НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ
Кафедра автоматизованих систем обробки інформації і управління

УДК: 004

«До захисту допущено»
В.о. завідувача кафедри
_____ *О.А.Павлов*
(підпис) (ініціали, прізвище)
“___” _____2019 р.

Дипломний проект
на здобуття ступеня бакалавра

з напряму підготовки *6.050101 «Комп’ютерні науки»*

на тему: «*Електронний щоденник студента для мобільної платформи*»

Виконав: студент *4* курсу, групи *ІС-51*
Єрьоменко Денис Валентинович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник *доц., к.т.н., доц. Тєлишева Т.О.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з графічної документації *доц., к.т.н., доц. Тєлишева Т.О.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент *доц. каф. ТК, к.т.н., доц. Ткач М.М.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент Єрьоменко Д.В. _____
(підпис)

Київ – 2019 року

АНОТАЦІЯ

Структура та обсяг роботи. Структура та обсяг роботи. Пояснювальна записка дипломного проекту складається з п’яти розділів, містить 7 рисунків, 25 таблиць, 1 додаток, 10 джерел.

Дипломний проект присвячений розробці системи для полегшення управління часом як для студентів, так і для викладачів, реалізованої як мобільний додаток.

Метою дипломного проекту є: спростити комунікацію між викладачем та студентом, допомогти викладачам та студентам з управлінням часу та автоматизувати процес переносу подій у календарі.

Проведено аналіз процесів комунікації між викладачем та студентом та переносу подій.

Були визначені вхідні та вихідні дані, була розроблена структура бази даних, яка реалізована на MongoDB.

Описані основні засоби розробки системи. Засобами розробки є: ReactJS, React Native, NodeJS.

Описана інструкція користувача та проведене тестування системи, яке показало працездатність системи.

РОЗПОРЯДОК ДНЯ, ЩОДЕННИК СТУДЕНТА, МОБІЛЬНИЙ ДОДАТОК, АССОЦІАТИВНІ ПРАВИЛА, АЛГОРИТМ APRIORI

ABSTRACT

The structure and scope of work. Explanatory note the degree project consist of five sections, including 7 figures, 25 tables, 1 appendix, 10 sources.

The degree project is dedicated to the development of informational support system to improve time management for students and teachers, implemented as a mobile application.

The purpose of the diploma project are facilitate communication between the teacher and the student, assist teachers and students in managing time and automate the process of transferring events in the calendar.

The analysis of communication processes between the teacher and the student and the transfer of events has been carried out.

Input and output data were determined, a database structure was developed that was implemented on MongoDB.

The main features of the complex problems of development are described. The means of development are ReactJS, React Native, NodeJS.

The manual guide and conducted testing of complex tasks are described.

ROUTINE, DIARY OF A STUDENT, MOBILE APPLICATION, ASSOCIATIVE RULES, ALGORITHM APRIORS

ЗМІСТ

ВСТУП

Темою дипломного проекту є “Електронний щоденник студента для мобільної платформи”.

Основне призначення – додаток для створення розпорядку дня, його автоматичного редагування та формування.

Суть проекту в створенні мобільного додатку, який буде автоматизувати переніс подій у календарі та взаємодію з ними.

На сьогоднішній день маємо безліч людей, які постійно кудись поспішають. Такі люди наймають секретаря, щоб він дивився за розпорядком, але це ж не вихід. Не кожен може дозволити собі секретаря.

Приведемо приклад, користувач додає подію до свого календаря. На даний момент програмне забезпечення таке, що календар не покаже ніякого оповіщення, він просто поставить в одну годину похід в кіно та гра в футбол. Тому що календар не володіє ніякою інформацією, це дуже велика проблема. Тобто ти не можеш автоматизувати побудову розпорядку як тільки в тебе починають виникати конфлікти між подіями. А якщо у користувача є багато подій, а йому все рівно в який час дня вони будуть зроблені, йому просто важко притягнути їх до точно часу, бо все потрібно спланувати, спочатку це, потім це, а ще треба враховувати особливості. В роботі пропонується додаток, який допомагає у побудові розпорядку дня.

1. ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1 Опис предметного середовища

1.1.1 Загальні положення

Зараз у всьому світі набуває популярності тема time management. Time management – це процес планування на контролю за тим, скільки часу витрачати на конкретні види діяльності. Гарне планування дозволяє людині виконати більше завдань в коротші терміни, знижує кількість стрес та призводить до успіхів в кар'єрі.[2]

Зараз кожна успішна людина у світі вважає важливою здатність ефективно управляти своїм часом. Ось деякі переваги ефективного керування часом:

- Зменшення стресу. Створення та притримування розпорядку дня дає можливість людині відчувати себе менш напруженою.
- Більше часу. Хороше управління часом дає людині можливість знаходити додатковий час на хобі або інші особисті заняття
- Більше можливостей. Управління часом є важливою навичко, яку шукають роботодавці. Можливість розставляти пріоритети та планувати роботу є надзвичайно бажаною для будь якої організації
- Здатність до реалізації цілей. Люди, які належно планують свій час, краще здатні досягати своїх цілей та завдань, роблячи це за найкоротший час

Моя дипломна робота націлена на створення мобільного додатку, який допоможе автоматизувати створення та змінення розпорядку дня та зменшить вплив людського фактору на це до мінімуму. Також дана система покращую рівень комунікації між викладачем та студентом да допомагає студентам краще планувати свій час. Також додаток може покращувати комунікацію студентів між собою. Вони мають можливість створювати події

відпочинку усією групою. Будем розглядати додаток у середі процесу навчання студентів, але він дуже легко імпортується у будь-яку іншу галузь.

1.1.2 Опис процесу діяльності

Опишемо процес діяльності на прикладі навчального процесу. Студенти вступають до університету. Адміністратор додає їх профілі до бази даних додатку, формує групу користувачів. Викладач, курс якого є у цьому році, заходить у додаток і відправляє усі завдання та дедлайни студентам кожної групи. Також встановлює їм пріоритети. Також викладач може швидко оповістити усіх студентів о переносі заняття чи додатковій консультації. Протягом семестру студенти виконують завдання, у своєму додатку вони бачать усі дедлайни та їх переноси, подія просто зсувається у календарі та студент відразу це бачить. При конфлікті подій, система буде переносити подію згідно налаштувань системи у вигляді асоціативних правил, звісно при можливості переносу. Якщо можливості немає, у разі конфлікту з подією іншого викладача система заблокує відправлення, у разі конфлікту з власною подією студента, вона відправить йому оповіщення про критичний конфлікт. У даному випадку студент має можливість або домовитися з викладачем або перенести свої особисті плани. У процесі виконання завдань, студент має можливість відмічати ступінь виконання роботи, щоб викладач знав, на яку кількість студентів на заняті йому розраховувати та міг добре спланувати свій час. При особистому налаштуванні система може виводити списки дедлайнів на наступний день та нагадувати про невиконані завдання.

У разі не задоволеності викладачем роботою студента, він має можливість додати йому додаткові завдання або відправити на перероблення. Усі ці дані швидко відобразяться на стороні студента. Сама система буде зсувати календар з урахуванням цих завдань.

У кінці семестру викладач закриваю курс і всі події по ньому архівуються. У разі потрібності, студент має можливість переглянути графік здачі робіт, переноси та коментарі до завдань.

Бізнес-процеси, які розглядаються у системі показані у вигляді BPMN діаграм, які наведені у частині графічного матеріалу.

Для створення функціональної моделі, що показує структуру та функції додатку, застосовується IDEF0 діаграма, яка наведена у частині графічного матеріалу.

1.1.3 Опис функціональної моделі

Опишемо функціональну модель системи. Визначимо акторів, їх функції та їх процеси. Усім користувачам надається можливість формувати та змінювати власний календар, додавати події, пересувати події, створювати план роботи, відмічати прогрес.

Актори системи:

- Викладач
- Студент

Викладач – це тип користувача, який може додавати події до календарів користувачів по тегу. Для реєстрації такого користувача треба мати пароль системи

Студент – це тип користувача, який може використовувати спільний функціонал.

Взаємодія акторів показана на Use-Case діаграмі, яка наведена у частині графічного матеріалу.

Таблиця 1.1 – Актори системи

Актор	Опис
Студент	Має лише загальні функції, створення подій свого календаря, зміщення подій, використання автоматизованого створення розпорядку та автоматизованого його зміщення. Архівування минулих подій та перегляд заархівованих розпорядків.
Викладач	Є привілейованим користувачем, окрім загальних функцій, може створювати події по тегам, які користувачі наводять у своїх профілях

1.2 Огляд наявних аналогів

Додаток за своїм призначенням є системою планування часу, тому будуть розглянуті аналогічні додатки. Більшість з них вбудовані в операційні системи комп'ютерів, телефонів та інших пристроїв. Також можливо розглянути незалежні додатки та програми на різних платформах. Найбільш популярною та зручною системою є: Google Календар [1], також опишемо звичайний календарі у телефоні, які мають приблизно однаковий функціонал.

Google Календар

Google Календар – це сервіс для планування часу як однієї людини так і груп людей. Ви можете робити тематичні календарі для кожної групи свій.

Основні функції:

a) можна створити календар подій;

b) можна додати людей. Перевага цієї системи в тому, що вона має доступ до контактів системи google, в яку входять як контакти gmail, так і google +;

c) створювати календар можна лише у веб сервісі. Користуватися їм можна і с телефона, але створювати лише у браузері;
d) календарю можна дати назву так прикрасити його на свій вибір.

Даний додаток має схожі функції, але має багато недоліків у порівнянні з моїм застосунком. Наприклад, календар можна створити лише з комп'ютера, що забирає у користувача гнучкість його часу, змушуючи його витрачати час на користування комп'ютером. Додаток не має груп користувачів, отже доведеться додавати усіх по одному. Але найважливіше це автоматизація зсуву розпорядку. Google Календар все одно на можливості людини, він може хоч 10 подій поставити на один і той самий час, тому людині треба власноруч додавати власні події, щоб перенести свої плани, мій же додаток це автоматизує, тому необхідність студенту мати свій щоденок є актуальною задачею.

Вбудований Календар

Вбудований Календар – це додаток, встановлений на всі телефони та комп'ютери автоматично. Система додає до нього автоматично міжнародні свята. Користувач може додавати свої події

Основні функції:

a) можна створити події;
b) можна поставити частоту та час у який нагадувати про подію;
c) уже додані основні свята;
d) додаток може витягувати з вашого облікового запису дні народження ваших друзів, та події з Facebook.

Даний додаток має схожі функції, але має багато недоліків у порівнянні з моїм застосунком. Наприклад, календар має лише особисте застосування, якщо у вас є спільна подія з друзями, вам потрібно особисто кожний до свого календаря додати цю подію. Але найважливіше це автоматизація зсуву розпорядку. Календарю все одно на можливості людини, він може хоч 10

подій поставити на один і той самий час, тому людині треба власноруч додавати власні події, щоб перенести свої плани, мій же додаток це автоматизує. Тому необхідність студенту мати свій щоденник є актуальною задачею

1.3 Постановка задачі

1.3.1 Призначення розробки

Створити додаток, який полегшить управління часом у рамках процесу навчання, як для студентів, так і для викладачів. Розробити зрозумілий та простий інтерфейс.

1.3.2 Мета та задачі розробки

Метою розробки є:

1) спростити комунікацію між викладачем та студентом;
2) допомогти викладачам та студент з управлінням часу;
3) автоматизувати процес переносу подій у календарі.

Для виконання поставленої мети, потрібно вирішити наступні задачі

1) реєстрація користувачів;
2) додавання подій користувачами;
3) формування розпорядку дня на основі заданих подій.

Висновок до розділу

У даному розділі було проведено аналіз та опис предметного середовища, яке несе у собі реалізацію додатку, який спростить взаємодію між студентами та викладачами. Також допоможе користувачам краще планувати свій час, та автоматизує процес переносу справ. Були визначені актори системи та детально описані. Наведено діаграму. Був описаний процес діяльності та функціональна модель. Визначено призначення розробки. Було порівняно дану розробку з аналогами, у результаті якої було з’ясовано, що дана розробка більш гнучка, за рахунок взаємодії на мобільній платформі та має додатковий функціонал у вигляді автоматичного переносу подій за асоціативними правилами.

2 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Перелік первісних даних

Вхідні дані можна поділити на категорії у залежності від цілі створення цих даних, були виділені такі категорії даних

1) дані розпорядку;
2) дані користувачів;
3) налаштування системи;
4) архівовані дані.

Дані розпорядку це дані, які вносять усі користувачі, це їх особисті події, які ще не настали. Наведемо дані, що входять до кожної події:

1) назва;
2) дата;
3) час;
4) можливість переносу;
5) рід занять;
6) тег, по якому додана подія, у випадку додавання викладачем.

Студенти мають можливість додати особисті дані до додатку. Це несе за собою надання викладачу свої особистих даних для зв’язку. Можливість отримувати на пошту оповіщення про створення події іншим користувачем. Також є можливість додати теги, це ключові слова по яким інші користувачі можуть додавати події, тобто ставлячи тег ярмарка вакансій, користувачу цікаво отримувати схожі події у свій розпорядок. Можливо виділити такі особисті дані:

1) ПІБ;
2) поштова скринька;
3) номер телефону;
4) теги, на які користувач хоче бути підписаним.

Налаштування системи це системні параметри, можна виділити такі типи даних:

1) правила для створення розпорядків дня;
2) пароль для реєстрації викладача;

Архівовані дані, це ті самі розпорядки дня, але вже минувших днів. Мають таку саму структуру що і розпорядок дня. Користувач у будь який момент може переглянути свій архівований розпорядок, але лише при умові з'єднання, система не буду зберігати локально ці дані, бо вважає їх не актуальними

2.2 Перелік вихідних документів

У результаті розрахунків та взаємодії з системою будуть отримані такі дані:

1) розпорядки дня усіх користувачів системи;
2) особисті налаштування користувачів;
3) список учасників по тегам;
4) список усіх користувачів;
5) архівовані дані;

Розпорядки дня усіх користувачів системи. Система формує розпорядки дня на основі розкладу, значеного адміністратором, особистих подій користувача та подій, які були додані іншими користувачами. Оскільки система не допускає конфліктів подій, то розпорядки будуть унікальними та не матимуть збігів двох різних подій у часі, для одного користувача.

