НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ
КАФЕДРА АВТОМАТИКИ ТА УПРАВЛІННЯ В ТЕХНІЧНИХ СИСТЕМАХ

«На правах рукопису»
УДК _004.4_____

«До захисту допущено»

Завідувач кафедри
_____ О.І Ролік____
(підпис) (ініціали, прізвище)
“___”_____20__ р.

Магістерська дисертація

зі спеціальності (спеціалізації) 121, інженерія програмного забезпечення (інженерія програмного забезпечення комп’ютерних систем) _____
(код і назва спеціальності)

_____

на тему: Інтерактивна система підтримки проведення навчальних занять _____

_____

Виконав (-ла): студент (-ка) __2__ курсу, групи ___ІТ-83мп_____
(шифр групи)

Калитюк Назарій Вікторович _____ _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник доцент кафедри АУТС, к.т.н, доцент Полторак В.П. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант _____ _____ _____
(назва розділу) (науковий ступінь, вчене звання, , прізвище, ініціали) (підпис)

Рецензент в.о. зав. каф. СПіСКС, д.т.н, доцент Романкевич В. О. _____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Київ – 2019 року

РЕФЕРАТ

Магістерська дисертація містить 113 сторінок пояснювальної записки, 28 рисунків, 49 таблиць, 8 креслеників та 28 посилань на використані літературні джерела.

Актуальність даної роботи полягає в тому тому, щоб розв’язати проблему недостатньої кількості інструментів для навчаняя комп’ютерної криптографії, яка є стрімко поширюється.

Мета дисертації – розробка нової системи для вивчення роботи шифрування, дешифрування та цифрового підпису.

Об’єктом дослідження є процес навчання користувачів комп’ютерної криптографії за допомогою веб-додатку.

Предметом дослідження є алгоритми шифрування, дешифрування та цифрового підпису, а також існуючі системи для навчання.

Методами дослідження є спостереження, порівняння та аналіз. За допомогою цих методів було проаналізовану ситуацію на ринку та у країні в цілому. Досліджено існуючі аналоги та порівняно їх між собою та з розроблюваною системою.

Отримані результати можуть використовуватись в галузях онлайн та дистанційного навчання. Також окремі модулі можуть бути застосовані для цифрового обілку документів, цифрових підписів та побудови захищених каналів зв’язку.

Ключові слова: криптографія, інтерактивна система, веб-додаток, криптоаналіз, цифровий підпис, односторінковий додаток.

SUMMARY

The master's thesis contains 113 pages of explanatory note, 28 drawings, 49 tables, 8 drawings and 28 references to used literature sources.

The urgency of this work is to solve the problem of the lack of tools for learning computer cryptography, which is rapidly spreading.

The purpose of the dissertation is to develop a new system for studying the work of encryption, decryption and digital signature.

The object of the study is the process of educating computer cryptography users using a web application.

The subject of the study is encryption, decryption, and digital signature algorithms, as well as existing training systems.

Research methods are observation, comparison and analysis. Using these methods, the situation in the market and in the country as a whole was analyzed. Existing analogues were investigated and compared with each other and with the developed system.

The results can be used in online and distance learning. Individual modules can also be used to digital document, digital signature and secure communication channels.

Keywords: cryptography, interactive system, web application, cryptanalysis, digital signature, one-page application.

Національний технічний університет України
«Київський політехнічний інститут
імені Ігоря Сікорського»

Факультет (інститут) Інформатики та обчислювальної техніки _____
(повна назва)

Кафедра Автоматики та управління в технічних системах _____
(повна назва)

Рівень вищої освіти – другий (магістерський) за освітньо-професійною (освітньо-професійною) програмою

Спеціальність (спеціалізація) 121 інженерія програмного забезпечення комп'ютерних систем _____
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О. І. Ролік
(підпис) (ініціали, прізвище)
«___»_____20__ р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Калитюку Назарію Вікторовичу _____
(прізвище, ім'я, по батькові)

1. Тема дисертації «Інтерактивна система підтримки проведення навчального процесу» _____
_____
_____
науковий керівник дисертації Полторак Вадим Петрович, к.т.н, доцент__ ,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «___»_____ 20__ р. №_____

2. Строк подання студентом дисертації _____

3. Об'єкт дослідження процес навчання користувачів комп'ютерної криптографії за допомогою веб-додатку _____
_____

4. Предмет дослідження (вихідні дані для магістерської дисертації за освітньо-професійною програмою) алгоритми шифрування, дешифрування та цифрового підпису, існуючі системи для навчання _____
_____

5. Перелік завдань, які потрібно розробити ознайомитися із алгоритмами шифрування, проаналізувати існуючі рішення та виявити їхні сильні та слабкі сторони, сформувати вимоги до системи та сценарії використання, реалізувати алгоритми шифрування за допомогою програмних кодів, використовуючи отримані дані розробити зручну систему для навчання комп'ютерної криптографії

6. Орієнтовний перелік ілюстративного (графічного) матеріалу діаграма компонентів, діаграма послідовності авторизації, діаграма розгортання, діаграма бази даних, діаграма пакетів, діаграма прецедентів

7. Орієнтовний перелік публікацій Автентифікація зображень на основі методів цифрового підпису

8. Консультанти розділів дисертації*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

9. Дата видачі завдання 24 жовтня 2018

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів магістерської дисертації	Примітка
1	Порівняльний аналіз існуючих рішень	02.09.19 – 06.09.19	
2	Визначення основних вимог до системи	09.09.19 – 13.09.19	
3	Розроблення сценаріїв використання системи	16.09.19 – 20.09.19	
4	Вибір технологій для розробки	23.09.19 – 27.09.19	
5	Розроблення структурної схеми	30.09.19 – 04.10.19	
6	Розроблення ER-діаграми	07.10.19 – 11.10.19	
7	Реалізація бізнес-логіки	14.10.19 – 18.10.19	
8	Розроблення інтерфейсу користувача	21.10.19 – 25.10.19	
9	Розроблення стартап-проекту	28.10.19 – 02.11.19	
10	Оформлення текстового матеріалу	04.11.19 – 25.11.19	
11	Оформлення графічного матеріалу	26.11.19 – 02.12.19	
12	Подача дисертації на перевірку	03.12.19 – 18.12.19	

Студент _____ Н.В. Калитюк
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ В.П. Полторак
(підпис) (ініціали, прізвище)

* Консультантом не може бути зазначено наукового керівника

ЗМІСТ

ПЕРЕЛІК ТЕРМІНІВ ТА СКОРОЧЕНЬ

API – Application Programming Interface.

CLI – Command Line Interface.

DSA – Digital Signature Algorithm.

HTTPS – HTTP over secure.

IDE – Integrated Development Environment.

JSON – JavaScript Object Notation.

Markdown – полегшена мова розмітки даних, яку створено з ухилом на прочитність та зручність у публікації з подальшим перетворенням її на structurally valid XHTML або HTML.

SPA – Single Page Application.

UX – User Experience.

Автентифікація – процедура встановлення наженості користуваченві інформації в системі пред’явленого ним ідентифікатора.

ЕЦП – Електронний Цифровий Підпис.

Криптоаналіз – розділ криптографії, що займається математичними методами порушення конфіденційності і цілісності інформації без знання ключа.

ВСТУП

Актуальність криптографії важко переоцінити, а її історія починається ще 2 тисячі років до нашої ери. Головним застосуванням у всі часи була передача захищених послань, в основному у військових цілях. Наразі криптографія набула ще більшої розповсюдженості та продовжує поширюватись. Неможливо уявити сучасний світ без криптографії, адже саме завдяки їй існує банківська система, захищені лінії зв'язку, включаючи Інтернет, цифрові підписи документів, сертифікати, хешування даних та ін. Це все можливо завдяки розвитку комп'ютерних можливостей та асинхронного шифрування. Враховуючи необхідність криптографії в світі, наявність навчання комп'ютерної криптографії в вищих навчальних закладах та зручність навчання за допомогою комп'ютерів доцільно розробити інформаційну систему, яка буде підтримувати у проведенні навчальних занять із криптографії. Під інформаційною розумітимемо будь-яку систему, яка за допомогою технічних засобів виконує одну або кілька таких функцій, як збирання, передавання, перетворення, накопичення, зберігання та оброблення інформації [1]. Сучасне суспільство все більшою мірою спирається на Інформаційні процеси (ІП), які стають рушійною силою економіки, суспільних відносин, військової справи. ІП – це процеси збору, підготовки, передачі, обробки, перетворення та використання інформації в різних сферах суспільства. Інформація, за одним із продуктивних визначень, є корисні, або ж, нові відомості про навколишній і внутрішній світ. Вона не дається нам безпосередньо, через органи відчуттів. Вона дана нам опосередковано, через фізичні носії - знаки, символи, сигнали (збурення фізичного стану середовища розповсюдження), тощо. Зазвичай їх називають даними[2].

Метою дисертації є розробка нової системи для вивчення роботи шифрування, дешифрування та цифрового підпису. Для досягнення поставленої мети необхідно вирішити наступні задачі:

- ознайомитися із алгоритмами роботи за застосуванням основних методів шифрування;

- проаналізувати існуючі рішення, виявити їх сильні та слабкі сторони, які впливають на процес навчання;

- сформувати вимоги до системи та сценарій використання користувачами;

- реалізувати алгоритми шифрування за допомогою програмних кодів;

- використовуючи отримані дані, розробити зручну систему для навчання комп'ютерної криптографії.

Об'єктом дослідження є процес навчання користувачів комп'ютерної криптографії за допомогою веб-додатку.

Предметом дослідження є алгоритми шифрування, дешифрування та цифрового підпису, а також існуючі системи для навчання.

Методами дослідження є спостереження, порівняння та аналіз. За допомогою цих методів було проаналізовану ситуацію на ринку та у країні в цілому. Досліджено існуючі аналоги та порівняно їх між собою та з розроблюваною системою.

Отримані результати можуть широко використовуватись в галузях онлайн-навчання та пов'язаних із криптографією сферах. На сьогодні Міністерство цифрової трансформації активно займається інтеграцією онлайн-навчання[3], також, за словами заступника Міністра з питань розвитку публічних послуг Людмили Рабчинської, «унікальний цінний досвід України у протистоянні найсучаснішим та цілеспрямованим атакам слід використовувати для створення інновацій у сфері кібербезпеки. І таким чином зробити Україну надійним джерелом рішень, послуг і талантів у світовій індустрії кібербезпеки, що швидко зростає»[4]. Таким чином досліджені в дисертації питання та розроблені рішення можуть бути застосовані не тільки в навчанні, а й для прикладного застосування у різних сферах.

1 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

Для формування вимог до системи, що розробляється, потрібно проаналізувати зразки подібних систем. Детальний аналіз допоможе виявити переваги та недоліки подібних систем. На основі проведеного аналізу можна буде сформувати вимоги до функціоналу майбутньої системи.

1.1 Огляд програми SEZAR.EXE

SEZAR.EXE – програма призначена для навчання шифру Цезаря. Для роботи потрібні мінімальні вимоги до обчислювальної техніки та операційна система Windows. Інтерфейс несучасний та погано адаптований для сучасних моніторів. Віконця маленький, шрифт також. Погано читаються символи. Текст у вікнах російською мовою, та часом трапляється англійською. Програма дозволяє переглянути теорію, подивитись демонстрацію роботи шифрування та дешифрування. Для виконання лабораторної роботи студенту необхідно пройти тест з 5 питань, вибраних із заданого переліку. Важливо дати правильну відповідь на всі 5 питань поспіль, аби пройти далі до практичної частини. Проте відповіді ніде не записуються і кількість спроб не обмежена. Програма відразу каже в якому питанні студент відповів вірно, а в якому ні. Отже, з часом студент дізнається всі відповіді на питання і зможе проходити вірно. Це не недолік, оскільки таким чином студент запам’ятає інформацію.

Далі необхідно зашифрувати і розшифрувати 3 речення. В програмі є вбудований шифратор і дешифратор, тому користувач може нічого не робити, а

використати програму, запустивши її одночасно 2 рази. Це серйозний недолік. Ще одним мінусом є неправильний алгоритм дешифрування, через що іноді просто неможливо дати правильну відповідь самому, проте програма рахує однаково в прикладі і в завданні. Очевидно, правильний варіантом має бути «ТИШЕ ЕДЕШЬ ДАЛЬШЕ БУДЕШЬ», проте через помилку, символ Ш замінений на О (рис 1.1).

В кінці проходження лабораторної роботи програма виводить результат, схожий на протокол. Проте протокол потрібно зберегти вручну як скриншот програми, аби потім показати викладачу. В протоколі зазначено які були дані відповіді на тестові питання та з якої спроби бути пройдені завдання з шифрування та дешифрування. Таким чином в теорії і формується оцінка студента, проте, студент

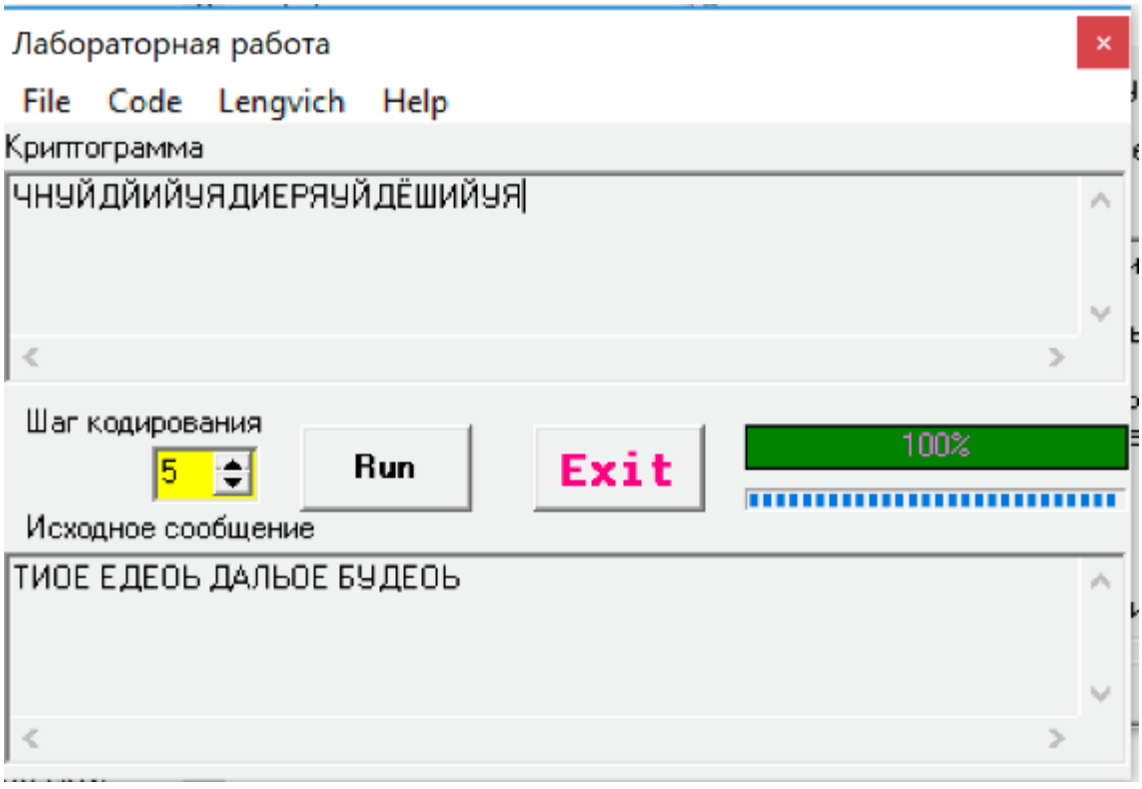

Рис. 1.1 Програма SEZAR.EXE

може перезапускати програму і перездавати тест необмежену кількість разів, поки не отримає бажану оцінку.

Переглянувши програму SEZAR.EXE, можна виділити наступні недоліки та переваги.

Недоліки:

- погано читаються символи у вікнах;
- підтримка лише в операційній системі windows;
- інтерфейс російською мовою з домішками англійською;
- немає збереження результатів роботи;
- немає ідентифікації користувачів;
- програма легко взламується.

Переваги:

- є демонстрація роботи;
- після скачування програми не потрібний інтернет для роботи.

1.2 Огляд програми TRITEM.EXE

TRITEM.EXE – програма призначена для навчання шифру тритеміуса. Програма містить добре сформовану демонстрацію по кроках як шифрувати і дешифрувати. Для виконання лабораторної необхідно пройти тест із 20 питань. При чому на кожне питання дається всього 20 секунд. Несвоєчасна відповідь прирівнюється до неправильної. На перший погляд це непоганий спосіб боротьби з заучуванням відповідей, проте через малу кількість варіантів неефективний. Блок шифрування і дешифрування окремий. Задля стійкості від взлому кожне завдання шифру комплектується своїм алфавітом з російських букв та символу пробіл. Проте програма містить функцію шифрування-дешифрування на заданому алфавіті і функції ключі. Тому студент може легко пройти тестування, відкривши програму ще раз для виконання шифрування-дешифрування за нього. Випадковий шифр для

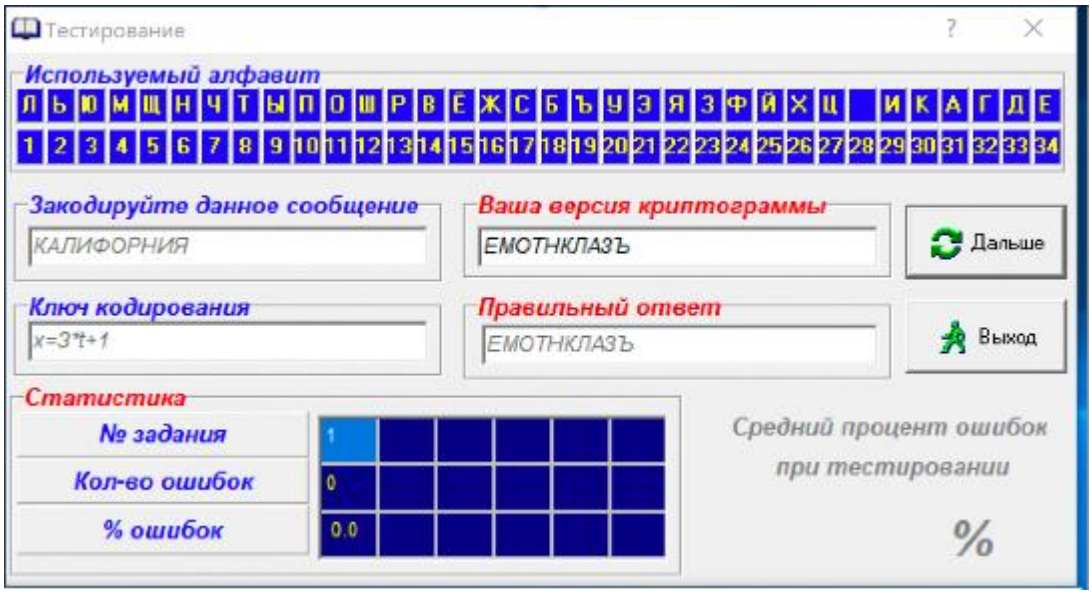

Рисунок 1.2 – Тестування в програмі TRITEM.EXE

кожного завдання просто створює студенту незручність у вигляді створення нового алфавіту для кожного завдання, але не критично (Рис 1.2).

Щодо теоретичних даних, то вони виконані в форматі windows довідки .hlp. Цей формат не підтримується в версії 10 і не буде підтримуватись далі. Отже теорію неможна подивитись.

Переглянувши програму TRITEM.EXE, можна виділити наступні недоліки та переваги.

Недоліки:

- Відсутність теоретичних відомостей на Windows 10;
- Підтримка лише операційною системою windows;
- Інтерфейс російською мовою;
- Відсутнє збереження результатів;
- Немає ідентифікації користувачів;
- Програму можливо взламати.

Переваги

- Є демонстрація роботи шифру;
- Після скачування програми вона працює без доступу в інтернет;

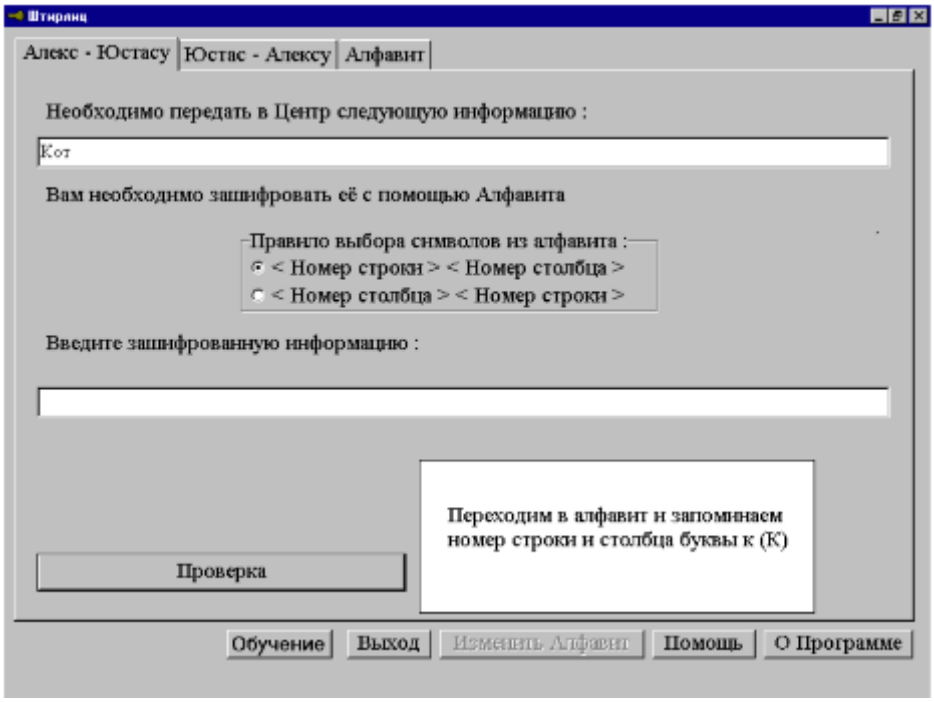

Рисунок 1.3 – Шифрування в програмі SHTIRL.EXE

- Є захист від взлому, проте він може лише трохи ускладнити процес;
- Зрозумілий і досить зручний інтерфейс.

1.3 Огляд програми SHTIRL.EXE

SHTIRL.EXE – програма для навчання шифру Штірліца (Рис 1.3). Програма виконана в простому інтерфейсі з 3 вкладок. Шифрування, дешифрування та алфавіт. Алфавітом є текст, в даному випадку вірш Пушкіна. Також є вікно з демонстрацією. Демонстрація виконана з кількох скриншотів з послідовними діями. В цілому приклад для такого простого коду непоганий. Дешифрування виконується аналогічно шифруванню. Алфавіт береться із 6 прикладів, які виконані у вигляді окремих файлів, які можна редагувати, отже таким чином можна зломати. Проте це не має ніякого сенсу, адже заміна алфавіту лише трохи спростить знаходження букви, проте алгоритм залишається тим самим. По суті це заміна ключа на більш простий.

Переглянувши програму SHTIRL.EXE, можна виділити наступні недоліки та переваги.

Недоліки:

- відсутність теоретичних відомостей;
- підтримка лише операційною системою windows;
- застарілий неадаптований інтерфейс російською мовою;
- відсутнє збереження користувача і результатів.

Переваги:

- є демонстрація роботи шифру;
- програма стійка до взлому;
- простий інтерфейс.

1.4 Огляд програми GAMMIR.EXE

GAMMIR.EXE – програма для Windows призначення для вивчення шифру гамуванням або шифру XOR. Інтерфейс головного вікна складається з кількох кнопок: теоретичні відомості; навчання; тест; кодування файлу; вихід. Теоретичні відомості виконані у вигляді довідки Windows. Текст інтерфейсу дрібний і погано читається. Навчання покрокове, проте лише на 1 прикладі, проте можна вручну змінити ключ псевдовипадкової послідовності, але не вказано як вона генерується. Тест виконаний у вигляді 10 теоретичних питань у випадковому порядку. Питання цікаві та не важкі. Лише після того, як студент правильно відповість на всі 10 питань, він може перейти до практичного завдання. В практичному завданні потрібно зашифрувати і розшифрувати кілька текстів. Наявний алфавіт, з номерами літер і послідовність псевдовипадкових чисел для гамування (рис 1.4). Після закінчення тестування та практичного завдання можна отримати результат у вигляді звіту, де

вказано з якої спроби та з якою кількістю помилок були пройдені тест і завдання.

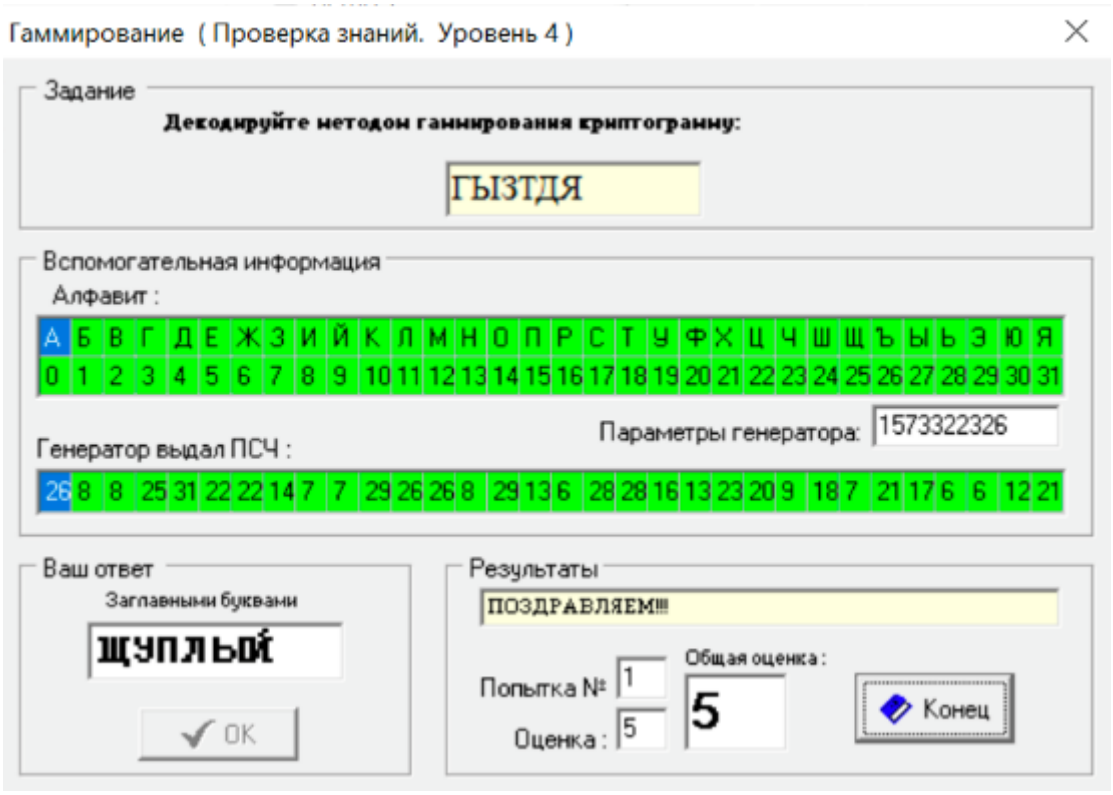

Рисунок 1.4 – Вікно практичного завдання додатку GAMMIR.EXE

Проте запис стирається після повторного відкриття програми.

