without relying on external servers. This approach enhances the application's autonomy while ensuring strict compliance with privacy principles, a critical consideration in today's data security landscape [25, 26].

Thus, the Cordova plugin system plays a crucial role in bridging web-based code with the native capabilities of mobile devices. It enables the implementation of core functionalities that not only enhance the user experience but also ensure the application's relevance to its target audience by addressing the unique requirements of its religious theme and supporting the user's daily spiritual practices.

2.4.2 Additional libraries and tools

In developing a cross-platform mobile application – particularly one focused on religious functions and intended for regular daily use – the foundational platform infrastructure alone is not sufficient. The integration of additional libraries and tools plays a vital role in enhancing development efficiency, improving user interaction, and ensuring high-quality rendering of complex content. While these libraries are not strictly mandatory, they often become essential for meeting key non-functional requirements such as internationalization, interface adaptability, data processing precision, and smooth visual performance.

A critical aspect of the application's functionality is managing calendar information that supports both the Gregorian and Hijri calendar systems. To accurately process dates – including displaying religious holidays and prayer times based on the Islamic calendar – it is advisable to utilize libraries capable of handling time zones, localization, and date formatting. While Moment.js has historically been a popular choice, modern alternatives are increasingly preferred due to their smaller size and improved performance. These libraries offer dynamic date formatting, interval calculations between events, and seamless alignment with local time display standards.

Equally important is the issue of internationalization, especially for an application designed for the global Muslim community. In this context, it is advisable to use specialized libraries that enable multilingual support and allow flexible language switching without requiring an application restart.

Among these solutions, the i18next library stands out for its flexible configuration, support for asynchronous loading of language resources, and seamless integration with most popular UI frameworks. Utilizing i18next allows the adaptation of text content and interface elements to the cultural and linguistic preferences of users, which is essential for enhancing usability and inclusivity.

From the perspective of creating an effective visual environment and optimizing user experience, the use of specialized mobile UI frameworks is essential. These frameworks enable the development of interfaces that comply with modern usability standards, including support for gesture navigation, dark mode, adaptive grid layouts, and smooth animated transitions between screens. Notably, Framework7 and Onsen UI offer tools specifically tailored for mobile devices and ensure seamless compatibility with Cordova, making them especially suitable for hybrid app development. By simplifying the layout process, these frameworks allow developers to concentrate more on business logic, thereby reducing the effort required for manual design and styling.

2.5 Application architecture

The architectural design of a mobile application intended for daily religious use must ensure comprehensive functional coverage alongside stability, flexibility, and high performance across a variety of devices. This project is distinguished by its implementation as a cross-platform application leveraging Apache Cordova technology, which encapsulates a web-based interface within a native container. Consequently, the software architecture requires a tailored approach that harmonizes web development principles with the specific demands of the mobile environment.

The modular architecture diagram shown in Figure 2.4 visually represents the structure and interactions among the primary components of the mobile application.

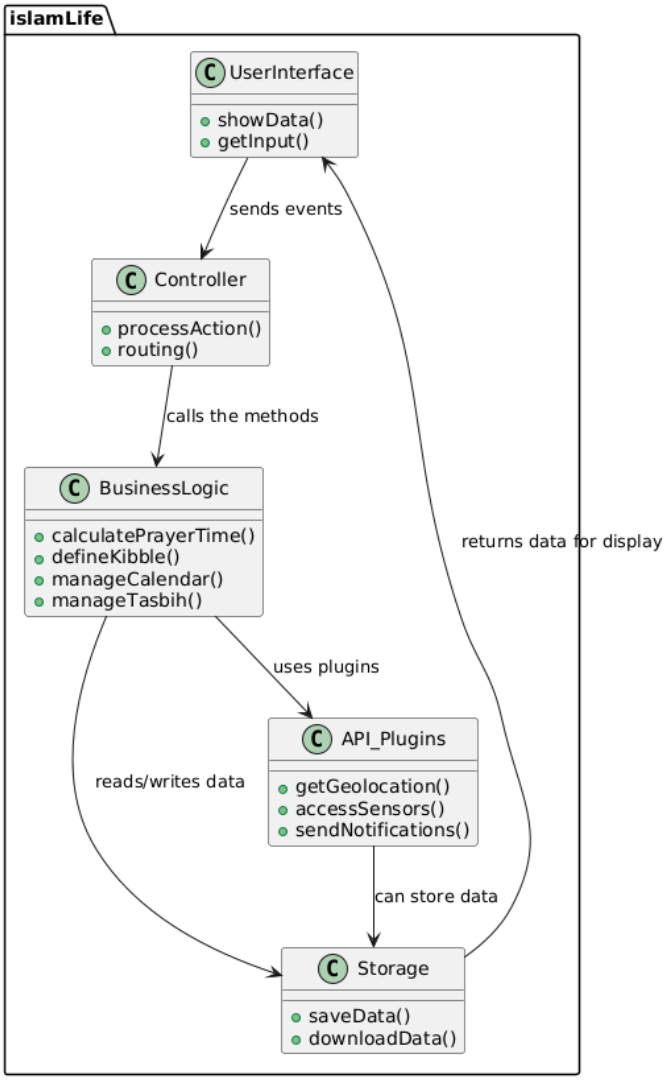


Figure 2.4 – Diagram of the modular architecture of the application

The diagram illustrates the main modules of the application:

– User Interface (UI): Responsible for displaying data to the user, managing interactive elements, supporting multilingualism, and adapting to various screen sizes;

– Controller: Handles user inputs and actions, orchestrates communication between the UI and business logic, and manages routing within the application;

– Business Logic: Contains the core functionalities such as calculating prayer times, determining Qibla direction, processing calendar data, managing tasbih counters, and other religious features;

– Storage: Provides local data persistence for settings, user history, cache, and religious content using technologies like IndexedDB, localForage, JSON, or SQLite;

– API/Plugins: Serves as the bridge between the application and Cordova plugins, enabling access to device hardware resources including GPS, compass, push notifications, and file storage.

The central architectural principle guiding the implementation is modularity. The application is structured around independent functional blocks, each responsible for highly specialized tasks such as calculating prayer times, processing geolocation data, managing local caching, building the user interface, synchronizing data, generating push notifications, and providing access to various device capabilities. This modular approach reduces cognitive complexity during code maintenance, simplifies testing of individual components, and enables flexible scalability when expanding the application's functionality. Each module encapsulates its own logic in isolation, minimizing inter-module dependencies and easing the integration of new technologies or migration to other frameworks as needed.

The logical organization of components follows the MVC (Model–View–Controller) pattern, adapted to the dynamic requirements of mobile web interfaces. The Model serves as the single source of truth, managing all core data including prayer schedules, religious calendars, local settings, user language preferences, and temporary tasbih counts. It also handles access to local storage using technologies such as IndexedDB or localForage, implementing caching mechanisms that enable offline functionality. The Controller coordinates interactions between the user and the model by responding to user inputs and external events, such as geolocation updates, sensor data (e.g., compass readings), language switching, and push notification triggers.

Additionally, it manages routing between different sections of the application's interface. The View represents the interactive user interface, which stays synchronized with the model's state. It is implemented using HTML templates, styled with CSS3, and enhanced with animations, adaptive layouts, and localized content to ensure a seamless and engaging user experience.

Mobile-friendliness poses a significant architectural challenge due to the wide variety of devices on which the application must run. The architecture must be fully responsive to different screen sizes, pixel densities, aspect ratios, and available hardware sensors. While development primarily targets the Android platform, maintaining compatibility with iOS is essential. This requires the use of CSS media queries, layout frameworks such as Flexbox and Grid, and conditional initialization of plugins based on the device's supported features. Additionally, the application incorporates accessibility considerations, including scalable fonts, high-contrast modes, and integration with native operating system accessibility settings.

Equally important is the implementation of fault tolerance within the architecture. To achieve this, the system employs auto-save mechanisms for critical user data – such as the current position in a surah, the status of the tasbih counter, and selected notification preferences. These parameters are securely stored in local storage and automatically restored upon the next application launch, even after unexpected crashes or forced closures by the operating system. Additionally, a robust error-handling system notifies users of any temporary unavailability of certain features without causing major disruptions to the overall application workflow.

The resulting architecture delivers comprehensive functional coverage tailored to the specific needs of a religious application while ensuring a high degree of modularity. This modular design enables scalability across diverse user groups and usage contexts, laying the foundation for a long project lifecycle, operational stability, and seamless evolution in response to future challenges and shifting user requirements.

Conclusion on Chapter 2

The second section offers a comprehensive analysis of the architectural and technological choices involved in developing a cross-platform religious mobile application. It outlines the functional and non-functional requirements that shape the technical and cultural profile of the product, ensuring its relevance and practical value for the Muslim community.

A comparative analysis of the primary mobile development approaches – native, hybrid, and cross-platform – led to the selection of a hybrid model based on Apache Cordova as the optimal choice for implementing a multifunctional, accessible, autonomous, and scalable solution. The chosen technology stack – HTML5, CSS3, and JavaScript, complemented by the Cordova plugin system – delivers the full range of required features, including precise prayer time calculations, Islamic calendar integration, multilingual support, a flexible user interface, and adherence to privacy standards.

A crucial stage involved developing a modular architecture based on the MVC pattern, ensuring flexibility, scalability, and stability amid evolving functional and non-functional requirements. Particular emphasis was placed on integrating religious content: guaranteeing the accuracy and reliability of sacred texts, accommodating diverse schools of thought and doctrines, implementing cultural and linguistic adaptations, and respecting the ethical norms of Islamic tradition throughout the design.

The practical stages of the project development are outlined – from initializing the application structure to integrating plugins, configuring settings, and compiling for target platforms. This section thus establishes a solid theoretical and practical foundation for the subsequent implementation, testing, and enhancement of a mobile application that aligns with modern technological standards and the spiritual needs of the Muslim community.

CHAPTER 3. MOBILE APPLICATION IMPLEMENTATION

The third section details the practical implementation of the mobile application developed according to the requirements and architectural decisions established in the previous stage. It covers the setup of the development environment, structuring of software modules, integration of religious content, and the implementation of mechanisms to ensure the application's functionality, security, and accessibility. Special emphasis is placed on the realization of key features with considerations for usability, compliance with non-functional requirements, multilingual support, offline capabilities, and performance testing across diverse devices.

3.1 General development organization

The mobile application development process was conducted following principles of phased development, transparency, and manageability. Initially, the project structure was set up using the Cordova CLI, enabling rapid creation of a basic template with essential configuration files. The core web resources and application code reside in the www directory, while the config.xml file serves as the central configuration point, defining key project properties such as the package identifier, application name, required permissions, integrated plugins, and the list of target platforms (fig. 3.1).

```
<widget xmlns="http://www.w3.org/ns/widgets" xmlns:cdv="http://cordova.apache.org/ns/1.0"
 xmlns:android="http://schemas.android.com/apk/res/android" id="org.islamlife.lessons" version="7.1.1">
   <preference name="loadUrlTimeoutValue" value="700000"/>
   <name>islamLife</name>
   <description>Application that provides useful informations and functions for muslims around the world</description>
   <author email="info@islamic-lessons.org" href="https://islamic-lessons.org"> Islamic lessons team </author>
   <preference name="DisallowOverscroll" value="true"/>
   <config-file target="AndroidManifest.xml" parent="/manifest">
     <uses-permission android:name="android.permission.SCHEDULE_EXACT_ALARM"/>
   </config-file>
   <icon src="www/img/icon.png" platform="android" density="mdpi"/>
   <content src="index.html"/>
   <access origin="*"/>
   <allow-intent href="http://*/*"/>
   <allow-intent href="https://*/*"/>
 </widget>
```

Figure 3.1 – Configuration file for managing project properties

After establishing the basic project structure, the next step was to add Android as the target platform, prioritizing access to the largest segment of the potential user base. This was accomplished using the cordova platform add android command, which automatically configures the environment for integration with the Android SDK and generates the necessary build and testing scripts.

An important component of the development organization was the integration of the necessary Cordova plugins, which provide access to the hardware capabilities of the device. Plugins were added via the command interface, and the project included both official Cordova Community solutions and third-party extensions adapted to specific functional needs – in particular, working with geolocation, sensors, the file system, push notifications and other services. All connected components were recorded in the project configuration file, which provided version control and simplified further support. The structure of such a project is shown in Figure 3.2.

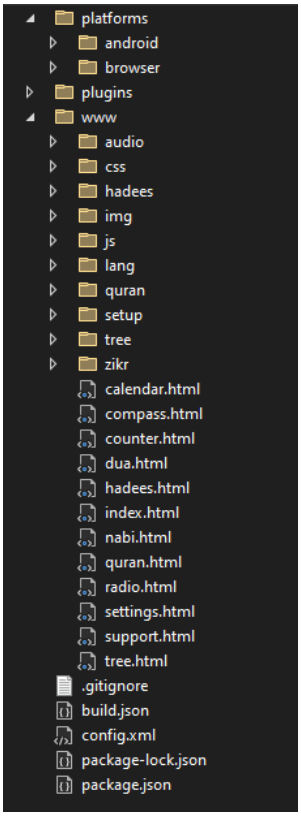


Figure 3.2 – Project structure

Team collaboration was structured through the use of Git for version control, complemented by project management tools such as Trello and Jira. This setup enabled efficient task allocation, continuous progress monitoring, and prompt issue resolution, while allowing the team to adapt quickly to evolving requirements and shifting priorities.

After integrating the core functional components, comprehensive testing was performed on both emulators and physical devices across various Android versions to ensure correct interface behavior, seamless hardware API interaction, and overall application stability. The final development phase involved building the release APK using the cordova build android command, configuring the application's digital signature to safeguard against unauthorized modifications, and verifying compliance with Google Play Store requirements, including support for the latest Android versions.

This development approach enabled effective project management, fostered flexibility in decision-making, and ensured that all functional and non-functional requirements were met within the allocated resources and timeline.

3.2 Deploying the project environment

At the project environment deployment stage, a primary objective was to ensure development convenience and efficiency, while creating an infrastructure that supports rapid integration of new components and simplifies workflow. Visual Studio Code was selected as the primary development environment due to its extensive support for JavaScript, HTML5, and CSS3 extensions, along with built-in tools for seamless Cordova integration. Additionally, Android Studio emulators were employed to test the application across various OS versions and device types.

The Cordova project was initialized using the Cordova CLI, which generates a basic application template with an optimized directory structure.

At this stage, the main folders are created: www (containing all web resources such as HTML, CSS, JavaScript, and images), platforms (automatically generated after adding target platforms, containing platform-specific settings for Android or iOS), plugins (storing the connected Cordova plugins), and configuration files including config.xml, which defines the global properties of the project.

An important step was integrating essential plugins and third-party libraries to enable access to the device's hardware resources and enhance the application's functionality (fig. 3.3). The main plugins incorporated into the project include:

– cordova-plugin-geolocation for obtaining user coordinates;

– cordova-plugin-device-orientation to work with the compass and determine the Qibla direction;

– cordova-plugin-local-notifications for managing local notifications;

– cordova-plugin-inappbrowser to open external links securely in an in-app browser.

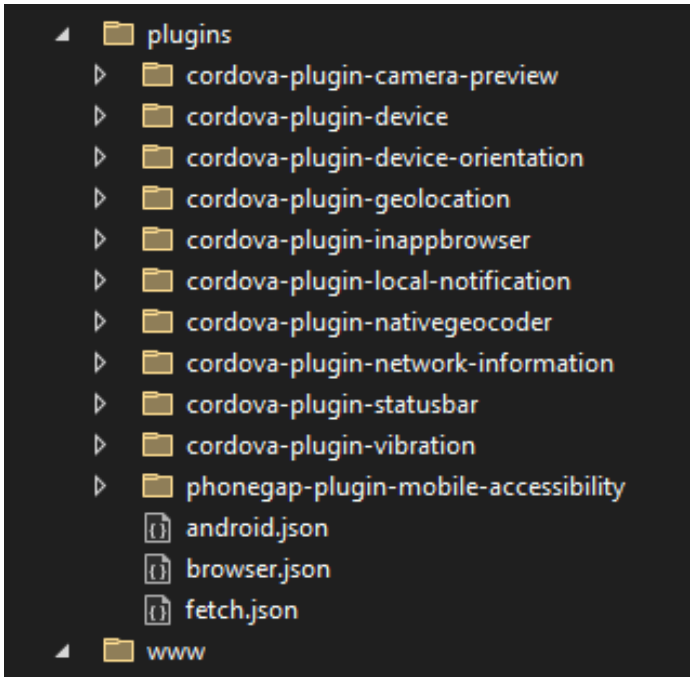


Figure 3.3 – Structure of plugins integrated into the project

In addition to plugins, the project integrates libraries for date handling, multilingual support (using i18next), and UI frameworks that ensure an adaptive and user-friendly interface. The directory structure and source files were organized following principles of modularity and scalability. Separate subfolders and files were created for each functional module – prayer, Qibla, Quran, calendar, tasbih, and notifications – simplifying code navigation and facilitating collaborative development. Special attention was given to the multilingual interface resources: all translation text files are stored as separate JSON files within a dedicated directory, making it easy to add new languages and maintain localization support (fig. 3.4).