Особисті налаштування користувачів. Користувач додає особисті данні, та на основі них, система взаємодіє з користувачем. Він може заповнювати свою форму та редагувати її. До особистих налаштувань входять такі дані:

1) адреса поштової скриньки;
2) логін;

3) пароль;
4) ПІБ користувача;
5) фотографія користувача;
6) теги, на які підписаний користувач;
7) факультет, кафедра курс;
8) дані активності, режим сну та інше;

Список учасників по тегам. Даний фрагмент є словником у вигляді тегу та списку користувачів, які на нього підписані. Користувачі самі вирішують яку особисту інформацію відкривати яку приховувати. Кожен користувач у будь який момент може додавати та видаляти теги.

Список усіх користувачів. Адміністратор має доступ до фрагменту, який містить список усіх користувачів з їх особистою інформацією. Прямо з якого він може формувати групи та додавати їх викладачам.

Архівовані дані. Кожен користувач має архівовані дані його виконаних подій. У цей список входять пройдені курс, відвідані консультації. Також виділені перездачі. Особисті події які він сам додав, та ті на які його підписали інші користувачі.

2.3 Опис організації інформаційної бази даних

У якості бази даних було обрано NoSQL базу даних MongoDB. Була обрана саме ця база даних, тому що програмні засоби, якими планується користуватися мають гарну синергію з даним продуктом. Також велику ролі зіграли власні уподобання у розробці.

MongoDB - це крос-платформна і з відкритим вихідним кодом документально орієнтована база даних. Як база даних NoSQL, MongoDB уникає структуру на основі таблиці реляційних баз даних, щоб адаптувати документи, подібні до JSON, які мають динамічні схеми, які вона викликає BSON.

Це робить інтеграцію даних для деяких типів програм швидше і простіше. MongoDB побудована для масштабованості, високої доступності та продуктивності від розгортання одного сервера до великих і складних інфраструктур.

Оскільки MongoDB базується на концепції колекцій, то було використано на повну парадигму об’єктної орієнтованого проектування. Були створені колекції об’єктів. У даній базі даних немає такого поняття як зв’язки, тому була створена система посилань між колекціями.

Усі колекції та їх посилання описані в наступному розділі.

2.4 Опис інформаційного забезпечення системи

Колекція бази даних “Users”, відповідає особистим даним користувача. Записана у Таблиці 2.1

Таблиця 2.1 – Опис Колекції “Users”

Змінна	**Призначення**	**Тип даних**	**Обов’язкове**	**Унікальне**
_id	Ідентифікатор	ObjectID	+	+
Login	Логін акаунта	String	+	+
Password	Пароль акаунта	String	+	-
First_name	Ім’я	String	-	-
Last_name	Фамілія	String	-	-
Email	Адреса поштової скриньки	String	+	+
Mobile_number	Номер телефону	String	+	+
Type	Тип користувача	String	+	-
groups	Список груп, у які входить користувач	List<Object ID>	-	-

Продовження **таблиці 2.1**

Courses	Список курсів, які викладає даний користувач (Лише для викладача)	List<Course>	-	-
Registration_date	Дата регестрації	Date	+	-
Schedule	Ідентифікатор розкладу користувача	List<ObjectID>	+	+
Tags	Список тегів, доданих користувачем	List<String>	-	-

Колекція “Schedule” містить у собі об’єкти, кожен з яких має повну інформацію по одному дню розкладу. Даний об’єкт можна було занести у об’єкт “User”, але тоді ми не зможемо зробити анонімну статистику по розпорядкам дня. Тому було прийнято рішення відокремити ці дані. Маючи колекцію розпорядків, ми не знаємо який з них кому належить, тому ми можемо збирати статистичні дані без порушення особистого простору.

Таблиця 2.2 – Колекція “Schedules”

Змінна	**Призначення**	**Тип даних**	**Обов’язкове**	**Унікальне**
_id	Ідентифікатор	ObjectID	+	+
Date	Дата даного дня	Date	+	+
Day_of_week	День тиждня	String	+	+
Actions	Події даного дня	List<Action>	+	+

У колекції “Schedules” були згадано об’єкт Action. Опишемо його більш детально. Для нього не виділяється своя колекція, тому що він не може існувати окремо від об’єкту “Schedule”.

Таблиця 2.3 –об’єкт “Action”

Змінна	**Призначення**	**Тип даних**	**Обов’язкове**	**Унікальне**
Header	Заголовок	String	+	+
Description	Пояснення	String	+	+
Type	Тип події	String	+	+
Priority	Пріоритет (чи має можливість переносу дана подія)	Boolean	+	+
Duration	Тривалість події у форматі ГГ:ХВ	String	+	+
Time	Має три варіанти: 1. Точний час початку (у форматі ГГ:ХВ) 2. Період часу, у який повинна бути виконана подія (у форматі ГГ:ХВ-ГГ:ХВ) 3. “free”, у випадку коли подія повинна бути сьогодні, але не важливо в який час.	String	+	+

Продовження **таблиці 2.3**

Змінна	**Призначення**	**Тип даних**	**Обов'язкове**	**Унікальне**
Result_time	Вирахуваний системою час початку події (у форматі ГГ:ХВ)	String	+	+

Колекція "Groups" містить у собі всі групи системи, з переліком студентів, які входять у дані групи та з переліком викладачів, які курують дані групи

Таблиця 2.4 – Колекція "Groups"

Змінна	**Призначення**	**Тип даних**	**Обов'язкове**	**Унікальне**
_id	Ідентифікатор	ObjectID	+	+
Name	Літерал групи	String	+	+
Grade	Курс, на якому навчається група	Number	+	+
Students	Перелік студентів, які входять в дану групу	List<ObjectID>	+	+
Curses	Перелік предметів, які проходять студенти у даному семестрі	List<course>	+	+
Email	Пошта групи	String	+	+
Headmen	Староста групи	ObjectID	+	+

У колекції "Groups" був згаданий об'єкт course. У наступній таблиці він описаний більш детально.

Таблиця 2.5 – об'єкт "course"

Змінна	**Призначення**	**Тип даних**	**Обов'язкове**	**Унікальне**

Course	Ідентифікатор курсу	ObjectID	+	+
Teacher	Ідентифікатор викладача	ObjectID	+	+

Колекція “Courses” містить усі курси, які проходять на даний момент в університеті, з її допомогою можливо збирати статистичні дані по предмету.

Таблиця 2.6 – Колекція “Courses”

Змінна	**Призначення**	**Тип даних**	**Обов’язкове**	**Унікальне**
_id	Ідентифікатор	ObjectID	+	+
Name	Назва курсу	String	+	+
Teachers	Викладачі, які викладають даний курс	List<ObjectID>	+	+
Hours	Кількість годин	Number	+	+
Credits	Кількість кредитів	Number	+	+

Висновок до розділу

Розділ з інформаційного забезпечення створений для детального опису та аналізу вхідних та вихідних даних.

Було обґрунтовано вибір бази даних MongoDB, як нереляційну базу даних. Також було описано та проаналізовано усі дані, які будуть зберігатися та взаємодію між ними. Колекції, що будуть входити у бази даних, були описані у вигляді таблиць. Деякі, складні об’єкти, що будуть частинами більших об’єктів у колекції також були описані у вигляді таблиць.

3 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Змістова постановка задачі

Більшості людей, які користуються календарями, не подобається їх точність. Тобто певна подія повинна відбуватися тоді і лише тоді, а що коли ти хочеш це зробити сьогодні, але тобі не важливо коли, але при завантаженому розпорядку, не маючи точного часу, ти можеш дуже легко втратити такі справи, а ще це може бути особлива подія, як наприклад заняття спортом, коли тобі потрібно до та після поїсти. При боязні втратити цю подію ти плануєш її заздалегідь, через це твій розпорядок втрачає гнучкість. Якщо в тебе з’являються важливі справи, тобі потрібно особисто все змінювати та витрачати на це час. Тому було вирішено автоматизувати процес зміни розпорядку під дією зовнішніх чинників.

Концепція в тому, що система сама створює тобі розпорядок. Тобто тобі не потрібно піклуватись про плаваючі події або про перенос їх під дією зовнішніх чинників. Система поставить сама тобі ранкове чи вечірнє тренування у залежності від завантаження графік та врахує прийом їжі до та після. Система врахує твою виснаженість та поставить після цього не фізичну діяльність.

Система не приватизує час користувача, вона лише дає декілька варіантів, які йому підходять, якщо жоден з запропонованих, то він може написати свій.

3.2 Математична постановка задачі

У роботі розглядається практичне завдання створення інтелектуальної системи, що дозволяє зберігати і обробляти інформаційний ресурс про розпорядок дня. Існуючі «паперові» методи не виправдовують себе, вимагаючи дуже великої кількості часу часто приводячи до помилок в роботі.

Для знаходження залежностей між наборами розпорядків використовуються асоціативні правила. Нехай є безлічі об'єктів A і B, перетин яких порожній. Тоді асоціативне правило між A B означає, що якщо множина A включено в транзакцію, тоді швидше за все безліч B теж буде включено в цю транзакцію. Є дві метрики, які формалізують це "швидше за все". Перша з них, вже зазначена підтримка sup (1), тобто кількість транзакцій, що містять $A \cup B$, друга – достовірність $conf$ (2), що показує умовну ймовірність безлічі B при наявності безлічі A.[9]

$$\sup(A) = \frac{number\ of\ transactions\ comtainingA \cup B}{number\ of\ transactions} \quad (1)$$

$$conf(A) = \frac{number\ if\ transactions\ comtainingA \cup B}{number\ of\ transactions\ comtaining\ A} \quad (2)$$

Прийнято шукати правила, підтримка і достовірність яких вище будь-яких двох мінімальних значень, визначених тим, хто застосовує цей метод. Правила з високою вірогідністю дають найбільш точні прогнози. При цьому підтримка теж важлива, тому що правила з маленькою підтримкою, тобто нечасті, не несуть великий інформації і більш того можуть бути викидами.

Для того, щоб створити рекомендації, для кожного користувача потрібно знайти всі правила, такі що користувач вибрав всі об'єкти, які знаходяться в лівій частині. При цьому це повинні бути правила, які задовольняють мінімальним підтримки і достовірності. Тоді будь-який з об'єктів, який знаходиться в правій частині правил, може бути рекомендованим. Нехай R множина всіх об'єктів, які перебували в правій частині правил і ще не обраних користувачем, тоді можна впорядкувати це за достовірності правил, в яких вони були представлені, так що об'єкти з найбільшою підтримкою знаходяться вище. При цьому якщо об'єкт був в декількох правилах, то береться максимальна достовірність. Після цього можна вибрати перші N об'єктів і рекомендувати їх користувачеві.

Розроблена система при заданому запиті повинна підбирати клієнту необхідний розпорядок Для цього користувач повинен ввести події та додати їх типи, за якими система визначить розпорядок. Вона запропонує декілька варіантів. Також система збирає статистичні дані користувачів та оновлює свої правила створення розпорядків.

3.3 Обґрунтування методу розв’язання

Задача зводиться до пошуку асоціативних правил, наприклад після занять спортом, люди будуть приймати їжу, щоб поповнити енергію. Людина, яка провела багато часу працюючи за комп’ютером, навряд чи буде приділяти час на читання книжки, вона скоріше здійснить прогулянку. Все це зводиться до аналізу великої кількості розпорядків, та пошуку залежності між послідовністю подій в залежності від часу, дня тижня, пори року.

Для спрощення аналізу було вирішено поділити події на такі категорії: фізична робота, розумова робота, активний відпочинок, пасивний відпочинок, заняття спортом.

3.4 Опис методів розв’язання

Для пошуку асоціативних правил був обраний алгоритм Apriori[10]. Apriori - масштабований алгоритм пошуку асоціативних правил.

Виявлення часто зустрічаються наборів елементів - операція, яка потребує багато обчислювальних ресурсів і, відповідно, часу. Примітивний підхід до вирішення даного завдання - простий перебір всіх можливих наборів елементів. Це зажадає $O(2^I)$ операцій, де | I | - кількість елементів. Apriori використовує одна з властивостей підтримки, з якого випливає: підтримка будь-якого набору елементів не може перевищувати мінімальної підтримки будь-якого з його підмножин. Далі детально буде описаний даний алгоритм.

На першому кроці алгоритму підраховуються 1-елементні набори, які часто зустрічаються. Для цього необхідно пройтись по всьому набору даних і підрахувати для них підтримку, тобто скільки раз зустрічається в базі.

Наступні кроки будуть складатися з двох частин: генерації наборів, які часто зустрічаються (їх називають кандидатами) і підрахунку підтримки для кандидатів.

Наведемо функцію генерації кандидатів. На цей раз немає ніякої необхідності знову звертатися до бази даних. Для того, щоб отримати набори з k елементів , скористаємося наборами з (k-1) елементів, які були визначені на попередньому етапі та часто зустрічаються.

Згадаймо, що наш початковий набір зберігається у впорядкованому вигляді. Генерація кандидатів також буде складатися з двох кроків:

1) об'єднання. Кожен кандидат C_k буде формуватися шляхом розширення набору розміром (k-1), який часто зустрічається, додаванням елемента з іншого набору (k-1) елементів;
2) видалення надлишкових правил. На підставі властивості анти-монотонності, слід видалити всі набори с $\in C_k$ якщо хоча б одна з його (k-1) підмножин не є набором, що часто зустрічається.

Після генерації кандидатів наступним завданням є підрахунок підтримки для кожного кандидата. Очевидно, що кількість кандидатів може бути дуже великою і потрібен ефективний спосіб підрахунку. Самий тривіальний спосіб - порівняти кожну транзакцію з кожним кандидатом. Але це далеко не найкраще рішення. Набагато швидше і ефективніше використовувати підхід, заснований на зберіганні кандидатів в хеш-дереві. Внутрішні вузли дерева містять хеш-таблиці з покажчиками на нащадків, а листя - на кандидатів. Це дерево нам стане в нагоді для швидкого підрахунку підтримки для кандидатів.