Переглянувши додаток GAMMIR.EXE, можна виділити наступні недоліки та переваги.

Недоліки:

- теоретичні відомості в застарілому форматі;
- підтримка лише операційною системою windows;
- застарілий незручний інтерфейс російською мовою;

Переваги:

- доволі зручна демонстрація роботи шифру;
- стійкість системи до взлому;
- генерація звіту після проходження.

1.5 Огляд програми Diffie-Hellman.exe

Програма Diffie-Hellman.exe розроблена для навчання студентів протоколу Діффі-Хеллмана (рис 1.5). Алгоритм Діффі-Хеллмана призначено для розповсюдження єдиного секретного ключа сеансу K (з метою використання його у подальшому в симетричній криптосистемі) серед кількох суб'єктів, через незахищене, відкрите середовище комунікацій, наприклад, мережу спільного доступа. Звісно, секретні дані і ключі у відкритому середовищі не передаються. Передаються певні дані, математично пов'язані із секретними даними, а саме – відкриті, публічні ключі. Можливо, тому у стандартах цю дію називають «узгодження ключа». Відкриті ключі можуть бути перехоплені в середовищі криптоаналітиком, але за ними надзвичайно важко (чи взагалі нереально) за осяжний час дізнатися ключ сеансу, якщо секретні дані не потрапили йому до рук[5]. Разом в програмою, виконаною у вигляді одного .exe файлу, розповсюджується .doc файл з відомостями про програму та алгоритмом роботи в програмі. Для початку рекомендують ознайомитись із теоретичними матеріалами. Вони представлені у вигляді вікна програми. Матеріали російською мовою, проте актуальні та зрозумілі. Текст погано читається, тому що це схоже на фото чи відсканову сторінку. Основне вікно програми містить в собі форму для обчислення закритих та відкритих ключів на основі вхідних даних. Також, для простоти виконання робіт є вікно з простими числами від 0 до 15000. Проте при спробі відкрити вікно, програма видає помилку ділення на 0. Вікно з навчанням протоколу виконане у вигляді послідовних вікон з покроковим виконанням процесу генерації ключів. Генерація завжди відбувається на одних і тих же вхідних даних. Послідовні вікна не найкращий варіант, тому що користувач втрачає минулі

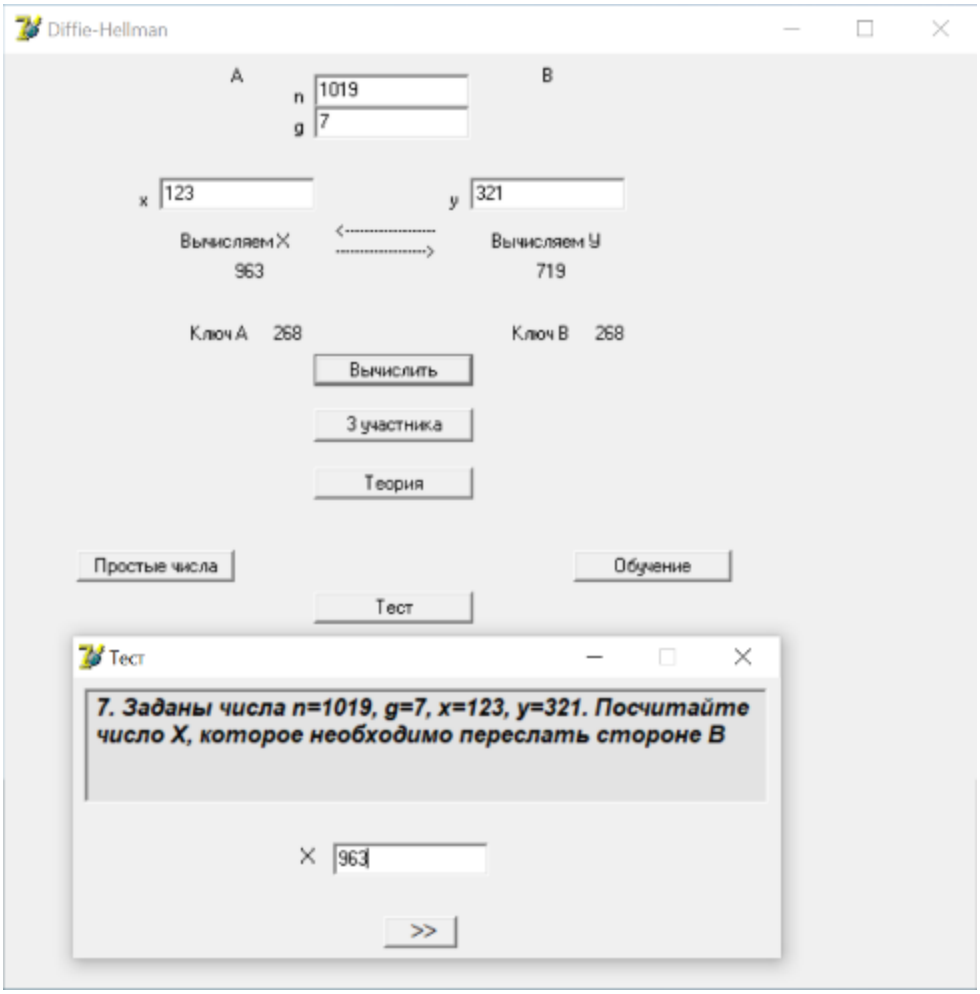

Рисунок 1.5 – Тест та головне вікно додатку Diffie-Hellman

обчислення, було б краще аби інформація з’являлась на одному вікні послідовно. Тестування складається з тестових питань та пратичних питань, де потрібно вказати в якості відповіді числовий ключ. Тестові питання актуальні та цікаві, проте практичні завдання можна виконати в самому додатку, в головному вікні, де є форма генерації ключів.

Всього в тесті є 9 питань. 6 теоретичних і 3 практичних. Аби здати потрібно набрати 8 або 9 правильних відповідей. Питання однакові, проте для практичних існує певний рандом для вхідних даних. Відомості про проходження тесту і кількість балів наведена в кінці, проте звіт відсутній.

Проаналізувавши додаток Diffie-Hellman.exe можна виділити наступні недоліки та переваги.

Недоліки:

- підтримка лише операційною системою windows;
- незручний інтерфейс, який погано виглядає на сучаних дисплеях;
- відсутній звіт про проходження;
- незручна демонстрація алгоритму протоколу;
- програма незахищена від нечесного проходження тестування.

Переваги:

- добре оформленні теоретичні відомості;
- справка по роботі з програмою.

1.6 Огляд програми EL-GAMAL.EXE

EL-GAMAL.EXE – програма для Windows, призначена для навчання студентами протоколу Діфі-Хеллмана та шифру Ель-Гамаля (рис 1.6). На головному екрані додатку знаходяться кілька кнопок для навігації. В додатку присутні теоретичні відомості, проте у вигляді довідки Windows. Процес тестування

Рисунок 1.6 – Головне вікно додатку EL-GAMAL.EXE

складається з 10 тестових питань.

Питання нескладні та цікаві. На кожне питання є 20 секунд на відповідь. Для доступу до шифрування/дешифрування немає необхідності проходити тестові питання. Після проходження програма видасть оцінку, по 0.5 бала за кожну правильну відповідь, та округлить в більший бік. Процес шифрування/дешифрування включає в себе режим демонстрації, оскільки будь який крок можна довірити на обчислення комп'ютеру. При цьому, якщо хоч 1 раз використати цю функцію, то у звіті буде написано, що шифрування/дешифрування було пройдено в режимі демонстрації. В будь який момент можна відкрити звіт, де вказано чи були пройдені тестування та практичні завдання, та на яку оцінку. Інтерфейс програми російською мовою. Зручний, текст добре читається, проте в режимі шифрування та дешифрування шрифт замалий. Алфавіт в цьому режимі представлений у вигляді малих літер російського алфавіту.

Проаналізувавши додаток EL-GAMAL.EXE, можна виділити наступні недоліки та переваги.

Недоліки:

- теоретичні відомості в застарілому форматі;
- підтримка лише операційною системою windows;
- інтерфейс російською мовою.

Переваги:

- простий і зручний інтерфейс;
- наявність звіту, якиий можна відкрити в будь який час;
- стійкість до нечесного проходження.

1.7 Огляд додатків для шифру RSA

Для навчання та виконання практичних завдань з шифрування на прикладі коду RSA існує 3 додатки: rsa, rsa1, rsa2. Розглянемо їх в цілому. Всі 3 додатки виконані у вигляді віконних додатків для операційної системи Windows. Додатки російською мовою.

Перший додаток містить теоретичні відомості у форматі довідки Windows. Вікно навчання складається з форми, де вже введені вхідні значення, проте їх можна змінити на некоректні, що призведе до помилки. Після генерації ключів нам показують процес шифрування якихось даних. Нажаль не покроково по формулі, а просто якось, що може бути не зрозуміло. Далі таким же чином показане

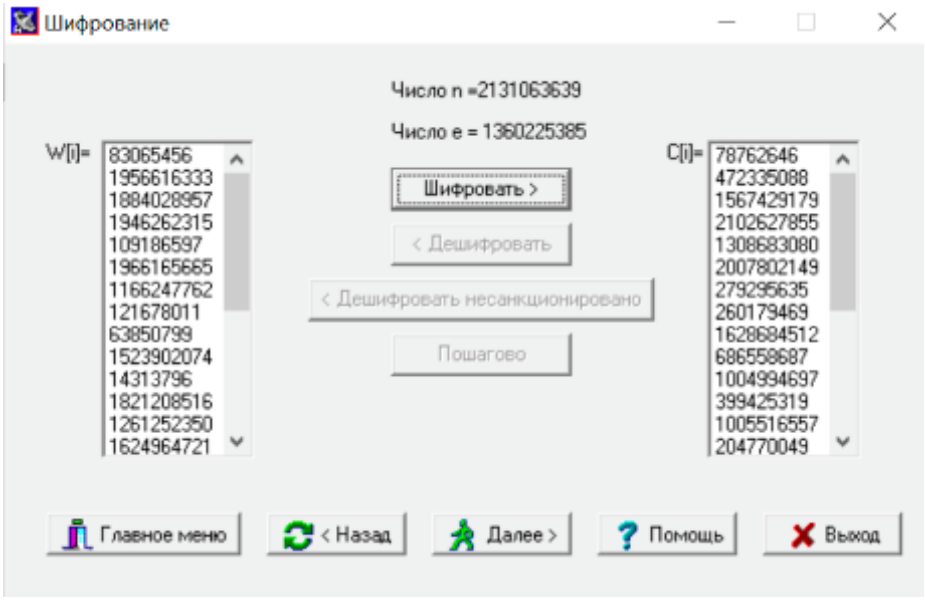

Рисунок 1.7 – Вікно демонстрації шифрування RSA

дешифрування (рис 1.7). Після цього демонстрація несанкціонованого дешифрування шляхом підбору ключів. Мабуть це останнє вікно демонстрації, тому що шифр RSA достатньо стійкий до взлому. Тому мабуть далі можна закрити демонстрацію.

Тестування складається з 20 питань, на кожне питання виділено певний час, мабуть 20 секунд. Іноді за цей час важко відповісти на питання, якщо воно велике та відповіді складаються зі схожих складних формул. Ще одним мінусом є те, що питання та варіанти відповідей погано читаються через низьку чіткість на малий шрифт. Для доступу до практичних питань, потрібно правильно відповісти на 15 питань із 20. Практичне завдання складається з 5 завдань, де потрібно зашифрувати кілька текстів заданими вхідними даними. Аби вдало скласти практичне завдання потрібно також набрати 75% правильних відповідей. Також програма містить

фукнцію шифрування дешифрування файлів, просто для ознайомлення з шифром. Проте цю фукцію можна використати для проходження практичного завдання, якщо записувати в файл завдання із практики, що дає студентам змогу нечесного проходження практичного завдання. Генерація будь якого звіту про проходження тестових та практичних завдань відсутня.

Проаналізувавши перший додаток можна виділити наступні недоліки та переваги.

Недоліки:

- підтримка лише операційною системою windows;
- застарілий інтерфейс російською мовою;
- погана демонстрація роботи шифру;
- відсутність цифрового підпису RSA;
- вразливість до нечесного проходження практичного завдання.

Переваги:

- добре сформовані тестові питання;
- наявність деякого звіту.

Наступний додаток створення для навчання шифрування алгоритмом rsa. Програма містить простий інтерфейс для шифрування повідомлень та файлів по заданим вхідним значенням. По суті програма є гарною демонстрацією роботи

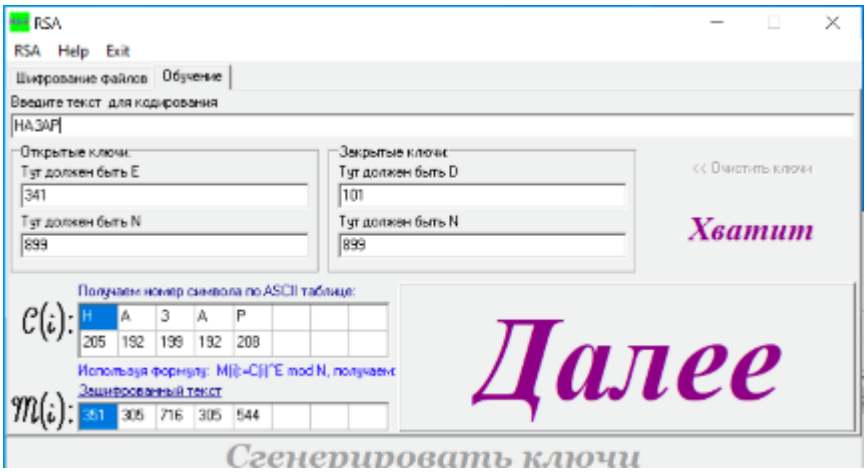

Рисунок 1.8 – Демонстрація роботи алгоритму RSA

шифру, із наочним зображенням шифрування кожного символу та цікавою анімацією роботи (рис 1.8).

В меню програми можна знайти таблицю простих чисел до 200. Також є теоретичні відомості про алгоритм, проте виконані не найкращим чином і погано читаються. Також присутній тест. Всього питань 10. Тестові питання погані, та інколи не відносяться до шифру RSA. До того ж питання неоднозначні, та можуть бути задані кілька разів за одне проходження.

Проаналізувавши другий додаток можна виділити наступні недоліки та переваги.

Недоліки:

- підтримка лише операційною системою windows;
- інтерфейс російською мовою;
- погані тестові запитання;
- відсутність цифрового підпису RSA;
- погані теоретичні відомості.

Переваги:

- добре сформована демонстрація роботи алгоритму;
- наявність деякого звіту;
- таблиця простих чисел.

Третій додаток складається із 6 форм. Кожна форма виконує свою функцію. Перша форма це головне меню, де знаходиться 5 кнопок для переходу до іншим форм.

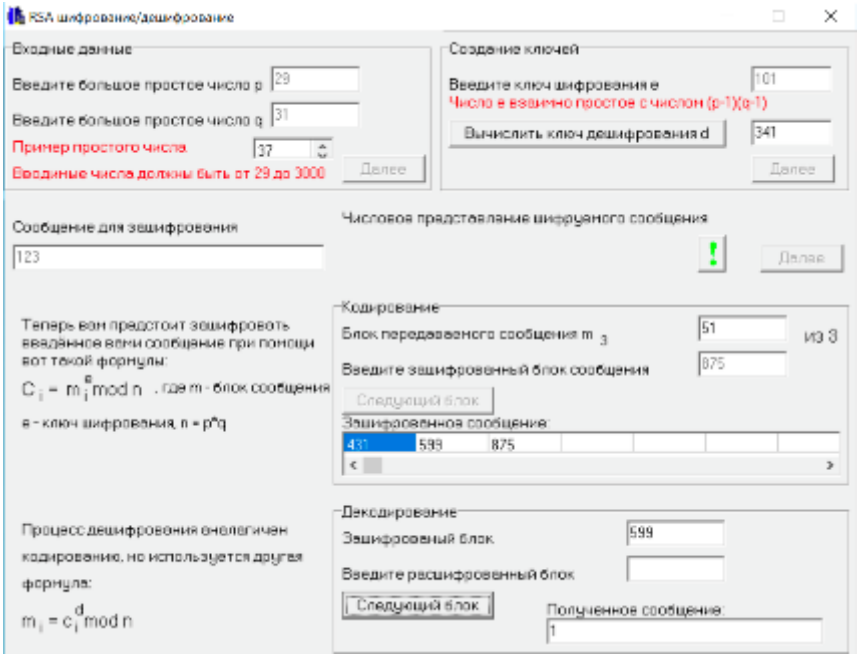

Рисунок 1.9 – Шифрування та дешифрування RSA

Форму шифрування/дешифрування (рис 1.9) дозволяє користувачу пройти практичне завдання з шифрування та дешифрування тексту.

Форма дуже зручна, прості числа можна вибрати із поля за допомогою стрілочок, що виключає помилки від неправильних чисел. Після вибору ключів, треба ввести повідомлення дл шифрування та покроково його зашифрувати. Форма допомагає це робити, проте основна робота по алгоритму лежить на користувачі. Далі потрібно розшифрувати повідомлення. Проте розшифрувати потрібно те саме повідомлення, яке користувач щойно зашифрував, що дає змогу не робити цього. Після проходження шифрування та дешифрування є невеликий звіт. Проблемою даної форми є те, що після її перезапуску кодування може буте неактивне. Ця проблема вирішується перезапуском програми. Вікно з тестуванням мість 10 питань. На кожне питання є 20 секунд на відповідь. Питання актуальні та цікаві. Після проходження програма дає результат, який потрібно зберегти за допомогою

скриншоту. Разом з програмою є скриншоти її роботи для ознайомлення, а також файли з описом функцій та алгоритмом роботи програми.

Проаналізувавши 3 додаток для навчання коду RSA, можна виділити наступні недоліки та переваги.

Недоліки:

- підтримка лише операційною системою windows;
- інтерфейс російською мовою;
- помилки у роботі додатку;
- відсутність цифрового підпису RSA;
- дешифрування можна не виконувати.

Переваги:

- наявність звіту;
- гарні тестові питання;
- зручне проходження практичної частини;
- інформація про роботу з додатком.

1.8 Огляд додатку DSA.exe

DSA.exe – додаток для навчання алгоритму цифрового підпису. Додаток містить в собі демонстрацію роботи на прикладі скриншотів самого додатку, що є не дуже зручно. Проте під час виконання практичного завдання інтерфейс додатку допомагає користувачу краще зрозуміти алгоритм роботи та виконати завдання (рис 1.10). Генерація вхідних даних також спрощує процедуру роботи, не зменшуючи користь від навчання алгоритму. Також є окремий додаток для виконання операції піднесення числа в степінь та ділення по модулю, тому що звичайні калькулятори не можуть впоратись з настільки великими числами, як наприклад 228 в 63912 степені. Це сильно спрощує користувачу роботу з алгоритмом, і погано що такого немає в інших програмах, алгоритми яких основані на труднощі обчислення дискретних логарифмів в кінцевому полі. Окрім практичного завдання, програма містить тестові завдання. Тестових завдань 10. На відповідь дається 20 сек. Питання актуальні та по

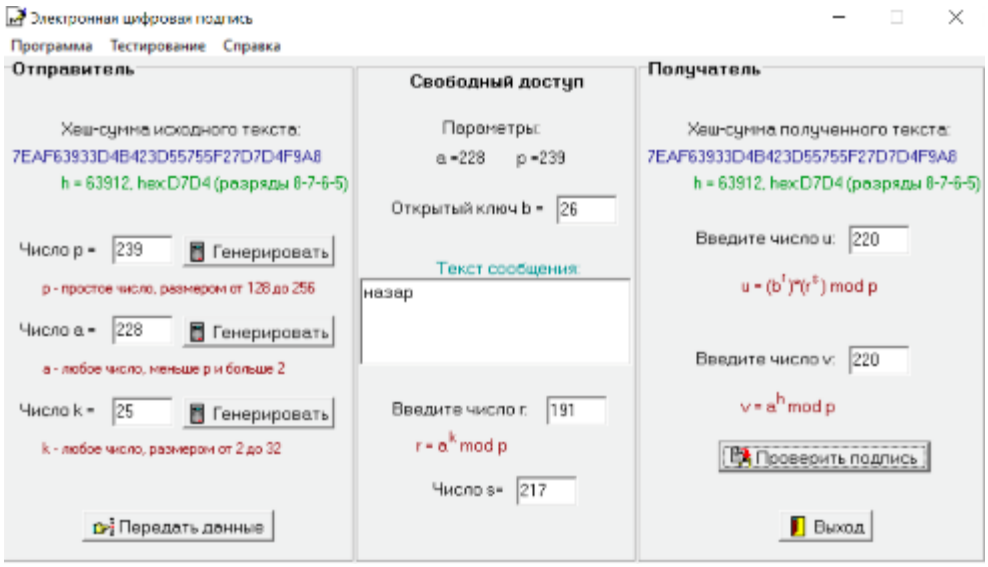

Рисунок 1.10 – Практичне завдання по ЕЦП

добре розкривають тему. Також варто виділити інформацію про роботу в програмі та теоретичні відомості. У вікні довідки є докладний опис роботи з програмою, про те які є можливості та алгоритм роботи. Також добре описані відомості про сам алгоритм та приклад реалізації.

Проаналізувавши додаток DSA.exe, можна виділити наступні недоліки та переваги.

Недоліки:

- підтримка лише операційною системою windows;
- інтерфейс російською мовою;
- відсутність генерації звіту.

Переваги:

- зручний інтерфейс;
- захист від нечесного проходження;
- добре продумані тестування та практичне завдання;
- наявність додатку для піднесення в степінь по модулю.

1.9 Порівняльний аналіз вищезгаданих додатків

Аналіз вищезгаданих додатків показав їх слабкі та сильні сторони, помилки та особливості. Використовуючи ці фактори можна сформувати порівняльні таблиці аналогів (таблиці 1.1-1.2).

Таблиця 1.1 – Таблиця порівняння аналогів

Критерій оцінки	SEZAR.EXE	TRITEM.EXE	SHTIRL.EXE	GAMMIR.EXE
Кросплатформенність	-	-	-	-
Захищеність від нечесного проходження	-	-	+	+
Зручний інтерфейс	-	+	+	-
Підтримка української мови	-	-	-	-
Теоретичні відомості	+	-	-	-
Генерація звіту	+	-	-	-
Автентифікація користувача	-	-	-	-
Онлайн доступ	-	-	-	-
Демонстрація алгоритму	-	+	+	+
Робота в парах	-	-	-	-
Криптоаналіз	-	-	-	-
Експорт даних про проходження	-	-	-	-
Розширюваність системи	-	-	-	-

Таблиця 1.2 – Таблиця порівняння аналогів

Критерій оцінки	Deifie-Hellman.exe	El-Gamal.exe	RSA.exe	DSA.EXE
Кросплатформенність	-	-	-	-

Таблиця 1.2 – Таблиця порівняння аналогів

Захищеність від нечесного проходження	-	-	-	+
Зручний інтерфейс	-	+	-	+
Підтримка української мови	-	-	-	-
Теоретичні відомості	-	+	-	-
Генерація звіту	-	+	+	-
Автентифікація користувача	-	-	-	-
Онлайн доступ	-	-	-	-
Демонстрація алгоритму	+	+	+	+
Робота в парах	-	-	-	-
Криптоаналіз	-	-	-	-
Експорт даних про проходження	-	-	-	-
Розширюваність системи	-	-	-	-

У розділі були оглянуті та проаналізовані існуючі рішення для навчання та тестування з предмету комп’ютерної криптографії. Основним недоліком даних систем є те, що для кожного коду існує окрема незалежна програма, або й кілька. Всі ці додатки не схожі один на одного, не надають спільного інтерфейсу взаємодії.

Користувацький інтерфейс несучасний та часом непередбачуваний. В більшості випадків теоретичні відомості виконані у непідтримуваному зараз форматі довідки, а іноді теоретичні відомості відсутні. Також, зважаючи на вік додатків, користувацький інтерфейс може бути замалий на сучасних моніторах, а текст погано читати.

Ще одним недоліком багатьох систем є відсутність захисту від нечесного проходження практичних завдань. Часто додаток містить функції шифрування та

дешифрування відкритих для користувача, який може підміняти дані в корисних цілях. Після проходження тестування та практичної частини, далеко не всі додатки дають змогу переглянути результат, а ті що дають мають малу кількість інформації.

Додатки були розроблені в період між 1997 та 2005 роками, підтримуються лише в операційній системі Windows. Можливо з часом, додатки взагалі перестануть підтримуватись системою, яка отримає критичні оновлення, як наприклад, в операційній системі OSX 10.15 повністю припинилась підтримка 32 бітних додатків. Також наявні додаткі містять низку багів системи, та видають помилки. Можливо це пов’язано з середовищем запуску.

Отже, було встановлено необхідність розробки нової системи для навчання та виконання робіт з комп’ютерної криптографії. Система має бути єдина для всіх шифрів та бути кросплатформенною. Повинна мати єдиний інтерфейс, бути захищеною від нечесного проходження, мати розгорнуті теоретичні відомості та демонстрації алгоритмів шифрування.

2 ФОРМУВАННЯ ВИМОГ ДО СИСТЕМИ

Вимоги до програмного забезпечення – набір вимог щодо властивостей, якості та функцій програмного забезпечення, що буде розроблено або знаходиться в розробці[6]. Формування вимог системи є важливим етапом, що в ітеративних методологіях[7] розробки програмного забезпечення присутній на кожній ітерації. Формування вимог дозволяє якісно поглянути на процес розробки та скоротити непередбачувані моменти. За свої характером вимоги можна розідлити на функціональні та нефункціональні.

Функціональний характер – вимоги до поведінки системи:

- бізнес-вимоги – визначають високорівневі цілі організації або клієнта – замовника ПЗ;

- вимоги користувача – описують цілі/завдання користувачів системи, які повинні виконуватись користувачами з допомогою розроблюваної системи. Ці вимоги часто представляють у вигляді діаграми варіантів використання;

- функціональні вимоги – визначають функціональність програмної системи, яка повинна бути створена розробниками для представлення можливості виконання користувачами своїх обов’язків в рамках бізнес вимог і в контексті користувацьких вимог.

Нефункціональний характер – вимоги до характеру поведінки системи:

- бізнес правила – містять або зв’язані з корпоративними регламентами, політиками, стандартами, законодавчими актами, внутрішніми корпоративними ініціативами, звітними практиками, алгоритмами обчислень і тд;

- системні вимоги – вимоги до програмних інтерфейсів, системи, обладнання, надійності;

- артибути якості – описують додаткові характеристики продукту в різних параметрах, важливих для користувача і/або розробників. Атрибути якості вказують на питання портованості, кросплатформенності, цілосності, стійкості і тд;

- зовнішні системи та інтерфейси – часто плутаються з «Користувацькими інтерфейсами». Насправді це конкретизація аспектів взаємодії з іншими системами, операційним середовищем, можливості моніторингу при експлуатації[8].