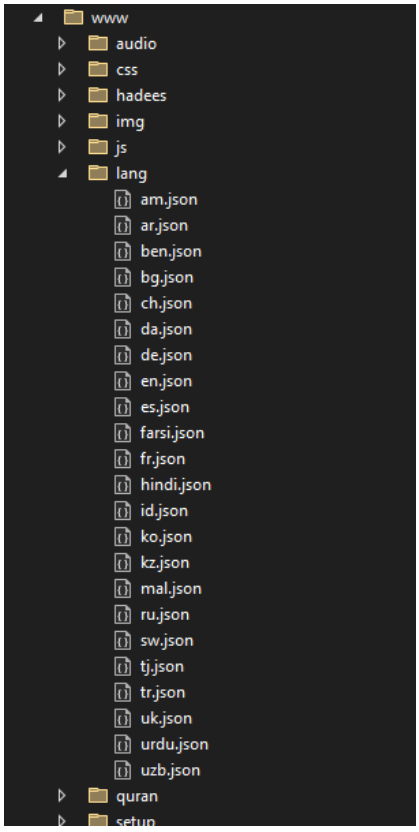


Figure 3.4 – Resource structure for multilingual interface

Thanks to this approach in setting up the project environment, a well-organized codebase was achieved, facilitating easy system scalability and providing a solid foundation for efficient team collaboration and individual development throughout all subsequent stages.

3.3 Implementation of main functional modules

The development of the core functional modules of the mobile application constitutes a pivotal aspect of the project's practical implementation. These components directly influence the app's utility, adherence to religious principles, and overall user experience. This section provides a detailed examination of the creation of individual modules responsible for key tasks: calculating prayer times, determining the Qibla direction, displaying and searching Quranic text, managing the Islamic calendar, operating a digital tasbih counter, and implementing a notification system. Special emphasis is placed on the principles of integrating these modules within the app's overall architecture, leveraging appropriate plugins and libraries, and ensuring flexibility, scalability, as well as compliance with contemporary security and localization standards.

3.3.1 Prayer time calculation module

One key function of the developed mobile application is the prayer time calculation module, which accurately and reliably determines the time windows for the five obligatory prayers: Fajr, Dhuhr, Asr, Maghrib, and Isha. This module is implemented using astronomical algorithms that consider the Sun's position, along with the user's geographical location and local time settings.

Prayer Time Determination Algorithm. The calculation of prayer times is based on accurately identifying specific astronomical events involving the Sun's position. Each prayer time corresponds to a distinct solar criterion:

– Fajr begins when the Sun is at a fixed angle below the horizon before dawn, typically between -15° and -20°;

– Isha is marked by a similar solar angle below the horizon after sunset;

– Maghrib corresponds to the moment of true sunset;

– Dhuhr is determined by the time of astronomical noon, when the Sun reaches its highest point in the sky;

– Asr calculation depends on the length of an object's shadow relative to its height, which varies according to the selected madhhab.

The algorithm uses formulas incorporating solar declination, the equation of time, and the user's geographic coordinates to precisely compute these moments.

Consideration of Islamic Schools of Thought and Calculation Methods. A key aspect of developing a religious mobile application is the adaptation of its logic to the diverse Islamic doctrines and legal schools, known as madhhabs. Although Islam is united by its spiritual foundations, it encompasses multiple authoritative traditions that interpret religious norms differently. These include the main Sunni schools – Hanafi, Maliki, Shafi'i, Hanbali – and Shiite traditions such as Ja'fari.

Each school holds distinct views on prayer times, permissible actions during specific times of day, and calendar calculations. For example, the method of calculating the Asr prayer time differs between the Hanafi and other Sunni schools, while Shiite practices may follow alternative parameters. To respect these variations, the application must allow users to select their preferred school of thought and adjust the calculation algorithms accordingly, ensuring both religious compliance and user personalization.

A clear example of these differences is the determination of Asr (afternoon) prayer time. According to the Hanafi school, Asr begins when the shadow of an object is twice its length, whereas most other schools consider it to start when the shadow equals the object's height. For practicing Muslims, adhering to the correct calculation is a crucial matter of religious accuracy. To accommodate this, the application architecture incorporates a parameterized prayer time calculation module designed with dependency inversion and template configuration principles, enabling flexible adaptation to various jurisprudential methods.

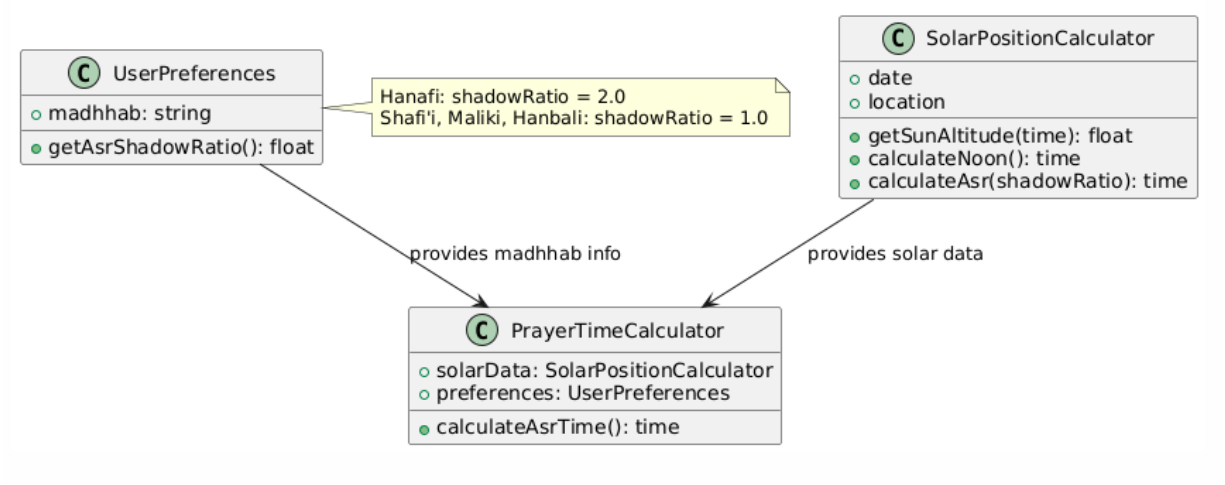
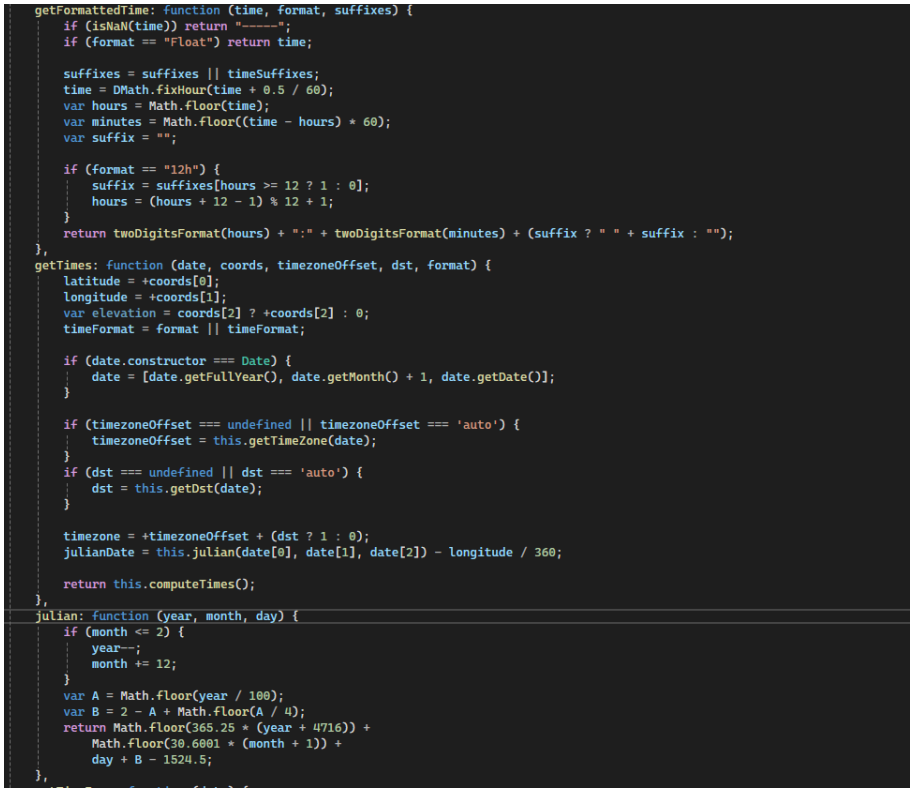


Figure 3.5 – The dependence of the beginning of the Asr prayer on the length of the shadow

Each calculation method – whether it originates from Umm al-Qura University, the Egyptian Geophysical Institute, the Islamic Society of North America (ISNA), or a specific madhhab variant – is defined by a set of parameters: the sun's angle below the horizon for Fajr and Isha prayers, the rule for determining Asr time, and regional adjustments such as those for high latitude locations (fig. 3.5). Users select their preferred method in the application settings upon first launch or may change it at any time later, with all prayer time calculations immediately updated according to the chosen parameters.

While the user interface allows selecting a school of thought or calculation method with a simple choice, this selection triggers a complex system that translates these preferences into precise mathematical functions. These functions incorporate not only angular parameters but also consider factors such as the method for calculating day length, adjustments between true and mean solar time, the presence of daylight saving time, and more. The underlying calculations rely on established astronomical formulas, including the equation of time, solar declination, the hour angle of the equinox, and the solar elevation angle (fig. 3.6).



```
getFormattedTime: function (time, format, suffixes) {
    if (isNaN(time)) return "-----";
    if (format == "Float") return time;

    suffixes = suffixes || timeSuffixes;
    time = DMath.fixHour(time + 0.5 / 60);
    var hours = Math.floor(time);
    var minutes = Math.floor((time - hours) * 60);
    var suffix = "";

    if (format == "12h") {
        suffix = suffixes[hours >= 12 ? 1 : 0];
        hours = (hours + 12 - 1) % 12 + 1;
    }
    return twoDigitsFormat(hours) + ":" + twoDigitsFormat(minutes) + (suffix ? " " + suffix : "");
},
getTimes: function (date, coords, timezoneOffset, dst, format) {
    latitude = +coords[0];
    longitude = +coords[1];
    var elevation = coords[2] ? +coords[2] : 0;
    timeFormat = format || timeFormat;

    if (date.constructor === Date) {
        date = [date.getFullYear(), date.getMonth() + 1, date.getDate()];
    }

    if (timezoneOffset === undefined || timezoneOffset === 'auto') {
        timezoneOffset = this.getTimeZone(date);
    }
    if (dst === undefined || dst === 'auto') {
        dst = this.getDst(date);
    }

    timezone = +timezoneOffset + (dst ? 1 : 0);
    julianDate = this.julian(date[0], date[1], date[2]) - longitude / 360;

    return this.computeTimes();
},
julian: function (year, month, day) {
    if (month <= 2) {
        year--;
        month += 12;
    }
    var A = Math.floor(year / 100);
    var B = 2 - A + Math.floor(A / 4);
    return Math.floor(365.25 * (year + 4716)) +
        Math.floor(30.6001 * (month + 1)) +
        day + B - 1524.5;
},
```

Figure 3.6 – Part of the prayer time calculation code

Thanks to this approach, the application enables users to configure calculation modes according to their specific school of faith, ensuring both high accuracy and adherence to Sharia principles. This prevents the imposition of a single, uniform standard that might conflict with local or denominational practices. Consequently, the system achieves not only technical flexibility but also religious legitimacy, making it suitable for use within the diverse, multicultural Islamic community.

3.3.2 Qibla direction determination module

The Qibla direction module is a core functional element of an Islamic mobile application, providing users with precise guidance to face Mecca during prayer. Its implementation requires seamless integration of the device's hardware sensors, coordination with appropriate Cordova plugins, and the creation of an intuitive, user-friendly compass interface.

Working with Device Sensors. To accurately determine the Qibla direction, the application utilizes data from the mobile device's built-in sensors, specifically the magnetometer (compass) and accelerometer. The magnetometer measures the device's orientation relative to the Earth's magnetic field, while the accelerometer tracks its tilt and position. By combining data from these sensors, the app can precisely calculate the azimuth, ensuring the correct Qibla direction regardless of how the user holds their phone.

The module's algorithm begins by obtaining the user's geographical coordinates (latitude and longitude), typically acquired via GPS or other geolocation methods. Using these coordinates, the azimuth – the angle between true north and the direction to the Kaaba (Mecca) – is calculated by applying a well-known spherical trigonometry formula:

$$\text{Qibla} = \arctan2(\sin(\Delta\lambda), \cos(\phi1)\tan(\phi2) - \sin(\phi1)\cos(\Delta\lambda)) \qquad (3.1)$$

Qibla — Qibla azimuth; $\Delta\lambda$ — difference in longitude;

$\phi1$ — latitude of the observer; $\phi2$ — latitude of the Kaaba;

Interaction with Cordova Plugins. Sensor data is integrated into the mobile application using specialized Cordova plugins. For working with the magnetometer and determining device orientation, the cordova-plugin-device-orientation plugin is employed, providing a standardized JavaScript interface to access real-time azimuth angle, tilt, and rotation data. The cordova-plugin-geolocation plugin is utilized to automatically obtain the user's geographic coordinates (fig. 3.3).

Thanks to these plugins, the application can obtain the necessary data in real time without the need for complex native development. Importantly, all operations are performed locally on the device, with no transmission of geolocation data to external servers, thereby ensuring compliance with modern security and privacy standards crucial for religious applications. Interaction with the plugins is organized through an event-driven model: whenever the device orientation or geolocation changes, the Qibla direction calculations are automatically updated.

Compass Display. A key element of the user interface for determining the Qibla direction is the visualization of the compass. The graphical compass shows the device's current orientation relative to true north and prominently points an arrow towards the Qibla. This functionality is implemented using a specialized UI component designed with adaptability and multilingual support in mind. The compass interface is intended to be intuitive and user-friendly, featuring a clearly distinguishable Qibla arrow, an indication of the north direction, and the option to display additional details such as the azimuth angle in degrees and the distance to Mecca (fig. 3.7).

```
document.addEventListener('deviceready', onDeviceReady, false);

1 reference
function onDeviceReady() {
    load();

    if (localStorage.getItem('color')) {
        const color = localStorage.getItem('color');
        const mode = localStorage.getItem('compass');
        if (mode === 'runcamera') {
            $('html').css('background', 'transparent');
        } else {
            $('html').css('background', color);
        }
        $('.compass').css('background-color', color);
        $('.compassBody').css('box-shadow', `
            0px -5px 10px 0px ${color} inset,
            0px 5px 10px 0px ${color} inset,
            -15px -30px 60px 20px ${color},
            0px 30px 100px 80px ${color}`);
    }

    const compassCircle = document.querySelector('.compass');
    let currentQiblaAngle = null;
    let compassWatchId = null;
    let lastHeading = null; // To track the heading for stabilization checking

    // Check if the modal has already been shown to the user
    const isModalShown = localStorage.getItem('compassCalibrationShown') === 'true';

    // Use stored location if available
    const storedLat = localStorage.getItem('latCoords');
    const storedLon = localStorage.getItem('lonCoords');
    if (storedLat && storedLon) {
        currentQiblaAngle = calcDegreeToPoint(storedLat, storedLon);
        startCompass(currentQiblaAngle);
    }

    // Then request fresh location
    navigator.geolocation.getCurrentPosition(
        function (position) {
            const lat = position.coords.latitude;
            const lon = position.coords.longitude;

            localStorage.setItem('latCoords', lat);
            localStorage.setItem('lonCoords', lon);

            const updatedQibla = calcDegreeToPoint(lat, lon);
            currentQiblaAngle = updatedQibla;
```

Figure 3.7 – Part of the compass code

The compass display is implemented using HTML5 Canvas, enabling smooth animation of the arrow as the device orientation changes. The application continuously captures device movements and updates the graphic in real time, providing a natural and responsive user experience.

Special attention is given to accuracy: since magnetic sensors can be affected by external interference, the application includes calibration instructions and applies filtering techniques to reduce signal noise and prevent erroneous readings.

Overall Integration and Significance of the Module. Through the seamless integration of hardware sensors via Cordova plugins and a thoughtfully designed user interface, the Qibla direction module enables users to accurately and conveniently determine the direction of prayer in any environment – whether indoors or outdoors, and with or without an Internet connection. This solution not only guarantees technical reliability and precision but also enhances the religious value of the application, positioning it as a trusted companion for the user's daily spiritual practice.

3.3.3 Quran and Hadith display module

The module responsible for displaying the Quran and Hadiths is a central component of the mobile application, addressing one of the primary spiritual needs of the user – easy and respectful access to sacred texts. Its implementation must consider the specific challenges of storing and presenting textual content, including support for multiple languages and the unique requirements of displaying Arabic script accurately. Additionally, the module should provide efficient search and navigation capabilities to help users quickly find relevant passages within extensive collections of religious texts (fig. 3.8).

```
document.addEventListener('deviceready', this.onDeviceReady.bind(this), false);

3 references
function onDeviceReady() {
    document.addEventListener('backbutton', onBackKeyDown, false);

    localStorage.setItem('lastsurah', window.location.pathname.split('/').pop().split('.')[0]);
    StatusBar.backgroundColorByHexString('#3B5D7D');

    $('.ayah').click(function () {
        var audio = $(this).attr('id');
        localStorage.setItem('lastayah', audio);
        var reader = localStorage.getItem('reader');

        $('#player').attr('src', 'https://cdn.islamic.network/quran/audio/' + reader + '/' + audio + '.mp3');
        $('#player')[0].play();
        $('#footer').css('bottom', '0');

        var nextAyah = Number(localStorage.getItem('lastayah'));
        $('#' + nextAyah).addClass('recent');
        $('.ayah').not('#' + nextAyah).removeClass('recent');

        $('#player').on('ended', function () {
            var reader = localStorage.getItem('reader');
            nextAyah += 1;

            $('#player').attr('src', 'https://cdn.islamic.network/quran/audio/' + reader + '/' + nextAyah + '.mp3');
            localStorage.setItem('lastayah', nextAyah);
            $('#' + nextAyah).addClass('recent');
            localStorage.setItem('ayanumber', $('#' + nextAyah).attr('data'));
            $('.ayah').not('#' + nextAyah).removeClass('recent');

            if (Number(localStorage.getItem('lastayah')) > Number($('.suratext .ayah').last().attr('id'))) {
                $('#player')[0].pause();
            } else {
                $('#player')[0].play();
                document.getElementById(nextAyah).scrollIntoView({ behavior: "smooth", block: "center" });
            }
        });
    });

    var scrollToAya = localStorage.getItem('lastayah');
    if (scrollToAya) {
        $('#' + scrollToAya).css('background', '#f4b4583d');
    }

    var previousScroll = 0;
    $('body').scroll(function () {
        var currentScroll = $(this).scrollTop();
        if (currentScroll > previousScroll) {
            $('#footer').css("bottom", "-100%");
        } else {
            $('#footer').css("bottom", "0");
        }
        previousScroll = currentScroll;
    });

    $('.pause').click(function () {
        $('#player')[0].pause();
    });
```

Figure 3.8 – Part of the Quran software implementation

To ensure fast access and support offline functionality, the texts of the Quran and Hadiths are stored locally on the user's device. A well-structured data organization is employed: all texts are divided into dedicated files containing necessary attributes for display, search, and navigation. For the Quran, this typically involves a hierarchical structure where metadata – such as surah number, name, and verse count – is stored separately, while the verses themselves are represented as arrays of objects that include verse numbers, the original Arabic text, and translations into multiple languages. A similar structure is applied for Hadiths, with each record containing a unique identifier, source reference, Arabic text, and translations (fig. 3.9).