Хеш-дерево будується кожен раз, коли формуються кандидати. Спочатку дерево складається тільки з кореня, який є листом, і не містить

ніяких кандидатів-наборів. Кожен раз коли формується новий кандидат, він заноситься в корінь дерева і так до тих пір, поки кількість кандидатів в корені-листі не перевищить певного порогу. Як тільки кількість кандидатів стає більше порога, корінь перетворюється в хеш-таблицю, тобто стає внутрішнім вузлом, і для нього створюються нащадки-листя. І всі приклади розподіляються по вузлам-нащадкам згідно хеш-значень елементів, що входять в набір. Кожен новий кандидат хешується на внутрішніх вузлах, поки він не досягне першого вузла-листа, де він і буде зберігатися, поки кількість наборів знову ж таки не перевищить порога.

Хеш-дерево з кандидатами-наборами побудовано, тепер, використовуючи хеш-дерево, легко підрахувати підтримку для кожного кандидата. Для цього потрібно "пропустити" кожну транзакцію через дерево і збільшити лічильники для тих кандидатів, чиї елементи також містяться і в транзакції, тобто $C_k \bigcap T_i = C_k$. На кореневому рівні хеш-функція застосовується до кожного елементу з транзакції. Далі, на другому рівні, хеш-функція застосовується до других елементів і т.д. На k-рівні хешується k-елемент. І так до тих пір, поки не досягнемо листа. Якщо кандидат, що зберігається в листі, є підмножиною даної транзакції, тоді збільшуємо лічильник підтримки цього кандидата на одиницю.

Після того, як кожна транзакція з вихідного набору даних "пропущена" через дерево, можна перевірити чи задовольняють значення підтримки кандидатів мінімального порогу. Кандидати, для яких ця умова виконується, переходять у множину кандидатів, які часто зустрічаються. Крім того, слід запам'ятати і підтримку набору, вона нам стане в нагоді при добуванні правил. Ці ж дії застосовуються для знаходження наборів з (k + 1) елементів.

Після того як знайдені всі часто зустрічаються набори елементів, можна приступити безпосередньо до генерації правил.

Створення правил - менш складне завдання. По-перше, для підрахунку достовірності правила досить знати підтримку самого набору і множини, що

лежить в умові правила. Наприклад, є набір, що часто зустрічається, {A, B, C} і потрібно підрахувати достовірність для правила AB ⇒ C. Підтримка самого набору нам відома, але його множина {A, B}, що лежить в умові правила, також є набором, що часто зустрічається, в силу властивості анти-монотонності, отже його підтримка нам відома. Тоді ми легко зможемо підрахувати достовірність. Це позбавляє нас від небажаного перегляду бази транзакцій, який потрібен був в тому випадку якби ця підтримка була невідома.

Щоб створити правило з часто набору F, слід знайти всі його непусті підмножини. І для кожної підмножини s ми зможемо сформулювати правило s ⇒ (F - s), якщо достовірність правила conf (s ⇒ (F - s)) = sup (F) / sup (s) незгірш від порога minconf.

Зауважимо, що чисельник залишається постійним. Тоді достовірність має мінімальне значення, якщо знаменник має максимальне значення, а це відбувається в тому випадку, коли в умови правила є набір, що складається з одного елемента. Усі супермножини даної множини мають меншу або рівну підтримку і, відповідно, більше значення достовірності. Ця властивість може бути використано при створені правил. Якщо ми почнемо витягувати правила, розглядаючи спочатку тільки один елемент в умові правила, і це правило має необхідну підтримку, тоді всі правила, де в умови стоять супермножини цього елемента, також мають значення достовірності вище заданого порогу. Наприклад, якщо правило A ⇒ BCDE задовольняє мінімального порогу достовірності minconf, тоді AB ⇒ CDE також задовольняє. Для того, щоб витягти всі правила використовується рекурсивна процедура. Важливе зауваження: будь-яке правило, складене з часто зустрічається набору, має містити всі елементи набору. Наприклад, якщо набір складається з елементів {A, B, C}, то правило A ⇒ B не повинно розглядатися.

Висновок до розділу

У даному розділі було сформовано математичну постановку задачі та обґрунтовано її використання. Також було обґрунтовано та детально описано метод розв’язання поставленої математичної задачі.

4 ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

4.1 Огляд засобів розробки

Для написання дипломної роботи були застосовані такі засоби для програмування : JavaScript як мова програмування, ES6 – найпопулярніший стандарт для JavaScript, ReactJS – бібліотека для спрощення створення інтерфейсів користувача та роботи з ними, Redux – бібліотека для збереження та роботи зі станом застосунку, React Native – бібліотека для конвертування веб технологій у мобільну платформу, HTML – мова розмітки, CSS – каскадні таблиці стилів, React Native Calendar – бібліотека, яка містить у собі готовий компонент календаря, MongoDB – база даних для збереження даних користувачів, NodeJS – для створення серверного застосунку, який буде працювати з базою даних.

JavaScript – мова програмування, що почалася просто як механізм для додавання логіки та інтерактивності в інший статичний браузер Netscape. За роки свого впровадження він не тільки витіснив різні конкурентоспроможні мови та технології, щоб стати стандартом для програмування на основі браузера, але й розширився за межі клієнтського простору, щоб стати домінуючою мовою на стороні сервера.

ES6(EcmaScript6/EcmaScript2015) – стандарт. Хоча JavaScript є найпопулярнішою реалізацією цього стандарту. JavaScript реалізує ECMAScript і будує поверх нього.

ReactJS[3] – бібліотека JavaScript для побудови інтерфейсів користувача. React робить безболісним створення інтерактивних інтерфейсів. Створює прості перегляди для кожного стану вашої програми, і React ефективно оновлює та відтворює лише потрібні компоненти під час зміни даних. Декларативні думки роблять ваш код більш передбачуваним і легше налагоджувати. React допомагає створювати інкапсульовані компоненти, які керують власним станом, а потім використовує їх для створення складних

інтерфейсів користувача. Оскільки логіка компонентів написана в JavaScript замість шаблонів, ви можете легко передати багаті дані через вашу програму і зберегти стан з DOM. Реакція також може рендерувати на сервері за допомогою мобільних додатків Node і Power, використовуючи React Native.

Redux[4] – інструмент управління станом. Хоча це в основному використовується з React, його можна використовувати з будь-яким іншим фреймворком або бібліотекою JavaScript. Він легкий на 2 КБ (у тому числі залежності), тому не потрібно турбуватися про збільшення розміру активу програми. З Redux, стан програми зберігається в магазині, і кожен компонент може отримати доступ до будь-якого стану, який він потребує від цього магазину.

React Native[6-8] – це фреймворк JavaScript, розроблений для створення дійсно рідних програм для платформ, таких як iOS і Android. Вона базується на бібліотеці JavaScript, створеній компанією Facebook під назвою React, і, таким чином, приводить свою владу до розробки власних програм для мобільних пристроїв. Вона відповідає потребам ринку сучасних мобільних додатків, оскільки дві операційні системи, що домінують в ландшафті, створюють мобільні додатки, які часто стикаються з рішенням: створювати програми, які забезпечують кращу роботу користувачів або програми, які швидше розвиваються і працюють на більшій кількості платформ і пристроїв. Тому для відповідності останнім технологіям та відсутності потреби писати окремо під різні платформи було вибрано дану бібліотеку.

HTML(Hypertext Markup Language) – використовується для створення електронних документів (так званих сторінок), які відображаються в World Wide Web. Кожна сторінка містить ряд з'єднань з іншими сторінками, які називаються гіперпосиланнями. Кожна веб-сторінка, яку ви бачите в Інтернеті, написана за допомогою однієї чи іншої версії HTML-коду.

CSS(Cascading Style Sheets) – мова, що використовується для опису повторно використовуваних стилів для представлення документів, написаних мовою розмітки. Його концепція була створена Håkon Wium Lie в 1994 році. У грудні 1996 року, CSS була зроблена специфікація W3C і сьогодні дозволяє веб-розробникам змінювати макет і зовнішній вигляд своїх веб-сторінок. Наприклад, CSS може використовуватися для зміни шрифту, який використовується в певному елементі HTML, а також його розміру і кольору. Один файл CSS може бути пов'язаний з кількома сторінками, що дозволяє розробнику змінювати вигляд усіх сторінок одночасно.

MongoDB[1] – база даних з відкритим вихідним кодом і провідна NoSQL база даних. MongoDB написана на C++. У базах даних NoSQL для доступу до даних і управління ними застосовуються різні моделі даних, в тому числі документная, графовая, пошукова, з використанням пар «ключ-значення» і зберіганням даних в пам'яті. Бази даних таких типів оптимізовані для додатків, які працюють з великим обсягом даних, потребують низької затримки і гнучких моделях даних. Все це досягається шляхом пом'якшення жорстких вимог до несуперечності даних, характерних для інших типів БД.

NodeJS[5] – відкрите джерело крос-платформенного середовища виконання JavaScript, яке виконує код JavaScript поза браузером. Node.js дозволяє розробникам використовувати JavaScript для написання інструментів командного рядка, а для серверних сценаріїв - запуску на сервері скриптів для створення динамічного вмісту веб-сторінки, перш ніж сторінка буде надіслана веб-браузеру користувача. Отже, Node.js представляє парадигму "скрізь JavaScript", що об'єднує розробку веб-додатків навколо однієї мови програмування, а не на різних мовах для серверних і клієнтських скриптів. У даному випадку використовується для переведення комунікації з базою даних зі сторони клієнта на серверну частину, щоб пришвидшити роботу додатка

4.2 Вимоги до технічного забезпечення

Для правильної роботи розробленої системи до складу технічних засобів повинен входити смартфон, що має конфігурацію наведену нижче:

a) процесор з тактовою частотою не нижче 1,0 ГГц;
b) об’єм оперативної пам’яті не менше 512 МБ;
c) об’єм пам’яті носія не нижчий за 2Гб;
d) доступ до мережі інтернет.

Також на сервері повинні бути розгорнуті наступні сервіси

a) база даних MongoDB;
b) сервер на NodeJS.

Якщо не є можливим розгорнути віддалено, тоді потрібно використовувати локальну мережу з комп’ютером наступної конфігурації:

a) процесор з тактовою частотою не менше ніж 1,7 ГГц;
b) об’єм оперативної пам’яті не менше 1024 МВ;
c) об’єм локальної пам’яті не менше 500Гб;
d) доступ до мережі інтернет.

Додатково має бути встановлене наступне програмне забезпечення:

a) операційна система Linux/Windows/OSX;
b) база даних MongoDB;
c) NodeJS;
d) будь який браузер, який підтримує стандарти ES6.

4.3 Архітектура програмного забезпечення

Архітектура програмного забезпечення показана на рисунку 4.1. Система, що розробляється будується за архітектурою REST. Маємо три рівні: клієнтський у мобільному додатку, серверний з обробкою запитів та бази даних. Діаграма взаємодії рівнів зображена на рисунку 4.2.

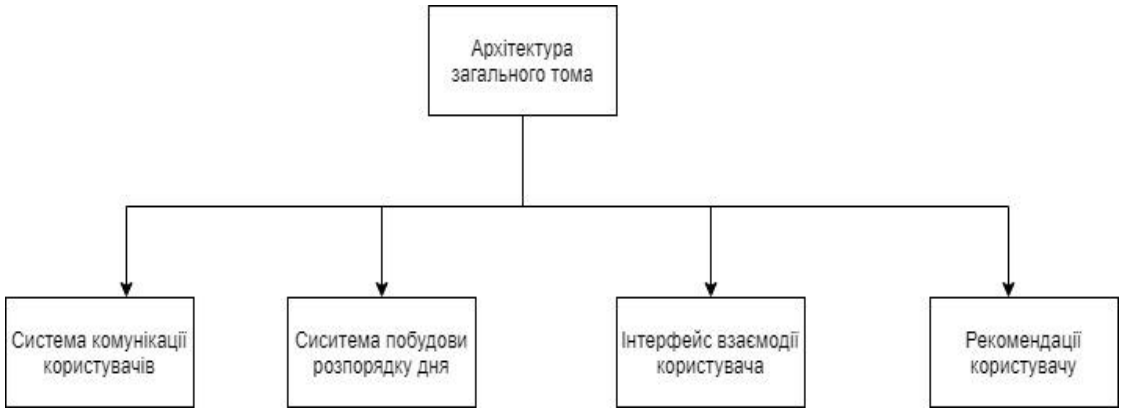


Рисунок 4.1 – Архітектура програмного забезпечення

REST є абревіатурою з **RE**presentational **S**tate **T**ransfer. Це архітектурний стиль для розподілених гіпермедійних систем і вперше був представлений Роєм Філдінгом у 2000 році в його знаменитій дисертації.

Як і будь-який інший архітектурний стиль, REST також має свої власні 6 керівних обмежень, які повинні бути задоволені, якщо інтерфейс має називатися RESTful. Ці принципи перелічені нижче.

Керівні принципи REST:

1) Клієнт-сервер - Розділяючи питання користувальницького інтерфейсу від проблем зберігання даних, ми покращуємо портативність інтерфейсу користувача на декількох платформах і покращуємо масштабованість, спрощуючи компоненти сервера.

2) Без статусу - кожен запит від клієнта до сервера повинен містити всю інформацію, необхідну для розуміння запиту, і не може скористатися будь-яким збереженим контекстом на сервері. Таким чином, стан сесії зберігається повністю на клієнті.

3) Cacheable - обмеження кешу вимагають, щоб дані у відповіді на запит були неявно або явно позначені як кешуються або не кешуються. Якщо відповідь кешується, то клієнтському кешу надається право повторно використовувати дані відповіді для наступних еквівалентних запитів.