2.1 Функціональні вимоги

Для даної системи були визначені наступні функціональні вимоги:

- система повинна авторизувати користувача по його логіну і паролю;
- система повинна відкрити сторінку авторизації, якщо користувач не авторизований в системі;
- система повинна відкрити головну сторінку, що користувач авторизований;
- система повинна зберегти дані користувача в базу даних при першій авторизації;
- система повинна дати користувачу право вийти із системи;
- система повинна відкрити сторінку авторизації після виходу;
- система повинна дати користувачу можливість перейти до будь якого шифру;
- система повинна мати теоретичні відомості про кожен шифр;
- система повинна мати тестові питання для кожного шифру;
- система повинна мати демонстрацію роботи алгоритму для кожного шифру;
- система повинна мати сторінку для криптоаналізу шифру, де це можливо (в простих шифрах);
- система повинна мати сторінку для парного практичного застосування шифру та електронного цифрового підпису;
- система має показувати оцінку прогресу проходження для кожного шифру;
- система має надавати звіт про проходження кожного шифру;
- система повинна містити єдиний алфавіт для використання в шифрах;
- система повинна мати список фраз для використання в шифруванні.

2.2 Нефункціональні вимоги

Для даної системи були визначені наступні нефункціональні вимоги:

- після натиснення на кнопку авторизації має бути показана анімація очікування;
- завантаження основної частини додатку має займати не більше 10 секунд;
- процес авторизації має займати не більше 5 секунд;
- під час авторизації система має отримати токен користувача із зовнішнього сервісу;
- під час авторизації система має отримати дані користувача із зовнішнього сервісу;
- інтерфейс додатку має бути виконаний у стилі material design;
- навігація між сторінками має займати не більше 2 секунд;
- навігація між сторінками має виконуватись без перезавантаження веб сторінки;
- система має автентифікувати користувача на основі токена;
- система має генерувати токен користувача на основі його логіну по технології JWT;
- система має закріплювати видані практичні завдання за користувачами;
- система має вибирати 10 випадкових тестових тестових питань із переліку для кожного шифру розміром 20 питань;
- система має випадково видавати практичні завдання із загального переліку фраз розміром 100 фраз;

У розділі було сформовано основні вимоги до системи, які були розділена на функціональні та нефункціональні. При формуванні вимог основну увагу було приділено вимогам, які ставлять розроблювану систему вище аналогів, показують її переваги. Також було зосереджено вимоги щодо швидкості роботи системи та зручності користуванням.

3 СЦЕНАРІЙ ВИКОРИСТАННЯ СИСТЕМИ

Перед описом сценарії використання системи було визначено типу користувачів (акторів), які можуть взаємодіяти із системою. Відповідно до сформованих вимог, користувачів було розподілено на наступні ролі:

- «Гість» – неавторизований користувач;
- «Студент» – авторизований користувач;
- «Викладач» – адміністратор курсів.

Для користувача «Гість» доступний лише прецедент авторизації в системі (табл. 3.1).

Таблиця 3.1 – Авторизація в системі

Назва	Авторизація в системі
ID	1
Опис	Користувач вводить дані облікового запису для входу в систему
Актори	Студент
Організаційні переваги	Користувач може користуватись системою
Частота використання	100%
Причини виникнення	Користувач бажає отримати доступ до функціоналу системи
Передумова	Користувач не авторизований
Постумова	Користувач авторизований в системі. Знаходиться на головній сторінці системи
Головний шлях виконання	1. відкрити систему; 2. в поле «Логін» ввести логін; 3. в поле «Пароль» ввести пароль; 4. натиснути кнопку «Вхід».
Виключення	Користувач не зареєстрований в системі 1. у авторизації відмовлено; 2. повернення до 2 пункру головного шляху. Користувач ввів невірний пароль

Для авторизованого користувача «Студент» доступно багато прецедентів, що наведені в таблицях 3.2-3.12.

Таблиця 3.2 – Перегляд доступних тем

Назва	Перегляд доступних тем
ID	2
Опис	Користувач переглядає список тем для ознайомлення
Актори	Студент
Організаційні переваги	Користувач може переглянути теми
Частота використання	100%
Причини виникнення	Користувач бажає отримати доступ до списку тем
Передумова	Користувач авторизований в системі
Постумова	Користувач відкрив сторінку з темами
Головний шлях виконання	1. користувач на будь-якій сторінці системи; 2. користувач натискає на кнопку «Теми» в панені навігації; 3. користувач переходить на сторінку з темами.
Виключення	При виникненні помилки при завантажені сторінки користувач буде перенаправлений на сторінку із відображенням повідомлення про помилку

Таблиця 3.3 – Перегляд свого прогресу проходження теми

Назва	Перегляд свого прогресу проходження теми
ID	3
Опис	Користувач переглядає свій прогрес проходження теми

Таблиця 3.3 – Перегляд свого прогресу проходження теми

Актори	Студент
Організаційні переваги	Користувач може оцінити свій прогрес по проходженню певної теми
Частота використання	70%

Таблиця 3.4 – Відкриття теми

Назва	Відкриття теми
ID	4
Опис	Користувач переходить на сторінку із темою
Актори	Студент
Організаційні переваги	Користувач може перейти на сторінку теми для подальшої роботи
Частота використання	100%
Причини виникнення	Користувач бажає відкрити тему для подальшої роботи
Передумова	Користувач авторизований в системі. Користувач на сторінці із вибором тем
Постумова	Користувач на сторінці із темою
Головний шлях виконання	1. користувач на сторінці із вибором тем; 2. користувач натискає на кнопку «Перейти» на карточці вибраної теми; 3. користувач переходить на сторінку з темою.

Продовження таблиці 3.3

Причини виникнення	Користувач бажає перегляну свій прогрес проходження теми
Передумова	Користувач авторизований в системі
Постумова	Користувач переглянув свій прогрес проходження теми
Головний шлях виконання	1. користувач на будь-якій сторінці системи; 2. користувач натискає на кнопку «Теми» в панені навігації; 3. користувач переходить на сторінку з темами; 4. користувач бачить свій прогрес на карточці із темою.
Виключення	При виникненні помилки при завантажені сторінки користувач буде перенаправлений на сторінку із відображенням повідомлення про помилку

Таблиця 3.4 – Відкриття теми

Виключення	При виникненні помилки при завантажені сторінки користувач буде перенаправлений на сторінку із відображенням повідомлення про помилку

Таблиця 3.5 – Ознайомлення із теоретичними відомостями теми

Назва	Ознайомлення із теоретичними відомостями теми
ID	5
Опис	Користувач переходить на сторінку із теоретичними відомостями та читає їх
Актори	Студент
Організаційні переваги	Користувач ознайомиться із представленими в дотатку відомостями
Частота використання	100%

Таблиця 3.5 – Ознайомлення із теоретичними відомостями теми

Причини виникнення	Користувач бажає підготуватись до тестування та вивчити теоретичні матеріали
Передумова	Користувач авторизований в системі. Користувач на сторінці теми
Постумова	Користувач ознайомився із теоретичними відомостями
Головний шлях виконання	1. користувач на сторінці теми; 2. користувач натискає на кнопку «Теорія» на боковому меню; 3. користувач переходить на сторінку із теоретичними відомостями; 4. користувач ознайомлюється із теоретичними відомостями.
Виключення	При виникненні помилки при завантажені сторінки користувач буде перенаправлений на сторінку із відображенням повідомлення про помилку

Таблиця 3.6 – Ознайомлення із демонстрацією алгоритму роботи шифру

Назва	Ознайомлення із демонстрацією алгоритму роботи шифру
ID	6
Опис	Користувач переходить на сторінку із демонстрацією алгоритму шифру та покроково ознайомлюється
Актори	Студент
Організаційні переваги	Користувач ознайомиться із представленим в дотатку алгоритмом роботи шифру
Частота використання	100%

Таблиця 3.7 – Проходження тестування

Назва	Проходження тестування
ID	7
Опис	Користувач проходить тестування із вибраної теми
Актори	Студент
Організаційні переваги	Користувач пройшов тестування

Продовженя таблиці 3.6

Причини виникнення	Користувач бажає підготуватись до практичного завдання
Передумова	Користувач авторизований в системі. Користувач на сторінці теми
Постумова	Користувач ознайомився із алгоритмом роботи шифру
Головний шлях виконання	1. користувач на сторінці теми; 2. користувач натискає на кнопку «Демонстрація» на боковому меню; 3. користувач переходить на сторінку із демонстрацією роботи шифру; 4. користувач знайомиться із вхідними даними; 5. користувач знайомиться з 1 кроком роботи шифру; 6. користувач натискає кнопку «Далі»; 7. користувач знайомиться із наступним кроком; 8. пункти 6-7 повторюється необхідну кількість разів; 9. користувач натискає кнопку готов в останньому кроці.
Виключення	При виникненні помилки при завантажені сторінки користувач буде перенаправлений на сторінку із відображенням повідомлення про помилку. При виникненні помилки в обчисленні якогось із кроків буде показано повідомлення про помилку та демонстрація почнеться заново

Таблиця 3.8 – Виконання практичного завдання

Назва	Виконання практичного завдання

Таблиця 3.8 – Виконання практичного завдання

ID	8
Опис	Користувач виконує практичне завдання із відповідної теми дотримуючись інструкції
Актори	Студент
Організаційні переваги	Користувач виконав практичне завдання
Частота використання	100%

Продовження таблиці 3.7

Частота використання	100%
Причини виникнення	Користувач бажає пройти тестування
Передумова	Користувач авторизований в системі. Користувач на сторінці теми
Постумова	Користувач пройшов тестування. Користувач отримав оцінку
Головний шлях виконання	1. користувач на сторінці теми; 2. користувач натискає на кнопку «Тест» на боковому меню; 3. користувач переходить на сторінку із тестовими запитаннями; 4. користувач знайомиться із вхідними даними; 5. користувач відповідає на всі питання шляхом вибору 1 правильної відповід; 6. користувач натискає кнопку «Далі»; 7. користувач отримує результат свого проходження тестування
Виключення	При виникненні помилки при завантажені сторінки користувач буде перенаправлений на сторінку із відображенням повідомлення про помилку.

Таблиця 3.9 – Виконання практичного завдання з криптоаналізу

Назва	Виконання практичного завдання із криптоаналізу
ID	9

Таблиця 3.9 – Виконання практичного завдання з криптоаналізу

Опис	Користувач виконує практичне завдання із криптоаналізу відповідно до теми дотримуючись інструкції
Актори	Студент
Організаційні переваги	Користувач виконав практичне завдання з криптоаналізу. Здобув відповідне розуміння предметної області

Продовження таблиці 3.8

Причини виникнення	Користувач бажає виконати практичне завдання
Передумова	Користувач авторизований в системі. Користувач на сторінці теми
Постумова	Користувач виконав практичне завдання. Отримав оцінку
Головний шлях виконання	1. користувач на сторінці теми; 2. користувач натискає на кнопку «Практика» на боковому меню; 3. користувач переходить на сторінку із практичним завданням; 4. користувач знайомиться із вхідними даними; 5. користувач виконує всі необхідні кроки для виконання завдання; 6. користувач вводить кінцеву відповідь у поле «Відповідь»; 7. користувач натискає кнопку «Завершити»; 8. користувач отримує результат свого проходження практичного завдання.
Виключення	При виникненні помилки при завантажені сторінки користувач буде перенаправлений на сторінку із відображенням повідомлення про помилку. При неправильному обчислення кроку користувачу буде показано відповідне зауваження

Таблиця 3.10 – Перегляд довідкової інформації

Назва	Перегляд довідкової інформації
ID	10

Таблиця 3.10 – Перегляд довідкової інформації

Опис	Користувач за необхідності та наявності посилання може переглянути додаткову довідкову інформацію
Актори	Студент

Продовження таблиці 3.9

Частота використання	100%
Причини виникнення	Користувач бажає виконати практичне завдання із криптоаналізу
Передумова	Користувач авторизований в системі. Користувач на сторінці теми
Постумова	Користувач виконав практичне завдання із криптоаналізу.
Головний шлях виконання	1. користувач на сторінці теми; 2. користувач натискає на кнопку «Криптоаналіз» на боковому меню; 3. користувач переходить на сторінку із практичним завданням із криптоаналізу; 4. користувач знайомиться із вхідними даними; 5. користувач знайомиться із інструкцією та методом; 6. користувач вводить кінцеву відповідь у поле «Відповідь»; 7. користувач натискає кнопку «Завершити»; 8. користувач отримує результат свого проходження практичного завдання із криптоаналізу. Результат не впливає на кінцеву оцінку
Виключення	При виникненні помилки при завантажені сторінки користувач буде перенаправлений на сторінку із відображенням повідомлення про помилку. Пунк «Криптоаналіз» може бути відсутній для певних тем.

Таблиця 3.11 – Виконання завдання із практичного застосування шифру

Назва	Виконання завдання із практичного застосування шифру
ID	11
Опис	Користувач може спробувати свої знання шифру на практиці разом з іншим студентом
Актори	Студент
Організаційні переваги	Користувач має можливість отримати знання застосування шифру на реальній практиці
Частота використання	100%
Причини виникнення	Користувач бажає застосувати знання з шифру на практиці
Передумова	Користувач авторизований в системі. Користувач на сторінці теми

Продовження таблиці 3.10

Організаційні переваги	Користувач має можливість отримати додаткові довідкові відомості в системі
Частота використання	100%
Причини виникнення	Користувач бажає ознайомитись із додатковою довідковою інформацією
Передумова	Користувач авторизований в системі. На сторінці доступне посилання на довідкову інформацію
Постумова	Користувач переглянув довідкову інформацію
Головний шлях виконання	1. користувач на сторінці з наявним посиланням на довідкову інформацію; 2. користувач натискає відповідне посилання; 3. користувач переходить на сторінку із довідковою інформацією; 4. користувач переглядає довідкову інформацію;
Виключення	При виникненні помилки при завантажені сторінки користувач буде перенаправлений на сторінку із відображенням повідомлення про помилку.

Таблиця 3.12 – Вихід із системи

Назва	Вихід із системи
ID	12
Опис	Користувач може вийти із системи, аби захистити свій обліковий запис від несанкціонованого доступу, у разі спільного використання пристрою із сторонніми особами
Актори	Студент

Продовження таблиці 3.11

Постумова	Користувач застосував свої знання на практиці
Головний шлях виконання	1. користувач на сторінці теми; 2. користувач натискає кнопку «Практичне застосування» в боковому меню; 3. користувач переходить на сторінку із завданням; 4. користувач знайомиться із памяткою та вхідними даними; 5. користувач виконує покроково завдання та відправляє повідомлення; 6. користувач отримує повідомлення та виконує завдання; 7. користувач натискає кнопку «Завершити»;
Виключення	При виникненні помилки при завантажені сторінки користувач буде перенаправлений на сторінку із відображенням повідомлення про помилку. При виникненні помилки при передачі повідомлення буде показано оповіщення.

Таблиця 3.12 – Вихід із системи

Організаційні переваги	Користувач має можливість захистити свій обліковий запис від дій сторонніх осіб
Частота використання	20%
Причини виникнення	Користувач бажає призупинити користування системою на даному пристрої
Передумова	Користувач авторизований в системі.
Постумова	Користувач деавторизований в системі

Продовження таблиці 3.12

Головний шлях виконання	1. користувач на будь якій сторінці; 2. користувач натискає кнопку зі своїм логіном на головній панелі; 3. користувач натискає кнопку «Вихід» у випадаючому меню; 4. користувач спрямований на сторінку авторизації
Виключення	При виникненні помилки при завантажені сторінки користувач буде перенаправлений на сторінку із відображенням повідомлення про помилку.

Для користувача «Викладач» доступна функціональність взаємодії із API системи для імпорту курсу та результатів проходження, сценарії описані у таблицях 3.13-3.14.

Таблиця 3.13 – Імпорт даних про курс

Назва	Імпорт даних про курс
ID	13
Опис	Викладач може отримати дані про курс, його назву, теми та іншу інформацію
Актори	Викладач
Організаційні переваги	Викладач отримує дані про курс для реєстрації у своїй системі
Частота використання	100%
Причини виникнення	Користувач бажає використовувати курс
Передумова	Користувач авторизований в системі авторизації.
Постумова	Користувач отримав дані про курс

Таблиця 3.13 – Імпорт даних про курс

Головний шлях виконання	1. авторизуватись в системі авторизації; 2. зробити запит на отримання даних про курс; 3. отримати дані про курс.
Виключення	При виникненні помилки при завантажені даних буде надано відповідне повідомлення про помилку

Таблиця 3.14 – Імпорт даних про успішність студентів

Назва	Імпорт даних про успішність студентів
ID	14
Опис	Викладач може отримати дані про успішність студентів в проходженні всіх частин курсу
Актори	Викладач
Організаційні переваги	Викладач отримує дані про успішність студентів для комплексної оцінки
Частота використання	100%
Причини виникнення	Користувач бажає отримати дані про успішність студентів
Передумова	Користувач авторизований в системі авторизації. Курс імпортований для викладача.
Постумова	Користувач отримав дані про успішність студентів
Головний шлях виконання	1. авторизуватись в системі авторизації; 2. зробити запит на отримання даних про успішність; 3. отримати дані про успішність;
Виключення	При виникненні помилки при завантажені даних буде надано відповідне повідомлення про помилку

В даному розділі були визначені основні актори системи. Дії які вони можуть чи не можуть виконувати. Для цього були розроблені та описані прецеденти. Під час опису кожного прецеденту було встановлено необхідні вхідні умови та послідовності дій користувача, як основні так і альтернативні сценарії. Також для графічного зображено побудовано діаграму прецедентів (додаток А).

4 ВИБІР ТА ОБҐРУНТУВАННЯ ЕЛЕМЕНТІВ ТА ТЕХНОЛОГІЙ

Зважаючи на сформовані вимоги до системи потрбіно визначитись із типом системи та технологій розробки. Це дозволить доцільно обрати саме ті технології, які необхідні, аби можливостей обраних технологій вистачило на всі необхідні функціональні та нефункціональні вимоги системи, проте не було занадто багато, тому що, зазвичай, це може збільшити бюджет розробки системи, навантаження на обчислювальну техніку користувачів та сладну підтримку продукту.

Оцінивши вимоги, можна дійти висновку, що кінцева система має складатись із клієнтського SPA[9], серверного додатку, бази даних для збереження інформації, та механізму для миттєвого обміну данними між користувачами.

4.1 Вибір мови розробки

Враховуючи структуру системи із клієнтського і серверного додатків варто обрати мови програмування для їх написання. На клієнтській частині вибирати нема з чого. Повноцінну підтримку має лише JavaScript, проте зараз існує кілька мов, які можуть бути транспільовані в JavaScript, наприклад TypeScript та Dart. Обидві ці мови активно розвиваються компаніями гігантами Microsoft та Google, та є відносно новими. Вони дозволяють суттєво розширити можливість розробника, та надаються йому таку функціональність під час розробки, як декоратори, геттери та сеттери, генератори та ін, які ще не прийняті стандартом ECMAScript. Звісно цей код є так званим, синтаксичним цуктом і в кінцевому варіанті після транспіляції перетворюється на стандартизований варіант, який підтримується всіма актуальними браузерами. Таким чином, розробнику не потрібно реалізовувати складні конструкції, його код виглядає більш лаконічно та просто, при цього виростають можливості та швидкість розробки.

Отже, для розробки клієнтської частини можна виділити 3 мови програмування JavaScript, TypeScript та Dart. Необхідно їх розглянути для вибору оптимальної мови.

JavaScript одна з найпопулярніших мов програмування. Це не дивно, оскільки популярність веб додатків дуже велика. Мова мультипарадигменна, що означає, що можна використовувати її для написання в різних парадигмах як об'єктно-орієнтоване програмування та функціональне програмування. Мова високорівнева з автоматичним управлінням пам'яттю. Має динамічну слабку типізацію. Всі ці характеристики роблять мову досить простою для новачків, проте у великих додатках може викликати плутанина у великій кількості нетипізованого коду. Саме для уникнення таких проблем були створені надстройки над JavaScript, такі як TypeScript.

Typescript мова програмування, яка є надмножиною над мовою JavaScript. Фактично, код написаний на JavaScript повністю сумісний з кодом написаним на TypeScript, що дозволяє поступово переходити на TypeScript, додаючи все нові функції. Головною відмінністю є наявність типів, що робить мову типізованою, проте вони не обов'язкові. Мова надає розробниками низку функціональності, таку як декоратори, оператори перевірки на null, генератори, інтерфейси, кортежі, модулі та ін. Проте з часом всі ці функції, або деякі з ним будуть прийняті стандартом ECMAScript, як це вже сталося наприклад з async/await, а в планах стандартизація декораторів. Проте в TypeScript розробники можуть користуватися цими функціями вже зараз. Мова надає не тільки нові функції, а й стилістичну та синтаксичну перевірку коду на можливі помилки, що спрощує процес розробки та тестування. Зараз багато бібліотек написані на TypeScript, навіть якшо для кінцевого користувача потрібна лише транспільована версія.

Dart не схожий на попередні мови. Це окрема мова, яка була створена також як аналог JavaScript, який має виправити фундаментальні недоліки[10] і зараз є одною з основних мов для розробки додатків для Android та iOS. Проте існує компілятор і набір бібліотек для створення веб додатків для браузера. Таким чином розробники мобільних додатків, які вже знайомі з Dart, можуть розробляти веб додатки. Мова сучасна і надає великі можливості для розробки. В 2014 році ECMAScrtipt затвердили стандарт для Dart[11]. Мова оптимізована для сучасного використання в браузері, де переважають асинхронні операції та роботи з http та даними. Мова добре підійде

розробникам, які вже знайомі з C++, Java та іншими об’єктно-орієнтованими мовами. Також розробники надають зручне апі для роботи з браузером.

Отже, для розробки клієнтської частини системи варто обрати мову TypeScript, порівняно з іншими варіантами вона має наступні переваги:

- розширені можливості порівняно з JavaScript;
- гарна підтримка спільнотою розробників та компанією;
- легший вхід ніж в Dart;
- велика кількість бібліотек;
- гарні можливості для розробки великих додатків;
- підтримка текстовими редакторами;
- велика кількість плагінів для розробки;
- вбудована система перевірки коду на помилки;
- транспіляція коду по стандарту ecmascript;
- більш читаємий код порівняно з JavaScript та Dart;
- можливість дебага коду в популярних IDE;
- інструменти для уніфікації підходу до написання коду в команді.

4.2 Технології клієнтського додатку

Сьогодні односторінкові додатки набули великої популярності. Це пов’язано з розвитком клієнтської обчислювальної техніки, потребами користувачів, та розвитком клієнтського WEB в цілому. Ці додатки не завантажують кожну сторінку повністю із сервера, а завантажують лише необхідні дані, та формують сторінку вже в браузері користувача. Такий підхід дозволяє зменшити кількість трафіку та пришвидшити переходи між сторінками додатку. Мінусом даного підходу, порівняно із завантаженням сторінки із серверу, є те, що перше завантаження додатку може зайняти більше часу, оскільки потрібно завантажити весь необхідний код для роботи механізму SPA.

Для розробки односторінкового додатку звісно можна обійтися і без сторонніх фреймворків та бібліотек, проте це не має ніякого сенсу, оскільки така розробка дуже

складна та довготривала. Тому використання бібліотеки або фреймворку є обов'язковим, більше того, повноцінні односторінкові додатки стали можливі лише після розробки відповідних фреймворків. Зараз для розробки SPA існують багато інструментів розробки, проте найбільш популярні це фреймворк від компанії Google Angular, бібліотека від FACEBOOK React та програміст із компанії Google Еван Ю. Всі ці інструменти є опен сорсними та розробляються зараз не тільки компаніями, а й сторонніми розробника зі всього світу, проте мають і окремі команди розробників, які вирішують напрямок розвитку продукту та займаються основним написання коду. Для вибору фреймворка чи бібліотеки потрібно коротко оглянути кожен із них. В цілому фреймворки та бібліотеки схожі між собою, проте мають ключові відмінності, саме на них і варто зосередитись.

Vue – це бібліотека для розробки односторінкових додатків. Містить інструменти для призв'язки даних та подій, методи життєвого циклу компонента та ін. Проте із коробки бібліотека не містить інструменту для роботи із навігацією, який потрібно встановлювати окремо, інструмента для роботи із формами, для роботи із http запитами. Саме цим відрізняється бібліотека від фреймворку. Проте, Vue має і свої переваги над фреймворками. Основна з них це те, що в'ю можна вбудувати у все існуючу систему із серверним шаблонізатором, для контролю над окремими частинами сторінки. Це дуже зручно при переході від багатосторінкового додатку до односторінкового, тому що немає необхідності створювати додаток з нуля. Також Vue дуже легкий для завантаження, оскільки не несе в собі багато модулів, які можуть бути необхідні, а можуть і не бути. Також кажуть, що в'ю легкий для новачків, проте це суб'єктивно і залежить від розробника, та з чим він працював раніше. За замовчуванням код потрібно писати на мові JavaScript, проте існують методи зміни мови на TypeScript.

React дуже схожий за своїм функціоналом на Vue. Головною відмінністю реакту від інших це те, що для написання інтерфейсів потрібно використовувати мову розмітки JSX (декларативний XML подібний синтаксис). Таких підхід має кілька переваг, наприклад підтримку функцій JavaScript поряд к кодом розмітки та перевірку типів, автодоповнення в редакторі і тд. Проте в такому випадку потрібно писати не

на звичному HTML, а на JSX. React також не містить із коробки бібліотеки для роботи із формами, маршрутизацією та HTTP запитами.

Angular це фреймворк, а не бібліотека, тому він із коробки містить все необхідне для повноцінної розробки для клієнтського веб додатку. Платформа включає в себе інструменти для роботи із формами, включаючи реактивні форми, http клієнт, маршрутизатор, модулі анімацій, модуль для роботи із додатком людьми із обмеженими можливостями, модуль для розробки багатомовних додатків, модуль для тестування, інстумент генерації та збірки додатку та ін. Структура коду розмежовує рівні html, css та ts на різні файли. Angular підтримує написання коду на JavaScript, на TypeScript та Dart.

Отже, для розробки клієнтського додатку доцільно було обрати фреймворк Angular, оскільки він має переваги над Vue та React в рамках сформованих раніше вимог системи, а саме:

- містить модулі для роботи з формами, HTTP, анімаціями та ін;
- можливість розробки на обраній мові typescript;
- підтримка unit та E2E тестування із коробки;
- розмежування файлів з логікою та представленням;
- наявність CLI системи для генерації файлів та збірки проекту;
- підтримка з коробки rxjs;
- наявність dependency injection;
- модульна архітектура додатку;
- активна розробка проекту, кожних пів року виходить нова версія;
- ідеально підходить для великих систем;
- велика кількість сторонніх бібліотек;
- повна підтримка сучасними IDE.