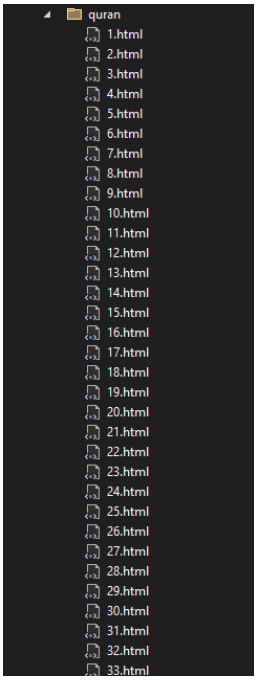


Figure 3.9 – Structured storage of the Quran text

Storing data in this structured format facilitates the easy addition of new languages, seamless text updates, and efficient caching, ensuring fast access even in offline mode. Utilizing local storage not only safeguards the content from unauthorized modifications but also helps preserve the religious authenticity of the texts by implementing integrity verification mechanisms such as checksums or digital signatures (fig. 3.10).

Figure 3.10 – JSON file with hadith texts

Multilingual Support and Arabic Text Rendering Features. Internationalization is a crucial requirement for a modern Islamic mobile application. The text display module facilitates multilingualism by storing translations of the Quran and Hadith within dedicated language sections in JSON files. To efficiently manage the multilingual interface, the application integrates the i18next library, enabling dynamic language switching without the need to reload the app.

Special attention is given to the correct rendering of Arabic text, which involves handling right-to-left (RTL) script direction, proper ligatures, and character shaping to ensure readability and aesthetic accuracy. This is achieved through the use of web technologies and CSS properties that support RTL layouts, as well as font selections optimized for Arabic calligraphy (fig. 3.11).

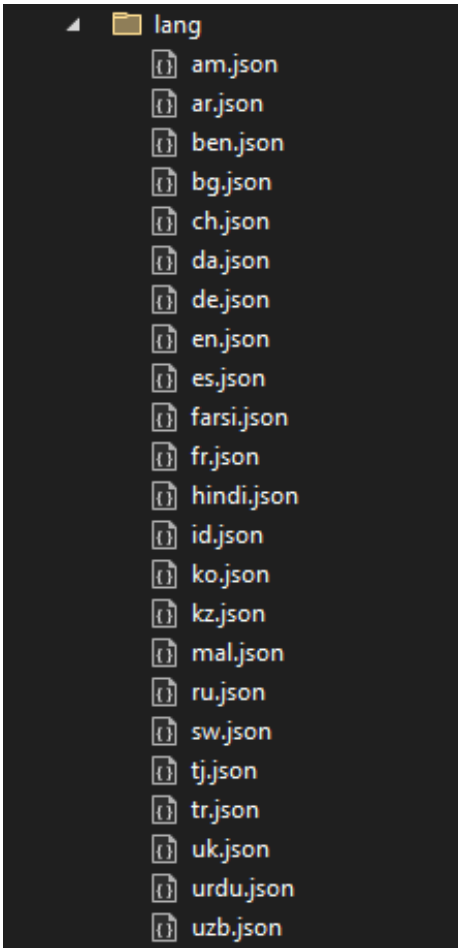


Figure 3.11 – Structured storage of the Quran text

Special attention is given to the rendering of Arabic text, which has unique characteristics: it is written from right to left, includes diacritical marks (harakat), and follows a calligraphic style with both religious and aesthetic significance. To ensure correct display, the application uses proper Unicode encoding alongside Arabic fonts specifically optimized for mobile screens. CSS rules are applied to set the text direction (direction: rtl), select appropriate fonts, and adjust scaling to enhance readability. Additionally, users can toggle between different display modes – showing or hiding harakat – catering to varying levels of language proficiency and personal preferences.

Additionally, the module offers customization features such as adjustable font size and the option to switch between different reading modes, including night mode for reduced eye strain. It also supports audio playback of Quranic verses with synchronized text highlighting, provided this feature is enabled in the settings and the device has sufficient memory capacity.

3.3.4 Islamic calendar module

The Islamic calendar module is a vital component of the mobile application, enabling users to track time according to Muslim tradition, monitor religious holidays, and plan spiritual activities. Its implementation demands accurate display of Hijri (lunar) dates, precise synchronization with the Gregorian calendar, and an intuitive interface for viewing and receiving reminders about significant events throughout the Islamic year (fig. 3.12).

```
3 references
function displayMonth(offset) {
    var lat = localStorage.getItem('latCoords');
    var lng = localStorage.getItem('lonCoords');
    var timeZone = 'auto';
    var dst = 'auto';
    var method = localStorage.getItem('namazmetod');
    var timeFormat = localStorage.getItem('timeformat');

    prayTimes.setMethod(method);
    if (localStorage.getItem('madhab') == 'hanafi') {
        prayTimes.adjust({ asr: 'Hanafi' });
    }

    currentDate.setMonth(currentDate.getMonth() + 1 * offset);
    var month = currentDate.getMonth();
    var year = currentDate.getFullYear();
    var title = monthFullName(month) + ' ' + year;
    $('#table-title').html(title);
    makeTable(year, month, lat, lng, timeZone, dst);
}

1 reference
function makeTable(year, month, lat, lng, timeZone, dst) {
    var items = {
        day: localStorage.getItem('day'),
        fajr: localStorage.getItem('fajr'),
        sunrise: localStorage.getItem('sunrise'),
        dhuhr: localStorage.getItem('zuhr'),
        asr: localStorage.getItem('asr'),
        maghrib: localStorage.getItem('magrib'),
        isha: localStorage.getItem('isha')
    };

    var tbody = document.createElement('section');
    var date = new Date(year, month, 1);
    var endDate = new Date(year, month + 1, 1);
    var format = localStorage.getItem('timeformat');

    while (date < endDate) {
        var times = prayTimes.getTimes(date, [lat, lng], timeZone, dst, format);
        times.day = date.getDate();

        var today = new Date();
        var isToday = (date.getMonth() == today.getMonth()) && (date.getDate() == today.getDate());
        var klass = isToday ? 'today-row' : '';
        tbody.appendChild(makeTableRow(times, items, klass));

        date.setDate(date.getDate() + 1);
    }

    removeAllChild(document.getElementById('timetable'));
    document.getElementById('timetable').appendChild(tbody);

    if (localStorage.getItem('color')) {
        var color = localStorage.getItem('color');
        $('.today-row div').css('background', color + '36');
    }
}
```

Figure 3.12 – Implementation of calendar functionality

The Islamic calendar module is based on the Hijri calendar, which follows lunar cycles and differs fundamentally from the Gregorian calendar used in everyday secular life. The module's main task is to accurately display dates in both calendar systems and provide seamless conversion between them.

To implement this feature, the application integrates a specialized library called moment-hijri for date handling. This library enables dynamic calculation of Hijri dates while accounting for the user's local time zone and automatically synchronizes them with the Gregorian calendar. When the calendar interface loads, both Gregorian and Hijri dates are displayed side by side for each day, facilitating easy navigation across both systems.

Particular attention is given to the accuracy of calculations, as the start of months in the Hijri calendar often depends on astronomical observations of the moon's phases, which can vary by region. Therefore, the module includes the ability to adjust dates regionally to align with local religious traditions. Additionally, the system accounts for leap year rules in the Hijri calendar, ensuring the correct number of days is displayed for each month.

Another key function of the calendar module is to inform users about major Islamic holidays and significant dates. To achieve this, the application maintains a database of important events such as Ramadan, Eid al-Fitr, Eid al-Adha, Mawlid, Ashura, and others. Each holiday is associated with either a fixed or calculated date in the Hijri calendar. This list is stored as a separate resource, allowing for easy updates and additions in accordance with changes or variations in religious calendars.

In the calendar interface, holiday dates are visually marked with special indicators or color highlights to attract the user's attention. When the user clicks on such a date, a detailed description of the holiday is displayed, including its historical background, religious significance, and, when applicable, related ritual prescriptions or hadiths. Additionally, the module features a reminder system that allows users to activate notifications for upcoming religious events, helping them prepare in a timely manner for important holidays and observances.

For greater flexibility and personalization, users can add their own events or mark dates with special significance, such as personal vows, Quran completion milestones, and other important occasions. All user-created events are stored locally on the device, ensuring full compliance with privacy requirements and enabling seamless offline access to personalized data.

The Islamic calendar module provides a unified platform for spiritual planning, enabling Muslims worldwide to stay informed about Hijri dates and important religious holidays. By promoting adherence to religious observances in daily life, it supports the user's spiritual journey. Integration with the notification system and multilingual interface ensures the calendar remains convenient, versatile, and accessible to a diverse international audience.

3.3.5 Tasbeeh module

The tasbih module serves as a digital counterpart to the traditional Muslim prayer beads used for reciting the names of Allah or repeating dhikrs during meditation and prayer. Within the mobile application, this module goes beyond simple counting functionality, preserving the ease, simplicity, and personal nature of spiritual practice by seamlessly blending modern technology with time-honored tradition.

The module features an intuitive counter interface that displays the current count of dhikr repetitions or other selected formulas. Its design closely resembles a classic "subha," with a prominent central display, a button to increment the count, and additional controls for resetting, pausing, or ending the session. The counter is implemented in JavaScript, ensuring immediate responsiveness to user touches and smooth animations during count updates (fig. 3.13).

```
4 references
function onDeviceReady() {
    if (localStorage.getItem('reload') === 'yes') {
        $('.tasbeehListPopup').show();
        localStorage.setItem('reload', 'no');
    }

    if (localStorage.getItem('teasbeehtext') !== null) {
        var tasbeehTxtt = localStorage.getItem('teasbeehtext');
        $('.tasbeehTextHm').html(tasbeehTxtt);
    } else {
        $('.tasbeehTextHm').html('');
    }

    var stockZikr = [
        { "title": "Alhamdu lillah", "number": "33" },
        { "title": "La ilaha illa Allah", "number": "33" },
        { "title": "Allahu Akbar", "number": "33" }
    ];

    if (localStorage.getItem('zikrlist') === null || localStorage.getItem('zikrlist') === "") {
        localStorage.setItem('zikrlist', JSON.stringify(stockZikr));
    }

    var newZikr = JSON.parse(localStorage.getItem('zikrlist'));

    $('#addToList').click(function () {
        var newZirkTxt = $('#zikrTxt').val();
        var newZirkNum = $('#zikrNumbr').val();
        newZikr.push({ "title": newZirkTxt, "number": newZirkNum });
        localStorage.setItem('zikrlist', JSON.stringify(newZikr));
        location.reload();
        localStorage.setItem('reload', 'yes');
    });

    for (var i = 0; i < newZikr.length; i++) {
        var elem = newZikr[i];
        $('.listFromJson').append(
            '<div class="ownZikr ownZkr' + i + '">' +
            '<div class="zikrTitleInlist" data="' + elem.title + '" number="' + elem.number + '">' + elem.title + '</div>' +
            '<div class="xclsSpn" data="' + i + '"><i class="fa fa-times-circle" aria-hidden="true"></i></div>' +
            '<div class="tasNubLst">' + elem.number + '</div>' +
            '</div>'
        );
    }

    var counter = Number(localStorage.getItem('counter'));
    if (localStorage.getItem('counter') === null) {
        localStorage.setItem('counter', 0);
    } else {
        $('.tasbeehNumberByUser').html(counter);
    }

    if (localStorage.getItem('allcounter') === null) {
        localStorage.setItem('allcounter', 0);
    }

    if (localStorage.getItem('maxcount') !== null) {
        $('.maxNumbByUser').html(localStorage.getItem('maxcount'));
    }
```

Figure 3.13 – Implementation of tasbeeh functionality

Special attention has been paid to ergonomics: the buttons are optimally sized for comfortable one-handed use, and users can enable tactile feedback such as vibration or a short beep to confirm each press. The interface also offers options to select different types of reminders, set session goals (e.g., 33, 99, or 100 repetitions), and receive notifications upon reaching intermediate milestones.

One of the advantages of a digital tasbeeh is the automatic saving of the user's progress. All data related to the current count, selected recitation formula, session goal, and history are stored locally on the device using mechanisms such as localStorage or IndexedDB. This enables users to pause and resume their sessions at any time without losing their progress.

Thanks to auto-save mechanisms, even if the application is accidentally closed or the device is rebooted, all information about the last tasbeeh session is reliably restored. Additionally, users can view statistics from previous sessions, set personal records, and access a spiritual practice calendar – features that foster self-discipline and enhance motivation for regular remembrance.

To ensure user privacy, all tasbeeh data is stored solely on the user's device and is never transmitted to third-party servers. Users have full control over their personal data, with options to manually reset the counter or delete their session history at any time.

The tasbih module seamlessly blends technological convenience with traditional spiritual significance, making the practice of dhikr accessible anytime and anywhere – whether at home, on the move, or at work. With its intuitive interface, reliable progress saving, and personalization options, this digital counter serves as a vital tool to support and enrich Muslim spiritual practice in today's digital age.

3.3.6 Notification system

The notification system is a key component of the mobile application designed to support the daily spiritual life of a Muslim. It delivers timely alerts about prayer times, important religious dates, tasbih progress, and other significant events, helping users stay connected to their faith amidst the pace of modern life.

The application supports two primary types of notifications: local and push notifications. Local notifications are generated directly on the user's device without requiring an Internet connection or communication with external servers. This makes them especially suitable for reminders such as prayer times, the start of fasting, the completion of a tasbih session, or upcoming religious holidays. To enable this functionality, the Cordova plugin cordova-plugin-local-notifications is integrated, allowing the application to programmatically create, schedule, and display notifications in the system tray, regardless of whether the app is active or running in the background.

Push notifications, in contrast, are sent from external servers to keep users informed about content updates, changes in the religious calendar, or important community announcements. Implementing push notifications requires integrating third-party services such as Firebase Cloud Messaging or OneSignal, along with configuring server-side components to manage message delivery. This approach enables dynamic communication with users and allows the application to promptly respond to timely events and updates.

By combining both local and push notifications, the application ensures maximum reliability and autonomy: users will not miss important events even without an Internet connection, while those online receive timely updates and the latest information.

The notification system is designed to accommodate diverse user needs by offering extensive customization options.

In a dedicated settings section, users can independently select which events trigger notifications – whether it be prayer times, major holidays, tasbeeh reminders, or all events from the Islamic calendar. Each notification type can be individually configured, allowing users to choose specific sounds, vibration patterns, alert times (such as 5, 10, or 30 minutes before an event), or even disable certain notification categories entirely.

Users can also easily customize the language of notifications, choose between concise or detailed message styles, and opt to display additional content such as the "Ayat of the Day" or relevant thematic hadiths. Furthermore, the application allows management of notification history, enabling users to review past alerts. All these settings are saved locally and can be exported along with other personal data, ensuring flexibility and convenience when switching devices.

The notification system goes beyond simple reminders, serving as a continuous source of spiritual support that helps users organize their time and stay connected with their religious traditions in daily life. Thanks to its flexibility, autonomy, and multilingual capabilities, this module stands out as one of the most valuable and appreciated features in a mobile application designed for the Muslim community.

3.4 Implementation of non-functional requirements

Successful mobile application development involves not only implementing core functionalities but also addressing a broad spectrum of non-functional requirements. These requirements shape the quality, usability, security, and versatility of the product, fostering user trust and ensuring compliance with modern technological standards. This section explores how key aspects – such as interface adaptability and accessibility, data security and protection, multilingual support, and offline operation – were incorporated into the practical implementation of the application.

The application interface was designed to accommodate the wide range of mobile devices on which it may run – from compact smartphones to larger tablets. To achieve this, modern CSS techniques were extensively utilized, including flexible grids (Flexbox and Grid Layout), media queries for responsive adaptation to various screen sizes, and relative units for scaling fonts and controls. This approach ensures that each screen maintains a harmonious and user-friendly appearance regardless of device specifications.

Special attention has been given to accessibility principles. The interface supports font enlargement and ensures sufficient color contrast to aid users with visual impairments. All interactive elements are appropriately marked with ARIA attributes, enabling comfortable interaction for users with varying physical abilities. This also guarantees compatibility with the built-in accessibility features of modern operating systems.

Given the sensitivity of personal information and religious content, the application employs modern data protection measures. All personal user data – such as settings, reading history, tasbih progress, and bookmarks – is stored locally on the device in encrypted form. The AES symmetric encryption algorithm is used via a JavaScript library, ensuring that unauthorized access is prevented even if the device falls into the wrong hands.