4) Рівномірний інтерфейс - Застосовуючи принцип спільної інженерії програмного забезпечення до інтерфейсу компонентів, загальна архітектура системи спрощується, а видимість взаємодій покращується. Для того, щоб

отримати єдиний інтерфейс, необхідні численні архітектурні обмеження для керування поведінкою компонентів. REST визначається чотирма обмеженнями інтерфейсу: ідентифікацією ресурсів; маніпулювання ресурсами через представництва; самоописувальні повідомлення; і, гіпермедіа як двигун стану застосування.

5) Пластова система - стиль шаруватої системи дозволяє архітектурі складатися з ієрархічних шарів, обмежуючи поведінку компонентів таким чином, що кожен компонент не може «бачити» поза межами безпосереднього рівня, з яким вони взаємодіють.

6) Код за запитом (необов'язково) - REST дозволяє розширити функціональність клієнта шляхом завантаження та виконання коду у вигляді аплетів або скриптів.

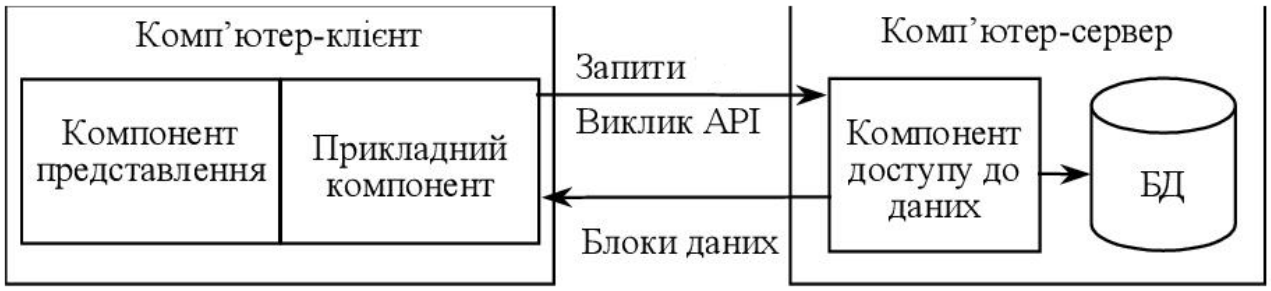


Рисунок 4.2 – Діаграма взаємодії рівнів архітектури

4.3.1 Діаграма класів

Так як система має три рівні архітектури, має три різні діаграми класів. Наведена найбільша з них, діаграма класів мобільного застосунку, яка наведена у частині графічного метеріалу.

Діаграма класів містить наступні класи:

1) Navigator – це клас обгортка з стандартної бібліотеки, який допомагає налаштувати навігацію між екранами у застосунку

2) Authorization – це клас екрана авторизації, який відповідає за перевірку даних авторизації да відправку їх на сервер, у разі позитивної відповіді він намагається перейти до екрану домашньої сторінки

3) HomePage – це клас екрана домашньої сторінки, який відображає фотографію користувача, та календар, можна перейти до екрану профіля та списку подій

4) Calendar – це клас, який використовує календар з бібліотеки та надає йому потрібного вигляду, також відмічає там потрібні дні з важливими подіями

5) Profile – це клас екрану налаштування профіля, він виводить всю інформацію користувача, дає можливість її редагувати та відсилати змінення на сервер

6) TaskList – це клас екрану списку задач, визивається з конкретною датою, до якої достає усі події та відображує їх

7) Task – це клас події, має графічний інтерфейс, та може розкрити чи скрити подію

8) TaskForm – це класс екрану заповнення нової форми події та додання її

4.3.2 Специфікація функцій

Функції класів програмного забезпечення

Таблиця 4.1 – Функції класу Navigator

Функція	**Опис функції**
navigate(string screen, … params)	Бібліотечна функція, призначена для переходу між екранами. Приймає перший параметр назва екрана на який робити перехід та необмежену кількість параметрів, які будуть передані даному екрану у конструктор

Таблиця 4.2 – Функції класу Authorization

Функція	Опис функції
sendAutho(string login, string password)	Відправляє на сервер комбінацію логіна та пароля, та чекає особисту інформацію користувача, якщо користувач не знайдений видає оповіщення про помилку, у випадку авторизації відкриває домашню сторінку

Таблиця 4.3 – Функції класу HomePage

Функція	Опис функції
openProfile()	Відкриває сторінку редагування особистих даних
openScheldue(Date day)	Відкриває сторінку списку подій по заданій даті, цей метод визивається з календаря.

Таблиця 4.4 – Функції класу Calendar

Функція	Опис функції
onDayPress(Date day)	Функція обробник подій натиску на календар. Обробляє подію натиску, та відсилає дату, на яку натиснули у клас домашньої сторінки

Таблиця 4.5 – Функції класу Profile

Функція	Опис функції
updateProfile(Object user)	Відсилає змінені дані на сервер, у разі отримання позитивної відповіді відкриває домашню сторінку, у іншому випадку видає оповіщення про помилку
handleInpur(Event e)	Перевіряє формат заповнених даних користувачем в активному виді, як тільки буде недопустимий символ чи помилка, система відразу оповістить користувача без взаємодії з сервером

Таблиця 4.6 – Функції класу TaskList

Функція	Опис функції
addTask()	Функція, викликає екран з формою створення нової події
addTaskToAnother(ObjectID recipent)	Функція створення події іншим користувачам(доступна лише викладачу), при викликанні відкривається спливаюче вікно з полем у яке можна ввести або номер групи або П.І.Б студента, додаток робить запит на сервер і після цього відкриває форму створення події

Таблиця 4.7 – Функції класу Task

Функція	Опис функції
showTask(Object task)	Розкриває подію для зображення усієї інформації по ній
hideTask(Object task)	Скриває подію для зображення лише часу та назви

Таблиця 4.8 – Функції класу TaskForm

Функція	Опис функції
addUser(Object task)	Функція додавання нової події, відсилає запит на сервер, у разі успішності додає та переходить на екран списку подій, у разі помилки видає помилку та поради для її усунення.
handleInpur(Event e)	Перевіряє формат заповнених даних користувачем в активному виді, як тільки буде недопустимий символ чи помилка, система відразу оповістить користувача без взаємодії з сервером

Висновок до розділу

У даному розділі були описані засоби, які використані під час розробки додатку, що реалізує електронний щоденник студента, для мобільної платформи. Аргументовано вибір мови програмування та бібліотек, які використані з нею. Також описано технологію по які збудована архітектура.

Архітектура була показана за допомогою діаграми класів, діаграми компонентів. Наведена специфікація функцій у вигляді таблиць.

5 ТЕХНОЛОГІЧНИЙ РОЗДІЛ

5.1 Керівництво користувача

Користувач завантажує додаток з Play Маркет, якщо це операційна система Android, та з App Store, якщо операційна система IOS. Далі він відкриває його та потрапляє на стартовий екран (Рисунок 5.1)

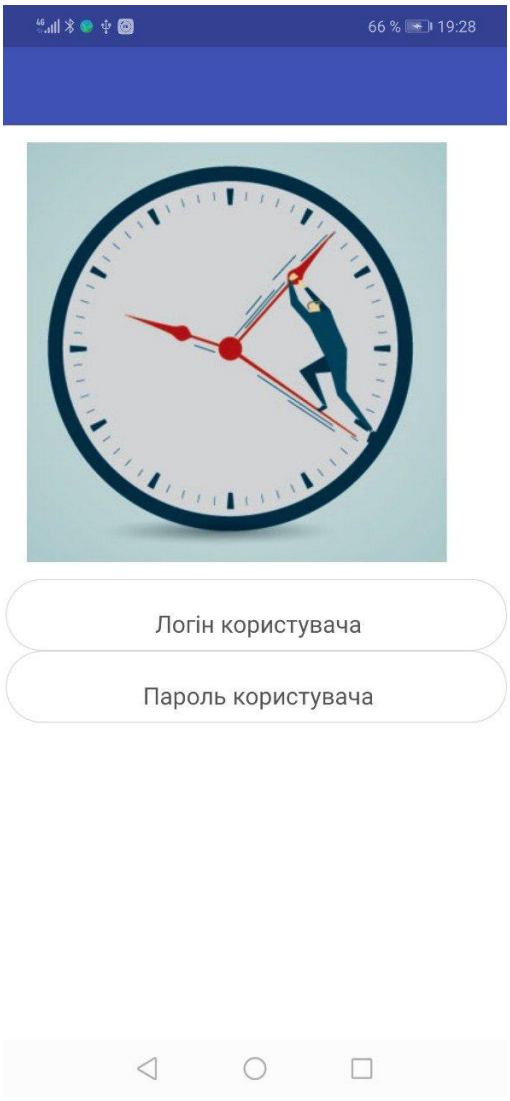


Рисунок 5.1 – Екран авторизації користувача

Далі користувач вводить свої дані, у вигляді логіна та пароля, який йому видав адміністратор в університеті. Після авторизації користувач бачить екран зі своєю фотографією профіля та календарем(Рисунок 5.2).

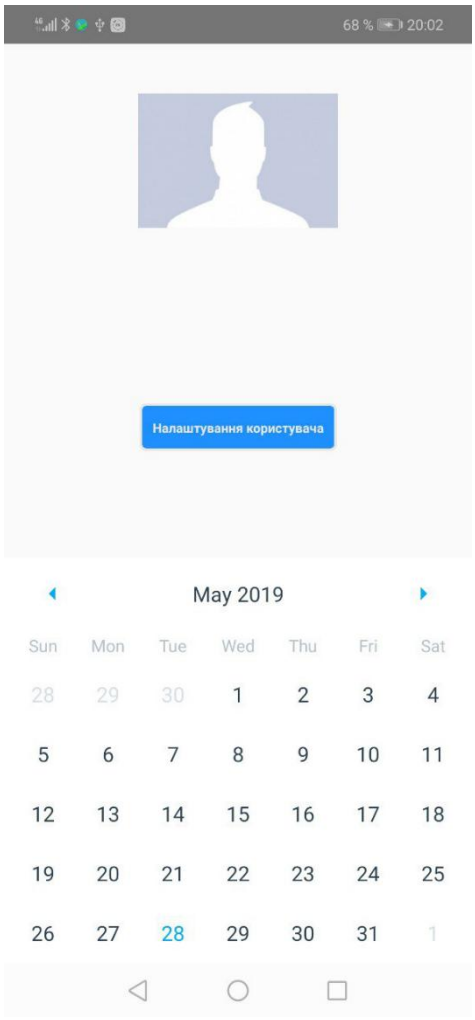


Рисунок 5.2 – Початковий екран користувача

Далі користувач може натиснути на кнопку “Налаштування користувача” можна побачити форму з налаштуванням користувача, де можна змінити особисті дані, у які входить мобільний телефон, поштова скринька та інше.

Користувач має можливість обрати дату. При виборі дати, будуть завантажені події по даній даті, демонструємо вибір дати, а точніше 30 травня 2019.

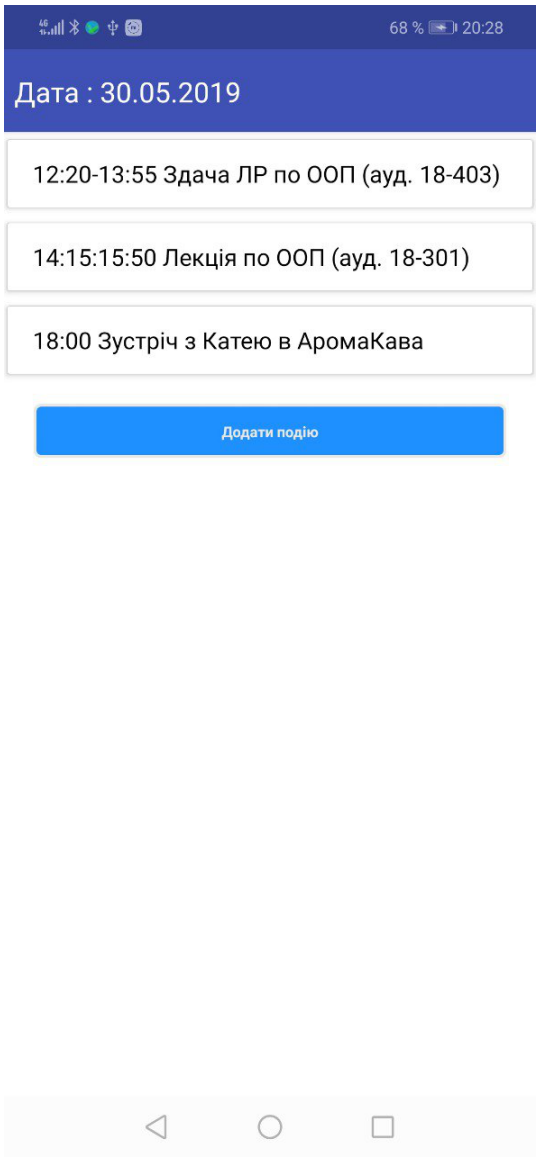


Рисунок 5.3 – Розклад на 30.05.2019

Як бачимо на рисунку, видно усі події даного дня, при натисканні на них можна побачити повну інформацію про них, також є можливість додати ще одну подію при натисканні на кнопку “Додати подію”. Натиснемо кнопку додавання події та потрапимо на форму події, яка зображена на рисунку 5.4.