4.3 Технології серверного додатку

Оскільки було визначено, що для клієнтського додатку буде використовуватись односторінковий додаток, який буде брати дані із сервера шлях

http запитів, то для побудови серверної системи необхідно використовувати WEB API. Для побудови WEB API сервера зараз існує безліч фреймворків на різних мовах програмування. Серед найпопулярніших варто зазначити мову Java та фреймворк Spring, C# та ASP.NET, Python та Django, Go та Revel, Ruby та Ruby on Rails, PHP та Laravel, Nodejs та Express і багато інших.

Для сформованих вимог нашої системи немає ніякої різниці у мові та фреймворку серверного застосунку, оскільки для клієнта потрібен лише інтерфейс API. Тому для вибору мови і фреймворку було обрано критерії зручності розробки та доцільності використання. Оскільки для розробки клієнтської частини було обрано мову програмування TypeScript, то ми можемо обрати для серверної частини також мову TypeScript, що дозволить нам спростити процес розробки. Для обраної мови існує платформа Nodejs, яка дозволяє запускати JavaScript локально на комп'ютері або на сервері. Оскільки TypeScript можна транспілювати в JavaScript, то ми можемо вести розробку серверного додатку на TypeScript. Таким чином у нас серверний та клієнтський додатки будуть написані однією мовою, що дозволить легко переключатись між ними та дасть змогу повторно використовувати код.

Nodejs платформа, яка була створена цілком для запуску JavaScript коду на сервері та містить всю необхідну функціональність, як роботу з файловою системою, потоками даних, мережею, кластерами, доменами та ін. Це означає, що можна легко створити сервер на Nodejs без застосування фреймворків та бібліотек. А Nodejs, на відміну від Java, C#, PHP та ін, не потребує сервера для системи, а сам виступає сервером. Таким чином спрощується розробка та масштабування системи.

Платформа Nodejs дуже активно розвивається розробниками в плані кількості модулів, бібліотек, плагінів, фреймворків та ін. Станом на 2019 рік npm (менеджер пакетів nodejs) налічує більше 1 мільйона пакетів. З цього випливає, що оскільки Nodejs дуже популярна для розробки серверних застосунків, і налічує багато пакетів, то для Nodejs існує багато фреймворків та бібліотек для розробки серверних додатків. Найпопулярнішими з них є Express, Koa та Nest.

Express – найвідоміший фреймворк для розробки серверних додатків на Nodejs. Він мінімалістичний та швидкий. Для написання додатків вистачить базових

знать мови JavaScript та платформи Nodejs. При цьому працює дуже швидко. Із коробки надає лише мінімум функцій, проте зважаючи на свою популярність, містить багато модулів розширення, які дозволяють розробляти додатки будь якого розміру, та легко масштабувати.

Nest – досить новий фреймворк, написаний на мові програмування TypeScript, розробниками, що були натхненні структурою Angular[12]. Фреймворк так само використовує декоратори для збільшення функціональності коду. Має модульну структуру, містить в собі Dependency Injection та інструменти для генерації коду і запуску додатку. Проте, зважаючи на великі можливості, обробка запитів може займати більше часу, ніж можливо. А поріг входження для розробників високий.

Koa – фреймворк, написаний паралельно з Express, аби покращити його. Як заявляють розробники, з філософської точки зору, ціль Koa «виправити та замінити Node», в тей час коли Express «розширює Node»[13]. В цілому Koa дуже схожий на Express з точки зору розробника, і ключовою відмінністю є використання асинхронних генераторів, для уникнення так званого «callback hell». Проте зараз це не має сенсу із приходим ES6 та TypeScript.

Проаналізувавши основні фреймворки для розробки серверних додатків на базі платформи Nodejs, було вирішено обрати фреймворк Express через його численні переваги над конкурентами, а саме:

- простота розробки;
- легка масштабованість;
- велика кількість бібліотек;
- підтримка ком’юніті;
- популярність (скачувань Express за тиждень в 20 разів більше ні у Koa та Nest разом узятих);
- швидкість, Express значно швидший за Nest, і такий само по швидкості як Koa;
- підтримка розробки на мові програмування TypeScript;
- широка документація різними мовами;
- легка інтеграція із сторонніми сервіса авторизації та базами даних.

4.4 Технології збереження даних

Для вибору технологій для збереження даних, спочатку було визначено, на основі вимог до системи, критерії даних, а саме:

- загальний середній розмір всіх даних в базі;
- приблизна кількість сутностей моделі;
- частота запитів до бази даних;
- відношення запитів читання та запису;
- необхідність підтримки великої кільсті з’єднань сутностей;
- підтримка обраної платформи розробки серверного додатку.

Зважаючи на вимоги до системи, за даними критеріями, оскільки розмір даних невеликий, запити в більшості на читання, сутністей не багато. Таким чином, врахувавши всі критерії, було вирішено використовувати нереляційну базу даних, оскільки вона має гарну підтримку в уже обраних технологіях розробки.

Із найпопулярніших варінтів нереляційних баз даних можна виділити MongoDb, CouchDb, HPCC, RavenDb.

Врахувавши стек розробки, що вже включає Angular, Nodejs, Express, найкращим рішенням було використати нереляційну базу даних Mongodb. Таким чином було обрано стек технологій розробки MEAN (Mongodb, Express, Angular, Nodejs). MondoDb в даному стеку має багато переваг перед іншими базами даних, як NoSQL так і SQL, а саме:

- кросплатформенність;
- наявність образів для інтеграції в контейнерах;
- наявність драйверів арі та фреймворків для роботи в Nodejs;
- докладна документація з прикладами роботи в Nodejs;
- велика швидкість та продуктивність;
- легка масштабованість;
- гнучкість у використанні;
- підтримка індексування;

- ціна, невеликі проекти можна хостити безкоштовно, а великі дешево.

4.5 Інші технології

Окрім основного стеку технологій розробки клінт-серверного додатку, для реалізації всіх вимог до програмного продукту, було обрано додаткові технології.

Враховуючи вимогу створення функціональності проходження практичного завдання шифрування та цифрового підпису студентами у парах, виникла необхідність використання технології для обміну повідомлень між студентами в реальному часі. Також результат їх взаємодії має бути збережений у сховищі. Таким чином для реалізації такого функціоналу необхідно використовувати технологію WebSocket. Ця технологія дозволяє передавати повідомлення між клієнтом та сервером в режимі реального часу. Таким чином сервер може не лише відповідати на HTTP запити із клієнта, а й з'єднуватись з клієнтом, використовуючи протокол WS. Ця технологія прекрасно підходить для даної функції, та використовується в різноманітних додатках з обміну повідомлень, на кшталт Telegram, Whatsup, соціальних мережах, банківських додатках та інших. Проте існує один ключовий мінус. Розробка додатку з використанням WebSockets займає набагато більше часу і ресурсів. Оскільки функціонал проходження практичних завдань не вимагає широкої бізнес логіки, було вирішено використати готове рішення у вигляді нереляційної бази даних реального часу Firestore.

Firestore – це модуль платформи для розробників Firebase, який надає функціональність бази даних реального часу. Основне застосування Firestore це синхронізація даних між клієнтською частиної і базою даних. Ключовими особливостями є:

- збереження даних у ієрархічній нереляційній структуі;
- розширюваність;
- гнучкість;
- офлайн підтримка;
- оновлення даних як в реальному часі, так і по запиті;

- наявність фреймворків для інтеграції в мобільних, веб та серверних додатках.

Враховуючи наявність офіційної підтримки та бібліотеки для використання Firestore в додатках на Angular та Nodejs, це рішення ідеально підходить для реалізації описаної функціональності системи.

Для реалізації вимог до побудови інтерфейсу, необхідно було обрати фреймворк для побудови користувацьких інтерфейсів. Використання фреймворку значно спростило розробку, надало інтерфейсу єдиний вигляд, та дозволило реалізувати всі необхідні вимоги, на кшталт адаптивності інтерфейсу, зручності та ін.

Оскільки клієнтська частина побудована з використанням фреймворку Angular, було необхідно обрати із бібліотек користувацького інтерфейсу для цього фреймворку. Найпопулярнішими з них є:

- Angular Material;
- PrimeNg;
- Ng-Bootstrap.

Всі ці фреймворки надають список готових елементів для побудови користувацьких інтерфейсів. Дають змогу використовувати налаштовану сітку розташування елементів, теми додатків та ін. Для розробки інтерфейсу користувача було обрано фреймворк Angular Material, через низку переваг, а саме:

- наявність всіх необхідних елементів;
- офіційна підтримка розробниками Angular та Google;
- наявність повної документації;
- швидкість;
- гнучкість налаштування;
- підтримка стандату Google Material[14];
- єдиний стандартизований задокументований інтерфейс;
- оптимізація для Angular;
- наявність шрифтів та іконок.

Таким чином, зважаючи на вимоги до системи, було розглянуто можливі мови та технології розробки. Визначившись з типом клієнтського додатку, було розглянуто варіанти фреймворків для односторінковий застосунків. Оскільки основною мовою

розробки клієнтською частини було обрано TypeScript та фреймворк Angular, вдалося визначитися із серверною платформою та мовою розробки. Завдяки гнучкості та розширюваності платформи Nodejs з'явилась можливість розробляти клієнтські і серверні додатки на одній мові програмування. Це значно скоротило витрати ресурсів на розробку та підтримку, дало можливість повторного використання коду та легкого розгортання додатків. При чому такий підхід надає можливість використання одних і тих же інструментів для розробки, дотримуватись єдиного стилю написання коду. Платформа Nodejs добре підходить для розробки WEB API додатків, а фреймворк Express значно додає можливостей розробникам для побудови масштабовних, продуктивних та швидких серверних рішень. Такий зв'язок клієнтських і серверних технологій підштовхує до використання MongoDb у якості сховища даних. Оскільки MongoDb, Express, Nodejs та Angular описують популярний стек технологій MEAN. Цей стек технологій має переваги великої цілісної підтримки з боку розробників, широкої документації, продуктивності та швидкості роботи та розробки. Що в кінцевому результаті, дає змогу реалізувати цілісну систему, проте із незалежних між собою рівнів. При цьому компоненти системи можуть спільно працювати з додатковими бібліотеками, такою як Firestore, для реалізації обміну миттєвими повідомленнями між користувачами та збереження даних про проходження парної практичної роботи. У зв'язку з численною кількістю розробників на платформі, існує багато готових рішень для реалізації типових завдань у розробці клієнт серверних додатків. З цього випливає, що під час підтримки системи та додавання нового функціоналу, завдяки легкій масштабованості та розширюваності, є можливість реалізації вимог з мінімальним використанням ресурсів.

5 СТРУКТУРНА СХЕМА СИСТЕМИ

Опираючись на технології для розробки системи, та також на вимоги до системи було розроблено структурну схему системи (рисунок 5.1). На схемі можна виділити наступні компоненти:

- Client;
- Server;
- Database;
- ExternalFileStorage;
- Firestore;

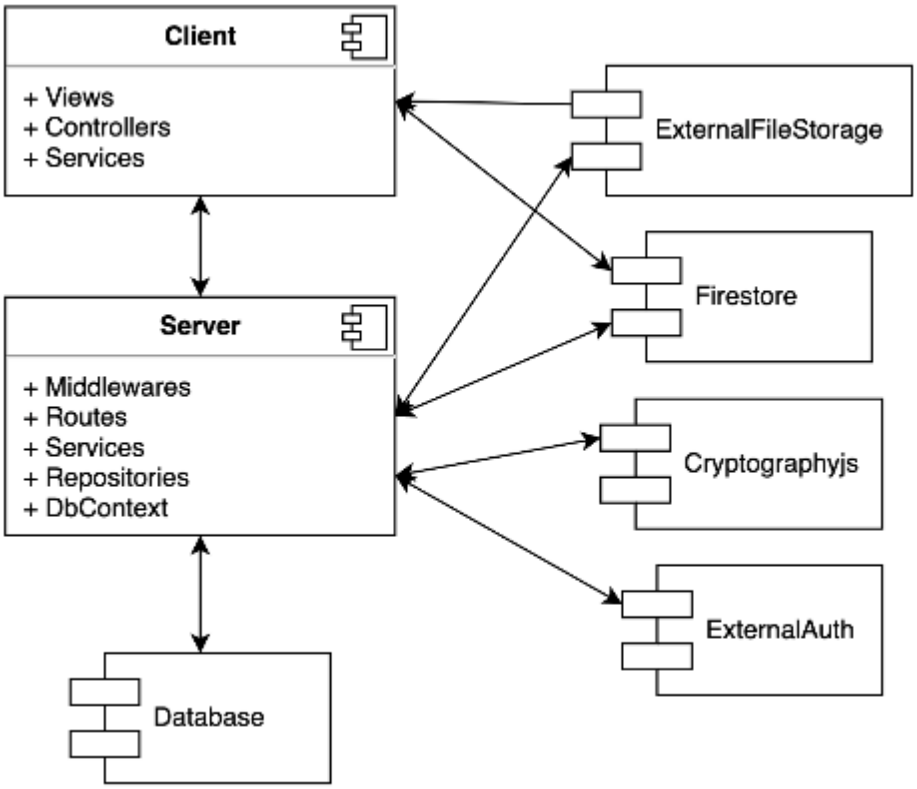

Рисунок 5.1 – Загальна структурна схема

- Cryptographyjs;
- ExternalAuth

Як видно із рисунку система складається із компонентів, які взаємодіють між собою.

Для зберігання даних використовуються одразу кілька компонентів.

5.1 Компонент Database

Компонент Database зображує основне сховище даних системи. Оскільки обраною базою даних є Mongodb, а платформою для розробки серверного рішення Nodejs, то для взаємодії між сервером та базою даних використовується спеціальний Nodejs MongoDB Driver[15]. З’єднання встановлюється по спеціальному протоколі, який зветься mongodb. З боку сервера за з’єднання з базою даних відповідає DbContext. Він за необхідності створює нове з’єднання чи використовує існуюче та надає назовні готову до взаємодії базу даних. Структура компоненту Database

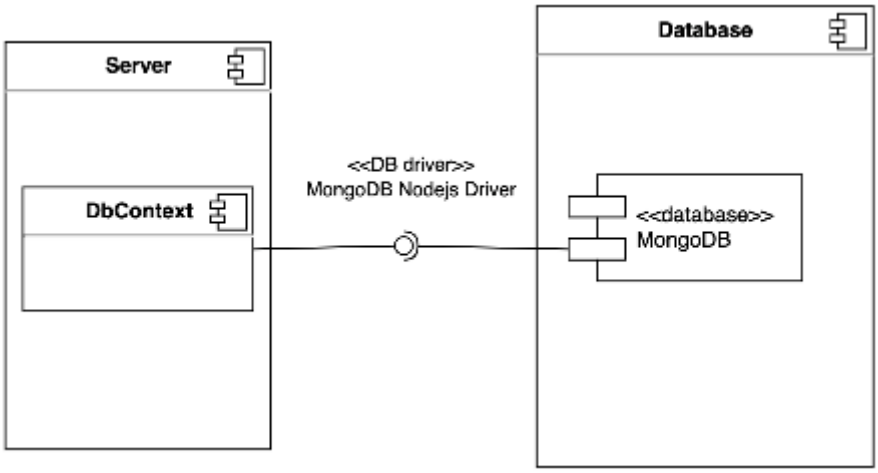

Рисунок 5.2 – Структура компоненту Database

зображена на рисунку 5.2.

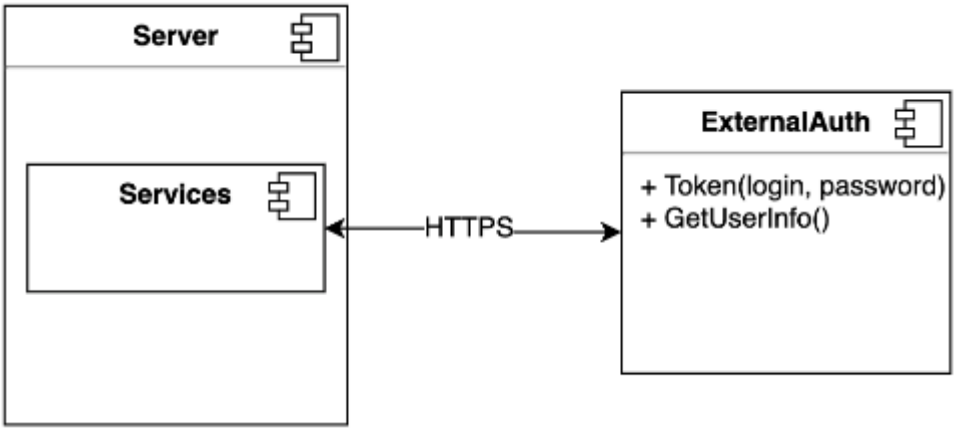

Рисунок 5.3 – Структура компоненту ExternalAuth

5.2 Компонент ExternalAuth

Компонент ExternalAuth використовується для авторизації користувача в системі. Згідно вимог реєстрації користувача відбувається в адміністративній системі викладача. Для авторизації в розроблюваній системі користувач повинен ввести виданий йому логін та пароль. Обмін повідомленнями відбувається по протоколу HTTPS. В серверній частині за взаємодію із компонентом ExternalAuth відповідає сервіс зовнішньої авторизації. Структура компоненту зображена на рисунку 5.3.

Таким чином компонент ExternalAuth забезпечує функціонал отримання токена для роботи із зовнішньою системо, а також отримання інформації про користувача. Компонент взаємодіє лише із своїм сервісом на серверній частині.

5.3 Компонент Cryptographyjs

Компонент Cryptographyjs призначений для всіх основних обчислень з шифрування, дешифрування та цифрового підпису. Компонент взаємодіє із компонентом сервісів. Таким чином вся робота із шифруванням інкаспулюється в окремому компоненті системи. Цей пакет винесений в репозиторій пакетів npm.

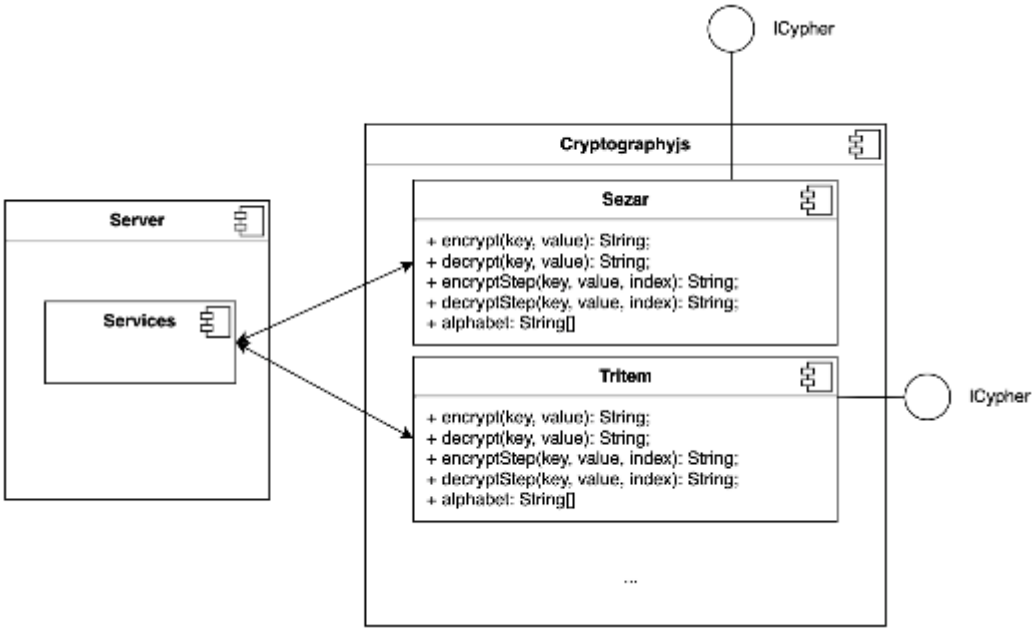

Рисунок 5.4 – Структура компоненту Cryptographyjs

Структура компоненту зображена на рисунку 5.4.

5.4 Компонент Firestore

Компонент Firestore призначений для тимчасового збереження даних необхідних для виконання студентами групових практичних завдань із кодування, декодування і цифрового підпису. Ключової особливістю даної бази даних є те, що взаємодіяти з нею на читання, запис, редагування та інші операції можна не лише із серверної частини системи, а й з клієнтської. Для роботи із базою даних на клієнті необхідні бібліотеки firebase та @angular/fire, які надають інтерфейс взаємодії зручний для розробника. Із клієнтською частиною пакети обмінюються даними за допомогою протоколів HTTPS та WS. Щодо серверної частини системи, то для цього, враховуючи обрані технології, необхідний пакет firebase-admin. Обмін даними із сервером виконується по протоколу HTTPS. Структура компоненту зображена на

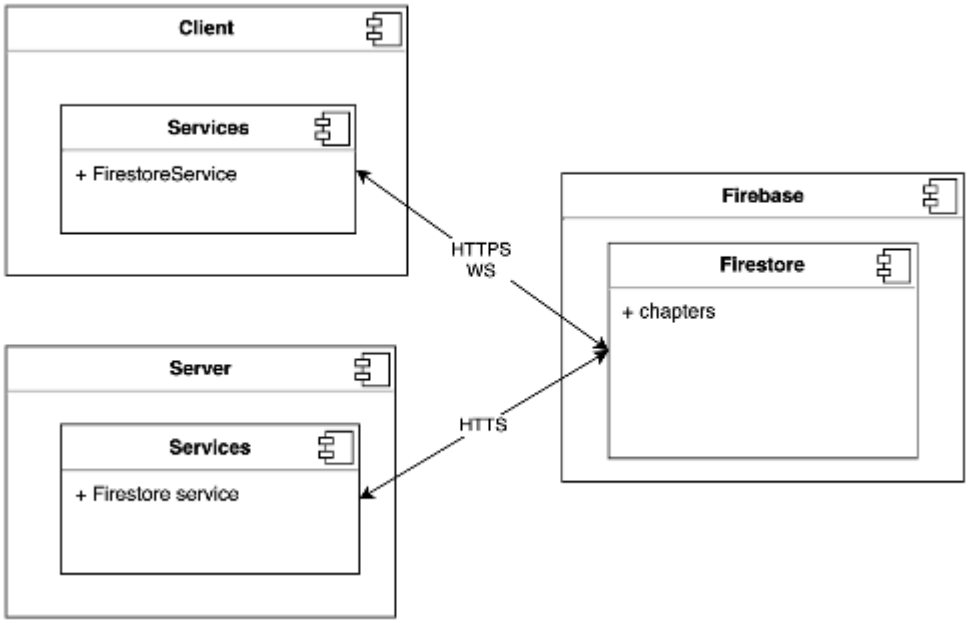

Рисунок 5.5 – Структура компоненту Firestore

рисунку 5.5.

5.5 Компонент Client

Компонент Client призначений для надання користувачу зручного інтерфейсу для роботи із системою. Компонент спілкується в основному із сервером по протоколу HTTPS для обміну даними про дії користувачів та надання користувачам даних. Окрім цього компонент напряму спілкується із базою даних Firestore по протоколу WS та HTTPS. Також компонент може отримувати файли із зовнішнього файлового сховища по протоколу HTTPS. Таким чином компонент Client має багато зв’язків з іншими компонентами, проте завдяки гнучким інтерфейсам взаємодії, або роботу через проміжні компоненти та бібліотеки компонент не зав’язаний на інших

компонентах, і їх можна замінити на необхідності іншими аналогічними за

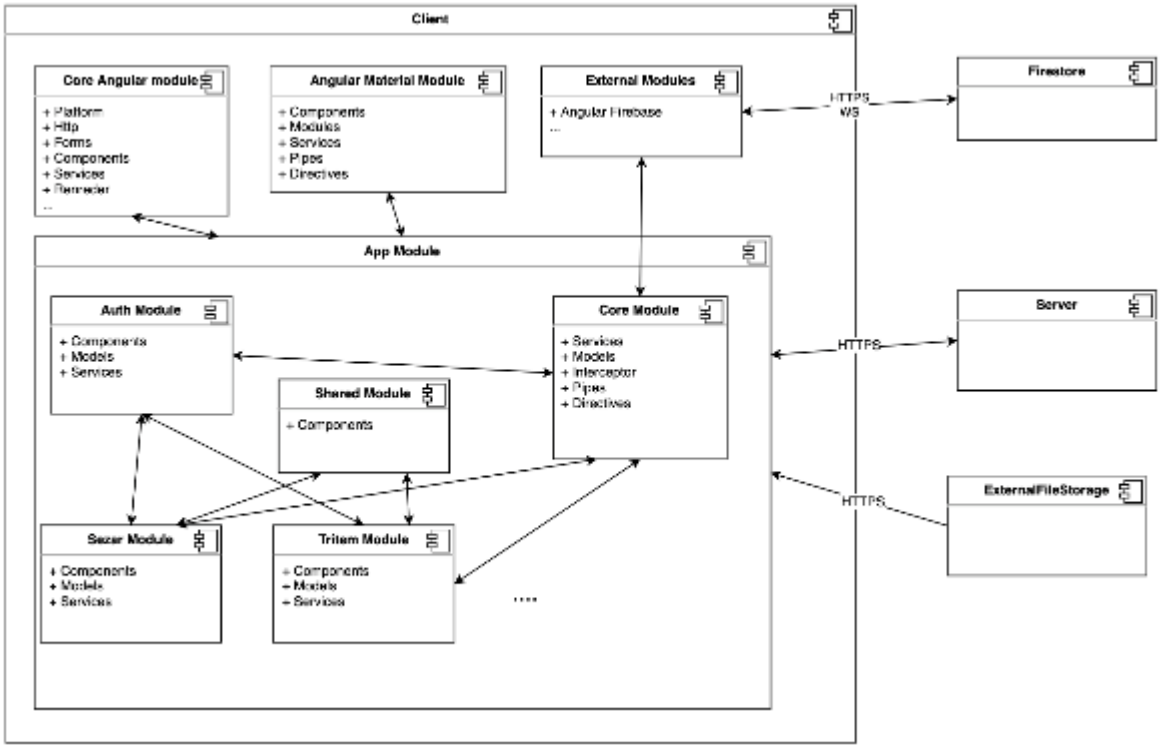

Рисунок 5.6 – Структура компоненту Client

функціоналом. Структура компоненту Client зображено на рисунку 5.6.