Additionally, the application does not transmit personal data to external servers without the user's explicit and informed consent, adhering strictly to the principle of data minimization. To safeguard critical religious texts such as the Quran and Hadith, integrity verification mechanisms – including checksums and digital signatures – have been implemented to prevent accidental or intentional tampering of content.

One of the core features of the application is its full autonomy – all essential functions remain accessible even without an Internet connection. To achieve this, all necessary content, including the Quran, Hadith, holiday calendar, audio files, and user settings, is stored locally on the device and systematically cached.

The application leverages local databases such as IndexedDB and localStorage, ensuring fast data retrieval and a seamless user experience under any conditions.

Even core functions – such as prayer time calculation, Qibla direction determination, tasbih counting, notifications, and text navigation – operate seamlessly without a constant Internet connection. This is achieved through Cordova plugins that provide direct access to the device's hardware resources (GPS, compass, local notifications) without the need to connect to external servers. As a result, the application ensures stable performance and safeguards user privacy regardless of location or network quality.

3.5 Integration of religious content

The integration of religious content is a crucial aspect of the practical implementation of a mobile application, as it defines the app's value, authenticity, and adherence to Islamic ethical and theological principles. This integration goes beyond simply loading canonical texts; it involves establishing robust systems for their validation, preservation, and protection against any form of distortion.

To ensure the authenticity and legitimacy of religious content, the application relies exclusively on verified and widely accepted sources. The foundation consists of the Quran in its original Arabic text, complemented by translations endorsed by authoritative Islamic scholars and supported by relevant fatwas. For hadiths, the application incorporates classical collections such as Sahih al-Bukhari, Sahih Muslim, and Sunan Abu Dawud. Each hadith entry includes its source reference, classification (e.g., sahih, hasan, da'if), and, where available, a concise sharh (interpretation) to aid understanding.

For automated text loading, the application integrates with reliable open APIs, including:

– AlQuran Cloud – providing access to Quranic verses, translations, and surahs;

– Prayer Times API – enabling dynamic calculation of daily prayer times based on user location;

– Hadith API – offering collections of hadiths with authenticated English translations.

After receiving the data, validation is performed to ensure each record conforms to the internal structure – checking for the presence of essential elements such as verse numbers, surah identifiers, transmitter names, and classifications for hadiths. Special attention is given to the formatting of Arabic texts, which must include the full set of diacritical marks, preserve canonical script appearance, and undergo preprocessing to ensure correct rendering across devices with varying language settings.

To ensure offline functionality and provide rapid access to religious content, all essential data is stored locally on the user's device. This is implemented using a JSON-based structure, where data collections are organized into separate files or modules for verses, surahs, hadiths, translations, and other relevant materials.

This organization enables:

– easy scalability by adding new translations or collections;

– fast search and seamless navigation through content;

– support for multilingualism without requiring data reloads;

– secure storage of personal marks, bookmarks, and reading history without compromising user privacy.

Additionally, to enhance user convenience, the application incorporates a night mode, adaptive typography, and a simplified interface designed in accordance with Islamic ethical norms. This includes the avoidance of images depicting living beings, the use of abstract and geometric patterns, calligraphic Arabic fonts, and a culturally appropriate color palette.

An important requirement for a religious application is ensuring the authenticity of sacred texts.

Even a minor error in an ayah or hadith can lead to theological disputes; therefore, several levels of protection have been implemented:

– Authenticity verification: Before loading or updating data, the content is cross-checked against canonical reference sources. For hadiths, the isnad (chain of transmitters) and classification of authenticity are explicitly indicated;

– Structural validation: All files undergo automatic verification to ensure compliance with a predefined schema, such as the mandatory presence of ayah numbers, Arabic text, translations, and relevant metadata for each content unit;

– Integrity control mechanisms: Checksums and digital signatures are generated for each file or content block. Upon application launch and with every content update, these signatures are verified to detect any attempts to alter or distort the texts. In case of discrepancies, the user is notified, and the content is blocked pending further investigation;

– Protection during transmission: When data is downloaded over the Internet, all operations are conducted via secure channels (HTTPS), with critical updates requiring additional authentication measures.

Thanks to this approach, the integration of religious content into the mobile application is carried out not only in accordance with modern technical standards but also in full compliance with Islamic ethical principles and theological requirements. This ensures users a secure, reliable, and aesthetically harmonious environment conducive to daily spiritual practice.

Conclusion on Chapter 3

The third section covers the practical implementation of a mobile application designed for a Muslim audience. Building on the functional and non-functional requirements established earlier, it details the complete development cycle – from project environment setup to the implementation of core modules and system performance testing.

Special attention is given to the software architecture, which is based on the principles of modularity, scalability, and flexibility. This approach significantly simplifies maintenance, updates, and future development of the application. Leveraging Apache Cordova and a modern web technology stack ensures cross-platform compatibility, autonomy, and rapid adaptation to evolving user requirements.

Each functional module – prayer time calculation, Qibla direction determination, Quran and Hadith display, Islamic calendar, tasbeeh, notification system – is implemented taking into account religious requirements, accuracy and personalization. It is important to implement a multilingual interface, offline support and security mechanisms, which guarantees both convenience and user data protection.

Special attention is given to verifying the authenticity of sources, validating data structures, ensuring data integrity, and protecting against unauthorized modifications.

An important aspect of the development was the interface aesthetics, ensuring compliance with Islamic ethical norms by implementing adaptive typography, a night mode, and a design free of images of living beings, with a focus on calligraphic and geometric elements.

CHAPTER 4. OVERVIEW OF THE DEVELOPED APPLICATION

4.1 Interface examples and usage scenarios

The user interface of a mobile application plays a critical role in facilitating effective interaction between the user and the software, directly influencing usability, accessibility of core features, and overall user satisfaction. In applications developed to serve the religious needs of Muslims, the interface must not only adhere to contemporary standards of mobile application design, but also reflect the cultural and spiritual values of its target audience. A key requirement is the support of intuitive usage scenarios that allow users to easily access essential religious functions without unnecessary complexity. The interface architecture should ensure clear navigation, adaptability across a wide range of devices, and the flexibility to personalize settings in accordance with individual preferences and the selected method for calculating religious parameters.

Therefore, the main screens of the application should be structured to ensure consistent navigation and provide quick, intuitive access to all essential features required for daily religious practice.

This section outlines the structure, functionality, and implementation aspects of the mobile application interface developed at the Department of Physics and Information Technologies. The presented solutions address both the user experience (UX) design and the technical methodologies employed throughout the software development process.

The application interface supports more than twenty languages, encompassing both widely spoken global languages and regionally significant ones. Multilingual support is implemented using standard localization practices, with all textual content dynamically retrieved from dedicated language resource files. Users can select their preferred language either during the initial setup of the application or at any time through the settings menu.

Figure 4.1– Interface language selection screen

The primary interface through which the user interacts with the application is a central screen displaying the current time, dates according to both the Gregorian and Islamic (Hijri) calendars, and the scheduled times for daily prayers (namaz). This functionality is implemented using both local and remote services for time synchronization, incorporating time zone data and performing astronomical calculations in accordance with the selected school of thought—specifically, the Shafi'i madhhab. The determination of prayer times relies on algorithms based on the solar position, which are adjusted using the user's geographic coordinates obtained through the device’s GPS module.

Figure 4.2 – Home screen with prayer times

A dedicated screen within the application provides the functionality of an electronic counter (tasbih), which facilitates the counting of repeated supplications (duas) or invocations (dhikr) in Islamic devotional practice. The counter is implemented using a reactive interface model, enabling real-time updates of the displayed repetition count as the user interacts with the component. The interface includes an intuitive tap button for incrementing the count, a visual progress bar, and the option to set a predefined target value. To ensure continuity, the application employs local data storage mechanisms to preserve the current counter state, allowing users to resume their session seamlessly even after closing or restarting the application.

At the top of the interface, the current dua or remembrance (dhikr) being recited by the user is displayed—for example, "Allahu Akbar." This text is dynamically generated from a local array that stores a list of commonly used dhikrs, allowing for flexible content management and personalization.

Beneath this, a progress indicator presents the counting status in the format "4 / 33," where the left-hand value ("4") reflects the current count, automatically updated with each press of the main button, and the right-hand value ("33") denotes the user-defined target. This target can be modified by selecting the pencil icon, which opens a dialog box with an input field for entering a new numeric value. Additional interface elements include an arrow button for resetting the counter and a plus button to add a new dhikr entry. All user interactions are managed through dedicated callback functions, with the component state persistently maintained using state management mechanisms.

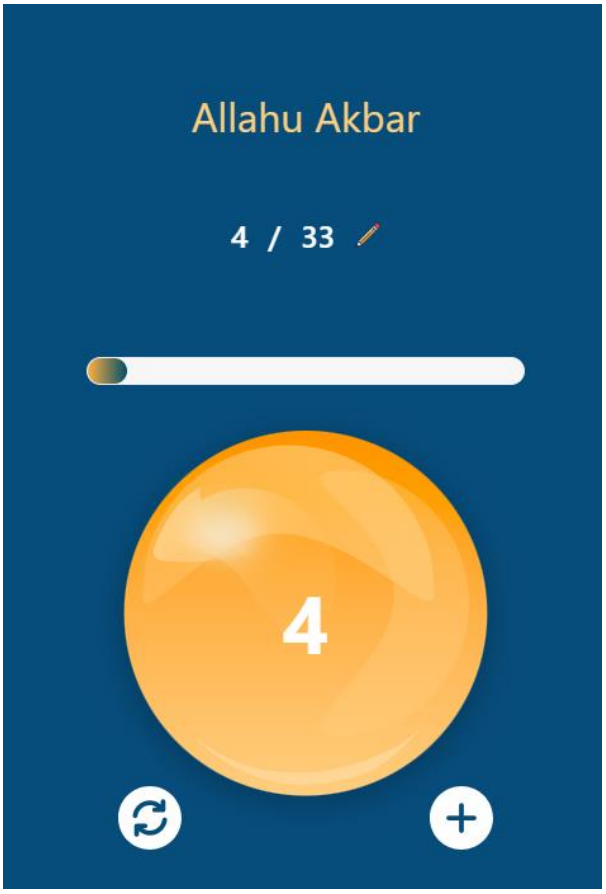


Figure 4.3 – Dua counter screen (tasbih)

The Qibla direction functionality is implemented via a dedicated screen that features an interactive compass interface, with the compass rose serving as the central visual element. The compass rose includes marked divisions at 30-degree intervals, displaying both the cardinal and intercardinal directions (e.g., N, NE, E, etc.), as well as numerical degree values ranging from 0 to 360.

This detailed visual layout enables users to accurately determine the orientation of their device relative to true north. The presence of degree markers enhances the precision of visual alignment with the Qibla direction. At the top of the screen, a location icon (geomarker) is displayed, signifying that the application is utilizing the device’s GPS module to retrieve the user's real-time geographical coordinates.

Specialized graphical components and animation libraries are employed to ensure the smooth and continuous rotation of the compass interface. The visualization dynamically updates in response to real-time sensor events—such as changes in device orientation—thereby delivering a realistic and intuitive user experience. The interface design adheres to principles of minimalism, deliberately excluding superfluous information in order to maintain focus on the primary task: determining the Qibla direction. The inclusion of visual elements such as angular markings and a clearly defined central point enhances the user's ability to align accurately, improving both usability and orientation precision.

Figure 4.4 – Compass for determining Qibla

The prayer times calendar is realized as an interactive table interface, enabling users to conveniently view the start times of daily prayers for the current day and subsequent days. The calendar generation relies on the same precise astronomical algorithms employed on the main screen, extended to perform bulk calculations over a monthly timeframe. To optimize performance and maintain functionality in offline scenarios, the computed prayer times are cached locally within the application.

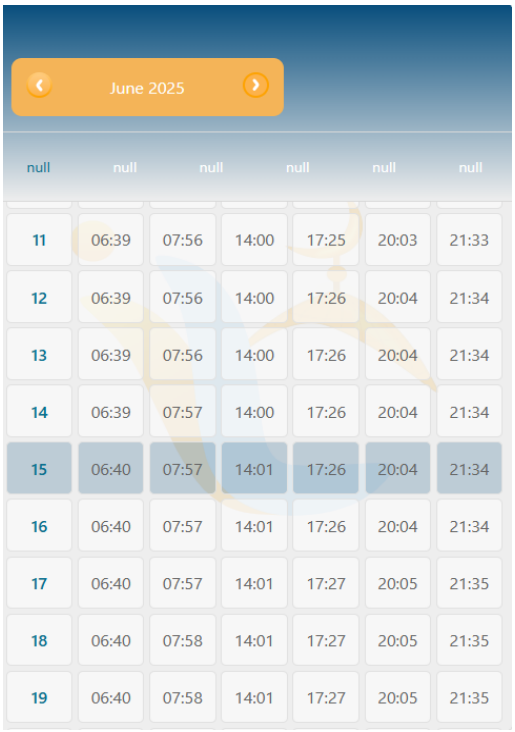


Figure 4.5 – Calendar with prayer times

The navigation screen for the Quranic surahs is implemented as a structured list, where each surah is represented by an individual rectangular button. These buttons display the surah's ordinal number, its name in Arabic, and its transliteration, facilitating rapid and intuitive access to any section of the Quran. The interface employs a tabular layout that dynamically adjusts to the width of the device display, ensuring usability across various screen sizes. The data source consists of a collection of objects, each encapsulating the surah's metadata, including its number, Arabic name, and Latin transliteration. Selecting a button navigates the user to the corresponding screen displaying the full text of the chosen surah.

Figure 4.6 – Contents of the Quran

The surah text display page presents the Arabic script with a clear segmentation into individual verses, each distinctly highlighted for ease of identification. Verse numbers are enclosed within specially designed circular markers, enhancing visual clarity. The currently active verse—whether navigated to manually or synchronized with audio playback—is automatically emphasized by a contrasting background color, enabling users to effortlessly follow along during reading or listening sessions. To ensure accurate rendering of Arabic text, a specialized font supporting diacritical marks and right-to-left script direction is employed. The textual content and verse structure are constructed using components that facilitate multi-level formatting and adaptive layout, thereby maintaining readability across different device sizes and orientations.

F

At the bottom of the interface, a control panel provides intuitive navigation elements, including icons for moving between verses, play and pause buttons for audio control, and a menu icon facilitating quick access back to the main content. All interface components are responsively scaled to accommodate various screen sizes, ensuring ergonomic usability across smartphones and tablets alike. Particular emphasis has been placed on the smoothness of animation transitions when switching verses, as well as on font adaptation features to support users with special needs, thereby enhancing both accessibility and overall user experience.

The technical implementation integrates efficient Arabic text processing with a flexible data structure and robust multimedia support, delivering a fully interactive experience for reading and studying the Quran.

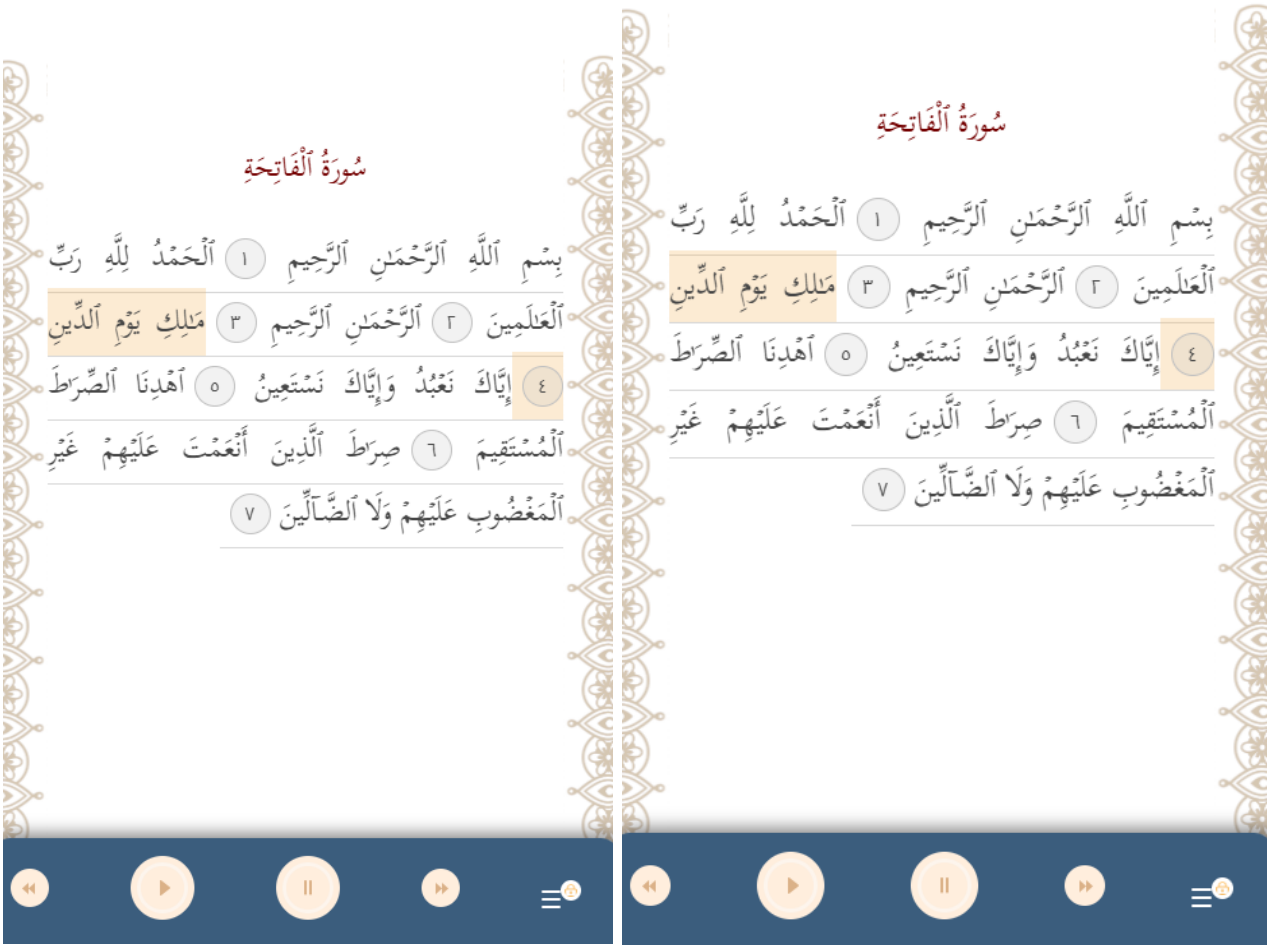


Figure 4.7 – Quran reading screen

The Quran listening module incorporates a built-in audio player that allows users to select from various reciters. Audio files are either streamed from the server or cached locally on the device to enable offline playback. Time markers associated with each verse synchronize the audio with the displayed text, providing a seamless listening and reading experience. Standard multimedia controls such as play, pause, and reciter selection facilitate intuitive user interaction with the audio content.