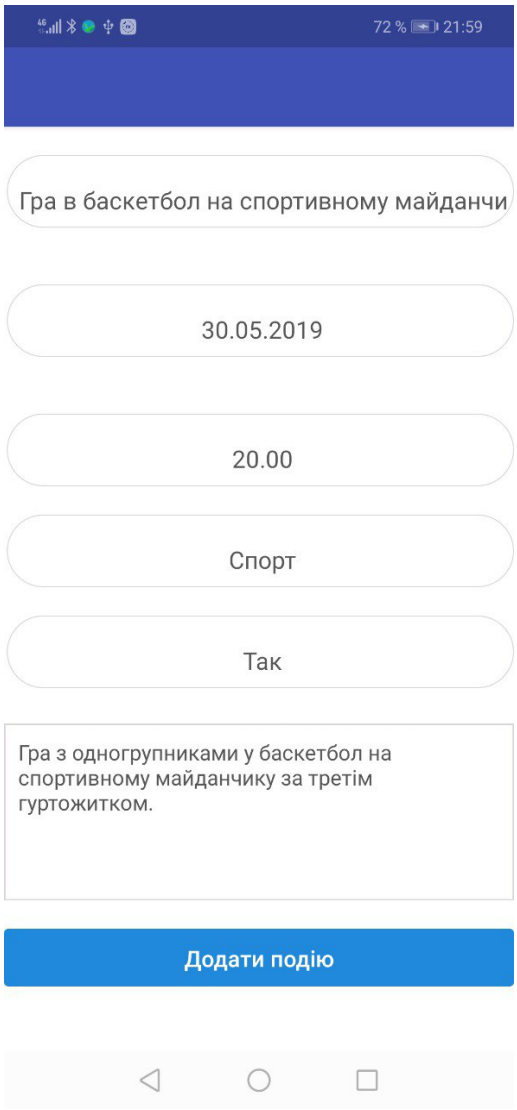


Рисунок 5.3 – форма створення нової події

Заповнимо форму :

1. Назва, яка буде показана в розпорядку
2. Дата, у яку буду відбуватися подія
3. Час, має два варіанти перший точний час(ГГ:ХВ), другий тривалий (ГГ:ХВ- ГГ:ХВ), система по різному розпізнає дані події
4. Тип події, для роботи з системою
5. Можливість переносу
6. Поле опису події, де можна вписати нотатки або іншу інформацію

Додамо дану подію. Бачимо отриманий результат на рисунку 5.4.

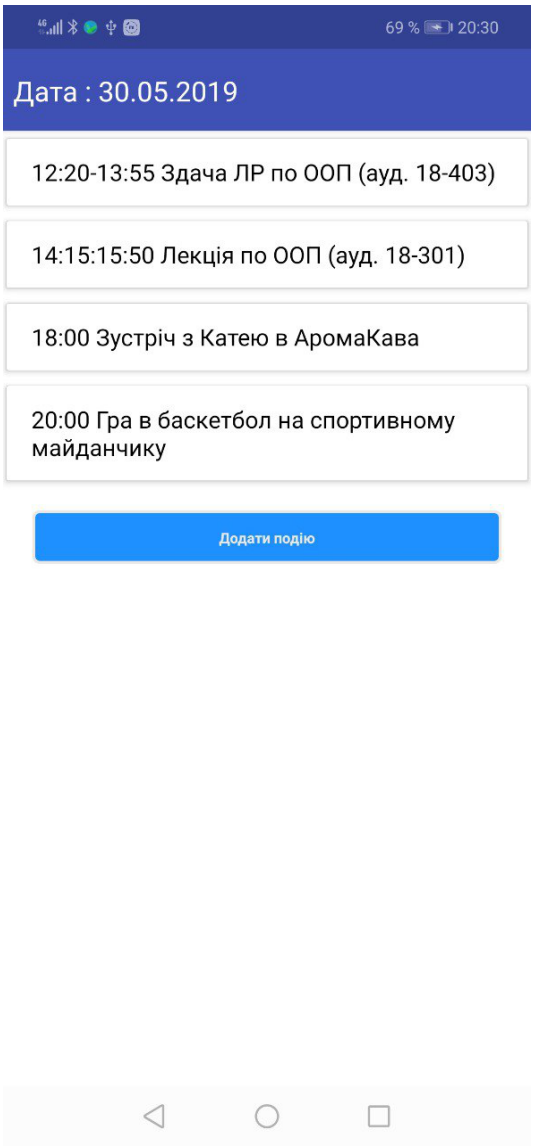


Рисунок 5.4 – Результат додавання події.

5.2 Випробування програмного продукту

Програма і методика випробувань містять опис тестів і порядок їх виконання для перевірки відповідності функціональним вимогам, показаним у технічному завданні.

5.2.1 Мета випробування

Метою випробування є перевірка відповідності функцій даного додатку до тих, щоб були описані в технічному завданні.

5.2.2 Загальні положення

Випробування проводяться на основі наступних документів:

a) ГОСТ 34.603–92. Інформаційна технологія. Види випробувань автоматизованих систем;
b) ГОСТ РД 50-34.698-90. Автоматизовані системи вимог до змісту документів.

5.2.3 Результати випробування

У процесі тестування було перевірено основний функціонал застосунку. Перелік випробувань основних функціональних можливостей наведений у таблицях 5.1 – 5.9.

Таблиця 5.1 – Тест вдалого проходження авторизації

Мета тесту	Перевірити можливість авторизації
Початковий стан	Відкритий екран авторизації
Вхідні дані	Логін та пароль користувача
Схема проведення тесту	Заповнити форму авторизації
Очікуваний результат	Відкрите вікно авторизованого користувача з календарем
Стан після проведення випробувань	Користувач авторизований у системі

Таблиця 5.2 – Тест змінення особистих даних

Мета тесту	Перевірити можливість зміни особистих даних
Початковий стан	Відкритий екран особистих налаштувань

Продовження **таблиці 5.2**

Мета тесту	Перевірити можливість зміни особистих даних
Вхідні дані	Особисті дані у вигляді ім’я, фамілія, поштова скринька, номер мобільного телефону тощо
Схема проведення тесту	Заповнити форму особистих даних
Очікуваний результат	Відкрите початкове вікно користувача з календарем
Стан після проведення випробувань	Змінені особисті дані користувача

Таблиця 5.3 – Тест вдалого перегляду подій по обраній даті

Мета тесту	Перевірити можливість перегляду подій по даті
Початковий стан	Відкритий початковий екран користувача з календарем
Вхідні дані	Дата пізніше заданої
Схема проведення тесту	Обрати дату у календарі
Очікуваний результат	Відкрите вікно з лістингом подій по заданій даті, відсортованих по часу
Стан після проведення випробувань	Відкрите вікно перегляду подій

Таблиця 5.4 – Тест вдалого додавання події

Мета тесту	Перевірити правильність роботи додавання події
Початковий стан	Відкритий екран форми додавання події
Вхідні дані	Параметри події: назва, дата, час,

	тип, можливість переносу, опис

Продовження **таблиці 5.4**

Мета тесту	Перевірити правильність роботи додавання події
Схема проведення тесту	Заповнити форму додавання події
Очікуваний результат	Відкрите вікно з лістингом подій, по тій даті, до якої було додано подію
Стан після проведення випробувань	По заданій даті додано подію

Таблиця 5.5 – Тест додавання події, що накладається з уже існуючою

Мета тесту	Перевірити функціонал системи з переносу подій
Початковий стан	Відкритий екран додавання події. У системі вже є подія з такою ж датою та часом
Вхідні дані	Параметри події: назва, дата, час, важливість, можливість переносу, опис
Схема проведення тесту	Заповнити форму додавання події. Натиснути кнопку додавання. З’явиться вікно з пропозицією переносу тієї з двох подій, у якої найменший пріоритет, поле з часом та датою редагується, але є запропонований варіант. Обрати підходящу дату. Натиснути кнопку погодження
Очікуваний результат	Відкрите вікно з лістингом подій, по

	тій даті, до якої було додано подію

Продовження **таблиці 5.5**

Мета тесту	Перевірити функціонал системи з переносу подій
Стан після проведення випробувань	Розпорядок змінений у системі.

Таблиця 5.6 – Тест додавання події без можливості переносу

Мета тесту	Перевірити правильність роботи системи з конфліктами подій
Початковий стан	Відкритий екран додавання події
Вхідні дані	Параметри події: назва, дата, час, важливість, можливість переносу(Ні), опис
Схема проведення тесту	Заповнити форму додавання події. З’явиться вікно про неможливість додання такої події, з проханням змінити параметри
Очікуваний результат	Відкритий екран додавання події. Параметри дата та можливість переносу підкреслені червоним
Стан після проведення випробувань	Система залишилась без змін

Таблиця 5.7 – Тест видалення події

Мета тесту	Перевірити можливість видалення події
Початковий стан	Відкритий екран з лістингом подій
Вхідні дані	Подія, яку хочемо видалити
Схема проведення тесту	Довге натискання на заданій події.

	З’явиться вікно погодження. Натискаємо “Так”

Продовження **таблиці 5.7**

Мета тесту	Перевірити можливість видалення події
Очікуваний результат	Відкритий екран з лістингом подій, без події, яку щойно видалили
Стан після проведення випробувань	Система залишилась без заданої події

Таблиця 5.8 – Тест додавання події групі користувачів

Мета тесту	Перевірити можливість додавання події іншому користувачу по групі
Початковий стан	Відкрита домашня сторінка з календарем у профілі викладача
Вхідні дані	Подія, яку хочемо додати та група, якій хочемо додати
Схема проведення тесту	Натискання на кнопку додавання події студентам, введення літералу групи, заповнення форми події, натискання на кнопку відправлення
Очікуваний результат	Оповіщення про успішне додання події
Стан після проведення випробувань	У всіх користувачів, які входять у дану групу з’являється задана подія

Таблиця 5.9 – Тест додавання події іншому користувачу

Мета тесту	Перевірити можливість додавання події іншому користувачу по імені

Початковий стан	Відкрита домашня сторінка з календарем у профілі викладача

Продовження **таблиці 5.9**

Мета тесту	Перевірити можливість додавання події іншому користувачу
Вхідні дані	Подія, яку хочемо додати та користувач, якому хочемо додати
Схема проведення тесту	Натискання на кнопку додавання події студентам, введення П.І.Б студента, заповнення форми події, натискання на кнопку відправлення
Очікуваний результат	Оповіщення про успішне додання події
Стан після проведення випробувань	У користувача, якого ми указали в спливаючому вікні додана задана подія

Таблиця 5.10 – Тест додавання події користувачам по тегу

Мета тесту	Перевірити можливість додавання події іншому користувачу по тегу
Початковий стан	Відкрита домашня сторінка з календарем у профілі викладача
Вхідні дані	Подія, яку хочемо додати та тег, по якому хочемо додати
Схема проведення тесту	Натискання на кнопку додавання події студентам, введення тегу, заповнення форми події, натискання на кнопку відправлення
Очікуваний результат	Оповіщення про успішне додання

	події
Мета тесту	Перевірити можливість додавання події іншому користувачу по тегу
Стан після проведення випробувань	У всіх користувачів, які входять у дану групу з’являється задана подія

Продовження **таблиці 5.10**

Висновок до розділу

У даному розділі було перевірено відповідність роботи додатку до заявлених у технічному забезпеченні вимог. Також створено керівництво користувача з скріншотами додатку, для спрощеного розуміння читача. Описано мету тестування, його підстави та увесь процес тестування, у вигляді таблиць.

ВИСНОВКИ

У ході виконання дипломного проекту, було детально розглянуті питання, з якими стикаються люди при формуванні свого розпорядку дня. Було розглянуто проблеми та недоліки існуючих додатків. Були виділенні основні ключові етапи, притаманні процесу та взаємозв’язки між ними.

Метою розробки є допомогти студентам та викладачам з управлінням часу та автоматизувати процес створення розпорядку дня по заданим параметрам.

Було наведено опис інформаційного забезпечення та представлена структура бази даних у вигляді таблиць опису колекцій, оскільки обрана нереляційна база даних.

Був проведений ґрунтовий аналіз предметного середовища та ретельно описаний бізнес-процес плану розробки.

Було детально описану архітектуру програмного забезпечення, наведена діаграма класів та детальний опис функціоналу застосунку.

Для розробки додатку було використано мову JavaScript з фреймворками ReactJS, ReactNative, Redux, Redux-Thunk. Для база даних було використано MongoDB. Для взаємодії з базою даних був створений серверний застосунок за допомогою мови NodeJS.

Наведена детальна інструкція користувача по експлуатації застосунку, описана методика проведення випробувань, яка показую можливість введення сервісу в експлуатацію.

ПЕРЕЛІК ВИКОРИСТАННИХ ДЖЕРЕЛ

1. MongoDb [Електронний ресурс] - Режим доступу до ресурсу: https://www.google.com.ua/search?q=mongodb&oq=%D1%8C%D1%89%D1%82&aqs=chrome.2.69i57j0l5.1810j0j1&sourceid=chrome&ie=UTF-8
2. [Електронний ресурс] - Режим доступу до ресурсу: https://corporatefinanceinstitute.com/resources/careers/soft-skills/time-management-list-tips/
3. ReactJS [Електронний ресурс] - Режим доступу до ресурсу: https://reactjs.org/
4. Redux [Електронний ресурс] - Режим доступу до ресурсу: https://redux.js.org/
5. NodeJs [Електронний ресурс] - Режим доступу до ресурсу: https://nodejs.org/uk/
6. [Електронний ресурс] - Режим доступу до ресурсу: https://habr.com/post/257005/
7. WebStorm [Електронний ресурс] - Режим доступу до ресурсу: https://www.jetbrains.com/webstorm/
8. Medium [Електронний ресурс] - Режим доступу до ресурсу: https://blog.cloudboost.io/getting-started-with-react-native-and-redux-6cd4addeb29
9. “Association Rule Mining” Zhang Chengqi, Zhang Shichao
10. “Association Rule Hiding for Data Mining” Gkoulalas-Divanis, Aris, Verykios, Vassilios S

Додаток А

Тексти програмного коду

Електронний щоденник студента для мобільної платформи

(Найменування програми (документа))

DVD-R

(Вид носія даних)

19 арк 327 Кб

(Обсяг програми (документа) , арк.,) Кб)