5.6 Компонент Server

Компонент Server містить в собі основну частину бізнес логіки системи. Оскільки серверна частина додатку написано на платформі Nodejs, то додаток виступає сам собі сервером. Оскільки, на сервері зосереджена основна частина бізнес логіки всеї системи, то компонент серверу взаємодіє з усіма іншими частинами системи. Для цього компонент серверу містить в собі необхідну структуру

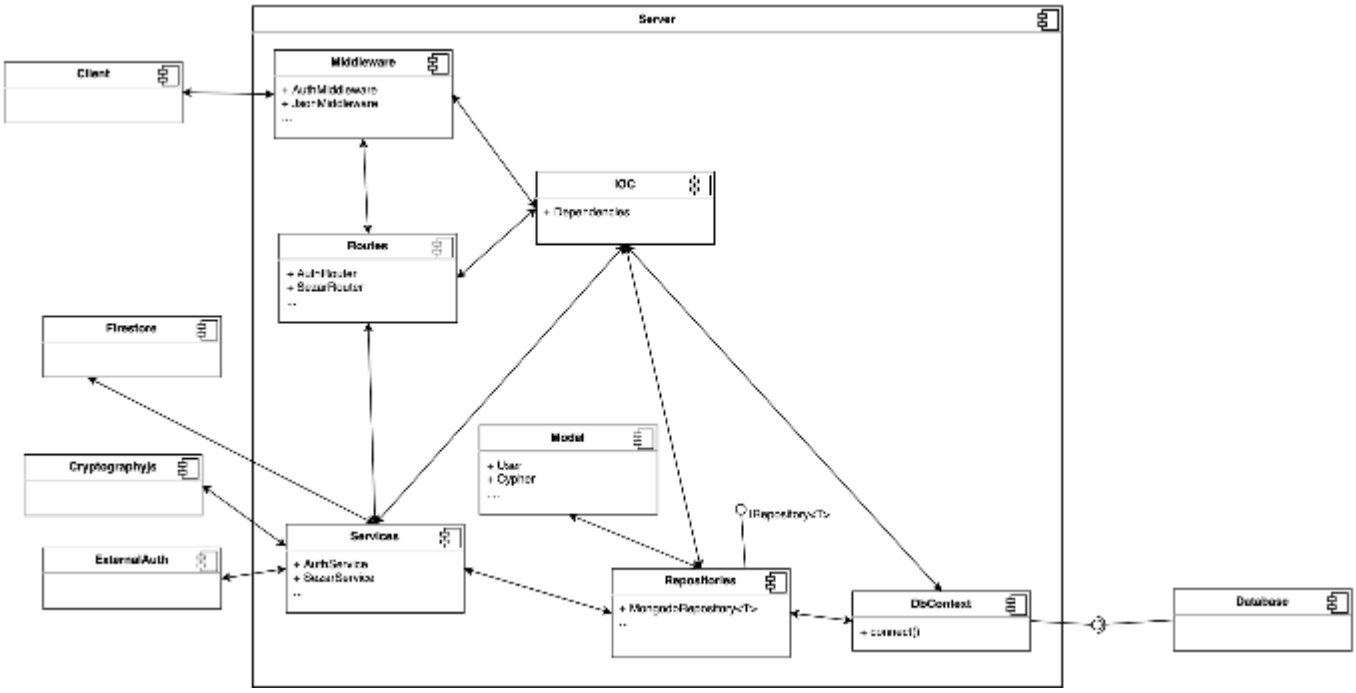

Рисунок 5.7 – Структура компоненту Server

компонентів і рівнів для взаємодії, що зображено на рисунку 5.7.

Отже, на основі визначених функціональних вимог було розроблено структурну компонентну схему системи (додаток Б). Система складається із кількох незалежних складних компонентів, які взаємодіють між собою в основному за допомогою протоколу HTTPS. Компоненти не мають сильно залежності один від одного, а отже можуть бути замінені на інші аналогічні, у випадку коли вони мають необхідний інтерфейс взаємодії. Також на основі компонентів було побудовано

діаграму розгортання (додаток В) Як видно із схеми, бізнес логіку поділяють між собою як клієнтський так і серверний додаток, а інші компоненти є допоміжними. Також варто виділити використання 2 різних баз даних, для виконання функціональних вимог, хоча Firestore може бути замінена реалізацією підтримки WebSocket кімнат на існуючому сервері.

6 СТРУКТУРА БАЗИ ДАНИХ

Зважаючи на вимоги до системи та сформований набір технологій для використання, в якості баз даних було обрано MongoDB та Firebase Firestore. Обидві ці бази даних є документо-орієнтованими нереляційними базами даних. Це означає, що структура збереження даних в таких базах відрізняється від реляційних. Кожен об’єкт, що називається «документ», представляє собою структуру із пар «ключ-значення». Значенням може бути примітив, масив документів або документ. У свою чергу документи знаходяться у колекціях. Для порівняння з реляційними базами даних документ це запис, а колекція це таблиця.

Для проектування структури бази даних необхідно опиратись на можливості бази даних, загальноприйнятні практики з проектування для конкретної бази даних, та специфіку самих даних.

В базі даних MongoDB дані зберігаються у дуже зручній формі для обробки. Дані зберігаються у форматі BSON[16], що дозволяє досягати викої пропускної здатності. Також це дає можливість використовувати дані без завчасної обробки, що зменшує кількість дій над даними. Таким чином при проектуванні структури бази даних варто враховувати як ці дані будуть використовуватись надалі, зв’язки між даними та об’єми. База даних MongoDB не підтримує транзакції, тож було розроблено структуру зважаючи на цей фактор. Також база даних дуже гнучка і дозволяє використовувати тей підхід до зв’язування даних, який розробник вважає за необхідний у його ситуації.

Основними засобами для забезпечення зв’язків між сутностями в MongoDB є вкладення та посилання. Вкладення це коли дані зберігаються в одному документі з іншими пов’язаними даними. Наприклад, коли коментарі до статті зберігаються разом із статтею. Посилання це коли дані зберігаються в окремих документах і посилаються один на інший за допомогою посилання на документ. Варто зазначити, що в цьому разі у випадку зв’язку один до багатьох чи багато до багатьох обидві сутності можуть мати посилання на зв’язану сутність, наприклад, стаття має масив

посилань на коментарі, і коментарі мають посилання на статтю. Це може бути корисно, оскільки немає транзакцій.

Обидва типи зв’язків повинні використовуватись там, де це має сенс. Вкладені сутності повинні бути тоді, коли:

- дочірні дані відносяться до батьківських як один до одного;
- піддокумент малого розміру;
- ієрархічна структура збереження даних.

У свою чергу посилання повинні бути тоді, коли:

- зв’язок багато до багатьох;
- піддокументи можуть рости в розмірі з часом;
- піддокументи великого розміру.

Також посилання мають свої недоліки:

- велика кількість посилань потребує запитів для кожного документу, що є затратним на ресурси;
- редагування піддокументів потрібно робити окремо від документа, що означає більшу логіку даних;
- збільшення дій програміста для задання обробки даних.

У вкладень також є недоліки:

- може бути великий об’єм непотрібних даних в запиті;
- зв’язок багато до багатьох можливий, проте жахливий.

6.1 Стуктура бази даних MongoDB

Враховуючи вимоги до програмного забезпечення та варіанти використання було виділено наступні основні сутності:

- користувач;
- прогрес проходження;
- розділ;
- тест;
- практичне завдання;

- довідка.

Для сутності «User» було визначено структуру документу, що наведена в таблиці 6.1.

Таблиця 6.1 – Структура документа User

Назва поля	Тип	Опис
_id	ObjectID	Унікальний ідентифікатор
uid	String	Зовнішній ідентифікатор
fullName	String	Повне ім’я
email	String	Пошта. Логін
groupCode	String	Код групи
progress	Object	Вкладений об’єкт із інформацією про проходження тем

У сутності User є вкладений об’єкт progress, з відношенням один до одного. Структура документу progress зображена в таблиці 6.2.

Таблиця 6.2 – Структура документа Progress

Назва поля	Тип	Опис
_id	ObjectID	Унікальний ідентифікатор
chapters	Array[ChapterProgress]	Масив розділів

Структура документу ChapterProgress наведено в таблиці 6.3.

Таблиця 6.3 – Структура документа ChapterProgress

Назва поля	Тип	Опис
_id	ObjectID	Унікальний ідентифікатор

Таблиця 6.3 – Структура документа ChapterProgress

Назва поля	Тип	Опис
chapterid	ObjectID	Посилання на розділ
theory	Object	
demo	Object	
test	Object	
practice	Object	
cryptanalysis	Object	
practice2	Object	

Таким чином в документі User зберігаються дані про користувача та прогрес користувача у проходженні розділів курсу.

Структура документу Chapter наведено в таблиці 6.4

Таблиця 6.4 – Структура документа Chapter

Назва поля	Тип	Опис
_id	ObjectID	Унікальний ідентифікатор
theory	String	Посилання на файл із теорією
test	Array[ChapterTest]	Об’єк із даними про тест

Структура документу ChapterTest наведено в таблиці 6.5.

Таблиця 6.5 – Структура документа ChapterTest

Назва поля	Тип	Опис
_id	ObjectID	Унікальний ідентифікатор
question	String	Питання тесту

Таблиця 6.5 – Структура документа ChapterTest

answers	Array[Answer]	Массив варіантів відповідей
rightAnswerID	ObjectID	Ідентифікатор правильної відповіді

Структуру документу Answer наведено в таблиці 6.6.

Таблиця 6.6 – Структура документа Answer

Назва поля	Тип	Опис
_id	ObjectID	Унікальний ідентифікатор
text	String	Відповідь

Таким чином в документі Chapter зберігається інформація про теорію та тестові завдання по розділу.

Структура документу Article зображено в таблиці 6.7.

Таблиця 6.7 – Структура документа Article

Назва поля	Тип	Опис
_id	ObjectID	Унікальний ідентифікатор
title	String	Назва, заголовок
text	String	Текст довідки

Документ Article містить в собі текст невеликих додаткових інформаційних статтей.

Структура документу PracticeTask наведено в таблиці 6.8.

Таблиця 6.8 – Структура документа PracticeTask

Назва поля	Тип	Опис
_id	ObjectID	Унікальний ідентифікатор
plainText	String	Відкритий текст для шифрування
takenBy	ObjectID	Посилання на користувача

Документ PracticeTask містить в собі відкритий текст для шифрування, а також посилання на користувача, який його вирішує в даний момент, що може бути null. Таким чином одне й те ж завдання не буде видане різним користувачам.

6.2 Структура бази даних Firestore

У базі даних Firestore зберігаються дані для виконання користувачами групових завдань. Для обміну захищеними повідомленнями потрібно передавати криптограму у випадку із симетричним шифруванням, а ключі обидві сторони повинні знати заздалегідь. У випадку із асиметричним шифруванням відкриті ключі також потрібно викласти у відкритий доступ. Таким чином потрібно зберігати відкриті ключі, для деяких розділів і повідомлення користувачів. Для цього зручно використати ієрархічну структуру.

В коллекції Chapters будуть зберігатись дані про групове шифрування в кожному розділі. Структура документу наведена в таблиці 6.9.

Таблиця 6.9 – Структура документа Chapter

Назва поля	Тип	Опис
id	String	Унікальний ідентифікатор
chapterId	String	Зовнішній ідентифікатор із основної БД
openKeys	Array[OpenKey]	Відкриті ключі абонентів. Може бути пустим

Таблиця 6.9 – Структура документа Chapter

messages	Array[Message]	Повідомлення між користувачами

Структура документа OpenKey зображено в таблиці 6.10.

Таблиця 6.10 – Структура документа OpenKey

Назва поля	Тип	Опис
id	String	Унікальний ідентифікатор
userId	String	Зовнішній ідентифікатор користувача із основної БД
Ключ	String	Відкриті ключі. Може бути декілька із зазначенням назви

Структура документа Message зображено в таблиці 6.11.

Таблиця 6.11 – Структура документа Message

Назва поля	Тип	Опис
id	String	Унікальний ідентифікатор
transmitterId	String	Зовнішній ідентифікатор відправника із основної БД
receiverId	String	Зовнішній ідентифікатор отримувача із основної БД
message	String	Зашифроване повідомлення

Загальна структура баз даних наведена в додатку Г.

Отже, враховуючи специфіку роботи з обраними базами даних MongoDB та Firebase Firestore, було розроблено структуру баз даних. Оскільки обрані бази є нереляційними та гнучкими у підходах до збереження даних це дозволило значно спростити структуру та уникнути складних конструкцій. Таким чином досягаються переваги щодо зменшення кількості запитів та бази даних та можливість до мінімальної обробки даних перед видачею, що в свою чергу зменшує навантаження на сервер та кількість роботи розробника. Структура вийшла розширювана та масштабована. Обрані бази даних дозволяють легко додавати нову інформацію до сутностей, що може знадобитись під час підтримки.

7 РЕАЛІЗАЦІЯ БІЗНЕС-ЛОГІКИ СИСТЕМИ

На основі вимог до системи та побудованої структурної схеми було розроблено загальну архітектуру системи. Для розробки системи було обрано монолітну архітектуру, оскільки необхідне легке розгортання системи на серверах, та за необхідності контейнеризація всього додатку. Для масштабування монолітні додатки зазвичай дублюються на різних серверах, що якраз підходить під вимоги до системи. Оскільки системи логічно складається із 2 частин: клієнтської та серверної, то бізнес логіка розділена в цих частинах. У кожній частині реалізована багаторівнева архітектура.

Завдяки використанню багаторівневої архітектури функціонал рівнів може багаторазово використовуватись. Такий підхід дає перевагу в меншому обсязі коду як взагалі, так і в окремих викликах дає змогу не повторювати код, що є одним із принципів розробки програмного забезпечення DRY[17].

У додатках з багаторівневою архітектуру встановлюються обмеження на взаємодію між рівнями. Кожен рівень має зв’язок тільки із сусідніми рівнями, а залежить тільки від нижчого рівня. Таким чином при змінах в одному із рівнів, які впливають на інтерфейс взаємодії із ним, вплим буде лише на один рівень, що дає змогу легко підлаштуватись під новий інтерфейс. Також це означає, що таким чином ми можемо легко замінювати рівні на аналогічні, з аналогічним інтерфейсом та з іншою логікою всередині. Це надає гнучкість при розробці системи, а також при тестуванні та розгортанні. Наприклад, додаток може використовувати базу даних OracleDB, робота з якою буде інкапсульована на рівні взаємодії із базою даних, а потім замінити цей рівень на аналогічний з аналогічним інтерфейсом, який в свою чергу вже буде взаємодіяти із базою даних MSSQL. Також під час тестування системи, використання багаторівневої архітектури дає можливість замінити рівень взаємодії з базою даних на фіктивну реалізацію, що дозволяє протестувати роботу рівня взаємодії із рівнем бази даних без втручання в реальну базу даних, чи навіть тестову, що у свою чергу підвищує швидкість виконання тестування, та знижує вартість використання ресурсів.

Поділ на логічні рівні широко поширений і допомагає організувати код.

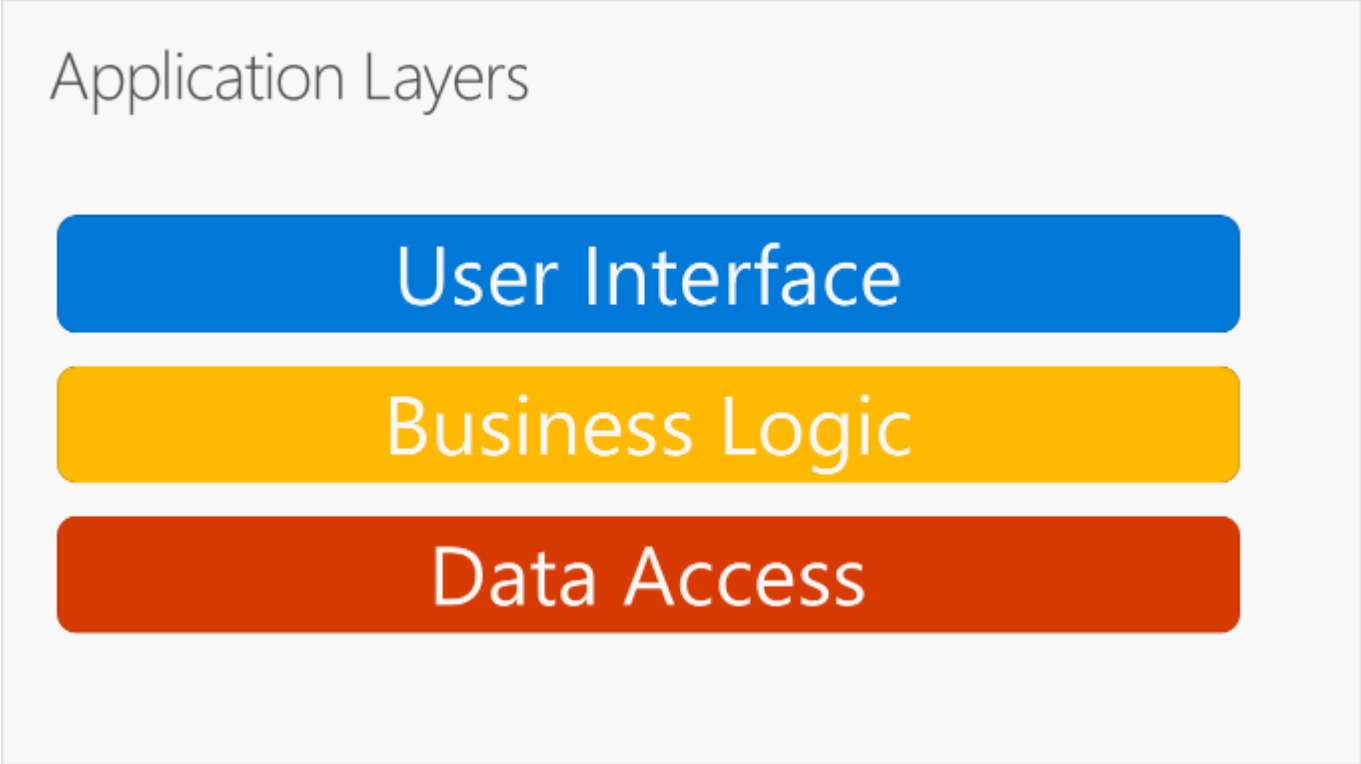

Рисунок 7.1 – Рівні типового додатку

Загальна організація логіки по рівнях зображена рисунку 7.1[18].

За такого поділу в додатку визначають рівні призначення для користувацького інтерфейсу, бізнес-логіки та рівень доступу до даних. За такої архітектури користувачі виконують запити через рівень користувацького інтерфейсу, який у свою чергу взаємодіє із рівнем бізнес логіки. Рівень бізнес логіки взаємодіє із рівнем представлення даних. При цьому користувач не може взаємодіяти із рівнем даних напряму.

Оскільки бізнес логіка розроблюваної системи розділена на клієнтську і серверну частини, то вони одночасно і входять у базові 3 рівні і мають кілька рівнів всередині.

7.1 Клієнтська частина

Оскільки клієнтська частина система використовує фреймворк Angular, який наполегливо рекомендує у розробці притримуватись своїх рекомендацій до розробки то архітектура клієнтської частини схожа на типову архітектуру Angular проектів

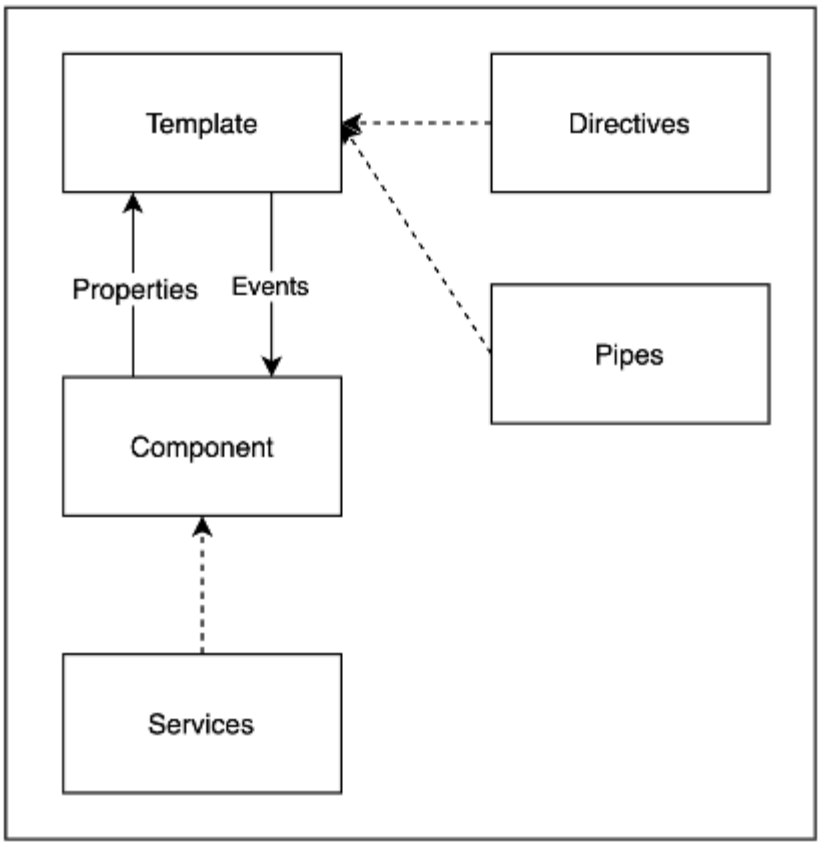

Рисунок 7.2 – Архітектура клієнтського додатку

(рисунок 7.2).

Додаток складається із модулів. Модуль це базова одиниця розробки. Всередині модуль містить інформацію про компоненти, сервіси, пайпи та інші залежності, які він має в собі, може віддати іншому модулю або імпортує в себе. Основними складовими модулів є компоненти. Компоненти відповідають за розмітку компонента, його стилі та логіку взаємодії. Таким чином компоненти складаються із 3 файлів відповідно. Основне завдання компонентів це передавати дані у шаблон, та отримувати дані із шаблону. Для обробки даних і бізнес-логіки компоненти використовують сервіси. Сервіси це класи, де зосереджена бізнес-логіка окремих функцій системи. Також шаблон може містити у собі директиви та пайпи, які відповідають за структурні, стильові та інші зміни представлення. Головним модулем додатку є AppModule.

AppModule містить в собі точку входу в додаток. В модулі вказано, що основним компонентом є AppComponent. Це означає, що при завантаженні сторінки буде показано AppComponent. Проте логіка в кореневому компоненті відсутня. Його завдання це надати контейнер для загальної маршрутизації додатку. Таким чином, окрім AppComponent, AppModule містить в собі AppRoutingModule, який керує лінивим завантаженням інших модулів при маршрутизації.

Ліниве завантаження модулів дозволяє завантажувати код модулів при переході на певний маршрут. Таким чином зменшується час для першого завантаження додатку. Окрім цього код після компіляції також буде розділений по модулях, що дає можливість проводити аналіз розміру модулів та залежностей і зменшувати розміри мініфікованих файлів для завантаження. Це особливо корисно, коли додаток складається із низки окремих великих модулів, які мають окрему бізнес-логіку, стилі та особливо доступи. Таким чином, для неавторизованого користувача

можна взагалі не завантажити код додатку, який доступний лише авторизованим користувачам. Схема маршрутизації системи наведена у додатку Д.

Після завантаження додатку, система за допомогою спеціального сервісу захисника перевіряє чи користувач авторизований в системі. Цей сервіс є складовою модуля AuthModule. Якщо користувач не авторизований, то система завантажить модуль авторизації, а якщо авторизований то головну сторінку.

AuthModule відповідає за авторизацію користувача та все що з цим пов’язано. При завантаженні цього модулю в маршрутизації додатку додається маршрут для сторінки авторизації. Сторінкою авторизації займається AuthComponent. Цей компонент містить форму для введення користувач логіну та паролю. Далі компонент викликає сервіс авторизації і передає туди логін та пароль. Сервіс у свою чергу при успішній авторизації повертає токен користувача. Далі токен та інші дані про користувача зберігаються у UserService. Таким чином AuthModule містить компонент з формою для авторизації, сервіс авторизації, який спілкується із сервером за допомогою HTTPS запитів, сервіс користувача, який зберігає дані про авторизованого користувача в системі та сервіс захисник, який контролює доступ до ресурсів системи авторизованими та неавторизованими користувачами. Схема авторизації наведена у додатку Е.

Також було розроблено спеціальний сервіс перехоплювач TokenInterceptor, який контролює кожен запит, що надсилає додаток. Створює його копію додаючи в заголовки запиту заголовок Authorization з токеном авторизованого користувача. Таким чином зникає необхідність вручну прикріплювати токен в кожен запит та використання сервісу AuthService де не потрібна робота з авторизацією напряму.

Для авторизованого користувача відкривається повний доступ до функціоналу системи і починається він із головної сторінки. Головна сторінка HomeComponent є складовою HomeModule. HomeComponent це сторінка контейнер, яка містить в собі головну панель для навігації та місце для контейнер для завантаження сторінок. Таким чином було уникнуто повторного написання та завантаження компонента головної панелі. Також це дає перевагу в правильній ієрархічній структурі навігації. Головна панель дозволяє швидко отримати доступ до

управління користувачем та навігації до розділів та довідки із будь-якого місця всередині навігації HomeComponent.

Основним модулем системи є модуль ChaptersModule. Це є кореневий модуль для роботи із розділами. Він містить в собі модуль для маршрутизації, модуль для списку розділів та модулі для кожного розділу. Кореневий компонент модулю слугує контейнером для маршрутизації сторінок розділів та сторінки зі списком розділів. ListComponent відповідає за відображення каталогу із доступних розділів, а також функціонал для переходу на сторінку потрібного розділу. ListComponent за допомогою UserService отримує інформацію про успішність користувача у проходженні розділів та відображає інформацію про відсоток проходження того чи іншого розділу. Далі користувач при переході завантажує модуль необхідного розділу в якому міститься вся необхідна логіка для роботи з ним.

Всі модулі для розділів схожі за своєю структурою. Кореневий компонент, наприклад CaezarComponent містить в собі панель навігації по сторінках розділу на контейнер для завантаження необхідних сторінок для роботи. За замовчуванням завантажується сторінка із теоретичними відомостями.

Для кожного пункту меню існує компонент який відображає відповідну сторінку, а саме:

- TheoryComponent;
- DemoComponent;
- TestComponent;
- PracticeComponent;
- CryptanalysisComponent;
- Practice2Component.

TheoryComponent відображає теоретичні відомості для відкритого розділу. Теоретичні відомості зберігаються у вигляді файлу формату markdown, посилання на який компонент отримує від сервісу CaezarService. Далі компонент відображає вміст файлу за допомогою директиви markdown із пакету ngx-markdown. Таким чином досягається уніфікації взаємодії із теоретичними відомостями. Оскільки вони зберігаються у відкритому форматі розмітки їх легко редагувати, можна описувати

складні математичці рівняння, вставляти посилання, картинки та інше. За домогою такого підходу було уникнуто написання великої кількості стилів для відображення теорії, а також отримано механізм універсальної та однакової для всіх сторінок відображення тексту із медіа вставками.

DemoComponent відповідає за демонстрацію користувачу роботи шифрування та дешифрування. Оскільки цей процес абсолютно різний для кожного розділу та повинен мати велику наочних елементів для демонстрації його неможливо отримати із серверу або якось згенерувати. На сторінці демонстрації показано покрокову роботу із процесом шифрування та дешифрування на реальному прикладі. Основною частиною бізнес логіки для цього компоненту займається DemoService, який завантажує із серверу необхідні дані, такі як алфавіт шифрування, ключ, відкритий текст. А також займається взаємодією з сервером щодо покрокового шифрування та дешифрування (більш докладно процес описаний в підрозділі серверної частини).