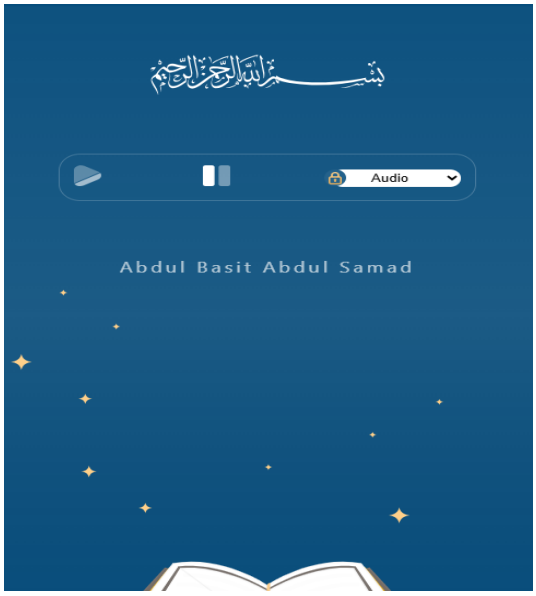


Figure 4.8 – Playing Quran audio

Conclusion on Chapter 4

The mobile application interface is designed in accordance with contemporary UX/UI principles and technical standards for multifunctional information systems. A comprehensive set of approaches has been employed to implement key modules, ranging from sensor data processing and astronomical calculations to integration with multimedia components and external services.

As a result, the application is intuitive, responsive, and adaptable to a diverse user base across multiple countries, thereby fulfilling modern requirements for mobile information systems in the realm of religious technologies.

CONCLUSIONS

This thesis focuses on the development of a cross-platform mobile application tailored for a Muslim audience, designed to comprehensively support religious practices while addressing modern technical, cultural, and ethical considerations. The project implements a suite of essential features, including precise calculation of prayer times based on geolocation and selected madhhab, integration of the Islamic calendar, multilingual access to the Quran and hadith collections, a digital tasbih counter, a robust notification system, as well as offline functionality and a multilingual user interface.

The first section of this work presents a comprehensive analysis of the relevance of digitalization within the religious sphere in today's globalized world. It reviews the current state and key trends in the development of mobile applications targeted at Muslim users. The section classifies various services, highlighting their advantages, limitations, and the primary challenges encountered by users. Particular emphasis is placed on critical issues such as security, localization, inclusivity, and the reliability of content.

The second section provides a rationale for selecting the architecture and technology stack for developing a universal mobile application. It includes a comparative analysis of native, hybrid, and cross-platform development approaches, concluding that Apache Cordova combined with modern web technologies (HTML5, CSS3, JavaScript) represents the optimal solution for the project. The section also explores the plugin system, integration of additional libraries, principles of modular architecture, and methods for accessing the device's hardware resources.

The third section details the practical implementation of the application, encompassing the initialization of the project environment and the integration of core functional modules.

It examines the organization of the project structure, the connection and configuration of plugins, and the implementation of modules for prayer time calculations, Qibla direction determination, Quran display, Islamic calendar, tasbih, and notifications. Special emphasis is placed on security considerations, personal data protection, multilingual support, offline functionality, and the validation of religious content.

The fourth section presents an overview of the developed application, showcasing its user interface and analyzing typical usage scenarios and core functionalities. It includes illustrative screenshots that demonstrate adherence to contemporary UX/UI design principles, cultural and religious considerations, and the fulfillment of requirements for adaptability, accessibility, and personalization.

As a result, a cross-platform mobile application was developed that complies with contemporary standards of security, internationalization, and technological robustness. Comprehensive testing validated the correct functioning of all features, while the modular project architecture ensures scalability and facilitates the integration of additional capabilities in future iterations.

LIST OF REFERENCES

1. Pew Research Center. (2017). The future of world religions: Population growth projections, 2010–2050. https://pewrsr.ch/2s2KmpU

2. Kabir, M. K., & Islam, R. S. (2024). Islamic lifestyle applications: Meeting the spiritual needs of modern Muslims. arXiv. https://doi.org/10.48550/arXiv.2402.02061

3. Yakubovych, M. (2010). Islam and Muslims in contemporary Ukraine: Common backgrounds, different images. Religion, State and Society, 38(3), 291–304. https://doi.org/10.1080/09637494.2010.499287

4. Khan, E. A., & Shambour, M. K. Y. (2018). An analytical study of mobile applications for Hajj and Umrah services. Applied Computing and Informatics, 14(1), 37–47. https://doi.org/10.1016/j.aci.2017.05.004

5. Balcı, Ş., & Özbaş, M. (2017). An analysis of Android mobile applications developed for Hajj and Umrah worship. Procedia Computer Science, 120, 450–457. https://doi.org/10.1016/j.procs.2017.11.275

6. Cheng, W., Yuan, Z., Zhou, P., Xu, Z., Li, R., & Wu, D. O. (2017). The security and privacy of mobile edge computing: An artificial intelligence perspective. Applied Computing and Informatics. https://doi.org/10.1016/j.aci.2017.05.004

7. Ahmed, A. (2019). Survey of Islamic Android apps. International Journal of Computer Applications, 177(25), 20–26.

8. Ismail, N., & Zainuddin, N. (2020). Security and privacy concerns in Islamic mobile applications. Journal of Islamic Studies and Culture, 8(1), 15–22.

9. Khan, R., & Alotaibi, S. (2021). Evaluating user experience of Islamic mobile apps. International Journal of Advanced Computer Science and Applications, 12(4), 233–240.

10. Alotaibi, B., & Alhassan, R. (2021). Muslim mobile apps: A review of features and challenges. Journal of Information Science, 47(4), 482–496. https://doi.org/10.1177/0165551520943198

11. Rahman, S. A., Ismail, W., & Hamat, A. (2018). The use of mobile apps among Malaysian Muslims: A review. Journal of Islamic Marketing, 9(2), 402–420. https://doi.org/10.1108/JIMA-06-2016-0044

12. Al-Emran, M., Malik, S. I., & Al-Kabi, M. N. (2018). A survey of Internet-based technologies in education. International Journal of Emerging Technologies in Learning, 13(3), 37–49. https://doi.org/10.3991/ijet.v13i03.8053

13. Alotaibi, B., & Alhassan, R. (2021). Muslim mobile apps: A review of features and challenges. Journal of Information Science, 47(4), 482–496. https://doi.org/10.1177/0165551520943198

14. Al-Saqqa, S., Abu-Shanab, E., & Mufleh, S. (2019). Mobile applications in Islamic contexts: An analytical study. International Journal of Interactive Mobile Technologies, 13(12), 153–164. https://doi.org/10.3991/ijim.v13i12.10725

15. Hameed, A., Ahmed, H. A., & Bawany, N. Z. (2019). Survey, analysis and issues of Islamic Android apps. Elkawnie: Journal of Islamic Science and Technology, 4(1), 22–34. https://doi.org/10.1016/j.procs.2019.08.019

16. Kabir, M., Kabir, M. R., & Islam, R. S. (2024). Islamic lifestyle applications: Meeting the spiritual needs of modern Muslims. arXiv. https://doi.org/10.48550/arXiv.2401.00001

17. Umar, I., & Tilli, S. F. (2023). The use of mobile apps for Islamic learning: A study on accessibility and learning outcomes. Journal of Computers for Science and Mathematics Learning, 13(2), 147–165. https://doi.org/10.1007/s40692-022-00234-5

18. «No Salvation from Trackers: Privacy Analysis of Religious Websites and Mobile Apps» (2023). In Data Privacy Management, Cryptocurrencies and Blockchain Technology (pp. 123–138). Springer. https://doi.org/10.1007/978-3-030-99544-7_10

19. Bawany, N. Z. (2020). Usability evaluation of Islamic learning mobile applications. Elkawnie: Journal of Islamic Science and Technology, 6(2), 50–62. https://doi.org/10.1016/j.aci.2020.05.0

20. Laird, B., Van Tongeren, D. R., Hook, J. N., Do, B., Hall, T., & Huberty, J. (2024). Exploring user perceptions of a mobile app for religious practices. Journal of Religion and Health, 63(3), 2068–2090. https://doi.org/10.1007/s10943-024-02004-9

21. Laird, B., Hook, J. N., Van Tongeren, D. R., Zuniga, S., Hall, T., & Huberty, J. (2024). The impact of using a faith and prayer mobile application, Pray.com, on mental health and well-being. Spirituality in Clinical Practice. Advance online publication. https://doi.org/10.1037/scp0000366

22. Müller, J., & Friemel, T. N. (2024). Dynamics of digital media use in religious communities—A theoretical model. Religions, 15(7), 762. https://doi.org/10.3390/rel15070762

23. Samarasinghe, N., Kapoor, P., Mannan, M., & Youssef, A. (2023). No salvation from trackers: Privacy analysis of religious websites and mobile apps. In J. Garcia-Alfaro, G. Navarro-Arribas, & N. Dragoni (Eds.), Data privacy management, cryptocurrencies and blockchain technology. DPM CBT 2022. Lecture Notes in Computer Science (Vol. 13619). Springer. https://doi.org/10.1007/978-3-031-25734-6_10

24. Magalhães, F. M. (2021, November 16). I built the same app with Flutter, React Native, and Ionic. Medium. https://medium.com/@fmmagalhaes/i-built-the-same-app-with-flutter-react-native-and-ionic-33ff8b358562

25. There’s a religious app for that: A framework for studying religious mobile applications. (n.d.). Academia.edu. https://www.academia.edu/21520260/Theres_a_religious_app_for_that_A_framework_for_studying_religious_mobile_applications

26. Exploring user perceptions of a mobile app for religious practices. (2024). ResearchGate. https://www.researchgate.net/publication/378236877_Exploring_User_Perceptions_of_a_Mobile_App_for_Religious_Practices

27. Exploring user perceptions of a mobile app for religious practices. (n.d.). OpenReview. https://openreview.net/forum?id=AvvVHACZuK

APPENDICES

APPENDIX A

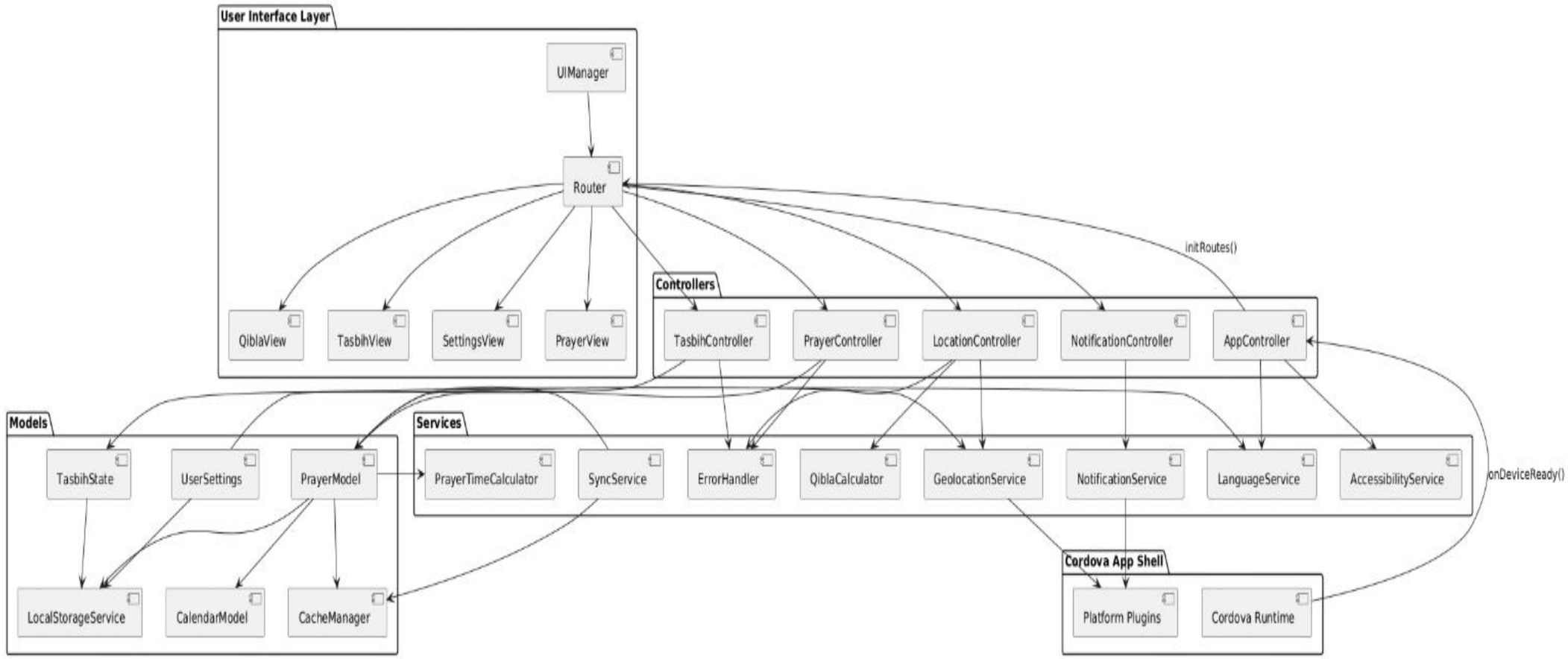
Application Architecture

ІАЛЦ. 467200.005 Д1

Lists 1

Kyiv - 2025

ІАЛЦ.467200.003 Д1



					ІАЛЦ.467200.003 Д1
CHG	Sheet	№ document.	Sign.	Date	Mobile Application for Supporting the Religious Practices of Muslims. Structural Diagram
Developed		Jamil D.			Type: T; Weight; Scale
Checked		Berdnyk Y.			Sheet 1 / Sheets 1
Reviewer		Shymkovych V.			FIOT "KPI" IO-17
Norm.control		Pavlov V.			
Approved		Novotarsky M.			

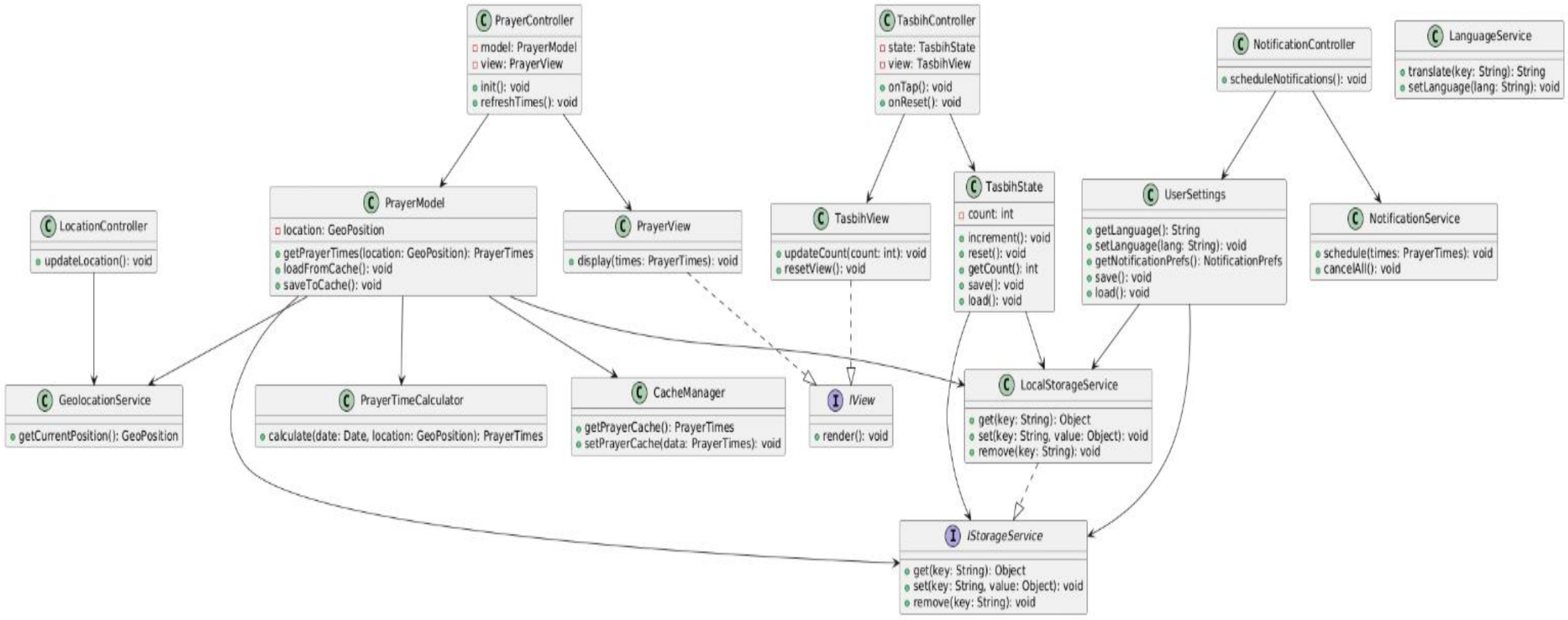
APPENDIX B

Class diagram

ІАЛІЦ. 467200.005 Д2

Lists 1

Kyiv - 2025



					ІАЛЦ.467200.004 Д2
CHG	Sheet	№ document.	Sign.	Date	Mobile Application for Supporting the Religious Practices of Muslims. Functional Diagram
Developed		Jamil D.			Type: T; Weight; Scale
Checked		Berdnyk Y.			Sheet 1 / Sheets 1
Reviewer		Shymkovych V.			FIOT "KPI" IO-17
Norm.control		Pavlov V.			
Approved		Novotarsky M.			