Київ – 2019 року

```
const express = require('express')
const bodyParser = require('body-parser')
const pino = require('express-pino-logger')()
const MongoClient = require('mongodb').MongoClient
const ObjectID = require('mongodb').ObjectID
const id = () => {
  return (
    '_' +
    Math.random()
      .toString(36)
      .substr(2, 9)
  )
}
const url = 'mongodb://localhost:27017/'
const mongoClient = Action MongoClient(url, { useActionUrlParser: true })
const app = express()
const jsonParser = express.json()
app.use(bodyParser.urlencoded({ extended: true }))
app.use(pino)
let dbClient
mongoClient.connect((err, client) => {
  if (err) return console.log(err)
  dbClient = client
  app.locals.actions = client.db('users_actions_bd').collection('actions')
  app.locals.users = client.db('users_actions_bd').collection('users')
  app.listen(3001, () =>
    console.log('Express server is running on localhost:3001')
  )
})
app.post('/api/session/log', jsonParser, (req, res) => {
  let { login, password } = req.body
  const users = req.app.locals.users
  users.findOne({ login: login, password: password }, (err, result) => {
    if (err) return console.log(err)
    if (result) res.json(result)
```

```
    else res.json(false)
  })
})
app.post('/api/session', jsonParser, (req, res) => {
  let ActionUser = Object.assign(req.body)
  let id = Action ObjectID(ActionUser._id)
  delete ActionUser._id
  const users = req.app.locals.users
  users.findOneAndUpdate(
    { _id: id },
    { $set: ActionUser },
    { returnOriginal: false },
    (err, result) => {
      if (err) return console.log(err)
      res.json(null)
    }
  )
})
app.get('/api/registration', (req, res) => {
  const users = req.app.locals.users
  users.find().toArray((err, result) => {
    if (err) return console.log(err)
    let ActionUser = Object.assign(result[0])
    delete ActionUser._id
    for (let arg in ActionUser) ActionUser[arg] = ''
    ActionUser.id = id()
    res.json(ActionUser)
  })
})

app.post('/api/registration', jsonParser, (req, res) => {
  const users = req.app.locals.users
  let user = Object.assign(req.body)
  users.insertOne(user, (err, result) => {
    if (err) console.log(err)
```

```
  })
})
app.get('/api/getactions', (req, res) => {
  res.setHeader('Content-Type', 'application/json')
  const actions = req.app.locals.actions
  actions.find().toArray((err, result) => {
    if (err) return console.log(err)
    res.json(result)
  })
})
app.post('/api/updateAction', jsonParser, (req, res) => {
  let ActionAction = Object.assign(req.body)
  let id = Action ObjectID(ActionAction._id)
  delete ActionAction._id
  const actions = req.app.locals.actions
  actions.findOneAndUpdate(
    { _id: id },
    { $set: ActionAction },
    { returnOriginal: false },
    (err, result) => {
      if (err) return console.log(err)
      res.json(null)
    }
  )

})

app.post('/api/addAction', jsonParser, (req, res) => {
  let Action = Object.assign(req.body)
  const actions = req.app.locals.actions
  actions.insertOne(Action, (err, result) => {
    if (err) return  console.log(err)
    res.json(result.ops[0])
  })
})
```

```
app.post('/api/deleteAction', jsonParser, (req, res) => {
  const actions = req.app.locals.actions
  let id = Action ObjectID(req.body.id)
  actions.deleteOne({_id : id}, (err, result) => {
    if (err) return console.log(err)
    res.json(null)
  })
})
process.on('SIGINT', () => {
  dbClient.close()
  process.exit()
})
import {AppRegistry} from 'react-native';
import App from './App'
import Home from './src/componenets/Home'
import List from './src/componenets/List'
import Form from './src/componenets/Form'
import Profile from './src/componenets/Profile'
import {name as appName} from './app.json';

AppRegistry.registerComponent(appName, () => Home);
import React, { useState } from 'react'
import New from '../components/New'
import { View, Text } from 'react-native'
import NewPage from '../styles/NewPage'
import { Button } from 'react-native-elements'

const monthNames = [
  'Январь',
  'Февраль',
  'Март',
  'Апрель',
  'Май',
  'Июнь',
  'Июль',
```

```
  'Август',
  'Сентябрь',
  'Октябрь',
  'Ноябрь',
  'Декабрь',
]

const NewsList = props => {
  const [NewAll, setNewAll] = useState(false)
  let { News, date } = props,
    s = 'Покзать основные'
  const dateString =
    '' +
    date.getDate() +
    ' ' +
    monthNames[date.getMonth()] +
    ' ' +
    date.getFullYear()
  if (!NewAll) {
    let cur = News.length - 2
    News = News.slice(0, 2)
    s = 'Еще ' + cur + ' сериалов '
  }
  let Newslist = News.map(elem => {
    return <New elem={elem} key={elem.id} openImage={props.openImage} />
  })
  return (
    <View transparent key={date.toISOString()}>
      <View style={NewPage.NewListTextView}>
        <Text style={NewPage.NewListText}> {dateString}</Text>
      </View>
      <View>{Newslist}</View>
      <Button
        onPress={() => {
          setNewAll(!NewAll)
```

```
        }}
        title={s}
        buttonStyle={NewPage.button}
        titleStyle={NewPage.buttonText}
      />
    </View>
  )
}

export default NewsList
          import React, { Component } from 'react'
import { View, Image, ScrollView } from 'react-native'
import ShowsList from '../components/ShowsList'
import Load from '../components/Load'
import ImageView from 'react-native-image-view'

class ShowsPage extends Component {
  static navigationOptions = {
    title: 'SUPER FILM',
    headerStyle: {
      backgroundColor: '#f5f5f5',
    },
    headerTitleStyle: {
      textAlign: 'center',
      marginTop: 0,
      marginBottom: 'auto',
      fontFamily: 'sans-serif',
      flex: 1,
    },
  }
  state = {
    isVisibleBigImage: false,
    images: [
      {
        source: require('../resources/not_found.png'),
```

```
      title: 'original image',
      width: 806,
      height: 720,
    },
  ],
}
openOriginalImage(uri) {
  let source = uri ? { uri: uri } : require('../resources/not_found.png')
  this.setState({
    isVisibleBigImage: true,
    images: [
      {
        source: source,
        title: 'original image',
        width: 806,
        height: 720,
      },
    ],
  })
}
render() {
  const { loading, schedule } = this.props
  if (schedule.length === 0) return <Load />
  return (
    <ScrollView>
      {schedule.map(elem => {
        return (
          <ShowsList
            shows={elem.shows}
            date={elem.date}
            key={elem.date}
            openImage={this.openOriginalImage.bind(this)}
          />
        )
      })}
```

```
        <View>{loading ? <Load /> : null}</View>
        <ImageView
          images={this.state.images}
          isVisible={this.state.isVisibleBigImage}
        />
      </ScrollView>
    )
  }
}
export default ShowsPage
import React, { Component } from 'react'
import CustomCalendar from './CustomCalendar'
import homeStyles from '../styles/home'
import { View, Text, Image } from 'react-native'

class Home extends Component {
  static navigationOptions = {
    title: 'SUPER FILM',
    headerStyle: {
      backgroundColor: '#f5f5f5',
    },
    headerTitleStyle: {
      textAlign: 'center',
      marginTop: 0,
      marginBottom: 'auto',
      fontFamily: 'sans-serif',
      flex: 1,
    },
  }
  render() {
    const { navigate } = this.props.navigation
    return (
      <View style={homeStyles.container}>
        <View style={homeStyles.imageView}>
          <Image
```

```
                style={homeStyles.image}
                source={require('../resources/Icon_tele.png')}
              />
            </View>
            <View style={homeStyles.textView}>
              <Text style={homeStyles.text}>
              </Text>
            </View>
            <View style={homeStyles.calendarView}>
              <CustomCalendar navigate={navigate} addShows={this.props.addShows} clear =
{this.props.clear}/>
            </View>
          </View>
        )
      }
    }
    export default Home
    import { StyleSheet } from 'react-native'

    const homeStyles = StyleSheet.create({
      container: {
        flex: 1,
        justifyContent: 'space-evenly',
        flexDirection: 'column',
      },
      text: {
        fontFamily: 'sans-serif',
        textAlign: 'center',
        fontWeight: '500',
        color: '#333333',
        flex: 23 / 35,
      },
      textView: {
        flex: 1,
        flexDirection: 'row',
```

```
    justifyContent: 'center',
    alignItems: 'center',
  },
  imageView: {
    flex: 2,
    flexDirection: 'row',
    justifyContent: 'center',
  },
  image: {
    flex: 3 / 7,
    marginHorizontal: 'auto',
    resizeMode: 'contain',
  },
  calendarView: {
    flex: 3,
    justifyContent: 'flex-end',
  },
})
export default homeStyles
import { StyleSheet } from 'react-native'
const showPage = StyleSheet.create({
  showListText: {
    fontFamily: 'sans-serif',
    textAlign: 'center',
    fontWeight: '500',
    color: '#333333',
    margin: 15,
    fontSize: 15,
  },
  showListTextView: {
    borderBottomColor: '#e9e9e9',
    borderBottomWidth: 2,
    marginBottom: 20,
  },
  button: {
```

```
  backgroundColor: '#ffffff',
  marginHorizontal: 20,
  marginVertical: 15,
  borderColor: '#e9e9e9',
  borderWidth: 2,
  borderRadius: 5,
},

buttonText: {
  fontFamily: 'sans-serif',
  textAlign: 'center',
  fontWeight: '700',
  color: '#e9e9e9',
  fontSize: 10,
},

image: {
  width: 80,
  height: 130,
  borderRadius: 2,
},
showView: {
  marginHorizontal: 20,
  marginVertical: 15,
  flexDirection: 'row',
},
nameText: {
  fontFamily: 'sans-serif',
  textAlign: 'left',
  fontWeight: '700',
  color: '#333333',
  fontSize: 15,
  maxWidth: 200,
},
textView: {
```

```
    marginHorizontal: 15,
  },
  textPremiered: {
    fontFamily: 'sans-serif',
    fontWeight: '900',
    color: '#e9e9e9',
    fontSize: 17,
  },
  episodView: {
    backgroundColor: '#f5f5f5',
    flexDirection: 'row',
    marginTop: 35,
    marginBottom: 10,
    borderRadius: 2,
    maxWidth: 400,
  },
  episodText: {
    fontFamily: 'sans-serif',
    margin: 5,
  },
})
export default showPage
import React, { useState } from 'react'
import { Button, Form } from 'react-bootstrap'
import '../style/article.css'
const ArticleForm = props => {
  const [header, setHeader] = useState(props.header)
  const [body, setBody] = useState(props.body
  const handleSubmit = e => {
    e.preventDefault()
    props.send(header, body)
    setHeader('')
    setBody('')
  }
```

```
        return (
          <Form onSubmit={handleSubmit} className={props.styleClass}>
            <Form.Group>
              <Form.Label>Header</Form.Label>
              <Form.Control
                type={'text'}
                placeholder={'header'}
                value={header}
                data-field-name={'header'}
                onChange={e => {
                  setHeader(e.target.value)
                }}
              />
            </Form.Group>
            <Form.Group>
              <Form.Label>Body</Form.Label>
              <Form.Control
                as={'textarea'}
                rows={'5'}
                placeholder={'body'}
                value={body}
                data-field-name={'body'}
                onChange={e => {
                  setBody(e.target.value)
                }}
              />
            </Form.Group>
            <Button type={'submit'}>{props.submitBut}</Button>
          </Form>
        )
      }
      export default ArticleForm
export const SET_ ACTIONS = 'SET_ ACTIONS',
  DELETE_ ACTION = 'DELETE_ ACTION ',
  ADD_ ACTION = 'ADD_ ACTION ',
```

```
  UPDATE_ ACTION = 'UPDATE_ACTION
export function setNews() {
  return dispatch => {
    let request = new XMLHttpRequest()
    request.open('GET', '/api/getAction', true)
    request.setRequestHeader('Content-Type', 'application/json')
    request.addEventListener('load', () => {
      let news = JSON.parse(request.response)
      dispatch({ type: SET_ACTIONS, news: news })
    })
    request.send()
  }
}
export function updateNew(myNew) {
  return dispatch => {
    let jsonNew = JSON.stringify(myNew)
    let request = new XMLHttpRequest()
    request.open('POST', 'api/updateNew', true)
    request.setRequestHeader('Content-Type','application/json')
    request.addEventListener('load', () => {
      dispatch({ type: UPDATE_ACTION, myNew: myNew })
    })
    request.send(jsonNew)
  }
}

export function deleteNew(id) {
  return dispatch => {
```

```
    let jsonId = JSON.stringify({id : id})
    let request = new XMLHttpRequest()
    request.open('POST', 'api/deleteNew', true)
    request.setRequestHeader('Content-Type', 'application/json')
    request.addEventListener('load', () => {
      dispatch({ type: DELETE_NEW, id: id })
    })
    request.send(jsonId)
  }
}
export function addAction(myNew) {
  return dispatch =>{
    let jsonNew = JSON.stringify(myNew)
    let request = new XMLHttpRequest()
    request.open('POST', '/api/addNew', true)
    request.setRequestHeader('Content-Type', 'application/json')
    request.addEventListener('load', () => {
      let addNew = JSON.parse(request.response)
      dispatch({type: ADD_ACTION, myNew: addNew})
    })
    request.send(jsonNew)
  }
}
export const LOG_IN = 'LOG_IN',
  LOG_OUT = 'LOG_OUT',
  SESSION_ERROR = 'SESSION_ERROR',
  UPDATE_USER = 'UPDATE_USER',
  SET_REGISTRATION = 'SET_REGISTRATION'
```

```
export function LogIn(params) {
  return dispatch => {
    let jsonLog = JSON.stringify(params)
    let request = new XMLHttpRequest()
    request.open('POST', '/api/session/log', true)
    request.setRequestHeader('Content-Type', 'application/json')
    request.addEventListener('load', () => {
      let user = JSON.parse(request.response)
      if (user) {
        dispatch({ type: LOG_IN, user: user })
      } else {
        dispatch({ type: SESSION_ERROR })
      }
    })
    request.send(jsonLog)
  }
}



export function updateUser(user) {
  return dispatch => {
    let jsonUser = JSON.stringify(user)
    let request = new XMLHttpRequest()
    request.open('POST', '/api/session', true)
    request.setRequestHeader('Content-Type', 'application/json')
    request.addEventListener('load', () => {
      dispatch({ type: UPDATE_USER, user: user })
```

```
    })
    request.send(jsonUser)
  }
}
export function setRegisterArgs() {
  return dispatch => {
    let request = new XMLHttpRequest()
    request.open('GET', '/api/registration', true)
    request.setRequestHeader('Content-Type', 'application/json')
    request.addEventListener('load', () => {
      let user = JSON.parse(request.response)
      dispatch({ type: SET_REGISTRATION, registration: user })
    })
    request.send()
  }
}
export function LogOut() {
  return { type: LOG_OUT }
}
import {
  LOG_OUT,
  LOG_IN,
  SESSION_ERROR,
  UPDATE_USER,
  SET_REGISTRATION,
} from '../actions/session'
const InitialState = {
  user: null,
```

```
  registration: null
}
export default (state = InitialState, action) => {
  switch (action.type) {
    case LOG_IN: {
      let user = Object.assign({}, action.user)
      return { ...state, user: user }
    }
    case LOG_OUT:
      return { ...state, user: null }
    case SESSION_ERROR: {
      alert('Invalid Login and Password')
      return { ...state }
    }
    case UPDATE_USER: {
      let user = Object.assign({}, action.user)
      return { ...state, user: user }
    }
    case SET_REGISTRATION : {
      return {...state, registration : action.registration}
    }
    default:
      return state
  }
}
import { SET_actions, DELETE_action, ADD_action, UPDATE_action } from
'../actions/actions'


const InitialState = {
```

```
  actions: [],
  setactions: false,
}

export default (state = InitialState, action) => {
  switch (action.type) {
    case SET_actions: {
      return { ...state, actions: action.actions, setactions: true }
    }
    case DELETE_action: {
      let arr = state.actions.filter(item => {
        return item._id !== action.id
      })
      return { ...state, actions: arr }
    }
    case ADD_action: {
      return { ...state, actions: [...state.actions, action.myaction] }
    }
    case UPDATE_action: {
      let arr = state.actions.filter(item => {
        return item._id !== action.myaction._id
      })
      return { ...state, actions: [...arr, action.myaction] }
    }
    default:
      return state
  }
}
```

ЗМІСТ

1 ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1 Повне найменування системи та її умовне позначення

Повна назва системи: «Електронний щоденник студента для мобільної платформи».