TestComponent містить в собі сторінку з тестовими питаннями для користувача. За допомогою TestService компонент завантажує перелік питань із сервера. На основі цього переліку динамічно генерується форма із набором контролів для кожного питання. У кожного питання може бути 1 правильний варіант відповіді, проте як питань так і варіантів може бути необмежено. Зазвичай це 10 питань і 3-4 варіанти відповіді. Для генерації форми було використано ReactiveFormsModule, що дозволяє генерувати форму на основі даних. Також за допомогою структурних директив було досягнуто динамічної кількості питань та варіантів відповіді. Після виконання користувачем тестових питань сервіс відправляє дані на сервер та отримує результат. Після цього результат показується користувачу. За необхідності користувач може пройти тест ще раз, проте питання можуть відрізнятись, а також буде врахована кількість спроб.

PracticeComponent один із ключових компонентів розділу, адже практика в навчанні криптографії дуже важлива. По структурі сторінки практика трохи схожа на демонстрації, адже також вказані алфавіт, ключ, відкритий текст. Проте тут користувачу доведеться більшість обчислень виконувати самому та заносити у

відповідні поля для перевірки. Згідно вимог, компонент практики повинен максимально допомагати користувачу у проходженні, проте не занадто. Де необхідно додаток може виконати обчислення, яке не підвласне більшості калькуляторів. Для взаємодії із сервером та виконанні обчислень розроблений PracticeService. Він містить в собі необхідну логіку для взаємодії із сервером щодо отримання завдання та відправлення результатів (більш докладно процес описаний в підрозділі серверної частини).

CryptoanalysisComponent є досить простим компонентом в плані своєї бізнес логіки. Спочатку надається інформація про способи взлому даного шифру, їх може бути кілька. Далі користувачу доступне завдання розшифрувати перехвачену криптограму. Для цього немає ніякої допомоги зі сторони додатку, адже існує багато варіантів взлому і користувач повинен сам обрати спосіб та виконати обчислення. Також при криптоаналізі обчислення можуть не бути зовсім точними, проте зрозумілими людині, тому для виконання завдання потрібно лише ввести результат виконаної роботи в форму. Для бізнес-логіки основним сервісом є CryptoanalysisService, який надає компоненту завданя та приймає відповідь.

Practice2Component – сторінка для інтерактивного виконання практичного завдання користувачами. Основною особливістю є оновлення даних на сторінці в режимі реального часу. Така взаємодіє реалізована за допомогою Firebase Firestore. Для взаємодії із базою даних із клієнту використовується пакет @angular/fire. Робота з інтерфейсом цього пакету та його налаштування інкапсульовані в сервісі FirestoreService. Цей сервіс разом з Practice2Service є основними, які виконують більшість процесів бізнес-логіки даної сторінки. Основними завданнями є завантаження та оновлення списку відкритих ключів користувачів та можливість передачі повідомлень між користувачами в режимі реального часу.

Діаграма пакетів клієнтської частини наведена у додатку Ж.

7.2 Серверна частина

На відміну від клієнтської частини для розробки додатку на платформі Nodejs та Expressjs немає жорстких вимог то проектування архітектури. Тому при проектуванні було дотримано загальні правила для побудови серверних REST API додатків. Оскільки Express реалізує паттерн chain of responsibility, коли запит послідовно проходить обробники, які задовольняють його вимоги то було вирішено використати багаторівневу архітектуру (рисунок 7.3).

7.2.1 Рівень проміжних обробників

Middlewares – рівень проміжних обробників запиту. В основному використовується для зміни запиту перед обробкою контролером або й після. Також може використовуватись для додаткових дій, таких як логування чи налаштування. Для розроблюваної системи було виконано кілька проміжних обробників, таких як:

- CORSMiddleware – займається додаванням в заголовки запиту заголовків пов’язаних із Cross-Origin Resource Sharing. Так як додаток є відкритим API, то необхідно було додати дозволи на використання API на різних доменах;

- AuthMiddleware – відповідає за авторизацію користувача на сервері по його токені. Спочатку обробник перевіряє токен на валідність і декодує його за допомогою AuthService і якщо токен хибний повертає помилку 401. Якщо все нормально то використовуючи UserService по розкодованих даних із токена отримує об’єкт користувача системи і записує його в об’єкт запиту. Таким чином при обробці запиту контролером, запит вже буде містити в собі готовий об’єкт користувача для зручної роботи;

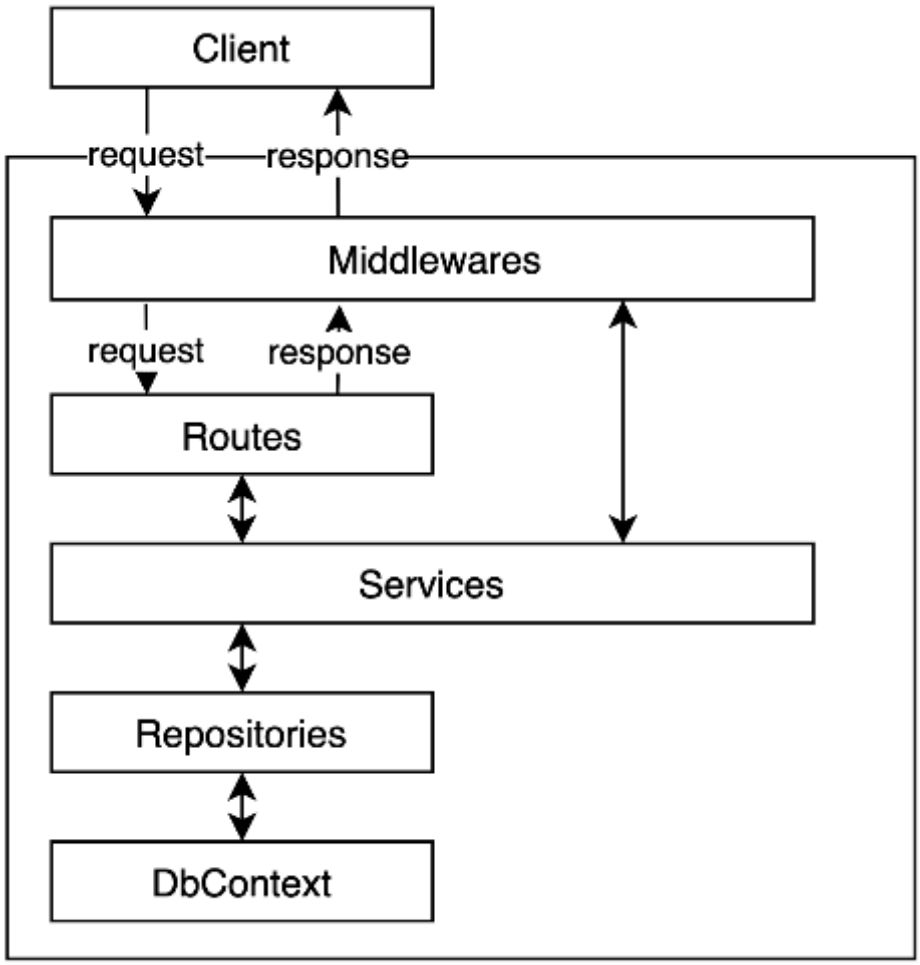

Рисунок 7.3 – Архітектура сервера

- TeacherMiddleware – обробник який використовується не на всіх маршрутах, а лише там де вказаний, а саме в маршрутах пов’язаних із імпортом даних про успішність студентів на зовнішній сервіс викладача. Використовує параметр teacherAPIKey за допомогою якого TeacherService перевіряє дозволи запиту на отримання інформації для викладача.

7.2.2 Рівень маршрутів

Routes або контролери відповідають за обробку запитів. По своїй суті відрізняються від проміжних обробників тим, що оброблюють запити по маршрутах. Оскільки серверний API додаток забезпечує підхід REST[19], то контролери реалізують маршрути для операцій GET, POST, PUT, DELETE та не зберігають стан

системи. Як і у випадку з контролерами на клієнтський частини, контролери сервера не тримають в собі багато бізнес-логіки, а делегують завдання сервісам. Усі маршрути однакову сигнатуру, яка показує що вони асинхронні, та приймають на вхід об’єкти запиту та відповіді. Із запиту дані читаються, а у відповідь дані записуються. Завдання контролера считати зані із запиту, віддати сервісам на обробку, отримати результат і в залежності від нього надати відповідь клієнту. Наприклад AuthCompoller має обробник для маршруту POST /api/auth. При запиті дані із запиту, такі як логін та пароль передаються в сервіс авторизації AuthService для зовнішньої авторизації. У відповідь обробник отримує Promise<ExternalAuth> в якому міститься токен для роботи із сервером зовнішньої авторизації. Далі за допомого сервісу авторизації отримуються дані про користувача від серву зовнішньої авторизації. Після цього за допомогою UserService перевіряється чи зареєстрований юзер в системі. Якщо так то за допомогою AuthService генерується токен і віддається клієнту, якщо ні то користувач реєструється і так само генерується токен. Таким чином у обробнику авторизації зосереджена велика частина бізнес-логіки, шо включає в себе роботу з базою даних, користувачами та запити на зовнішній сервер авторизації, проте завдяки багаторівневої архітектури, принципу єдиної відповідальності обробник лише взаємодії із двома сервісами, які повертають дані у зручному асинхронному форматі.

Аналогічно із клієнтською частиною на серверній частині розроблено контролер для кожного розділу системи. Ці контролери відповідають за надання інформації про шифри, тестування та практичні завдання. Кожен контролер містить перелік маршрутів для необхідного функціоналу.

getTheory – функція контролера, яка віддає посилання на теоретичні відомості про шифр. Для отримання даних використовується SezarService, в якому міститься велика кількість бізнес-логіки для роботи з цим шифром.

getTests – функція контролера для отримання тестових завдань. В базі даних знаходиться перелік із більше 10 запитань. Для тестування користувача вибираються випадкові 10 питань за допомогою CaezarService. На клієнтську частину не передається інформація про правильні відповіді, лише питання та варіанти.

postTests – функція контролера для отримання відповідей. Формат тіла запиту включає массив об’єктів питання та відповідь. Далі ці дані передаються в сервіс, який перевіряє не лише правильність відповідей на питання, а й звіряє отримані питання та питання, які були видані користувачу. Поточний тест зберігається в базі даних у користувача. Таким чином реалізується захист процесу тестування від нечесного проходження, адже якщо просто перевіряти питання і відповідь, то існує кілька варіантів обману системи. Так як система реалізована у вигляді відкритого REST API, то користувач може відправити результат на сервер у вигляді JSON об’єкту за допомогою POSTMAN[20] чи навіть командного рядку. Наприклад, користувач може взяти готовий об’єкт із пройденими тестами у іншого користувача та відправити від свого імені. Таким чином, якщо просто перевіряти тести на правильні відповіді, то користувачі зможуть обманувати систему. Іншим варіантом є дублювання питань. Так як питань 10, то користувач може вибрати 1, на яке він знає правильну відповідь, та відправити на сервер об’єкт із 10 копій одного питання. Звісно цей варіант можна знешкодити, якщо просто перевіряти питання на унікальність, але для повної впевненості в захищеності від обману тестування користувачів необхідно зберігати видані питання. Звісно навіть такий варіант безсильний проти списування та пошуку відповідей в Інтернеті, для цього є можливість завантажити в базу даних велику кількість запитань, аби була варіативність та аби ці питання були нетривіальні.

getDemo – функція контролера для отримання завдання на демонстрацію. Завдання на демонстрацію видаються із великого загального списку завдань. Таким чином демонстрація схожа на практику. Крім цього демонстрація різна у всіх користувачів. Це допомагає краще зрозуміти як працює шифр. Вже видане завдання не демонстрацію не може бути видане як завдання на практику будь-кому. Це ще один із основних механізмів захисту системи від нечесного проходження практичних завдань. Окрім цього завдання на демонстрацію записується як видане користувачу і користувач може отримати покрокове рішення та загальне за необхідності. API дозволяє це, так як виконати дії з шифрування чи дешифрування можна лише для завдань виданих для демонстрацій, та зробити запит може лише користувач який отримав це завдання для демонстрації. Таким чином, хоч API і надає інтерфейс для

шифрування та розшифрування текстів, виконати ці запити можна лише щодо демонстраційних завдань. В залежності від шифру запит на сервер повертає об’єкт, що включає відкритий текст, алфавіт, ключі та інші вхідні параметри. Для демонстрації шифрування та дешифрування відбувається на одному відкритому тексті, про для асинхронних шифрів можуть бути задані різні закриті ключі.

getPractice – функція контролера, яка видає користувачу завдання для його практичної роботи. Так само як і при отриманні завдання для демонстрації, для практичної роботи завдання береться із загального списку завдань та не може бути видане чи виконане іншим користувачем системи. Функція працює із сервісами CaezarService, OpenTextService, UserService та іншими в залежності від шифру. Видане завдання закріплюється за користувачем і змінити його не можна. Для практичного завдання видаються 2 тексти для шифрування та дешифрування. Обидва тексти спочатку відкриті і на сервері один із них шифрується і користувач отримує відкритий текст і параметри для його шифрування, та криптограму для розшифрування. Всі параметри завдання зберігаються в базі даних користувача.

submitPractice – обробник пост запитів для приймання результатів виконання практичного завдання користувачем. При перевірці результату користувач має прислати завдання та відповідь і завдання будуть порівняні з виданими. Таким чином користувач не може прислати чуже виконане завдання чи завдання з демонстрації.

getCryptoanalisys – функція контролера для відправки користувачу завдання для виконання криптоаналізу перехопленого тексту. Відкритий текст береться із openTextService та шифрується параметрами. Потім всі дані записуються в базу даних користувача, а віддається на клієнт лише перехоплена криптограма.

submitCryptoanalisys – функція контролера для отримання від клієнта виконаного завдання по криптоаналізу. Оскільки криптоаналіз не може бути гарантованим, то перевірка виконується шляхом порівняння присланого результату, і попередньо збереженого в базі даних.

7.2.3 Рівень сервісів

Сервіси в багаторівневому додатку містять в собі велику частину бізнес логіки. Якщо контролери взаємодіють із сервісами, то сервіси можуть працювати із іншими сервісами, репозиторіями, іншими системи, робити запити на інші сервери та інші дії. Сервіси отримують сирі дані та після виконання своєї логіки віддають практично готові дані для відправки клієнту. В розроблюваній системі логіка сервісів в основному заключається в роботі із репозиторіями даних, авторизації та роботі із Firestore. Всі функції сервісів є або синхронними, або асинхронними, що повертають об’єкт Promise<T>.

Сервіс авторизації AuthService виконує функції роботи із зовнішнім сервісом авторизації та управлінням токенами користувачів. Зовнішня авторизація виконується за допомогою HTTPS POST запиту на сервер авторизації. Функція повертає проміс із об’єктом інтерфейсу ExternalAuth, де є поле з токеном доступу до серверу авторизації. Завдяки токену авторизації зовнішнього сервісу сервіс авторизації надає функцію отримання даних про користувача за допомого запиту на сервер авторизації. Ця функція повертає проміс ExternalUser, де міститься інформація про користувача. Ця інформація може бути використана для формування об’єкту навантаження необхідного для створення токену авторизації на сервері розроблюваній системі. Незважаючи на зручність постійної перевірки токена користувача на стороні зовнішнього серверу авторизації, це дуже важкий процес в плані ресурсів, оскільки потрібно робити мережевий запит верифікації. Таким чином локальна авторизація, на основі зовнішньої авторизації значно пришвидшує процес верифікації користувача, а також розвантажує обидва сервери. Більше того, дані про користувача всерівно потрібно зберігати в базі даних розроблюваної системи для збереження додаткових даних.

JWT – відкритий стандарт для створення токенів доступу[21]. Він дозволяє легко ідентифікувати користувачів на сервері. Зазвичай при запиті до ресурсів сервера токен прикріплюють в HTTPS заголовок, або в кукі. Має перевагу над кукі-сесіями, оскільки можна авторизуватись на різних доменах. Також така авторизація не зберігається на сервері, а, отже є stateless. В розроблюваній системі використовується пакет jsonwebtoken для роботи із токенами, який надає інтерфейс

взаємодії для генерації, декодування, перевірки та ін. Всі ці функції наявні в сервісі авторизації.

Сервіс для управління користувачами займається створенням, редагуванням, видаленням та отримання даних про користувачів. Для цього сервіс використовує репозиторій.

Для кожного розділу системи є свій сервіс для бізнес-логіки. В них зосереджена не лише робота із репозиторіями, а й з пакетом cryptographyjs.

7.2.4 Cryptographyjs

Cryptographyjs – розроблений npm пакет для роботи із шифрами. Він включає в себе функції із наданна алфавіту, генерації ключів, шифрування, дешифрування та цифрового підпису. Коже шифр реалізує інтерфейс ICypher. Діаграма класів наведена

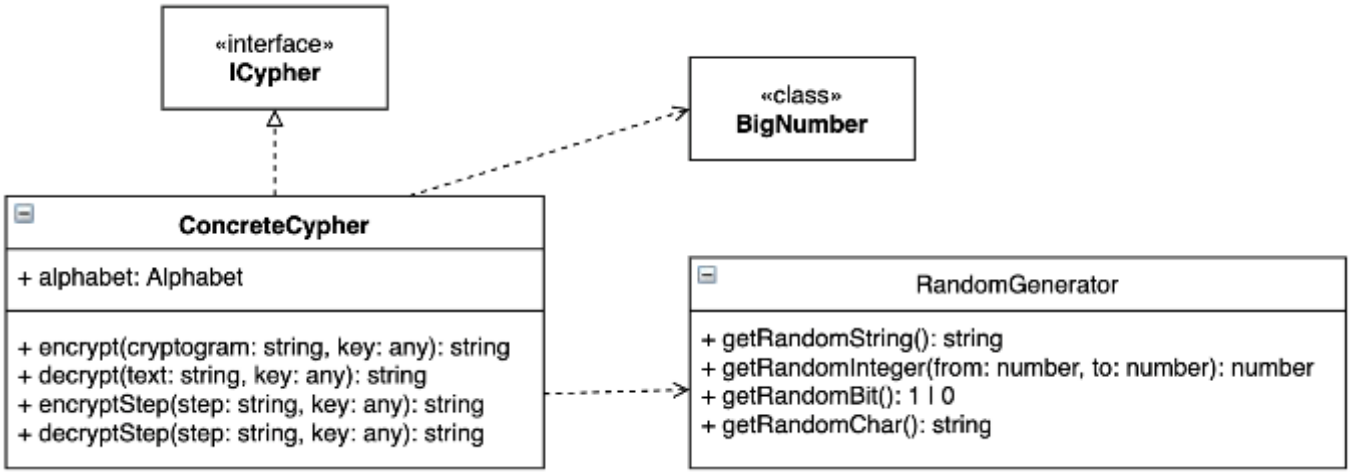

Рисунок 7.4 – Діаграма класів пакету Cryptographyjs

на рисунку 7.4.

Оскільки мова програмування JavaScript не підтримує із «коробки» операції із великим числами, пакет використовує клас BigNumber, який дозволяє працювати із числами будь-якого розміру. Також для деяких шифрів можеть бути необхідні випадкові числа чи символи, для цього розроблено клас RandomGenerator. Також кожен шифр містить алфавіт, з яким він працює. Алфавіт це набір символів і він може

бути будь-яким, проте пакет вже містить константи із латинським алфавітом на українським разом із додатковими символами. Ключі мають тип any, оскільки для деяких шифрів може бути більше одного ключа та вони можуть бути як числами так і строками, і будь-чим.

7.2.5 Рівень репозиторіїв

Репозиторій – це патерн який дозволяє інкапсулювати доступ до збереження даних. Таким чином можна за допомогою репозиторіїв контролювати з яким сховищем даних працювати те легко його замінювати при розробці та тестуванні. Зазвичай для реалізації цього патерну в системі використовують інтерфейс репозиторію, інтерфейс конкретного репозиторію та їх реалізації. Таким чином для

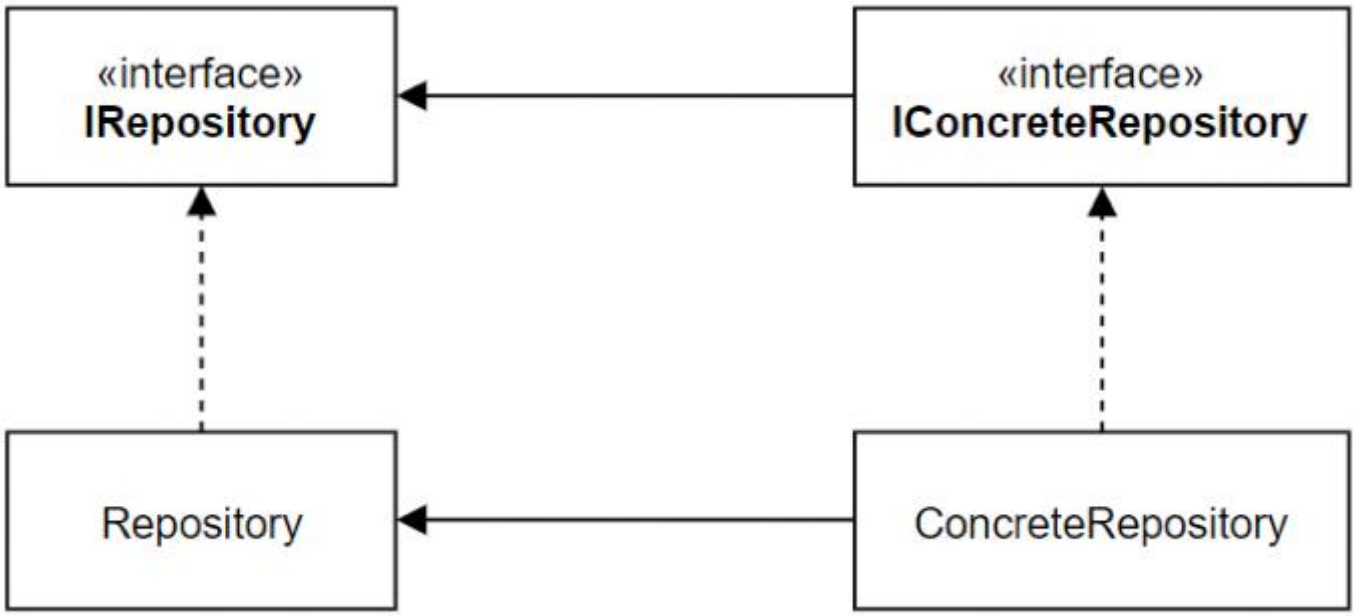

Рисунок 7.5 – Типова реалізація патерну репозиторій

кожної сутності є своя реалізація репозиторію (рисунок 7.5).

Проте в розроблюваній системі, завдяки використанню механізмів Dependency Injection[22], патерну фабричний метод[23] та гнучкості мови програмування TypeScript та бази даних MongoDB вдалося реалізувати фабрику репозиторіїв. Таким чином для кожного репозиторію створюється власна реалізація в

момент ін’єкції в сервіс. Це дає переваги в тому, що зменшує кількість коду, робить його простішим та скорочує роботу програмістів. Репозиторій має основні методи для роботи з даними, щоб отримати дані, змінити, записати чи видалити. Кожен метод репозиторію абстрагується від бази даних і надає зручний та уніфікований інтерфейс для роботи. Для цього кожна функція репозиторію асинхронна і провертає проміс із даними, хоча база даних повертає дані у своїй обгортці, а даному випадку результати, які реалізують групу інтерфейсів MongoDB.

7.2.6 Рівень контексту бази даних

Для реалізації роботи із базою даних було розроблено клас MongoDbContext. Головна задачу цього рівня організація доступу до бази даних. Основною роботою є з’єданння з базою даних. Для цього рівень використовує нативний драйвер для платформи Nodejs mongodb. За допомогою цієї бібліотеки можна приєднатись до бази даних та використовувати методи роботи з базою. Для управління з’єднанням з базою було реалізовано паттерн singleton[24], що дозволяє повторно використовувати одне з’єднання і не створювати нові на кожен запит до бази даних.

7.2.7 Ін’єкція залежностей (DI)

Для зручного використання серівсів, репозиторіїв і інших частин системи в розроблюваній системі було додано контейнер інверсії управління (IOC). Впровадження залежностей це один із патернів проектування програмного забезпечення, що надає зовнішні залежності за допомогою інверсії управління для розв’язання залежностей. В результаті цього код проекту отримує наступні переваги:

- легке модульне тестування з використанням ізоляції макетів компонентів;
- робота не з конкретними реалізаціями, а з наданим інтерфейсом, оскільки для ін’єкції непотрібно знати залежності;
- можливість паралельної і незалежної розробки;
- низька зв’язність між класами і залежностями.

Для реалізації ін’єкції залежностей в системі було використано бібліотеку inversify. Вона надає зручне апі для керування на впровадження залежностями. За допомогою бібліотеки було налаштовано залежності в контейнері. Для реалізації репозиторіїв було розроблено фабрику залежностей, яка віддає налаштовану залежність по параметру запиту. Цим параметром служить токен колекції бази даних. Також за допомогою контейнеру було налаштовано асинхронне підключення до бази даних при завантаженні системи, таким чином швидше за все база даних буде готова до роботи ще до першого запиту до неї. Діаграма пакетів серверної частини наведена у додатку И.

В даному розділі було розлянуто бізнес логіки системи. Оскільки система складається із клієнтської та серверної частин, то бізнес логіку присутня на обох. Загальна архітектура системи побудована із використанням рівнів. Всі рівні виконують свій функціонал і пов’язан лише із сусідніми.

Бізнес логіка клієнтської частини включає в себе роботу із користувачем системи. За допомогою фреймворку Angular було реалізовано односторінковий додаток із маршрутизацією, формами та сервісами взаємодії із сервером та базою даних реального часу. Архітектуру для розробки було обрано запропоновану офіційною документацією фреймворку. Всі компоненти системі розбиті на незалежні модулі. Присутня ін’єнкція залежностей для сервісів. Для роботи із зовнішньою базою даних було підключено спеціальну бібліотеку.

Серверна частина побудована за допомогою фреймворку Express. Логіка додатку зосереджена в сервісах. Контролери на проміжні обробники відповідають за обробку запитів на API. За допомогою репозиторіїв реалізований зручний доступ до даних. А зв’язком з базою даних займається контекст. Всі ключові елементи можна підключати за допомогою ін’єктора залежностей, що дозволило зробити систему гнучкою. Для авторизації користувачів було використано зовнішню систему, зв’язок з якою виконується за допомогою відкритого API. Для логіки шифрування, дешифрування та цифрового підпису було розроблено пакет cryptographyjs, який може працювати з великими числами та інкапсулює логіку роботи шифрів.

8 РОЗРОБЛЕННЯ ІНТЕРФЕЙСУ КОРИСТУВАЧА

Оскільки розроблюваний додаток має клієнтську частину було розроблено дизайн WEB-сайту. При відвідуванні сайту користувач в першу чергу звертає увагу на його графічне оформлення і зручність навігації. В більшості випадків, від того яке враження зробить сайт залежить чи користувач залишиться на сайті чи віддасть перевагу іншим ресурсам.

Дизайн сайту це не просто його зовнішнє оформлення, він також вирішує такі важливі завдання як функціональність і зручність, що часто називають UX[25]. Користувач повинен мати можливість легко орієнтуватись на сторінках сайту, знаходити необхідну інформацію, повертатись на головну сторінку.

Для розробки дизайну системи було визначено цільову аудиторію користувачів, сучасні тенденції WEB дизайну та вимоги до системи.