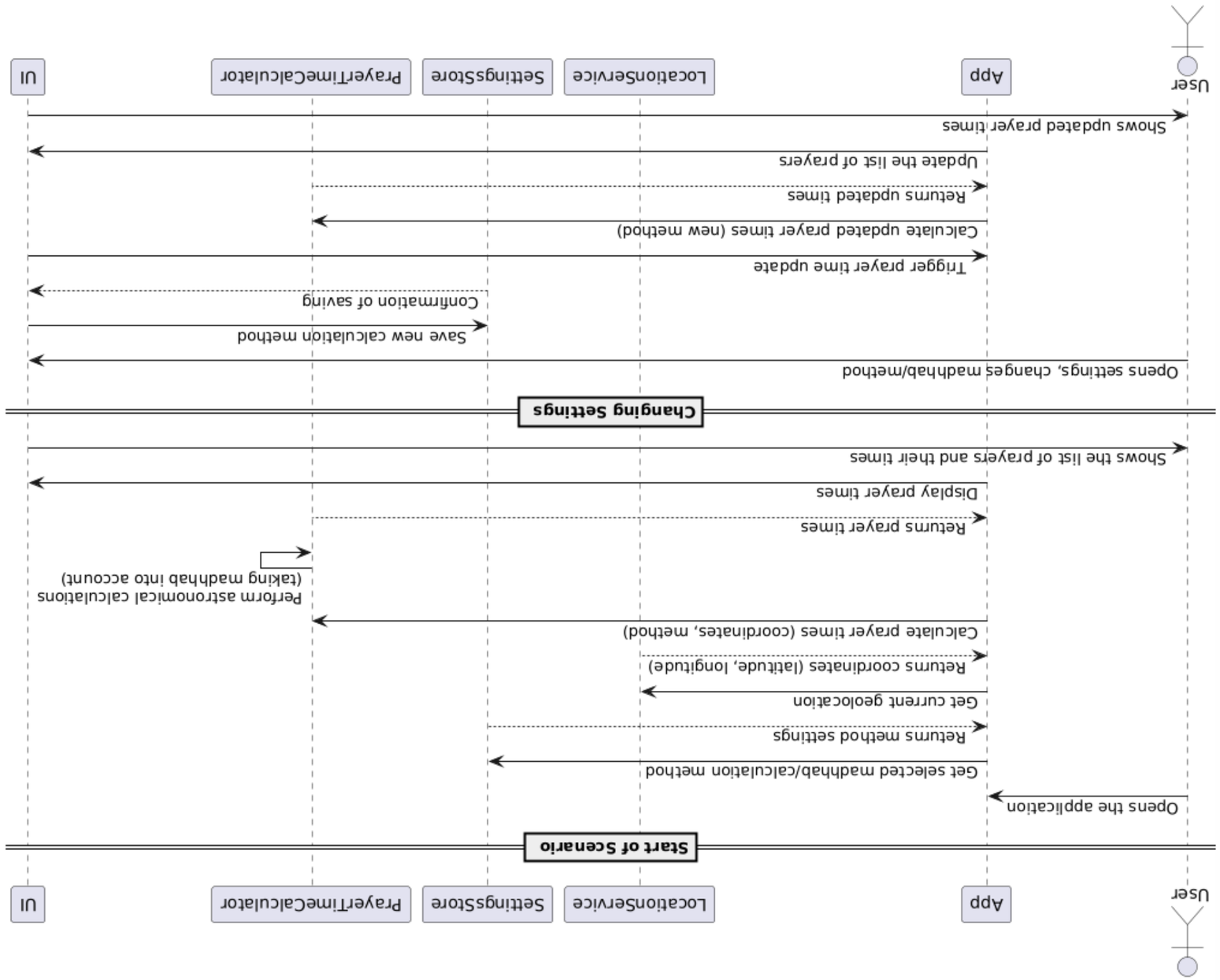
APPENDIX C

Sequence diagram

ІАЛІЦ. 467200.005 Д3

Lists 1

Kyiv - 2025



CHG	Sheet	№ document.	Sign.	Date	
Developed		Jamil D.			
Checked		Berdnyk Y.			
Reviewer		Shymkovych V.			
Nonm.control		Pavlov V.			
Approved		Novotarskyi M.			

		IAЛЦ.467200.004 Д3	
		Mobile Application for Supporting the Religious Practices of Muslims. Activity Diagram	
		Type	Weight	Scale
		T		
		Sheet 1	Sheets 1	
		FIOT "KPI" IO-17	

APPENDIX D

Program Codes

ІАЛІЦ. 467200.005 Д4

Lists 63

Kyiv - 2025

calendar.js

```
document.addEventListener("deviceready", onDeviceReady, false);

function onDeviceReady() {
    if (localStorage.getItem('color')) {
        $('.calendarHeader').css(
            'background',
            'linear-gradient(180deg, ' + localStorage.getItem('color') + ',
#eee)'
        );
        $('.today-row')
            .find('div')
            .css('background', localStorage.getItem('color') + '36
!important');
        $('.head-row').css('color', localStorage.getItem('color'));
    }

    $('.head-row').append(
        '<row><div>' + localStorage.getItem('day') + '</div>' +
        '<div>' + localStorage.getItem('fajr') + '</div>' +
        '<div>' + localStorage.getItem('zuhr') + '</div>' +
        '<div>' + localStorage.getItem('asr') + '</div>' +
        '<div>' + localStorage.getItem('magrib') + '</div>' +
        '<div>' + localStorage.getItem('isha') + '</div></row>'
    );

    var currentDate = new Date();
    var timeFormat = 1;
    switchFormat(0);

    function displayMonth(offset) {
        var lat = localStorage.getItem('latCoords');
        var lng = localStorage.getItem('lonCoords');
        var timeZone = 'auto';
        var dst = 'auto';
        var method = localStorage.getItem('namazmetod');
        var timeFormat = localStorage.getItem('timeformat');

        prayTimes.setMethod(method);
        if (localStorage.getItem('madhab') === 'hanafi') {
            prayTimes.adjust({ asr: 'Hanafi' });
        }

        currentDate.setMonth(currentDate.getMonth() + 1 * offset);
        var month = currentDate.getMonth();
        var year = currentDate.getFullYear();
        var title = monthFullName(month) + ' ' + year;
        $('#table-title').html(title);

        makeTable(year, month, lat, lng, timeZone, dst);
    }
```

```
    function makeTable(year, month, lat, lng, timeZone, dst) {
        var items = {
            day: localStorage.getItem('day'),
            fajr: localStorage.getItem('fajr'),
            sunrise: localStorage.getItem('sunrise'),
            dhuhr: localStorage.getItem('zuhr'),
            asr: localStorage.getItem('asr'),
            maghrib: localStorage.getItem('magrib'),
            isha: localStorage.getItem('isha')
        };

        var tbody = document.createElement('section');
        var date = new Date(year, month, 1);
        var endDate = new Date(year, month + 1, 1);
        var format = localStorage.getItem('timeformat');

        while (date < endDate) {
            var times = prayTimes.getTimes(date, [lat, lng], timeZone, dst,
format);
            times.day = date.getDate();

            var today = new Date();
            var isToday = (date.getMonth() === today.getMonth()) &&
(date.getDate() === today.getDate());
            var klass = isToday ? 'today-row' : '';

            tbody.appendChild(makeTableRow(times, items, klass));
            date.setDate(date.getDate() + 1);
        }

        removeAllChild(document.getElementById('timetable'));
        document.getElementById('timetable').appendChild(tbody);

        if (localStorage.getItem('color')) {
            var color = localStorage.getItem('color');
            $('.today-row div').css('background', color + '36');
        }
    }

    function makeTableRow(data, items, klass) {
        var row = document.createElement('row');
        for (var i in items) {
            var cell = document.createElement('div');
            cell.innerHTML = data[i];
            cell.style.width = (i === 'day') ? '2.5em' : '3.7em';
            row.appendChild(cell);
        }
        row.className = klass;
        return row;
    }

    function removeAllChild(node) {
        if (!node) return;
        while (node.firstChild) {
            node.removeChild(node.firstChild);
        }
    }

    function switchFormat(offset) {
        var formats = ['24-hour', '12-hour'];
        timeFormat = (timeFormat + offset) % 2;
        $('#time-format').innerHTML = formats[timeFormat];
        update();
    }
```

```
    function update() {
        displayMonth(0);
    }

    function monthFullName(month) {
        var monthName = [
            'January', 'February', 'March', 'April', 'May', 'June',
            'July', 'August', 'September', 'October', 'November', 'December'
        ];
        return monthName[month];
    }

    $('.monthPlus').click(function (e) {
        displayMonth(-1);
    });

    $('.monthMinus').click(function (e) {
        displayMonth(+1);
    });
}
```

compass.js

```
document.addEventListener('deviceready', onDeviceReady, false);

function onDeviceReady() {
    load();

    if (localStorage.getItem('color')) {
        const color = localStorage.getItem('color');
        const mode = localStorage.getItem('compass');
        if (mode === 'runcamera') {
            $('html').css('background', 'transparent');
        } else {
            $('html').css('background', color);
        }
        $('.compass').css('background-color', color);
        $('.compassBody').css('box-shadow', `
            0px -5px 10px 0px ${color} inset,
            0px 5px 10px 0px ${color} inset,
            -15px -30px 60px 20px ${color},
            0px 30px 100px 80px ${color}`);
    }

    const compassCircle = document.querySelector('.compass');
    let currentQiblaAngle = null;
    let compassWatchId = null;
    let lastHeading = null;

    const isModalShown = localStorage.getItem('compassCalibrationShown') ===
'true';

    const storedLat = localStorage.getItem('latCoords');
    const storedLon = localStorage.getItem('lonCoords');
    if (storedLat && storedLon) {
        currentQiblaAngle = calcDegreeToPoint(storedLat, storedLon);
        startCompass(currentQiblaAngle);
    }
```

```
    navigator.geolocation.getCurrentPosition(
        function (position) {
            const lat = position.coords.latitude;
            const lon = position.coords.longitude;

            localStorage.setItem('latCoords', lat);
            localStorage.setItem('lonCoords', lon);

            const updatedQibla = calcDegreeToPoint(lat, lon);
            currentQiblaAngle = updatedQibla;

            if (compassWatchId) navigator.compass.clearWatch(compassWatchId);
            startCompass(updatedQibla);

            nativegeocoder.reverseGeocode(userPos, () => {}, lat, lon, {
                useLocale: true,
                maxResults: 1
            });

            function userPos(pos) {
                const city = pos[0].locality;
                const code = pos[0].countryCode;
                localStorage.setItem("cityCoords", city);
                localStorage.setItem("code", code);
                $('.cityNow').html(city);
            }
        },
        function (e) {
            console.warn("Location error:", e.message);
        },
        {
            enableHighAccuracy: true,
            timeout: 10000,
            maximumAge: 0
        }
    );

    function calcDegreeToPoint(lat, lon) {
        const lato = 21.422487;
        const lngo = 39.826206;
        const a = lat - lato;
        const b = lon - lngo;
        let psi;

        if (a > 0 && b >= 0) {
            psi = 180 + Math.atan(b / a) * 180 / Math.PI;
        } else if (a <= 0 && b > 0) {
            psi = 270 + Math.atan(Math.abs(a) / b) * 180 / Math.PI;
        } else if (a >= 0 && b < 0) {
            psi = 90 + Math.atan(a / Math.abs(b)) * 180 / Math.PI;
        } else {
            psi = Math.atan(b / a) * 180 / Math.PI;
        }

        return Math.round(psi);
    }

    function startCompass(qiblaAngle) {
        const mode = localStorage.getItem('compass');
        const options = { frequency: 100 };
        const handler = mode === 'runcamera' ? handlerCam : handlerNormal;

        compassWatchId = navigator.compass.watchHeading(handler,
compassError, options);
```

```
        function handlerNormal(e) {
            CameraPreview.stopCamera();
            const heading = e.magneticHeading || Math.abs(e.alpha - 360);
            const kaabahR = heading - qiblaAngle;
            $('.degrees').html(Math.round(heading));
            compassCircle.style.transform = `rotate(${-heading}deg)`;

            checkCompassCalibration(heading);

            if (Math.abs(kaabahR) <= 2.5) {
                $('.compassBody').css('box-shadow', `
                    0px -5px 10px 0px rgb(43 255 39 / 90%) inset,
                    0px 5px 10px 0px rgb(43 255 39 / 90%) inset,
                    -15px -30px 60px 20px rgb(43 255 39 / 90%),
                    0px 30px 100px 80px rgb(43 255 39 / 90%)`);
                navigator.vibrate(200);
            } else {
                $('.compassBody').css('box-shadow', 'none');
            }

            lastHeading = heading;
        }

        function handlerCam(e) {
            $('html, body').css('background', 'transparent');
            $('.compassBody').hide();

            const heading = e.magneticHeading || Math.abs(e.alpha - 360);
            const kaabahR = heading - qiblaAngle;

            CameraPreview.startCamera({
                x: 0,
                y: 0,
                width: window.screen.width,
                height: window.screen.height,
                camera: CameraPreview.CAMERA_DIRECTION.BACK,
                toBack: true,
                tapPhoto: true,
                tapFocus: false,
                previewDrag: false,
                storeToFile: false,
                disableExifHeaderStripping: false
            });

            if (Math.abs(kaabahR) <= 2.5) {
                $('.carpet').show();
                $('body').css('box-shadow', '0px 0px 100px 10px #03A34E
inset');
                navigator.vibrate(200);
            } else {
                $('.carpet').hide();
                $('body').css('box-shadow', 'none');
            }
        }

        function compassError(err) {
            alert('Compass error: ' + err.code);
        }
    }
```

```
    function checkCompassCalibration(compassHeading) {
        const calibrationThreshold = 10;
        const stabilizationThreshold = 2;

        if (Math.abs(compassHeading - lastHeading) > calibrationThreshold &&
!isModalShown) {
            showCalibrationConfirm();
            localStorage.setItem('compassCalibrationShown', 'true'
        }

        if (Math.abs(compassHeading - lastHeading) <= stabilizationThreshold)
{
            lastHeading = compassHeading;
        }
    }

    function showCalibrationConfirm() {
        if ($('.calibration-modal').length === 0) {
            const modalHTML = `
                <div class="calibration-modal">
                    <div class="modal-content">
                        <img src="/img/compass/cal.gif" alt="Calibration
Instructions" style="max-width:100%"/>
                        <button id="calibrateButton">I Understand</button>
                    </div>
                </div>
            `;

            $('body').append(modalHTML);

            $('.calibration-modal').css({
                position: 'fixed',
                top: '50%',
                left: '50%',
                transform: 'translate(-50%, -50%)',
                width: '80%',
                maxWidth: '400px',
                backgroundColor: 'white',
                padding: '20px',
                borderRadius: '10px',
                boxShadow: '0px 0px 10px rgba(0, 0, 0, 0.5)',
                zIndex: '9999'
            });

            $('.modal-content').css({
                textAlign: 'center'
            });

            $('#calibrateButton').css({
                marginTop: '20px',
                padding: '10px 20px',
                backgroundColor: '#03A34E',
                color: 'white',
                border: 'none',
                borderRadius: '5px',
                cursor: 'pointer'
            });

            $('#calibrateButton').click(function () {
                $('.calibration-modal').remove();
                localStorage.setItem('compassCalibrationShown', 'true');
                location.reload()
            });
        }
    }

}
```