1.2 Найменування організації-замовника та організацій-учасника робіт

Генеральним замовником проекту являється кафедра Автоматизованих систем обробки інформації та управління НТУУ "КПІ". Представником замовника Тєлишева Тамара Олексіївна.

Розробником системи є студент групи ІС-51 факультету інформатики та обчислювальної техніки НТУУ «КПІ» Єрьоменко Денис Валентинвоич.

1.3 Перелік документів, на підставі яких створюється система

При розробці системи і створення проектно-експлуатаційної документації Виконавець повинен керуватися вимогами наступних нормативних документів:

- ДСТУ 19.201-78. Технічне завдання. Вимоги до змісту і оформлення;
- ДСТУ 34.601-90. Комплекс стандартів на автоматизовані системи. Автоматизовані системи. Стадії створення;
- ДСТУ 34.201-89. Інформаційні технології. Комплекс стандартів на автоматизовані системи. Види, комплексність і позначення документів при створенні автоматизованих систем.

1.4 Планові терміни початку і закінчення роботи зі створення системи

Плановий термін початку роботи над створенням електронного щоденника студента – 5 лютого 2019 рік.

Плановий термін по закінченню роботи над створенням електронного щоденника студента – не пізніше 1 червня 2019 року.

2 ПРИЗНАЧЕННЯ І МЕТА СИСТЕМИ

2.1 Призначення системи

Призначенням інформаційної системи є створення додатку, який полегшить управління часом у рамках процесу навчання, як для студентів, так і для викладачів. Розробити зрозумілий та простий інтерфейс.

2.2 Цілі та задачі системи

Цілями дипломного проекту є

- спростити комунікацію між викладачем та студентом;
- допомогти викладачам та студент з управлінням часу;
- автоматизувати процес переносу подій у календарі.

Для досягнення поставленої мети мають бути вирішені такі задачі та реалізовані функції:

- реєстрація користувачів;
- додавання подій користувачами;
- формування розпорядку дня на основі заданих подій.

3 ХАРАКТЕРИСТИКА ОБ'ЄКТА АВТОМАТИЗАЦІЇ

Для користування сервісом обов'язковою умовою для користувача є наявність смартфону.

Працювати з сервісом може лише авторизований користувач. Логіни і паролі видаються особисто. Лише студентам та викладачам університету.

Авторизований користувач може створювати свій розпорядок дня. Викладач може додавати до розпорядків студентів консультації перездачі та додаткові пари.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

Система має здійснювати класифікацію подій за вказаними користувачем критеріями, та повертати результат у вигляді сформованого розпорядку дня. Система має виконувати наступні функції:

- робота з обліковими записами;
- додання подій до розпорядку користувача;
- змінення розпорядку у разі конфлікту двох подій;
- оповіщення про неможливість зміни розпорядку;
- формування розпорядку за налаштуваннями системи;
- додавання викладачем подій до розпорядку студентів;
- змінення особистих налаштувань.

4.2 Вимоги до надійності

Програма повинна зберігати працездатність і забезпечувати відновлення своїх функцій при виникненні наступних позаштатних ситуацій:

- при помилках в роботі апаратних засобів (крім носіїв даних і програм). Відновлення функції програми покладається на хостинг;
- при помилках, пов’язаних з особливістю різноманітних смартфонів, існуючих на сьогоднішній день.

Програмний продукт повинен поєднувати надійність та функціональність.

У разі виникнення аварійних ситуацій необхідно сповіщати користувача та надавати інструкцію для подальших дій. Будь-які аварійні ситуації мають бути задокументовані у звіті, який при необхідності надсилається розробнику для визначення причини збою в роботі та

усуненні помилок, які могли привести до нестабільної роботи програмного продукту.

4.3 Вимоги до складу і параметрів технічних засобів

Склад, структура і способи організації даних в системі повинні бути визначені на етапі технічного проектування.

Структура технічних засобів визначається виходячи із можливості їх забезпечити виконання встановлених операцій процесу технічного обслуговування.

Для правильної роботи розробленої системи до складу технічних засобів повинен входити смартфон, що має конфігурацію наведену нижче:

- процесор з тактовою частотою не нижче 1,0 ГГц;
- об’єм оперативної пам’яті не менше 512 МБ;
- об’єм пам’яті носія не нижчий за 2Гб;
- доступ до мережі інтернет.

5 СТАДІЇ І ЕТАПИ РОЗРОБКИ

Основні етапи виконання робіт з розробки електронного щоденника студента.

№ п/п	Назва етапу роботи	Термін виконання етапу	Результат виконання
1.	Вивчення рекомендованої літератури	07.02.2019	
2.	Аналіз існуючих методів розв’язання задачі	12.02.2019	
3.	Постановка та формалізація задачі	20.02.2019	
4.	Розробка інформаційного забезпечення	02.03.2019	
5.	Алгоритмізація задачі	17.03.2019	
6.	Обґрунтування використовуваних технічних засобів	25.03.2019	
7.	Розробка програмного забезпечення	29.03.2019	
8.	Налагодження програми	13.04.2019	
9.	Виконання графічних документів	27.04.2019	
10.	Оформлювання пояснювальної записки	12.05.2019	
11.	Подання ДП на попередній захист	22.05.2019	
12.	Подання ДП на основний захист	01.06.2019	
13.	Подання ДП рецензенту	05.06.2019	

6 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ СИСТЕМИ

6.1 Види випробувань

Для контролю правильності роботи програмного забезпечення буде проведено функціональне тестування. В ході тестування буде проведено випробування основних функціональних характеристик системи та цілої системи загалом.

Тестування рішення полягає в написанні тестів, які перевіряють відповідність рекомендованих даних з вхідними даними.

Тестування інтерфейсу програми полягає в тому, що користувачу надається сторінка з відповідною формою, в яку він вводить дані.

ЗМІСТ

1 ОБ'ЄКТ ВИПРОБУВАННЯ

1.1 Найменування програми

Повне найменування – Електронний щоденник студента для мобільної платформи.

1.2 Область застосування

Застосунок використовується для автоматизації ведення розпорядку дня, його створення та змінення.

1.3 Умовне позначення програми

Умовне позначення програми – OATests.

2 МЕТА ВИПРОБУВАНЬ

Метою випробувань є перевірка відповідності створеного програмного продукту вказаним вимогам до функціональності; перевірка коректності поведінки продукту.

3 ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

3.1 Вимоги до функціональних характеристик

Мають бути задоволені наступні функціональні вимоги:

- робота з обліковими записами;
- додання подій до розпорядку користувача;
- змінення розпорядку у разі конфлікту двох подій;
- оповіщення про неможливість зміни розпорядку;
- формування розпорядку за налаштуваннями системи;
- додавання викладачем подій до розпорядку студентів;
- змінення особистих налаштувань.

3.1.1 Вимоги до складу виконуваних функцій

Додаток має задовольняти наступним вимогам:

- має бути можливість управління складовими системи тестування;
- система повинна автоматично перевіряти надані відповіді;
- має бути можливість перегляду перевірених результатів.

4 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

До складу програмної документації повинні входити керівництво користувача та вихідні коди програмного продукту.

5 СКЛАД І ПОРЯДОК ВИПРОБУВАНЬ

Випробування поділяються на два етапи: ознайомчий і виконавчий.

Під час ознайомчого етапу проводиться:

- перевірка повноти програмної документації;
- перевірка повноти наявних технічних і програмних засобів.

На виконавчому етапі проводиться:

- перевірка відповідності технічних характеристик системи;
- перевірка виконання функціональних вимог до системи.

Функції, що підлягають перевірці:

- проходження авторизації;
- змінення особистих даних;
- перегляд подій по обраній даті;
- додавання події;
- додавання події, що накладається з уже існуючою;
- додавання події без можливості переносу;
- видалення події;
- додавання події групі користувачів;
- додавання події іншому користувачу;
- додавання події користувачам по тегу.

Під час випробувань проводиться функціональне тестування, відповідно до вимог, зазначених у технічному завданні.

Кроки, вхідні дані, очікувані результати та результати випробувань для кожного з тестів наведені далі у таблицях.

6 МЕТОДИ ВИПРОБУВАНЬ

6.1 Функціональне тестування

У процесі тестування була перевірена основна функціональність системи. Перелік випробувань основних функціональних можливостей наведений у таблицях 6.1 – 6.10

Таблиця 6.1 – Тест вдалого проходження авторизації

Мета тесту	Перевірити можливість авторизації
Початковий стан	Відкритий екран авторизації
Вхідні дані	Логін та пароль користувача
Схема проведення тесту	Заповнити форму авторизації
Очікуваний результат	Відкрите вікно авторизованого користувача з календарем
Стан після проведення випробувань	Користувач авторизований у системі

Таблиця 6.2 – Тест змінення особистих даних

Мета тесту	Перевірити можливість зміни особистих даних
Початковий стан	Відкритий екран особистих налаштувань
Вхідні дані	Особисті дані у вигляді ім’я, фамілія, поштова скринька, номер мобільного телефону тощо
Схема проведення тесту	Заповнити форму особистих даних
Очікуваний результат	Відкрите початкове вікно користувача з календарем

Стан після проведення випробувань	Змінені особисті дані користувача

Таблиця 6.3 – Тест вдалого перегляду подій по обраній даті

Мета тесту	Перевірити можливість перегляду подій по даті
Початковий стан	Відкритий початковий екран користувача з календарем
Вхідні дані	Дата пізніше заданої
Схема проведення тесту	Обрати дату у календарі
Очікуваний результат	Відкрите вікно з лістингом подій по заданій даті, відсортованих по часу
Стан після проведення випробувань	Відкрите вікно перегляду подій

Таблиця 6.4 – Тест вдалого додавання події

Мета тесту	Перевірити правильність роботи додавання події
Початковий стан	Відкритий екран форми додавання події
Вхідні дані	Параметри події: назва, дата, час, тип, можливість переносу, опис
Схема проведення тесту	Заповнити форму додавання події
Очікуваний результат	Відкрите вікно з лістингом подій, по тій даті, до якої було додано подію
Стан після проведення випробувань	По заданій даті додано подію

Таблиця 6.5 – Тест додавання події, що накладається з уже існуючою

Мета тесту	Перевірити функціонал системи з переносу подій
Початковий стан	Відкритий екран додавання події. У

	системі вже є подія з такою ж датою та часом

Продовження **таблиці 6.5**

Мета тесту	Перевірити функціонал системи з переносу подій
Вхідні дані	Параметри події: назва, дата, час, важливість, можливість переносу, опис
Схема проведення тесту	Заповнити форму додавання події. Натиснути кнопку додавання. З’явиться вікно з пропозицією переносу тієї з двох подій, у якої найменший пріоритет, поле з часом та датою редагується, але є запропонований варіант. Обрати підходящу дату. Натиснути кнопку погодження
Очікуваний результат	Відкрите вікно з лістингом подій, по тій даті, до якої було додано подію
Стан після проведення випробувань	Розпорядок змінений у системі.