Існує 3 популярні стилі розробки дизайнів:

- скевоморфізм;
- плоский дизайн;
- матеріальний дизайн.

Скевоморфізм – це стиль дизайну, який запозичує певні функції із минулого, навіть якщо в цьому немає необхідності. Наприклад, при фотографуванні зі смартфону можна почути звук затвору камери, і цей звук йде із динаміків, а не реального затвору. В цьому дизайні графічні елементи нагадують елменти із реального життя, аби користувачу було зручно переходити від реальних предметів до віртуальних. Стів Джобс вважав, що комп’ютери і технології повинні бути настільки простими і зрозумілими, щоб ними міг оволодіти навіть новачок, поклавшись на свої інстинкти[26]. Таким чином даний дизайн більше підходить для старшої аудиторії, або для людей, які тільки знайомляться із комп’ютерними технологіями. Приклад дизайну наведено на рисунку 8.1.

Плоский дизайн – пропонує мінімалістичний підхід до дизайну інтерфейсів. Ставка зроблена на комфорт використання і підхід до зображення даних, а не

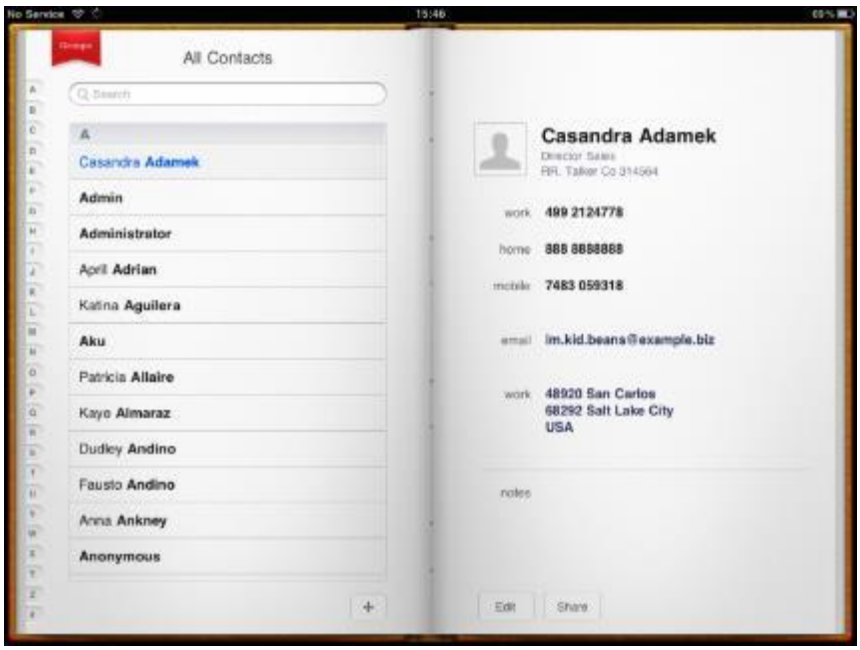

Рисунок 8.1 – Скевоморфізм

допоміжних елементів. Також такий дизайн зменшує навантаження на ресурси клієнту, тому що не потрібно зображувати складні анімації, тіні, переходи і тд.

Матеріальний дизайн – є наслідком протистояння скевормфізму та плоского дизайну. Підхід від компанії Google включає легкі натяки мінімалізму у поєднанні із чистотою і простотою плоского дизайну. Поверхні і краї елементів створюють візуальні образи, які передають підказки і допомагають інтуїтивно орієнтуватись. Material Design оснований на наступних принципах:

- тактильні поверхні. В Material Design інтерфейс складається із шарів так званого «цифрового паперу». Ці шари розташовані на різній висоті і відкидають тіні один на одного, що дозволяє користувачам краще розуміти анатомію інтерфейсу і принцип взаємодії з ним;

- поліграфічний дизайн. Якщо уявити шари шматками «цифрового паперу», то в тому, що стосується «цифрових чорнил» використовується підхід із традиційного графічного дизайну, наприклад, журнального та плакатного;

- осмислена анімація. У реальному світі предмети не з’являються нізвідки і не зникають в нікуди. Тому в Material Design за допомогою анімацій даються підказки користувачам у роботі із системою;

- адаптивний дизайн. Material Design адаптується під різні пристрої.

Приклад дизайну наведено на рисунку 8.2.

Для розробки системи було вибрано Material Design, як такий, що найкраще

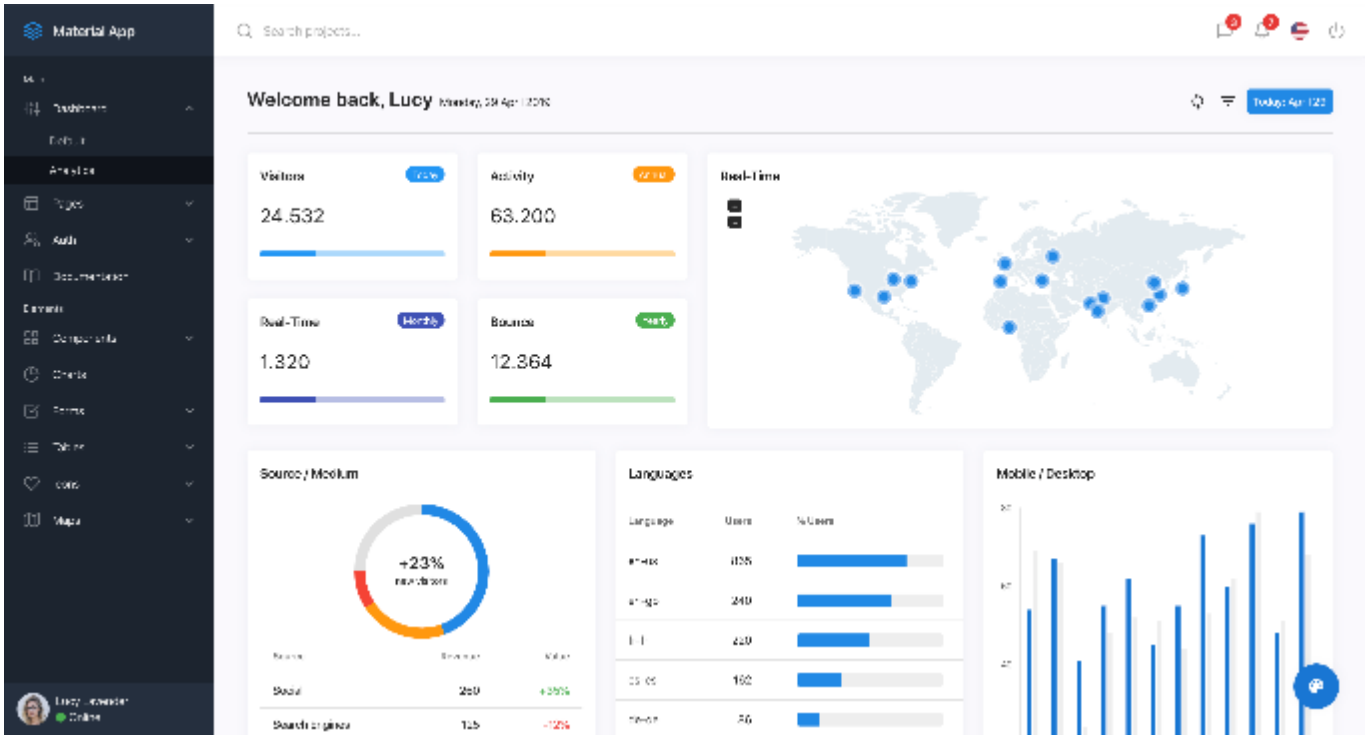

Рисунок 8.2 – Приклад Material Design[27]

задовольняє вимоги, є сучасним та найбільш сумісним із обраними технологіями.

8.1 Проектування компонентів

Оскільки неавторизованим користувачам немає доступу до системи, то перша сторінка, яку бачить гість системи це сторінка авторизації (рис. 8.3). Посередині сторінки авторизації знаходиться картка із формою авторизації. Фон сторінки виконаний у вигляді градієнту від світлосинього до світлозеленого кольору по діагоналі. Таким чином користувач повністю зосереджується на формі. У верхній частині форми знаходиться логотип додатку та назва. Завдяки цьому користувач знає куди він хоче залогінитись. Форма виконана у вигляді двох текстових полів і кнопки «ВХІД». Для авторизації користувачі потрібно ввести логін та пароль. При натиску на поле, воно зреагує та виділиться, аби користувач розумів що поле готове до вводу. Для найменування полів використано placeholder. При натисненні на кнопку «ВХІД»,

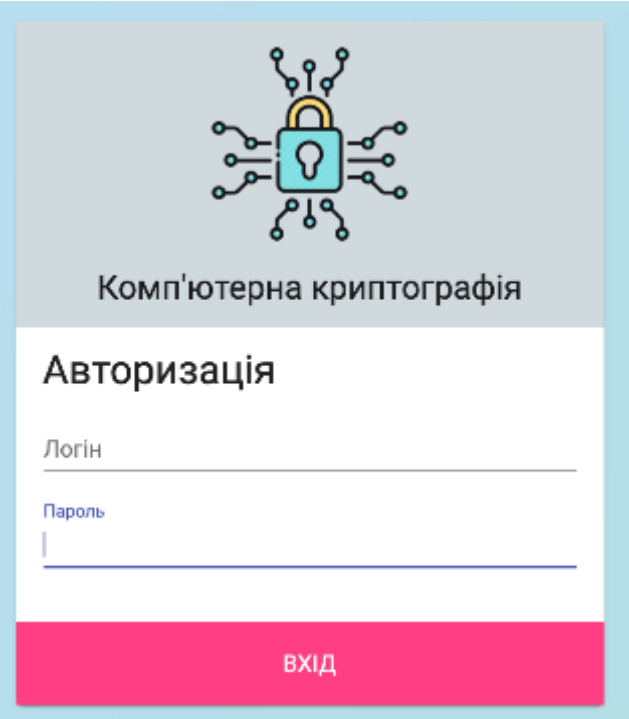

Рисунок 8.3 – Форма авторизації

оскільки процес авторизації може зайняти певний час в залежності від швидкості Інтернету та серверів, користувачу буде показано анімацію завантаження у вигляді неповного кільця, що обертається на місці кнопки «ВХІД». Таким чином система демонструє користувачу, що вона не «зависла», та попереджує повторні кліки на кнопку, поки не повернувся результат запиту.

Після авторизації користувач потрапляє на сторінку із списком розділів (рис. 8.4). На цій сторінці можна виділити головну панель зверху сторінки та плитки із розділами нижче.

Головна панель містить в собі кілька ключових елементів. На лівій частині це головне меню сайту, де можна перейти на головну сторінку, сторінку із розділами, про програму та довідку. На правій частині є кнопка із ім’ям користувача. Це випадаюче меню, де знаходиться кнопку «Вихід». Також такий підхід дозволяє додади функціонал в меню пізніше.

Перелік розділів має вигляд карток. На кожній картці є картинка, по якій легко зрозуміти суть розділу і це дозволяє швидше знайти необхідний розділ. Також під заголовком картки, тобто назві розділу, знаходиться лінія, що позначає поточний прогрес у проходженні розділу. Таким чином користувач може оцінити свій прогрес

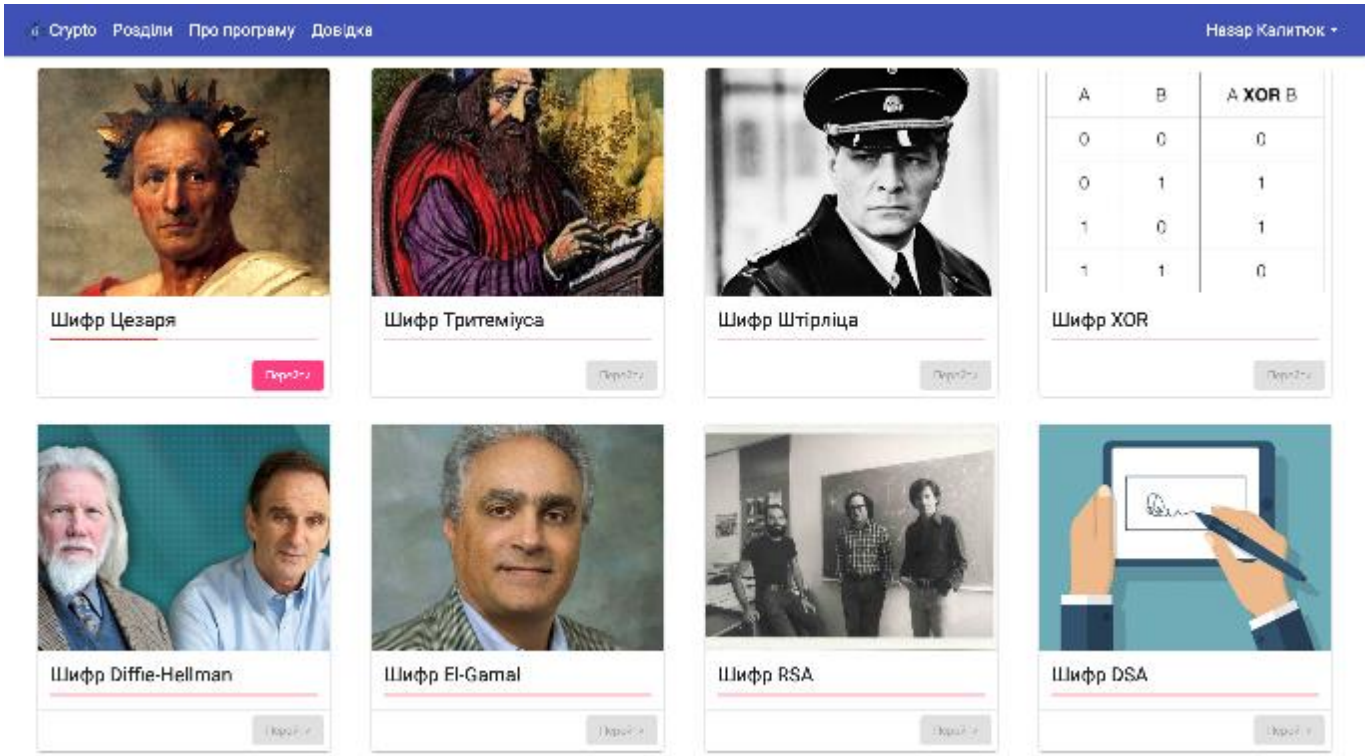

Рисунок 8.4 – Сторінка із списком розділів

по кожному розділу на сторінці із їх переліком. Також при повному проходженні лінія стає зеленою, що повідомляє про завершення розділу. Проте виконувати тести і завдання розділу можна і після його проходження. При зменшенні ширини дисплея кількість стовпчиків карток зменшується, що дозволяє сторінці гарно виглядати на будь-яких дисплеях. Аби перейти всередину розділу потрбіно натиснути кнопку «Перейти», проте для швидкості та інтуїтивності, для переходу можна натиснути на картинку. Це є загальноприйнятим правилом у дизайні таких елементів.

Після переходу всередину розділу користувача перенаправить на сторінку теорії (рис. 8.5). Сторінки розділу складаються із 3 елементів:

- головна панель;
- бокове меню навігації;
- контент.

Головна панель та сама, що й на всіх сторінка веб-сайту, окрім сторінки авторизації. Бокова панель слугує для навігації всередині розділу. Вгорі знаходиться назва шифру, а далі список із доступних сторінок. Поточна сторінка підсвічена

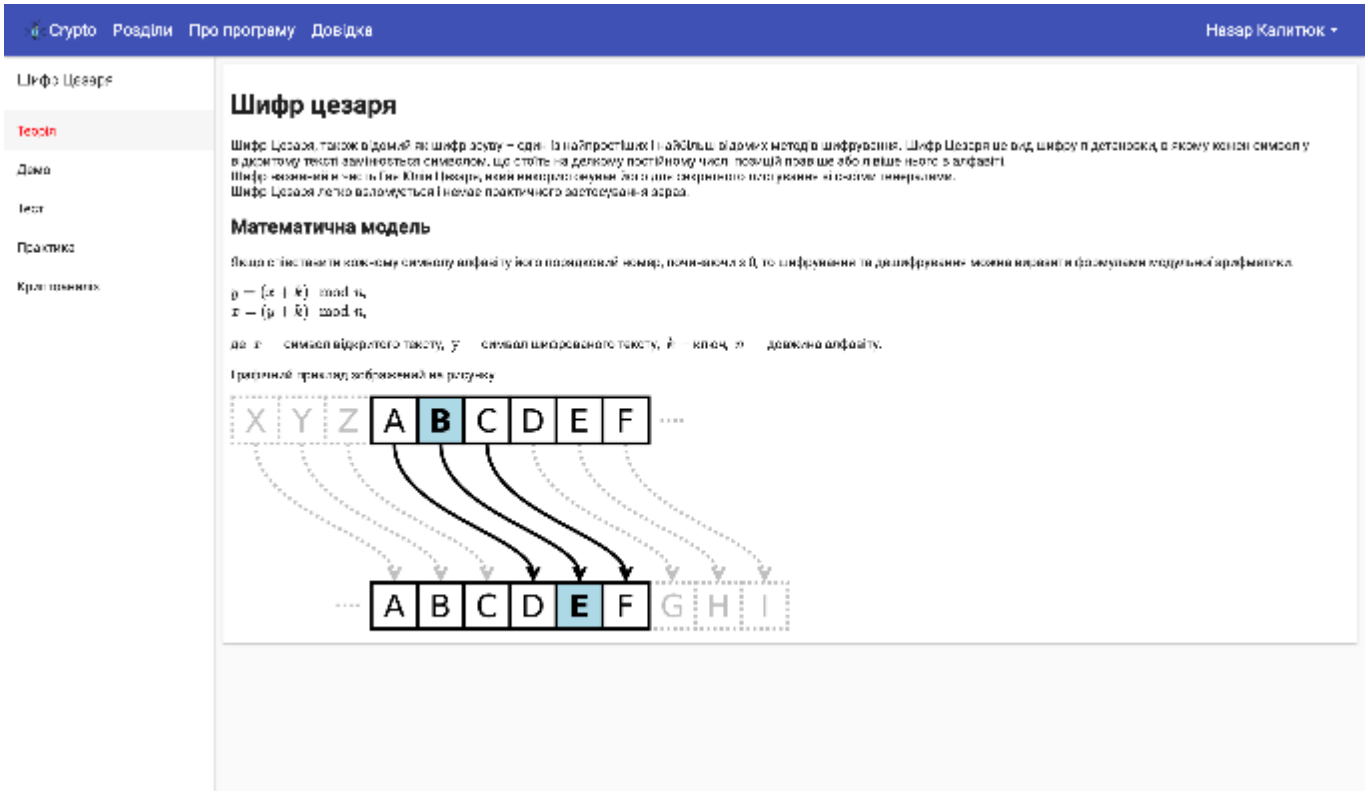

Рисунок 8.5 – Приклад сторінки теорії

червоним шрифтом та сірим фоном. При наведенні пункти підсвічуються сірим фоном. На маленьких екранах бокова навігація може бути прихована, аби на екрані вміщалося більше корисного контенту. Контент розміщується справа від навігації та займає основну частину екрану. Контент залежить від обраної сторінки. Для відображення теоретичних відомостей використовується формат markdown[28], який завантажується із зовнішнього файлового сховища. Таким чином теоретичні відомості можуть бути змінені без зміни вихідного коду додатку, і з іншого боку можуть бути написані в зрозумілому форматі, а не на мові розмітки HTML.

Сторінка тесту, складається із питань та варіантів відповіді. Варіанти вибираються за допомогою радіо-кнопки, що довзволяє явно вказати, що правильний варіант відповіді лише один. Також кнопка «Завершити тест» буде неактивною поки користувач не дасть відповідь на всі питання. Сторінка дуже проста і дозволяє ефективно зосередитись на тестуванні.

Сторінка криптоаналізу доступна на всіх шифрах. Це ще одна максимально проста сторінка, оскільки містить лише інформацію про криптоаналіз та посилання

на можливо корисну теорію, завдання та форми відповіді. Так як криптоаналіз це варіативний процес, то неможливо зробити його покроковим у веб-сайті.

Сторінками як відрізняються у різних розділах є сторінки «Демо» та «Практика». Оскільки кожен шифр різний, то і алгоритми різні, проте в цілому прос демонстрації розділений на певні етапи:

- вхідні дані;
- шифрування/дешифрування;
- результат.

Дизайн цих блоків було розроблено у вигляді акордеону. Таким чином блоки можна розкривати та закривати поступово, або й усі разом. Це дозволяє

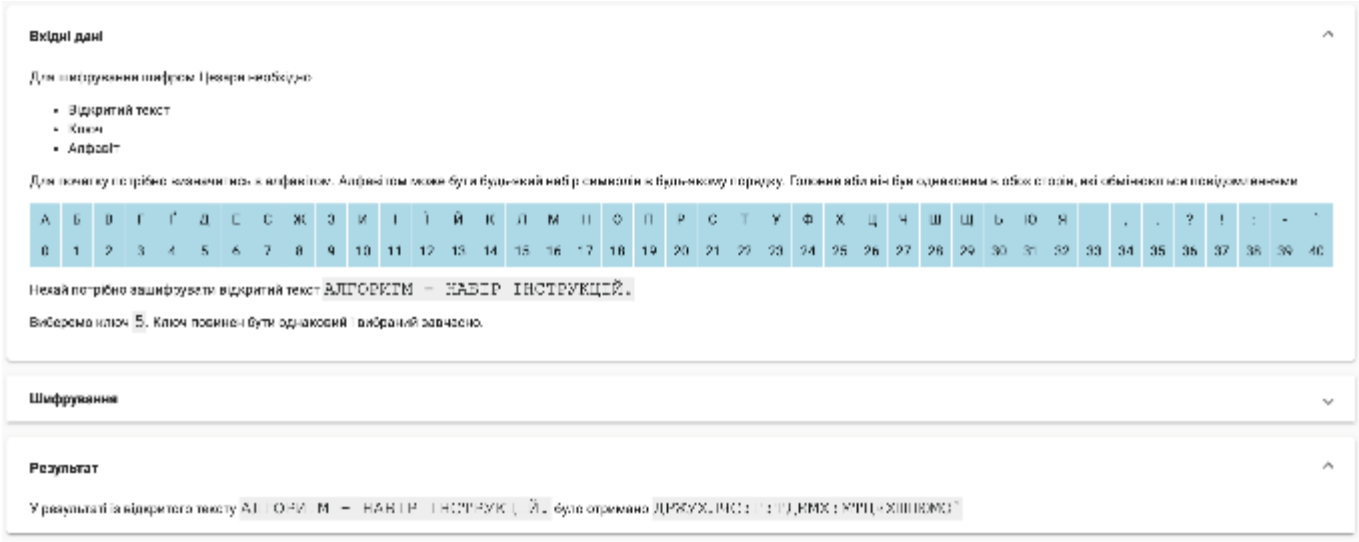

Рисунок 8.6 – Дизайн блоків для демонстрації

зосередитись на процесі та заховати опрацьовані непотрібні дані. Також дозволяє не скролити багато для переміщення між необхідними даними (рис. 8.6).

Блоків може бути більше за необхідності, якщо шифрування потребує більшої підготовки, або складається з великої кількості кроків.

Дизайн інтерфейсу користувача було розроблено з урахуванням всіх вимог до системи, сучасних тенденцій дизайну та з огляду на цільову аудиторію системи. Основна увага була приділена мінімалістичності дизайну та простоті донесення

інформації, що є ключовим пунктом в навчальній системі. Завдяки використанню бібліотеки @angular/material дизайн відповідає всим стандартам Material Design. Таким чином дизайн виглядає сучасно, інтуїтивно та зручно. Також сторінки адаптивні для різних дисплеїв та підтримують жести.

9 СТАРТАП-ПРОЕКТ

9.1 Опис ідеї стартап-проекту

Для опису ідеї стартап-проекту було проаналізовано наступні пункти:

- зміст ідеї, що пропонується;
- можливі напрямки застосування;

Таблиця 9.1 – Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача

- основні вигоди, що може отримати користувач товару;
- им відрізняється від існуючих аналогів та замінників.

Перші 3 пункти подані у таблиці 9.1 і дають цілісне уявлення про можливі потенційні ринки і групи для пошуку клієнтів.

До основних техніко-економічних переваг ідеї можна віднести:

- кросплатформенність;
- збереження даних про навчання;
- авторизацію;
- інтерактивність;
- тестування;
- українську локалізацію;

- теоретичні відомості;

- доступ до результатів викладачем.

Серед конкурентів присутній лише набір окремих програм для Windows. Ці програми за своїми загальними характеристиками схожі, тому для порівння були обрані TRITEM.EXE та EL-GAMAL.EXE. Для формування оцінки конкурентоспроможності розроблюваної системи були визначені сильні слабкі та нейтральні сторони відносно конкурентів та їх аналіз, що зображено у таблиці 9.2.

Таблиця 9.2 – Визначення сильних, слабких та нейтральних характеристик

		(потенційні) товари/концепції конкурентів					
		CryptoCourse	TRITEM.EXE	EL-GAMAL.EXE			
1	Кросплатформенність	Має	Немає	Немає	-	-	+
2	Збереження даних про навчання	Має	Немає	Немає	-	-	+
3	Авторизація	Має	Немає	Немає	-	-	+
4	Інтерактивність	Має	Немає	Немає	-	+	-
5	Тестування	Має	Має	Має	+	-	-
6	Українська локалізація	Має	Немає	Немає	+	-	-
7	Теоретичні відомості	Має	Немає	Має	+	-	-
8	Доступ до результатів викладачем	Має	Немає	Немає	-	+	-

9.2 Технічний аудит ідеї проекту

Для технічного аудиту ідеї проекту було проаналізовано технології реалізації та їх наявність та доступність (таблиця 9.3).

Висновок: згідно з технічним аудитом, необхідні технології вже доступні для використання, отже проект може бути реалізований з використанням обраних технологій.

Таблиця 9.3 – Технічний аудит ідеї проекту

Ідея проекту	Технологія реалізації	Наявність технології	Доступність технології
Авторизація користувача	JWT	Наявна	Доступна
Збереження даних	MongoDB	Наявна	Доступна
Обмін даними в реальному часі	Firebase	Наявна	Доступна
Обчислення великих чисел	BigNumber.js	Наявна	Доступна
Відкрите API	Expressjs	Наявна	Доступна
Клієнтський застосунок	Angular, Angular material, rxjs, html, css, js	Наявна	Доступна
Серверний застосунок	Nodejs, expressjs	Наявна	Доступна
Обрані технології реалізації ідеї проекту: JWT, MongoDB, Firebase, Bignumber.js, Expressjs, Angular, rxjs, html, css, js, nodejs			

9.3 Аналіз ринкових можливостей запуску стартап-проекту

Для рикового впровадження проекту було визначено ринкові можливості проекту на ринкові загрози проекту, які можуть перешкодити реалізації проекту. Було сплановано напрямки розвитку проекту із урахуванням розвитку ринку, потреб клієнтів та пропозицій конкурентів. Аналіз попиту зображений у таблиці 9.4.