counter.js

```
document.addEventListener("deviceready", onDeviceReady, false);

function onDeviceReady() {
    if (localStorage.getItem('reload') == 'yes') {
        $('.tasbeehListPopup').show();
        localStorage.setItem('reload', 'no');
    }

    if (localStorage.getItem('teasbeehtext') !== null) {
        var tasbeehTxtt = localStorage.getItem('teasbeehtext');
        $('.tasbeehTextHm').html(tasbeehTxtt);
    } else {
        $('.tasbeehTextHm').html('');
    }

    var stockZikr = [
        { "title": "Alhamdu lillah", "number": "33" },
        { "title": "La ilaha illa Allah", "number": "33" },
        { "title": "Allahu Akbar", "number": "33" }
    ];

    if (localStorage.getItem('zikrlist') == null ||
localStorage.getItem('zikrlist') == "") {
        localStorage.setItem('zikrlist', JSON.stringify(stockZikr));
    }

    $('#addToList').click(function () {
        var newZirkTxt = $('#zikrTxt').val();
        var newZirkNum = $('#zikrNumbr').val();
        newZikr.push({ "title": newZirkTxt, "number": newZirkNum });
        localStorage.setItem('zikrlist', JSON.stringify(newZikr));
        location.reload();
        localStorage.setItem('reload', 'yes');
    });

    var newZikr = JSON.parse(localStorage.getItem('zikrlist'));
    for (var i = 0; i < newZikr.length; i++) {
        var elem = newZikr[i];
        var objIndx = i;
        $('.listFromJson').append(
            '<div class="ownZikr ownZkr' + i + '">' +
            '<div class="zikrTitleInlist" data="' + elem.title + '" number="'
+ elem.number + '">' + elem.title + '</div>' +
            '<div class="xclsSpn" data="' + objIndx + '"><i class="fa fa-
times-circle" aria-hidden="true"></i></div>' +
            '<div class="tasNubLst">' + elem.number + '</div>' +
            '</div>'
        );
    }

    var counter = Number(localStorage.getItem('counter'));
    if (localStorage.getItem('counter') == null) {
        localStorage.setItem('counter', 0);
    } else {
        var savedCount = localStorage.getItem('counter');
        $('.tasbeehNumberByUser').html(savedCount);
    }
```

```
if (localStorage.getItem('allcounter') == null) {
    localStorage.setItem('allcounter', 0);
}

var maxCounter = Number(localStorage.getItem('maxcount'));
if (localStorage.getItem('maxcount') !== null) {
    var maxCount = localStorage.getItem('maxcount');
    $('.maxNumbByUser').html(maxCount);
}

$('#countChoose').click(function () {
    $('#tasbihPopup').show('slow');
});

$('.closeMaxCount').click(function () {
    $('#tasbihPopup').hide('slow');
});

$('#saveTasbihMaxCount').click(function () {
    var maxTypedCount = $('#maxCount').val();
    localStorage.setItem('maxcount', maxTypedCount);
    localStorage.setItem('grenbar', '0');
    var maxCon = localStorage.getItem('maxcount');
    var barRes = localStorage.getItem('grenbar');
    var barsze = Number(barRes) + Number(100 / maxCon);
    localStorage.setItem('grenbar', barsze);
    var barResized = localStorage.getItem('grenbar');
    $('.greenBar').css('width', barResized + '%');
    $('#tasbihPopup').hide('slow');
    location.reload();
});

if (localStorage.getItem('grenbar') == null) {
    localStorage.setItem('grenbar', 0);
} else {
    $('.greenBar').css('width', localStorage.getItem('grenbar') + '%');
}

$('.newTasbeeh').click(function () {
    var allcounter = Number(localStorage.getItem('allcounter'));
    var counter = Number(localStorage.getItem('counter'));
    counter += 1;
    allcounter += 1;

    var maxCounter = Number(localStorage.getItem('maxcount'));
    var barResizing = localStorage.getItem('grenbar');
    var barTtLsze = Number(barResizing) + Number(100 / maxCounter);
    localStorage.setItem('grenbar', barTtLsze);
    var barResized = localStorage.getItem('grenbar');
    $('.greenBar').css('width', barResized + '%');

    if (counter > maxCounter && maxCounter > 0) {
        counter = 1;
        navigator.vibrate(300);
    } else if (counter >= maxCounter) {
        localStorage.setItem('grenbar', 0);
        $('.greenBar').css('width', barResized + '%');
    }

    localStorage.setItem('counter', counter);
    localStorage.setItem('allcounter', allcounter);
    $('.tasbeehNumberByUser').html(counter);
```

```
            var ifTurnToZikr = $('.tasbeehTextHm').html();
            localStorage.setItem(ifTurnToZikr, counter);
        });

        $('.resetTasbihCounter').click(function () {
            localStorage.setItem('counter', 0);
            localStorage.removeItem('maxcount');
            localStorage.setItem('grenbar', 0);
            localStorage.removeItem('teasbeehtext');
            location.reload();
        });

        if (localStorage.getItem('color')) {
            $('.tasbeehListPopup').css('background',
    localStorage.getItem('color'));
        }

        $('#opnZkrLst').click(function () {
            $('.tasbeehListPopup').show('slow');
        });

        $('#closeTasbihList').click(function () {
            $('.tasbeehListPopup').hide('slow');
        });

        $('.zikrTitleInlist').click(function () {
            var title = $(this).attr('data');
            var tnumbbr = $(this).attr('number');

            if (localStorage.getItem(title) !== null) {
                localStorage.setItem('counter', localStorage.getItem(title));
                localStorage.setItem(title, tnumbbr);
                localStorage.setItem('maxcount', tnumbbr);
                localStorage.setItem('grenbar', '0');
                var maxCon = localStorage.getItem('maxcount');
                var barRes = localStorage.getItem('grenbar');
                var barsze = Number(barRes) + Number(100 / maxCon);
                localStorage.setItem('grenbar', barsze);
                var barResized = localStorage.getItem('grenbar');
                $('.greenBar').css('width', barResized + '%');
                localStorage.setItem('teasbeehtext', title);
            } else {
                localStorage.setItem(title, tnumbbr);
                localStorage.setItem('maxcount', tnumbbr);
                localStorage.setItem('teasbeehtext', title);
                localStorage.setItem('grenbar', 0);
                localStorage.setItem('counter', 0);
            }

            location.reload();
        });

        $('.xclsSpn').click(function () {
            var removeTheZir = $(this).attr('data');
            localStorage.removeItem(removeTheZir);
            newZikr.splice(removeTheZir, 1);
            localStorage.setItem('zikrlist', JSON.stringify(newZikr));
            localStorage.removeItem('teasbeehtext');
            localStorage.removeItem('counter');
            localStorage.removeItem('maxcount');
            $('.tasbeehTextHm').html('');
            localStorage.setItem('grenbar', 0);
            localStorage.setItem('reload', 'yes');
            location.reload();
        });
```

```
    $('.back').click(function () {
        window.plugins.nativepagetransitions.slide({
            backgroundColor: "#BBBBBB",
            direction: "right",
            duration: 300,
            iosdelay: 2,
            href: history.back(),
        });
    });
}
```

home.js

```
if (localStorage.getItem("color") !== null) {
    const o = localStorage.getItem("color");
    $("#homeL").css({ color: o, background: `${o}21` });
    $("#homeIc").attr("fill", o);

    $('#homePageBody').scroll(function () {
        if ($(this).scrollTop() > 50) {
            StatusBar.backgroundColorByHexString(o);
        } else {
            StatusBar.backgroundColorByHexString('#00000000');
        }
    });
} else {
    $('#homePageBody').scroll(function () {
        if ($(this).scrollTop() > 50) {
            StatusBar.backgroundColorByHexString('#074d7c');
        } else {
            StatusBar.backgroundColorByHexString('#00000000');
        }
    });
}

if (localStorage.getItem('citydumu') == null) {
    localStorage.setItem('citydumu', 'kiyv');
}
if (localStorage.getItem('compass') == null) {
    localStorage.setItem('compass', 'runcompass');
}

setTimeout(() => {
    $('.menuBox').css('transform', 'scale(1)');
}, 100);

$('.hijriDate').html(
    new Intl.DateTimeFormat('ar-u-ca-islamic-nu-latn', {
        day: 'numeric',
        month: 'long',
        year: 'numeric'
    }).format(Date.now())
);
$('.georgeDate').html(
    `${new Date().getFullYear()}-${new Date().getMonth() + 1}-${new
Date().getDate()}`
);

document.addEventListener("deviceready", onDeviceReady, false);
document.addEventListener("resume", () => location.reload());
```

```
function onDeviceReady() {
    load();
    StatusBar.backgroundColorByHexString('#00000000');

    if (window.MobileAccessibility) {
        window.MobileAccessibility.usePreferredTextZoom(false);
    }

    if (CameraPreview) {
        CameraPreview.stopCamera();
    }

    $('#homePryMetod').html(localStorage.getItem('namazmetod'));
    if (localStorage.getItem('madhab') == 'hanafi') {
        $('.shfHanaf').html('Hanafi');
    } else {
        $('.shfHanaf').html('Shafii');
    }

    if (localStorage.getItem('namazmetod') === 'dumu') {
        runDumu();
    } else {
        prayers();
    }

    userLocation();
}

function userLocation() {
    navigator.geolocation.getCurrentPosition(success,
GeolocationPositionError, { enableHighAccuracy: true });

    function success(position) {
        const Lat = position.coords.latitude;
        const Lon = position.coords.longitude;
        localStorage.setItem("latCoords", Lat);
        localStorage.setItem("lonCoords", Lon);
        prayers();

        nativegeocoder.reverseGeocode(userPos, failure, Lat, Lon, {
useLocale: true, maxResults: 1 });

        function userPos(pos) {
            const city = pos[0].locality;
            localStorage.setItem("cityCoords", city);
            $('#City').html(city);
        }
    }
}

function GeolocationPositionError() { }
function failure() { }

function prayers() {
    $('#City').html(localStorage.getItem('cityCoords'));
    const method = localStorage.getItem('namazmetod');
    prayTimes.setMethod(method);

    if (localStorage.getItem('madhab') == 'hanafi') {
        prayTimes.adjust({ asr: 'Hanafi' });
        $('.shfHanaf').html('Hanafi');
    } else {
        $('.shfHanaf').html('Shafii');
    }
```

```
    const prayers = prayTimes.getTimes(
        new Date(),
        [localStorage.getItem('latCoords'),
localStorage.getItem('lonCoords')],
        'auto', 'auto'
    );

    $('.voshudTime').html(prayers.sunrise);
    $('#alfajr').html(prayers.fajr);
    $('#alduhr').html(prayers.dhuhr);
    $('#alasr').html(prayers.asr);
    $('#almaghrib').html(prayers.maghrib);
    $('#alisha').html(prayers.isha);
    $('.timeLeftTxt').html(localStorage.getItem('timeleft'));

    function getTodayTimestampAt(hour, minute) {
        const now = new Date();
        return new Date(now.getFullYear(), now.getMonth(), now.getDate(),
hour, minute).getTime();
    }

    const zuhrTimestamp = getTodayTimestampAt(...prayers.dhuhr.split(':'));
    const asrTimestamp = getTodayTimestampAt(...prayers.asr.split(':'));
    const magribTimestamp =
getTodayTimestampAt(...prayers.maghrib.split(':'));
    const ishaTimestamp = getTodayTimestampAt(...prayers.isha.split(':'));
    const fajrTimestamp = getTodayTimestampAt(...prayers.fajr.split(':')) +
24 * 60 * 60 * 1000;

    const nowIs = localStorage.getItem('nowis');
    const duhrTitl = localStorage.getItem('zuhr');
    const asrTitl = localStorage.getItem('asr');
    const margibTitl = localStorage.getItem('magrib');
    const ishaTitl = localStorage.getItem('isha');

    cordova.plugins.notification.local.schedule([
        {
            id: 1,
            title: 'IslamLife',
            trigger: { at: zuhrTimestamp },
            text: `${nowIs} ${duhrTitl}`,
            androidChannelId: 'ring',
            androidChannelName: 'islamlife',
            androidChannelEnableVibration: true,
            sound: 'file://audio/ring.mp3',
            androidWakeUpScreen: true
        },
        {
            id: 2,
            title: 'IslamLife',
            trigger: { at: asrTimestamp },
            text: `${nowIs} ${asrTitl}`,
            androidChannelId: 'ring',
            androidChannelName: 'islamlife',
            androidChannelEnableVibration: true,
            sound: 'file://audio/ring.mp3',
            androidWakeUpScreen: true
        },
```

```
        {
            id: 3,
            title: 'IslamLife',
            trigger: { at: magribTimestamp },
            text: `${nowIs} ${margibTitl}`,
            androidChannelId: 'ring',
            androidChannelName: 'islamlife',
            androidChannelEnableVibration: true,
            sound: 'file://audio/ring.mp3',
            androidWakeUpScreen: true
        },
        {
            id: 4,
            title: 'IslamLife',
            trigger: { at: ishaTimestamp },
            text: `${nowIs} ${ishaTitl}`,
            androidChannelId: 'ring',
            androidChannelName: 'islamlife',
            androidChannelEnableVibration: true,
            sound: 'file://audio/ring.mp3',
            androidWakeUpScreen: true
        }
    ]);

    function updatePray() {
        const now = new Date();

        function isNowBetweenTimeRange(startStr, endStr) {
            const [startHour, startMinute] = startStr.split(':').map(Number);
            const [endHour, endMinute] = endStr.split(':').map(Number);
            const startTime = new Date(now.getFullYear(), now.getMonth(),
now.getDate(), startHour, startMinute);
            let endTime = new Date(now.getFullYear(), now.getMonth(),
now.getDate(), endHour, endMinute);
            if (endTime <= startTime) endTime.setDate(endTime.getDate() + 1);
            let adjustedNow = new Date(now);
            if (now < startTime && endTime > startTime) {
                adjustedNow.setDate(adjustedNow.getDate() + 1);
            }
            return adjustedNow >= startTime && adjustedNow < endTime;
        }

        if (isNowBetweenTimeRange(prayers.sunrise, prayers.dhuhr)) {
            $('.nextPrayName').html(localStorage.getItem('zuhr'));
            $('.nextPrayTime').html(prayers.dhuhr);
            $('#solnse').attr({ class: 'zuhrBoxSun', src:
'img/home/zuhrBox.png' });
        } else if (isNowBetweenTimeRange(prayers.dhuhr, prayers.asr)) {
            $('.nextPrayName').html(localStorage.getItem('asr'));
            $('.nextPrayTime').html(prayers.asr);
            $('#solnse').attr({ class: 'asrBoxSun', src:
'img/home/asrBox.png' });
        } else if (isNowBetweenTimeRange(prayers.asr, prayers.maghrib)) {
            $('.nextPrayName').html(localStorage.getItem('magrib'));
            $('.nextPrayTime').html(prayers.maghrib);
            $('#solnse').attr({ class: 'magribBoxSun', src:
'img/home/magribBox.png' });
        } else if (isNowBetweenTimeRange(prayers.maghrib, prayers.isha)) {
            $('.nextPrayName').html(localStorage.getItem('isha'));
            $('.nextPrayTime').html(prayers.isha);
            $('#solnse').attr({ class: 'ishaBoxSun', src:
'img/home/ishaBox.png' });
```

```
        } else if (isNowBetweenTimeRange(prayers.isha, prayers.fajr)) {
            $('.nextPrayName').html(localStorage.getItem('fajr'));
            $('.nextPrayTime').html(prayers.fajr);
            $('#solnse').attr({ class: 'fajrBoxSun', src:
'img/home/fajrBox.png' });
        } else if (isNowBetweenTimeRange(prayers.fajr, prayers.sunrise)) {
            $('.nextPrayName').html(localStorage.getItem('sunrise'));
            $('.nextPrayTime').html(prayers.sunrise);
            $('#solnse').attr({ class: 'sunriseBoxSun', src:
'img/home/sunriseBox.png' });
        }
    }

    updatePray();
    setInterval(updatePray, 1000);
}

$('#colorPick').click(() => {
    $('#colorMain').css('bottom', '0');
});
$('#closeColors').click(() => {
    $('#colorMain').css('bottom', '-100%');
});
$('#homePageBody').scroll(() => {
    if ($('#homePageBody').scrollTop() > 50) {
        StatusBar.backgroundColorByHexString(localStorage.getItem("color") ||
'#074d7c');
    } else {
        StatusBar.backgroundColorByHexString('#00000000');
    }
});
$('#goToHadees').click(() => location = 'hadees.html');
$('.otherLnk').click(() => {
    if (localStorage.getItem('namazmetod') === 'dumu') {

cordova.InAppBrowser.open('https://islam.ua/uk/bibliotekaukr/kalendar-
namaziv', '_system');
    } else {
        location = 'calendar.html';
    }
});
$('#bannerMain').click(() => {

cordova.InAppBrowser.open('https://play.google.com/store/apps/details?id=isla
mic.app.pro', '_system');
});
$('#lessons').click(() => {
    cordova.InAppBrowser.open('https://islamic-lessons.org', '_system');
});
$('.treelnk').click(() => location = 'tree.html');
$('.prayerSettings').click(() => location = 'settings.html');
```

interface.js

```
var lng = localStorage.getItem('lang');

function translation() {
    function Translate() {
        this.init = function (attribute, lng) {
            this.attribute = attribute;
            this.lng = lng;
        }
```

```
        this.process = function () {
             self = this;
            var xrhFile = new XMLHttpRequest();
            xrhFile.open("GET", "/lang/" + this.lng + ".json", true);
            xrhFile.onreadystatechange = function () {
                if (xrhFile.readyState === 4) {
                    if (xrhFile.status === 200 || xrhFile.status == 0) {
                        var LngObject = JSON.parse(xrhFile.responseText);
                        var allDom = document.getElementsByTagName("*");
                        var translation = [];

                        translation.push(JSON.stringify(LngObject));
                        console.log(translation);
                        localStorage.setItem('translation', translation);

                        localStorage.setItem('fajr', LngObject.Fajr);
                        localStorage.setItem('sunrise', LngObject.Sunrise);
                        localStorage.setItem('zuhr', LngObject.Zuhr);
                        localStorage.setItem('asr', LngObject.Asr);
                        localStorage.setItem('magrib', LngObject.Magrib);
                        localStorage.setItem('isha', LngObject.Isha);
                        localStorage.setItem('nowis', LngObject.nowis);
                        localStorage.setItem('day', LngObject.day);
                        localStorage.setItem('timeleft', LngObject.timeleft);

                        for (var i = 0; i < allDom.length; i++) {
                            var elem = allDom[i];
                            var key = elem.getAttribute(_self.attribute);
                            if (key != null) {
                                elem.innerHTML = LngObject[key];
                            }
                        }
                    }
                }
            }
            xrhFile.send();
        }
    }

    function load() {
        var storedLangByUser = localStorage.getItem("choesedlanguage");
        var translate = new Translate();
        var currentLng = storedLangByUser;
        var attributeName = 'text';
        translate.init(attributeName, currentLng);
        translate.process();
    }

    load();
}


translation();
```