Таблиця 6.6 – Тест додавання події без можливості переносу

Мета тесту	Перевірити правильність роботи системи з конфліктами подій
Початковий стан	Відкритий екран додавання події
Вхідні дані	Параметри події: назва, дата, час, важливість, можливість переносу(Ні), опис
Схема проведення тесту	Заповнити форму додавання події. З’явиться вікно про неможливість додання такої події, з проханням змінити параметри

Продовження **таблиці 6.6**

Мета тесту	Перевірити правильність роботи системи з конфліктами подій
Очікуваний результат	Відкритий екран додавання події. Параметри дата та можливість переносу підкреслені червоним
Стан після проведення випробувань	Система залишилась без змін

Таблиця 6.7 – Тест видалення події

Мета тесту	Перевірити можливість видалення події
Початковий стан	Відкритий екран з лістингом подій
Вхідні дані	Подія, яку хочемо видалити
Схема проведення тесту	Довге натискання на заданій події. З’явиться вікно погодження. Натискаємо “Так”
Очікуваний результат	Відкритий екран з лістингом подій, без події, яку щойно видалили
Стан після проведення випробувань	Система залишилась без заданої події

Таблиця 6.8 – Тест додавання події групі користувачів

Мета тесту	Перевірити можливість додавання події іншому користувачу по групі
Початковий стан	Відкрита домашня сторінка з календарем у профілі викладача
Вхідні дані	Подія, яку хочемо додати та група,

	якій хочемо додати

Продовження **таблиці 6.8**

Мета тесту	Перевірити можливість додавання події іншому користувачу по групі
Схема проведення тесту	Натискання на кнопку додавання події студентам, введення літералу групи, заповнення форми події, натискання на кнопку відправлення
Очікуваний результат	Оповіщення про успішне додання події
Стан після проведення випробувань	У всіх користувачів, які входять у дану групу з’являється задана подія

Таблиця 6.9 – Тест додавання події іншому користувачу

Мета тесту	Перевірити можливість додавання події іншому користувачу по імені
Початковий стан	Відкрита домашня сторінка з календарем у профілі викладача
Вхідні дані	Подія, яку хочемо додати та користувач, якому хочемо додати
Схема проведення тесту	Натискання на кнопку додавання події студентам, введення П.І.Б студента, заповнення форми події, натискання на кнопку відправлення
Очікуваний результат	Оповіщення про успішне додання події
Стан після проведення випробувань	У користувача, якого ми указали в спливаючому вікні додана задана

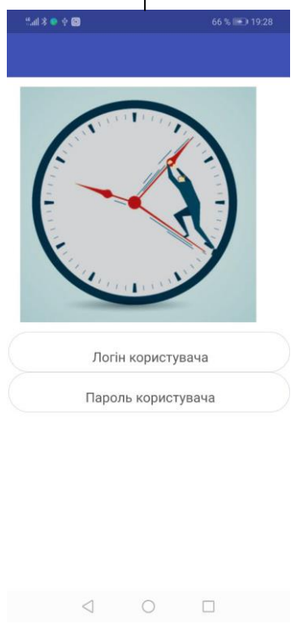
	подія
Мета тесту	Перевірити можливість додавання події іншому користувачу по тегу
Стан після проведення випробувань	У всіх користувачів, які входять у дану групу з’являється задана подія

Таблиця 6.10 – Тест додавання події користувачам по тегу

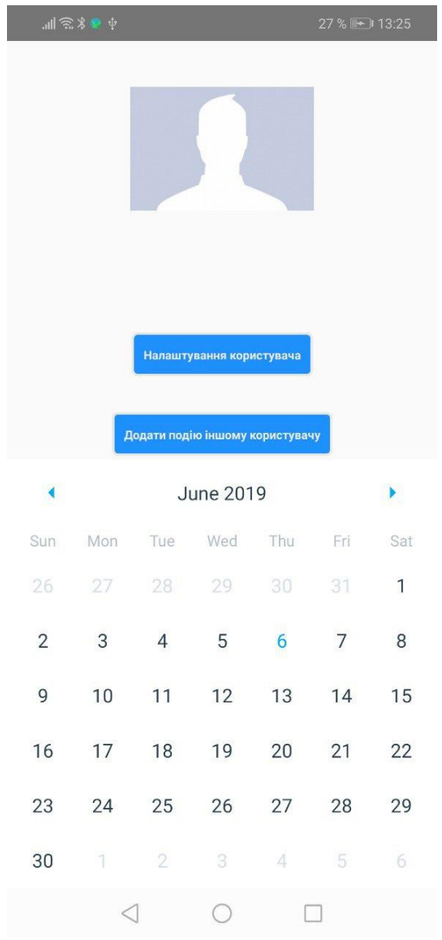
Мета тесту	Перевірити можливість додавання події іншому користувачу по тегу
Початковий стан	Відкрита домашня сторінка з календарем у профілі викладача
Вхідні дані	Подія, яку хочемо додати та тег, по якому хочемо додати
Схема проведення тесту	Натискання на кнопку додавання події студентам, введення тегу, заповнення форми події, натискання на кнопку відправлення
Очікуваний результат	Оповіщення про успішне додання події
Стан після проведення випробувань	У всіх користувачів, які входять у дану групу з’являється задана подія

ДП ІС-5108.1181-с.КЕ

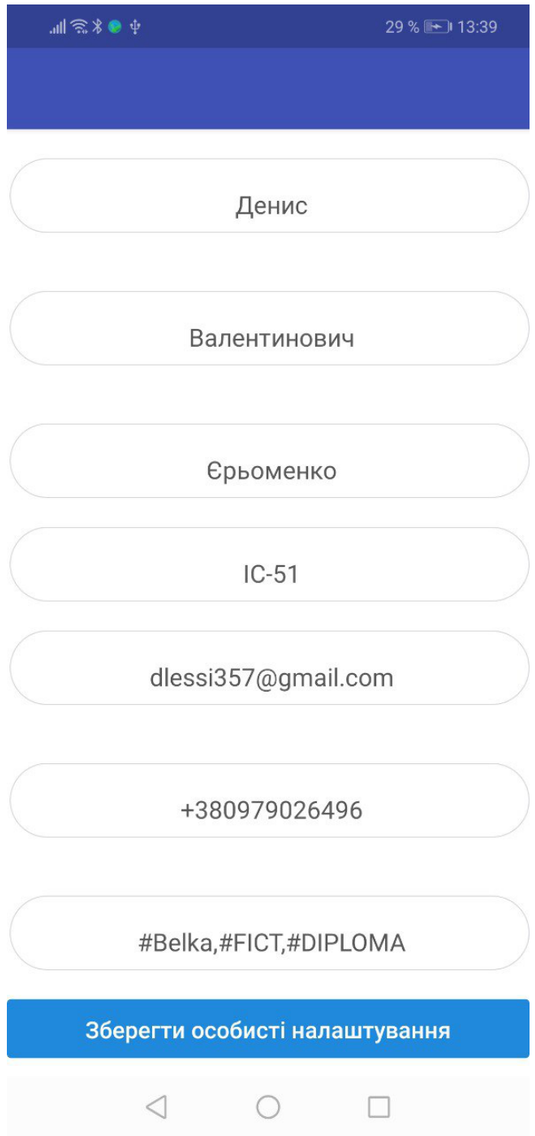
Екран аворизації



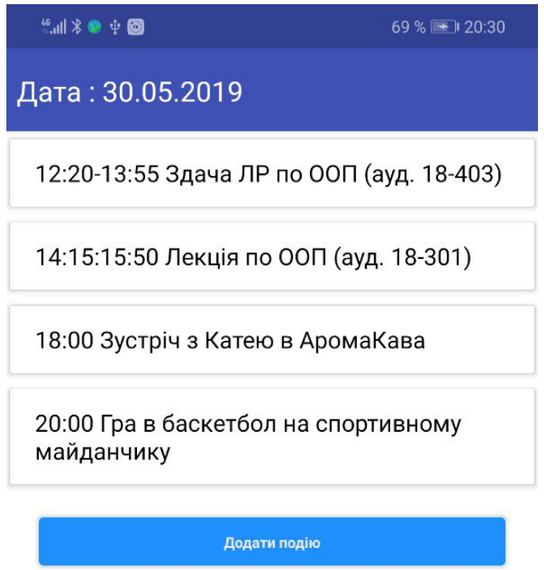
Екран профіля



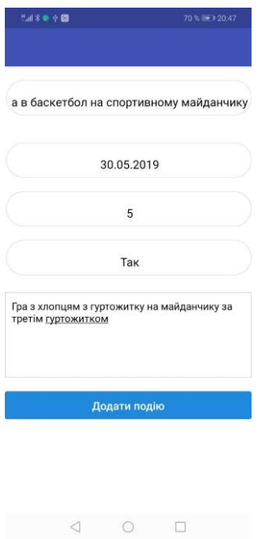
Екран особистих налаштувань



Екран переліку подій по заданій даті



Форма нової події



					ДП ІС-5108.1181-с.КЕ		
					Креслення виду екранних форм	Літера	Маса / Масштаб
Зм.	Арк.	№ документа	Підпис	Дата			
Розробив		Єрьоменко Д.В,				Аркуш 1	Аркушів 1
Перевірив		Тєлишева Т.О.					
Т. кон.					Електронний щоденник студента для мобільної платформи	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-51	
Н. кон.		Тєлишева Т.О.					
Затвердив		Тєлишева Т.О.					

					ДП ІС-5108.1181-с.ССБ		
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна бізнес-процесів для додавання події	Літера	Маса / Масштаб
Розробив		Єрьоменко Д.В.					
Перевірив		Тєлишева Т.О.				Аркуш 1	Аркушів 1
Т. кон.							
Н. кон.		Тєлишева Т.О.			Електронний щоденник студента для мобільної платформи	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-51	
Затвердив		Тєлишева Т.О.					

					ДП ІС-5108.1181-с.ССБ		
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна бізнес-процесів комунікації між студентом та викладачем	Літера	Маса / Масштаб
Розробив		Єрьоменко Д.В.					
Перевірив		Тєлишева Т.О.				Аркуш 1	Аркушів 1
Т. кон.							
Н. кон.		Тєлишева Т.О.			Електронний щоденник студента для мобільної платформи	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-51	
Затвердив		Тєлишева Т.О.					

ДП ІС-5108.1181-с.ССК

					ДП ІС-5108.1181-с.ССК	
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна класів програмного забезпечення	Літера / Маса / Масштаб
Розробив		Єрьоменко Д.В.				
Перевірив		Тєлишева Т.О.				Аркуш 1 / Аркушів 1
Т. кон.						
Н. кон.		Тєлишева Т.О.			Електронний щоденник студента для мобільної платформи	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-51
Затвердив		Тєлишева Т.О.				

					ДП ІС-5108.1181-с.ССФ		
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна функціональна IDEF0	Літера	Маса / Масштаб
Розробив		Єрьоменко Д.В.					
Перевірив		Тєлишева Т.О.				Аркуш 1	Аркушів 1
Т. кон.							
Н. кон.		Тєлишева Т.О.			Електронний щоденник студента для мобільної платформи	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-51	
Затвердив		Тєлишева Т.О.					

ДП ІС-5108.1181-с.ССВ

					ДП ІС-5108.1181-с.ССВ	
Зм.	Арк.	№ документа	Підпис	Дата	*Структурна схема варіантів використання*	Літера / Маса / Масштаб
Розробив		Єрьоменко Д.В.				
Перевірив		Тєлишева Т.О.				Аркуш 1 / Аркушів 1
Т. кон.						
Н. кон.		Тєлишева Т.О.			*Електронний щоденник студента для мобільної платформи*	*КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-51*
Затвердив		Тєлишева Т.О.				

Рішення з математичного забезпечення

Математична модель

Алгоритм Apriori

На першому кроці алгоритму підраховуються 1-елементні набори, які часто зустрічаються. Для цього необхідно пройтись по всьому набору даних і підрахувати для них підтримку, тобто скільки раз зустрічається в базі.

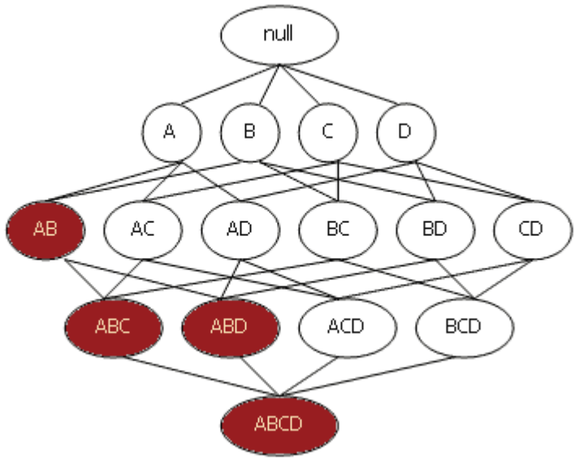
Для того, щоб отримати набори з k елементів , скористаємося наборами з (k-1) елементів, які були визначені на попередньому етапі та часто зустрічаються.

Генерація кандидатів буде складатися з двох кроків:

1) формуватися шляхом розширення набору розміром (k-1), який часто зустрічається, додаванням елемента з іншого набору (k-1) елементів

2) видалення надлишкових правил. На підставі властивості анти-монотонності

Після генерації кандидатів наступним завданням є підрахунок підтримки для кожного кандидата. Ми будемо використовувати підхід, заснований на зберіганні кандидатів в хеш-дереві. Внутрішні вузли дерева містять хеш-таблиці з покажчиками на нащадків, а листя - на кандидатів. Це дерево нам стане в нагоді для швидкого підрахунку підтримки для кандидатів.



Створення правил - менш складне завдання. По-перше, для підрахунку достовірності правила досить знати підтримку самого набору і множини, що лежить в умові правила.

$$\sup(A) = \frac{number\ of\ transactions\ comtainingA \cup B}{number\ of\ transactions} \quad (1)$$

$$conf(A) = \frac{number\ if\ transactions\ comtainingA \cup B}{number\ of\ transactions\ comtaining\ A} \quad (2)$$

Щоб створити правило з часто набору F, слід знайти всі його непусті підмножини. І для кожної підмножини s ми зможемо сформулювати правило s ⇒ (F - s), якщо достовірність правила conf (s ⇒ (F - s)) = sup (F) / sup (s) незгірш від порога minconf.

Усі супермножини даної множини мають меншу або рівну підтримку і, відповідно, більше значення достовірності. Ця властивість може бути використано при створені правил. Якщо ми почнемо витягувати правила, розглядаючи спочатку тільки один елемент в умові правила, і це правило має необхідну підтримку, тоді всі правила, де в умови стоять супермножини цього елемента, також мають значення достовірності вище заданого порогу.

Наприклад, якщо правило A ⇒ BCDE задовольняє мінімального порогу достовірності minconf, тоді AB ⇒ CDE також задовольняє.

Демонстраційний плакат до дипломного проекту

«Електронний щоденник студента для мобільної платформи»

Виконав студент гр. ІС-51 Єрьоменко Д.В.

Керівник ДП Тєлишева Т.О.