Таблиця 9.4 – Попередня характеристика потенційного ринку СП

№	Показники ринку (найменування)	Характеристика
1	Кількість головних гравців	4 (розробник, ВНЗ, Міністерство освіти, школи)
2	Загальний обсяг продаж, грн/ум.од	Планується продаж у 10 вищих навчальних закладах у наступному році на суму близько 500000 гривень

Продовження таблиці 9.4

3	Динаміка ринку (якісна ціна)	Зростає
4	Наявність обмежень для входу	Дотримання процедури введення додатку у навчальну програму закладу
5	Специфічні вимоги до стандартизації та сертифікації	Відповідність до стандартів впровадження проекту в курс навчання
6	Середня норма рентабельності в галузі (або по ринку), %	200%

Висновок: За результатами проведено дослідження отримано висновок, що ринок є привабливим для входження, а рентабельність є більшою за середній відсоток за вкладеннями приблизно в 10 разів, отже є сенс вкластись у стартап.

У таблиці 9.5 було визначено потенційні групи клієнтів, їх характеристики, та сформований основний перелік вимог.

Таблиця 9.5 – Характеристика потенційних клієнтів СП

Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару

Таблиця 9.5 – Характеристика потенційних клієнтів СП

Якісний програмний продукт для швидкого та глибокого вивчення криптографії комп’ютерних систем	Вищі навчальні заклади, Міністерство освіти, школи	Переваги над несучасними аналогами. Уніфікація додатку для всіх тем	Зручність у використанні. Висока якість товару.

У таблиці 9.6 описані фактори загроз стартап-проекту.

Таблиця 9.6 – Фактори загроз СП

№	Фактор	Зміст загрози	Можлива реакція компанії
1	Небажання змінювати підхід до навчання	Неможливість впровадження системи у навчальному закладі	Зменшення ціни, пошука інших навчальних закладів
2	Зміна програми навчання курсу	Окремі теми стають непотрібними, а деякихв не вистачає	Адаптація до нової програми навчання
3	Поява нового конкурента на ринку	Можливе зменшення долі стартапу на ринку	Аналіз конкурента, зменшення ціни, збільшення можливостей стартапу

Таблиця 9.6 – Фактори загроз СП

4	Зміни політики надання послуг щодо серверів баз даних та хостингу	Зміна ціни чи політики послуг для хмарних систем може створити загрозу для коректної роботи системи	Створення локальних серверів для розташування необхідних хмарних сервісів, оскільки всі технології, крім Firestore є безкоштовними
5	Зникнення курсу із навчальної програми	Повне зникнення курсу із комп’ютерної криптографії.	Зміна підходу до розповсюдження системи. Створення факультативної підписки для індивідуальних користувачів

У свою чергу в таблиці 10.7 наведено фактори можливостей стартап-проекту.

Таблиця 9.7 – Фактори можливостей СП

№	Фактор	Зміст можливості	Можлива реакція компанії
1	Зацікавленість у продукті іноземними навчальними закладами	Продаж продукту закордон	Додавання в систему необхідних модулів та локалізації для замовників
2	Зацікавленість у продукті окремих компаній для внутрішніх потреб	Продаж продукту компаніям	Створення корпоративних стандартів навчання

Таблиця 9.7 – Фактори можливостей СП

3	Відкриття продукту для індивідуальних користувачів	Можливість виходу на ринок платформ для самоосвіти	Створення відкритої реєстрації та самонавчання
4	Розширення системи додатковими курсами	Додавання нових курсів збільшить аудиторію системи, що дасть додаткові прибутки	Додавання нових курсів
5	Використання модулів системи в інших системах	Продаж окремих модулів для використання іншими розробниками	Розмежовування модулів системи на окремі компоненти

Для проекту були встановлені основні загрози та можливості. Загрози в основному пов’язані із виходом на ринок нових конкурентів та зміною вартості підтримки проекту. Проте всі ці загрози малоймовірні і потреують великого часу на реалізацію, отже проект встигне адаптуватись під них. Можливості для розвитку зосередженні навколо цілі росту проекту в своєму ринку та виходу на нові ринки, що може дати значний приріст у кількості клієнтів.

Загальні риси конкуренції на ринку зображено в таблиці 9.8.

Після проведення ступеневого аналізу конкуренції на ринку було визначено, що пропозиція бізнес продукту є актуальною, оскільки товар має сильні переваги на конкурентами в межах націленого ринку.

Після аналізу конкуренції був проведений більш детальний аналіз за М. Портером, що зображено в таблиці 9.9.

Таблиця 9.8 – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив діяльності підприємства
Тип конкуренції: олігополія	Існує збірка несучасних аналогів	Нейтралізація недоліків конкурентів та додавання своїх особливих переваг для збільшення якості продукту та конкурентоспроможності на ринку
За рівнем конкурентної боротьби: національний	Ринок охоплює 1 країну – Україну	Забезпечення продуктом відповідних стандартів та положень для інтеграції в навчальний процес
За галузевою ознакою: внутрішньогалузева	В межах 1 галузі	-
Конкуренція за видами товарів: товарно-видова	Товари 1 виду – програмне забезпечення	-
За характером конкурентних переваг: не цінова	Товар якісніший ніж у конкурентів	Зосередження процесу розробки і пітримки товару на забезпеченні високої якості продукту, зважаючи на загальні вимоги до програмного забезпечення, безпеки та окремі вимоги до навчальних продуктів
За інтенсивністю: не марочна	Товар тільки входить на ринок	Забезпечити легкий старт продукту на ринку для подальшого розвитку в рамках навчальних продуктів

Таблиця 9.9 – Аналіз конкуренції в галузі за М. Портером

	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
	Набір застаріх програм	Конкурентами можуть стати портали для самоосвіти	Орендодавці хмартих сервісів та хостингів, баз даних	Вищі навчальні заклади, Міністерство освіти, школи, викладачі, студенти	Немає, альтернатива це не використання спеціалізованого програмного забезпечення для навчання
Висновки	Конкуренція існує, проте вона незначна, а оскільки, програми закинуті, то вона будуть відсутні будь які дії з боку розробникі в несучасних програм	Конкурентів дуже мало і вони не планують вихід на ринок стартап-проекту, оскільки націлені на іншу цільову аудиторію та підхід	Постачальники великі та незалежні. Враховуючи стрімкий ріст ринку хмарних сховищ та обчислень, очікується зниження тарифів на оренду	Головна умова клієнтів це зручність використання продукту, відповідність стандартам та навчальній програмі, наявність необхідних компонентів для навчання	Прямих товарів замінників не існує

Аналіз за М. Портером показав, що прямі конкуренти не можуть протидіяти виходу ну ринок стартап-проекту, а потенційні конкуренти малоймовірні та їм потрібно багато часу для виходу на ринок. При цьому проект має надійних постачальників послуг, а ціна на послуги буде лише зменшуватись. Товари замінники відсутні, а клієнти будуть задоволені якістю проекту.

Обґрунтування факторів конкурентоспроможності описані в таблиці 9.10

Таблиця 9.10 – Обґрунтування факторів конкурентоспроможності

№	Фактор конкурентоспроможності	Обґрунтування факторів
1	Авторизація на автентифікація	Продукт забезпечує авторизацію та автентифікацію для збереження інформації про користувача, а також для можливості доступу до своїх даних з інших пристроїв
2	Захищеність від взлому	Продукт захищений від взлому та нечесного проходження тестування та практичних завдань, аби можна було об’єктивно оцінити знання студента
3	Зручний інтрефейс	Інтерфейс продукту є зручним для використання, сучасним, зрозумілим. Допомагає у проходженні завдань
4	Кросплатформенність	Продукт доступний на різних платформах, що дає змогу користувачам завжди мати доступ до продукту та не підлаштовуватись під вимоги
5	Якість	Продукт працює швидко та без помилок.

Таблиця 9.10 – Обгрунтування факторів конкурентоспроможності

6	Робота у парах	Продукт дає можливість студентам працювати у парах, що посилює засвоєння навчального матеріалу

Беручи до уваги фактори конкурентоспроможності було отримано висновок, що проект буде мати сильну позицію конкурентоспроможності. На основі визначених факторів було проведено порівняльний аналіз сильних та слабких сторін стартап-проекту, що зображено у таблиці 9.11.

Таблиця 9.11 – Порівняльний аналіз сильних і слабких сторін СП

		Рейтинг товарів-конкурентів порівняно із проектом						
		-3	-2	-1	0	+1	+2	+3
Авторизація та автентифікація	17	TRITEM.EXE, EL-GAMAL.EXE						
Захищеність від взлому	15		TRITEM.EXE	EL-GAMAL.EXE				
Зручний інтерфейс	18		TRITEM.EXE	EL-GAMAL.EXE				
Кросплатформенність	20	TRITEM.EXE, EL-GAMAL.EXE						
Якість	17	TRITEM.EXE						
Робота у парах	15	TRITEM.EXE, EL-GAMAL.EXE						

Порівняльний аналіз сильних та слабких сторін стартап-проекту показав, що проект переважає конкурентів по всіх факторах конкурентоспроможності, а отже може мати успіх на ринку.

Для повного аналізу можливостей впровадження проекту було складено SWOT-аналіз, що зображено в таблиці 9.12.

На основі SWOT-аналізу було оцінено ключові сторони проекту і ринкового середовища. Після цього було проведено аналіз альтернатив ринкового впровадження, що зображено в таблиці 10.13.

Таблиця 9.12 – SWOT-аналіз СП

Сильні сторони: 1. По маркетингу – презентації продукту на тематичних конференціях; 2. По виробництву – тестування нових технологій; 3. По персоналу – невелика команда висококваліфікованих спеціалістів; 4. По дослідженню і розробках – постійне оновлення продукції; 5. По фінансах – основна ума капіталу за рахунок власних коштів	Слабкі сторони: 1. По маркетингу – невідомість на ринку; 2. По виробництву – необхідність розгортання для кожного освітнього закладу; 3. По персоналу – висока заробітня плата розробників; 4. По дослідженню і розробках – необхідність постійних досліджень у зв’язку з розвитком тематики; 5. По фінансах – висока вартість розробки на початкових етапах.
Можливості: 1. Зростання попиту на продукцію; 2. Вихід на нові ринки; 3. Послаблення позиції конкурентів; 4. Використання висококваліфікованого персоналу; 5. Використання новітніх технологій.	Загрози: 1. Нестабільна політична та економічна ситуація в країні; 2. Можливість виходу на ринок сильних конкурентів в майбутньому.

Таблиця 9.13 – Альтернативи ринкового впровадження СП

№	Альтернатива (орієнтовний комплекс заходів) ринквої поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Індивідуальні підписки студентів	90%	3 місяці
2	Вільний доступ з рекламою, та платною частиною	90%	6 місяців

Обрана альтернатива ринкового впровадження №1. Ця альтернатива дозволить досить швидко вийти на ринок, та легко набрати критичну масу користувачів для подальшого розвитку проекту.

9.4 Розроблення ринкової стратегії проекту

Для успішного виходу на ринок потрібно було розобити ринкову стратегію – сукупність заходів для отримання запланованих об’ємів продажів і прибутків. Першим кроком було описано групи цільових споживачів, що наведено в таблиці 9.14.

Таблиця 9.14 – Вибір цільових груп потенційних споживачів

№	Опис профілю цільової групи потенційних споживачів	Готовність споживачів прийняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу в сегмент
1	Вищі навчальні заклади	Висока	Високий	Низька	Висока
2	Середні навчальні заклади	Середня	Високий	Низька	Середня
Які цільові групи обрано: Вищі навчальні заклади					

Серед цільових груп потенційних споживачів були розглянуті вищі навчальні заклади та середні навчальні заклади.

Оскільки вищі навчальні заклади демонструють кращу готовність для отримання проекту, то для початку було обрано орієнтуватись на них, проте забувати про середні навчальні заклади не варто.

Оскільки проект зосереджується на одному сегменті було обрано стратегію концентрованого маркетингу.

Далі було визначеного базову стратегію розвитку, що зображено в таблиці 9.15.

Наступним кроком було визначено базову стратегії конкурентної поведінки, що наведено в таблиці 9.16.

Таблиця 9.15 – Визначення базової стратегії розвитку

№	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентноспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
1	Індивідуальні підписки студентів	Концентрація на потребах одного цільвого сегменту	Надання клієнтам якісного і сучасного програмного продукту	Стратегія спеціалізації

Таблиця 9.16 – Визначення базової стратегії конкурентної поведінки

№	Чи є проект «першопрохідцем» на ринку	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки

Таблиця 9.16 – Визначення базової стратегії конкурентної поведінки

1	Ні, вже існує набір несучасних програм	Проект забере споживачів у конкурентів і далі буде шукати нових	Ні	Стратегія виклику лідера

Із можливих стратегій конкурентної поведінки була обрана стратегія виклику лідера, як найбільш підходяща. Стратегія полягає в активній конкуренції із лідером ринку з метою витіснити його і зайняти його місце, це обґрунтовано тим, що проект має значні переваги над конкурентом що дає впевненість у результаті.

На основі вимог споживачів до постачальника та продукту, а також в залежності від обраної стратегії розвитку та стратегії конкурентної поведінки була розроблена стратегія позиціонування, що полягає у формуванні ринкової позиції (таблиця 9.17).

Таблиця 9.17 – Визначення стратегії позиціонування

№	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного СП	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
1	Зручний інтерфейс, якісні теоретичні матеріали та демонстрація	Стратегія спеціалізації	Стратегія виклику лідера	Відповідність навчальним стандартам, кросплатформенність, авторизація

При визначенні стратегії позиціонування стартап-проекту були розглянуті вимоги цільової аудиторії до товару та попереднього зроблені дослідження, завдяки чому було досягнуто перевагу над конкурентами в межах обраного ринку.

9.5 Розроблення маркетингової програми стартап-проекту

Розроблення маркетингової програми є ключовим етапом підготовки стартап-проекту до виходу на ринок. Першим кроком є формування маркетингової концепції товару, який отримає споживач (таблиця 9.18).

Таблиця 9.18 – Визначення ключових переваг концепції потенційного товару

№	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі, або такі, що потрібно створити)
1	Потреба в програмному забезпеченню для навчання	Зручний, якісний програмний додаток	Кросплатформенність, авторизація, збереження даних, доступність
2	Потреба в контролі викладача за процесом	Спрощення процедури роботи	Формування звіту про роботу студентів для викладача, можливість віддаленої здачі

Ключовими перевагами концепції потенційного товару є те, що новий продукт кращий за наявних конкурентів та може бути легко впроваджений на ринок. (таблиця 9.19).

Таблиця 9.19 – Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові

Таблиця 9.19 – Опис трьох рівнів моделі товару

<table>
<tr><td>1. Послуга за задумом</td><td colspan="3">Опис базової потреби споживача, яку задовольняє товар (згідно концепції), її основні функціональні вигоди</td></tr>
<tr><td></td><td colspan="3">Потреба в якісному програмному забезпеченні для навчання комп’ютерної криптографії</td></tr>
<tr><td rowspan="5"></td><td>Властивості/характеристики</td><td>М/Нм</td><td>Вр/Тх/Тл/Е/Ор</td></tr>
<tr><td>1. Економічні: низька ціна на обслуговування;
2. Надійності: додаток надійний;
3. Технологічні: використання сучасних технологій;
4. Ергономічні: додатком зручно користуватись на різних платформах;
5. Естетичні: всі частини виконані в одному сучасному стилі;
6. Екологічні: використання продукту дозволяє перенести документацію в електронний вигляд;
7. Безпеки: дані про користувача надійно захищені</td><td></td><td></td></tr>
<tr><td colspan="3">Якість: продукт виконаний згідно стандартів навчання</td></tr>
<tr><td colspan="3">Марка: CryptoCourse</td></tr>
<tr><td colspan="3"></td></tr>
<tr><td rowspan="2"></td><td colspan="3">До продажу: продукт представлений клієнтам</td></tr>
<tr><td colspan="3">Після продажу: підтримка продукту</td></tr>
<tr><td colspan="4">За рахунок чого товар буде захищено від копіювання: код продукту закритий від сторонніх осіб, а копіювати користувацький інтерфейс немає сенсу.</td></tr>
</table>

Наступним кроком було визначення цінових меж, якими потрібно керуватись при становленні цін на продукт, що зображено в таблиці 9.20.

Таблиця 9.20 – Визначення меж встановлення ціни

Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар
10000 ум.од.	10000 ум.од.	8000 ум.од.	1000-5000 ум.од.

Аналіз цін на товари, а також доходів цільової групи споживачів показує, що продукт є доцільним для придбання, оскільки коштує девешве, а також пропонує безстрокову ліцензію, на відміну від аналогів, які в більшості працюють за щомісячною або річною підпискою.

Наступним кроком було визначення оптимальної системи збуту, що зображено в таблиці 9.21.

Таблиця 9.21 – Формування системи збуту

Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
Покупка ліцензії на використання	Розгортання проекту на серверах покупця, або надання своїх серверів. Підтримка протягом користування	Глибока	Всебічна робота із усіма рівнями освітніх закладів

У якості оптимальної системи збуту було вирішено розповсюджувати продукт власними силами, показуючи його на спеціалізованих виставках та конференціях.

Надалі покупці будуть самі розповсюджувати інформацію про продукт, що дозволить збільшувати об’єми продажів.

Однією із найважливіших складових маркетингової програми є розроблення маркетингових комунікацій, що наведено в таблиці 9.22.

Таблиця 9.22 – Концепції маркетингових комунікацій

Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
Цільові клієнти пов’язані з комп’ютерними технологіями і навчанням, отже відвідують багато тематичних заходів	Інтернет, зустрічі, виставки, конференції	Буде розповсюджуватись інформація про продукт на обраних майданчиках	Донести інформацію, що продукт більш якісний, підходящий та дешевший за аналоги	Звернення у вигляді виступів із презентаціями та електронними повідомленнями

Було досліджено, що основними каналами комунікацій потенційних клієнтів є Інтернет і тематичні заходи, тому там і буде розміщена реклама про продукт.

9.6 Висновки до розділу

Отже, для даного стартап-проекту є можливість ринкової комерціалізації проекту, оскільки попит на навчальні проекти такого типку високий, а конкуренція

низька. В цілому ринок освіти, а особливо таких напрямків як комп’ютерна освіта, віддалена освіта, самоосвіта стрімко зростає, що говорить про те, що даний стартап-проект після виходу на ринок буде користуватись великим попитом. Рентабельність веб-додатків висока, оскільки відсутні постійні закупки дорогої сировини, а потрібна лише оренда та/або покупка обчислювальної техніки для розгортання додатку. Також незважаючи на дороге виробництво на початкових етапах, в цілому, з часом рентабельність додатку буде стрімко зростати через постійне зменшення ціни на обслуговування та збільшення прибутку.

Перспективи впровадження стартап-проекту високі, оскільки проект займає вузьку нішу, і на най низикй рівень конкуренції, проте потенційним групам клієнтів потрібний якісний продукт. Таким чином проект буде користуватись популярністю серед клієнтів та мати високу конкурентоспроможність.

Серед альтернатив впровадження на ринок варто обрати стратегію продажів продукту окремим користувачам по підписці, це має свої переваги у вигляді швидшого зростання кількості користувачів та розповсюдження проекту на іноземні ринки, що збільшить витрати на підтримку продукту, проте збільшить і доходи за рахунок більших продажів.

Оцінивши всі характеристики даного стартап-проекту, було отримано висновок, що подальша імплементація є доцільною.

ВИСНОВКИ

У ході виконання магістерської дисертації було проведено дослідження предметної області. Проаналізовано вимоги до системи в цілому, окремих функцій та платформи розробки.

Для проведення аналізу існуючих рішень були обрані застаралі додатки для Windows. Досліджено їхній функціонал, виявлені всі переваги та недоліки. Після аналізу були обрані основні функції для реалізації.

На основі дослідження аналогів було сформовано функціональні та нефункціональні вимоги до системи, що є основним етапом підготовки до розробки. При формуванні вимог було взято до уваги можливості сучасних технологій розробки ПЗ та потреби користувачів.

Для розроблюваної системи була побудована діаграма прецедентів, а самі прецеденти описані у вигляді таблиць. При цьому були визначені основні роді користувачів в системі. На основі сформованих вимог були визначені варіанти використання системи.

На основі сформованих функціональних вимог, було досліджено представлені технології для реалізації системи. Оскільки розроблювана система складається із односторінкового додатку на клієнті на API серверу, були порівняні варіанти для розробки клієнтської та серверної частини. Серед популярних аналогів були обрані найбільш поєднувані між собою. Для збереження даних була обрана база даних, яка найкраще інтегрується із сервером та надає необхідну гнучкість для виконання вимог. Також для обміну повідомлень в реальному часу було обрано технологію Firestore, замість ресурсомістких сокетів на сервері, що довзолило розвантажити систему та розподілити.

Опираючись на обрані технології та вимоги до системи було розроблену структурну схему системи. Вона включає основні компоненти системи. Структурна

схема є досить гнучкою і дозволяє замінювати компоненти на аналогічні, що робить саму систему варіативною.

Завдяки використанню документо-орієнтованої бази даних, вдалося спроектувати схему бази дуже вільною. Існуючі документи легко розширити новими даними, оскілки користувач може зберігати в себе статистичні дані, та свою дії. Було описано основні поля та залежності документів.

Бізнес-логіка системи складається із двох частин: клієнтської та серверної. На клієнтській частині описана взаємодія системи із користувачем. Налаштовані маршрути за доступи. На серверній частині міститься логіка для роботи із даними, шифрами. На обох частина є робота із базою даних Firebase для обміну повідомлень. Було розроблено загальну архітектуру системи, що складається з багатьох шарів, які мають свої відповідальності. Завдяки використання паттернів проектування вдалося зменшити кількість коду, його повторюваність, підвищити якість.

Інтерфейс системи було розроблено максимально простим та зручним для користувача. В той же час інтерфейс користувача є сучасним, спроектованим із використанням бібліотеки @angular/material, що дозволило дотримуватись всіх правил та вимог до побудови інтерфейсу. Таким чином інтерфейс системи виглядає свіжо, адаптивний до різних екранів, має підтримку жестів, що дозволяє користуватись системою зручно і усюди.

Було досліджено можливості даного проекту реалізації у вигляді стартап проекту. За результатами дослідження було встановлено, що ринок готовий до нового продукту. Продукт є актуальним для споживачів. Динаміка росту ринку позитивна, що дає надію на швидкий та активний розвиток продукту з часом. За оцінкою рентабельності рентабельність висока.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Полторак В.П. Теорія інформації та кодування / Ю.П. Жураковський, В.П. Полторак // Підручник. – К.: Вища школа, 2001. – 255 с.: іл., укр.мовою (Гриф МОНУ №1/11-2367 від 21.05.2001).

2. Полторак В.П. Криптографічний захист даних в цифрових інформаційних системах (Частина 1) / Полторак В.П. // Телеком. Військовий звязок. Спеціальний випуск, №2/2018. - К.: Softpress. hi-Tech.ua, Жовт., 2018. - с. 98-104., Мова публікації:українська.

3. Мінцифра презентувала екосистему цифрових рішень для закладів середньої освіти [Електронний ресурс] – Режим доступу до ресурсу: https://thedigital.gov.ua/news/mintsifra-prezentuvala-ekosistemu-tsifrovikh-rishen-dlya-zakladiv-serednoi-osviti.

4. Кібербезпека: Україна може стати джерелом рішень світового рівня [Електронний ресурс] – Режим доступу до ресурсу: https://thedigital.gov.ua/news/kiberbezpeka-ukraina-mozhe-staty-dzherelom-innovaciy-svitovogo-rivnya.

5. Полторак В.П. Криптографічний захист даних в цифрових інформаційних системах (Частина 3) / Полторак В.П. // Телеком. Військовий звязок. Спеціальний випуск, №7/2019. - К.: Softpress. hi-Tech.ua, Жовт., 2019. - с. 98-101., Мова публікації:українська.

6. Software requirements [Електронний ресурс] – Режим доступу до ресурсу: https://www.tutorialspoint.com/software_engineering/software_requirements.htm.

7. Iterative and incremental development [Електронний ресурс] – Режим доступу до ресурсу: https://www.techopedia.com/definition/25895/iterative-and-incremental-development.

8. Функциональные и нефункциональные требования (Functional and Non-functional Requirements) [Електронний ресурс] – Режим доступу до ресурсу: https://studfile.net/preview/2152457/page:4/.

9. Single-page application vs. multiple-page application [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/@NeotericEU/single-page-application-vs-multiple-page-application-2591588efe58

10. Leaked internal google dart email [Електронний ресурс] – Режим доступу до ресурсу: https://gist.github.com/paulmillr/1208618.

11. Standard ECMA-408 Dart Programming Language Specification [Електронний ресурс] – Режим доступу до ресурсу: http://www.ecma-international.org/publications/standards/Ecma-408.htm.

12. introduction.md [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/nestjs/docs.nestjs.com/blob/master/content/introduction.md.

13. Koa vs Express [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/koajs/koa/blob/master/docs/koa-vs-express.md.

14. Material articles [Електронний ресурс] – Режим доступу до ресурсу: https://material.io/design/.

15. Node.js MongoDB Driver API [Електронний ресурс] – Режим доступу до ресурсу: http://mongodb.github.io/node-mongodb-native/3.2/api/.

16. JSON and BSON [Електронний ресурс] – Режим доступу до ресурсу: https://www.mongodb.com/json-and-bson.

17. Software Design Principles DRY and KISS [Електронний ресурс] – Режим доступу до ресурсу: https://dzone.com/articles/software-design-principles-dry-and-kiss.

18. Common web application architectures [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/common-web-application-architectures.

19. What is a RESTful API? [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/@lazlojuly/what-is-a-restful-api-fabb8dc2afeb.

20. Postman API Client [Електронний ресурс] – Режим доступу до ресурсу: https://www.getpostman.com/product/api-client.

21. JWT простым языком: что такое JSON токены и зачем они нужны [Електронний ресурс] – Режим доступу до ресурсу: https://proglib.io/p/json-tokens/.

22. A quick intro to Dependency Injection: what it is, and when to use it [Електронний ресурс] – Режим доступу до ресурсу: https://www.freecodecamp.org/news/a-quick-intro-to-dependency-injection-what-it-is-and-when-to-use-it-7578c84fa88f/.

23. Design Patterns | Set 2 (Factory Method) [Електронний ресурс] – Режим доступу до ресурсу: https://www.geeksforgeeks.org/design-patterns-set-2-factory-method/.

24. How to make the perfect Singleton? [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/@kevalpatel2106/how-to-make-the-perfect-singleton-de6b951dfdb0.

25. В чем отличие UI от UX? Подробный разбор часто используемых терминов [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/company/pixli/blog/324794/.

26. Скевоморфизм, или на чем базируется дизайн смартфонов [Електронний ресурс] – Режим доступу до ресурсу: https://www.bbc.com/ukrainian/ukraine_in_russian/2013/06/130613_ru_s_skeuomorphism_apple.

27. MUI Themes [Електронний ресурс] – Режим доступу до ресурсу: https://themes.material-ui.com/previews/material-app/.

28. ngx-markdown [Електронний ресурс] – Режим доступу до ресурсу: https://jfcere.github.io/ngx-markdown/.