pray.js

```
function PrayTimes(method) {
    var latitude, longitude, timezone, julianDate;
```

```
    var methods = {
        MWL: { name: "Muslim World League", params: { fajr: 14 } },
        ISNA: { name: "Islamic Society of North America (ISNA)", params: {
fajr: 15, isha: 15 } },
        Egypt: { name: "Egyptian General Authority of Survey", params: {
fajr: 19.5, isha: 17.5 } },
        Makkah: { name: "Umm Al-Qura University, Makkah", params: { fajr:
18.5, isha: "90 min" } },
        Karachi: { name: "University of Islamic Sciences, Karachi", params: {
fajr: 18, isha: 18 } },
        Tehran: {
            name: "Institute of Geophysics, University of Tehran",
            params: { fajr: 17.7, isha: 14, maghrib: 4.5, midnight: "Jafari"
}
        },
        ADMIN: { name: "Dumu", params: { fajr: 14 } }
    };

    var currentMethod = "Makkah";
    var settings = {
        imsak: "10 min",
        dhuhr: "0 min",
        asr: "Standard",
        highLats: "NightMiddle"
    };
    var timeFormat = "24h";
    var timeSuffixes = ["am", "pm"];
    var timeOffsets = {};
    var defaultParams = { maghrib: "0 min", midnight: "Standard" };

    for (var m in methods) {
        var params = methods[m].params;
        for (var def in defaultParams) {
            if (params[def] === undefined) {
                params[def] = defaultParams[def];
            }
        }
    }

    settings = { ...settings, ...methods[method || currentMethod].params };

    var prayerNames = {
        imsak: "Imsak",
        fajr: "Fajr",
        sunrise: "Sunrise",
        dhuhr: "Dhuhr",
        asr: "Asr",
        sunset: "Sunset",
        maghrib: "Maghrib",
        isha: "Isha",
        midnight: "Midnight"
    };

    for (var name in prayerNames) {
        timeOffsets[name] = 0;
    }

    function evalParam(val) {
        return +(val + '').split(/[^0-9.+-]/)[0];
    }

    function isMin(val) {
        return (val + '').indexOf("min") !== -1;
    }
```

```
        function timeDiff(time1, time2) {
            return DMath.fixHour(time2 - time1);
        }

        function twoDigitsFormat(num) {
            return (num < 10) ? "0" + num : num;
        }

        return {
            setMethod: function (m) {
                if (methods[m]) {
                    this.adjust(methods[m].params);
                    currentMethod = m;
                }
            },
            adjust: function (params) {
                for (var key in params) {
                    settings[key] = params[key];
                }
            },
            tune: function (offsets) {
                for (var key in offsets) {
                    timeOffsets[key] = offsets[key];
                }
            },
            getMethod: function () {
                return currentMethod;
            },
            getSetting: function () {
                return settings;
            },
            getOffsets: function () {
                return timeOffsets;
            },
            getDefaults: function () {
                return methods;
            },
            getFormattedTime: function (time, format, suffixes) {
                if (isNaN(time)) return "-----";
                if (format == "Float") return time;

                suffixes = suffixes || timeSuffixes;
                time = DMath.fixHour(time + 0.5 / 60);
                var hours = Math.floor(time);
                var minutes = Math.floor((time - hours) * 60);
                var suffix = "";

                if (format == "12h") {
                    suffix = suffixes[hours >= 12 ? 1 : 0];
                    hours = (hours + 12 - 1) % 12 + 1;
                }
                return twoDigitsFormat(hours) + ":" + twoDigitsFormat(minutes) +
(suffix ? " " + suffix : "");
            },
            getTimes: function (date, coords, timezoneOffset, dst, format) {
                latitude = +coords[0];
                longitude = +coords[1];
                var elevation = coords[2] ? +coords[2] : 0;
                timeFormat = format || timeFormat;

                if (date.constructor === Date) {
                    date = [date.getFullYear(), date.getMonth() + 1,
date.getDate()];
                }
```

```
            if (timezoneOffset === undefined || timezoneOffset === 'auto') {
                timezoneOffset = this.getTimeZone(date);
            }
            if (dst === undefined || dst === 'auto') {
                dst = this.getDst(date);
            }

            timezone = +timezoneOffset + (dst ? 1 : 0);
            julianDate = this.julian(date[0], date[1], date[2]) - longitude /
360;

            return this.computeTimes();
        },
        julian: function (year, month, day) {
            if (month <= 2) {
                year--;
                month += 12;
            }
            var A = Math.floor(year / 100);
            var B = 2 - A + Math.floor(A / 4);
            return Math.floor(365.25 * (year + 4716)) +
                Math.floor(30.6001 * (month + 1)) +
                day + B - 1524.5;
        },
        getTimeZone: function (date) {
            var year = date[0];
            var jan = this.gmtOffset([year, 0, 1]);
            var jun = this.gmtOffset([year, 6, 1]);
            return Math.min(jan, jun);
        },
        getDst: function (date) {
            return +(this.gmtOffset(date) != this.getTimeZone(date));
        },
        gmtOffset: function (date) {
            var localDate = new Date(date[0], date[1] - 1, date[2], 12, 0, 0,
0);
            var GMTString = localDate.toGMTString();
            var GMTDate = new Date(GMTString.substring(0,
GMTString.lastIndexOf(" ") - 1));
            return (localDate - GMTDate) / (1000 * 60 * 60);
        }
    };
}

var DMath = {
    dtr: function (d) { return (d * Math.PI) / 180; },
    rtd: function (r) { return (180 * r) / Math.PI; },
    sin: function (d) { return Math.sin(this.dtr(d)); },
    cos: function (d) { return Math.cos(this.dtr(d)); },
    tan: function (d) { return Math.tan(this.dtr(d)); },
    arcsin: function (x) { return this.rtd(Math.asin(x)); },
    arccos: function (x) { return this.rtd(Math.acos(x)); },
    arctan: function (x) { return this.rtd(Math.atan(x)); },
    arccot: function (x) { return this.rtd(Math.atan(1 / x)); },
    arctan2: function (y, x) { return this.rtd(Math.atan2(y, x)); },
    fixAngle: function (a) { return this.fix(a, 360); },
    fixHour: function (a) { return this.fix(a, 24); },
    fix: function (a, range) {
        a = a - range * Math.floor(a / range);
        return a < 0 ? a + range : a;
    }
};

var prayTimes = new PrayTimes();
```

quran.js

```
document.addEventListener('deviceready', this.onDeviceReady.bind(this),
false);

function onDeviceReady() {
    document.addEventListener('backbutton', onBackKeyDown, false);

    localStorage.setItem('lastsurah',
window.location.pathname.split('/').pop().split('.')[0]);

    StatusBar.backgroundColorByHexString('#3B5D7D');

    $('.ayah').click(function () {
        var audio = $(this).attr('id');
        localStorage.setItem('lastayah', audio);
        var reader = localStorage.getItem('reader');

        $('#player').attr('src', 'https://cdn.islamic.network/quran/audio/' +
reader + '/' + audio + '.mp3');
        $('#player')[0].play();
        $('#footer').css('bottom', '0');

        var nextAyah = Number(localStorage.getItem('lastayah'));
        $('#' + nextAyah).addClass('recent');
        $('.ayah').not('#' + nextAyah).removeClass('recent');

        $('#player').on('ended', function () {
            var reader = localStorage.getItem('reader');
            nextAyah += 1;

            $('#player').attr('src',
'https://cdn.islamic.network/quran/audio/' + reader + '/' + nextAyah +
'.mp3');
            localStorage.setItem('lastayah', nextAyah);
            $('#' + nextAyah).addClass('recent');
            localStorage.setItem('ayanumber', $('#' +
nextAyah).attr('data'));
            $('.ayah').not('#' + nextAyah).removeClass('recent');

            if (Number(localStorage.getItem('lastayah')) >
Number($('.suratext .ayah').last().attr('id'))) {
                $('#player')[0].pause();
            } else {
                $('#player')[0].play();
                document.getElementById(nextAyah).scrollIntoView({ behavior:
"smooth", block: "center" });
            }
        });
    });

    var scrollToAya = localStorage.getItem('lastayah');
    if (scrollToAya) {
        $('#' + scrollToAya).css('background', '#f4b4583d');
    }

    var previousScroll = 0;
    $('body').scroll(function () {
        var currentScroll = $(this).scrollTop();
        if (currentScroll > previousScroll) {
            $('#footer').css("bottom", "-100%");
        } else {
            $('#footer').css("bottom", "0");
        }
        previousScroll = currentScroll;
    });
```

```
$('.pause').click(function () {
    $('#player')[0].pause();
});

$('.play').click(function () {
    $('#player')[0].play();
});

var suraNumber = localStorage.getItem('lastsurah');
var nextSura = Number(suraNumber) + 1;
var prevSura = Number(suraNumber) - 1;

$('.nextSura').click(function () {
    if (nextSura !== 115) {
        location = '../quran/' + nextSura + '.html';
    }
});

$('.prev').click(function () {
    if (prevSura !== 0) {
        location = '../quran/' + prevSura + '.html';
    }
});

if (localStorage.getItem('fontsize') == null) {
    localStorage.setItem('fontsize', '24');
}

$('.BigFont').click(function () {
    var fontSize = Number(localStorage.getItem('fontsize'));
    if (fontSize < 36) {
        fontSize += 4;
        $('.sura').css('font-size', fontSize + 'px');
        localStorage.setItem('fontsize', fontSize);
    }
});

$('.smallFont').click(function () {
    var fontSize = Number(localStorage.getItem('fontsize'));
    if (fontSize > 23) {
        fontSize -= 4;
        $('.sura').css('font-size', fontSize + 'px');
        localStorage.setItem('fontsize', fontSize);
    }
});

let lastScale = 1;
let currentFontSize = 24;
const MIN_FONT = 24;
const MAX_FONT = 42;

document.addEventListener('touchstart', function (e) {
    if (e.touches.length === 2) {
        const dist = getDistance(e.touches[0], e.touches[1]);
        lastScale = dist;
    }
}, { passive: true });

document.addEventListener('touchmove', function (e) {
    if (e.touches.length === 2) {
        const newDist = getDistance(e.touches[0], e.touches[1]);
        const scale = newDist / lastScale;
```

```
            if (scale > 1.05) {
                changeFontSize(1);
                lastScale = newDist;
            } else if (scale < 0.95) {
                changeFontSize(-1);
                lastScale = newDist;
            }
        }
    }, { passive: true });

    function getDistance(touch1, touch2) {
        const dx = touch2.clientX - touch1.clientX;
        const dy = touch2.clientY - touch1.clientY;
        return Math.hypot(dx, dy);
    }

    function changeFontSize(delta) {
        currentFontSize = Math.min(MAX_FONT, Math.max(MIN_FONT,
currentFontSize + delta));
        document.querySelector('.suratext').style.fontSize = currentFontSize
+ 'px';
    }

    function onBackKeyDown() {
        let page = window.location.pathname.split('/').pop();
        if (page !== '../quran.html') {
            window.location.href = '../quran.html';
        } else {
            navigator.app.exitApp();
        }
    }

    $('#reader option').prop('disabled', true);
    $('.set').append('<img src="../img/lock.png">');
}
```

lang/uk.json

```
{
"Qibla":"Кибла",
"Tasbeeh":"Тасбіх",
"Quran": "Кур'ан",
"Islam":"Іслам",
"Fajr":"Фаджр",
"Sunrise":"Схід",
"Zuhr":"Зуhр",
"Asr":"Аср",
"Magrib":"Магріб",
"Isha":"Іша",
"kids":"Дітям",
"Settings":"Налаштування",
"Tree":"Пророки",
"Locateme":"Знайти мене",
"Go back":"Назад",
"Change language":"Змінити мову",
"Change city by gps":"Змінити розташування (GPS)",
"prayer times method": "Спосіб розрахунку часу Намазів",
"contacts": "Контакти",
"Count number": "Кількість повторень",
```

```
"show on map": "Показати на мапі ",
"donate":"Підтримка",
"menu":"Меню",
"Dua & zikr":"Дуа",
"daily inspiration":"Цитати",
"egyptian":"Єгипет",
"isna":"ІОСА",
"mekkah":"Мекка",
"DUMU":"Адміністація",
"quran24":"Куран24",
"notifications":"Сповіщення" ,
"privacy policy":"Політика конфіденційності",
"mwl":"MWL",
"personal info":"Особистий кабінет",
"city":"Місто",
"lang":"Мова",
"all tasbeeh count":"Кількість зікрів",
"last surah":"Остання сура",
"add your city":"Виберіть своє місто",
"next":"Далі",
"ready":"Готово",
"notifications off":"Вимкнути сповіщення",
"reset settings":"Змінити налаштування",
"last page":"Останній розділ",
"add to list":"Додати до списку",
"muslima":"Мусульманці",
"prayertimesapproximate":"Islam Life використовує GPS для визначення часу
молитви. Зверніть увагу-часи Намазів вказані приблизно.",
"geolocation info":"Islam Life використовує доступ до GPS розташування для
визначення часів намазів у вашому місті",
"video":"Відео",
"hadees":"Хадіс",
"attention":"Важливо!",
"choose city from list":"Вибрати місто зі списку",
"Own zikr":"Додайте свій зікр",
"nowis":"Зараз час",
"free app":"IslamLife - це некомерційний і безкоштовний додаток, що не
містить реклами або платного контенту",
"we relie":"Для зручності використання додатку та розробки нових функцій ,
наша команда покладається тільки на вашу підтримку",
"to donate":"Щоб підтримати команду IslamLife, будь ласка, зв'яжіться з нами
через будь-який з цих сервісів",
"hayd":"Тест на визначення місячних",
"nifas": "Тест на визначення післяпологових виділень",
"women info":"Корисні статті для жінок та сім'ї",
"yes":"Так",
"no":"Ні",
"back":"Назад",
"refresh":"Оновити",
"your answers":"Ваші відповіді",
"your results":"Результат",
"ramadantrack":"Рамадан трекер",
"locationdenied":"Ви відмовили в доступі до геолокації, необхідному для
визначення молитовних часів додатком. Будь ласка, перейдіть до налаштувань і
дозвольте додатку використовувати службу геолокації.",
"startday":"Почни свій день з Дуа",
"zakat":"Захід сонця",
"lessons":"Уроки на сайті",
"theme":"Тема",
"home":"Головна",
"calendar":"Календар",
"notification":"Сповіщення",
"allowNotifications":"Дозволити сповіщення",
```

```
"day":"День",
"ringtone":"Рінгтон",
"azan":"Азан",
"compass":"Компас",
"arrows":"Стрілки",
"dark":"Темна",
"light":"Світла",
"camera":"Камера",
"uaonly":"(Тільки для України)",
"profile":"Профіль",
"mainsettings":"Загальні налаштування",
"timeformat":"Формат часу",
"prayersettings":"Налаштування намазу",
"fajrnotify":"Сповіщення про Фаджр",
"support":"Контактна інформація Якщо у вас є запитання, пропозиції або ви
хочете зв’язатися з нами, ми завжди раді допомогти!",
"pro": "Підтримайте наш проєкт, придбавши один із наших платних додатків та
насолоджуйтеся розширеними функціями.<p>IslamLife-Pro: вибір кольорової схеми
додатку, додаткові налаштування для Кібли, календар молитв на рік, Азан,
більше читецiв у розділі Корану та багато іншого</p>",
"getnotifications": "IslamLife запросить дозвіл на відправку сповіщень про
часи молитви. Будь ласка, натисніть 'Дозволити' у наступному вікні, щоб
увімкнути сповіщення про час молитви.",
  "timeleft": "Залишилось часу: "
}
```


zikr/uk.json


```
[
  {
    "Title": "«О Аллаг! Просимо більше величі та пошани Пророку Мухаммаду, та
його дружинам - матерям віруючих. І його нащадкам і членам його сім\u0027ї,
так само, як Ти дарував пошану і велич Пророку Ібрагіму, його роду і
благочестивим мусульманам його громади. Воістину, Ти-Хвалебний, і ми Тебе
славимо»."
  },
  {
    "Title": "Ім\u0027я Бога «аль-Малік» означає - Володар. \r\nХто регулярно
повторює ім\u0027я Бога «аль-Малік», як зікр, протягом часу намазу аз-Зугр
100 разів, у того зникне сум і його серце буде спокійним."
  },
  {
    "Title": "«Починаю з Іменем Аллага згадуючи Якого не буде жодної шкоди ні
на землі, ні на небесах, і Аллаг Той, Хто все Чує і Той, Хто все Знає»."
  },
  {
    "Title": "«Покладаюся тільки на Аллага - немає творця, крім Нього. Я
сподіваюсь на Аллага. Він - Господь величезного ‘Аршу. \r\nЦі слова захищають
від хвилювань і\r\nзанепокоєння."
  },
  {
    "Title": "«Я задоволений тим, що Аллаг - мій Господь, Іслам - моя
Релігія, Мухаммад ﷺ- мій Пророк»."
  },
  {
    "Title": "Ім\u0027я Бога «Ар-Рахман» означає Милостивий до всіх на цьому
світі і лише до віруючих на Тому Світі.\r\nХто систематично читає «Ар-Рахман»
як зікр, той матиме співчуття до людей і стане милосердним."
  },
```

```
  {
    "Title": "Ім\u0027я «Аллаh» означає Той, Кому притаманна
божественність.\r\nОдним із благ зікра (згадування) цього Імені є те, що воно
зміцнює серце."
  },
  {
    "Title": "Читайте у п\u0027ятницю 1000 разів «وَلِيّ يا» «йа Уалій»\r\nЦе
може стати причиною прийняття Ду\u0027а."
  }

]
```