МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій

«До захисту допущено»
В.о. завідувача кафедри

_____ Світлана ВАЩЕНКО

____ _____ 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня магістр

зі спеціальності 122 «Комп'ютерні науки»,

освітньо-професійної програми «Інформаційні технології проєктування»

на тему: Мобільний додаток автоматизації процесу підготовлення до отримання водійського посвідчення

Здобувача (ки) групи ІТ.м-22 Омельченка Євгенія Олександровича
(шифр групи) (прізвище, ім'я, по батькові)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ Євгеній ОМЕЛЬЧЕНКО
(підпис) (Ім'я та ПРІЗВИЩЕ здобувача)

Керівник к.т.н., доц. Вікторія АНТИПЕНКО _____
(посада, науковий ступінь, вчене звання, Ім'я та ПРІЗВИЩЕ) (підпис)

Суми-2023

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Інформаційні технології проєктування»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри ІТ

Світлана ВАЩЕНКО
«_____» _____ 2023 р.

ЗАВДАННЯ

на кваліфікаційну роботу магістра студентові

Омельченку Євгенію Олександровичу
(прізвище, ім'я, по батькові)

1 Тема кваліфікаційної роботи Мобільний додаток автоматизації процесу підготовлення до отримання водійського посвідчення

затверджена наказом по університету від «08» листопада 2023 р. № 1249-VI

2 Термін здачі студентом кваліфікаційної роботи «___» ___грудня___ 2023 р.

3 Вхідні дані до кваліфікаційної роботи план робіт, технічна документація системи Android, технічна документація SQLite

4 Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) аналіз предметної області, постановка задачі, методи дослідження, моделювання мобільного додатку, розробка мобільного додатку, тестування мобільного додатку.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових слайдів презентації) актуальність, мета та задачі дослідження, огляд існуючих аналогів, результати проведеного аналізу аналогів, функціональні вимоги, засоби реалізації, структурно-функціональне моделювання, моделювання варіантів використання, фізична модель бази даних, програмна реалізація, демонстрація роботи мобільного додатку, тестування, висновок.

6. Консультанти випускної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

Дата видачі завдання _____.

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Підготовка специфікації	15.09.23 – 20.09.23	
2	Пошук інформації	10.10.23 – 29.10.23	
3	Систематизація інформації	17.10.23 – 29.10.23	
4	Розроблення теоретичної частини роботи	24.10.23 – 31.10.23	
7	Визначення засобів реалізації	31.10.23 – 02.11.23	
10	Проєктування мобільного додатку	02.11.23 – 10.11.23	
11	Розробка мобільного додатку	11.11.23 – 22.11.23	
12	Тестування мобільного додатку	26.11.23 – 30.11.23	

Магістрант _____ Євгеній ОМЕЛЬЧЕНКО

Керівник роботи _____ к.т.н., доц. Вікторія АНТИПЕНКО

АНОТАЦІЯ

Темою кваліфікаційної роботи магістра є «Мобільний додаток автоматизації процесу підготовлення до отримання водійського посвідчення».

Пояснювальна записка складається зі вступу, 4 розділів, висновку, списку використаних джерел із 30 найменувань, 2 додатки. Загальний обсяг роботи – 100 сторінки, у тому числі 58 сторінок основного тексту, 3 сторінки списку використаних джерел, 39 сторінок додатків.

Актуальність роботи полягає у зменшенні витрати часу на вивчення правил дорожнього руху та подальшого запису на складання теоретичного іспиту.

Мета роботи: Розробка мобільного додатку автоматизації процесу підготовлення до отримання водійського посвідчення. Даний програмний продукт повинен відповідати всім зазначеним вимогам та виконувати поставлені завдання.

У результаті виконання кваліфікаційної роботи було розроблено мобільний додаток автоматизації процесу підготовлення до отримання водійського посвідчення. Використання створеного програмного продукту дозволить зменшити час на вивчення правил дорожнього руху та зусилля для запису на складання теоретичного іспиту, що, у свою чергу, розгрузить сервісні центри.

Ключові слова: МОБІЛЬНИЙ ДОДАТОК, ВОДІЙСЬКЕ ПОСВІДЧЕННЯ, ТЕСТУВАННЯ, ПРАВИЛА ДОРОЖНЬОГО РУХУ, KOTLIN, ANDROID STUDIO ENTITY, DAO, JETPACK COMPOSER.

ЗМІСТ

ВСТУП

Актуальність. Наразі питання безпекової ситуації на дорогах є актуальним через великий попит на особисті транспортні засоби серед нашого населення. Особливо це стало нагальним під час повномасштабного вторгнення в Україну, коли кожен зрозумів, що за таких обставин у сім'ї повинно бути хоча б одне авто та водій. Як наслідок, збільшилася кількість охочих отримати право на керування транспортними засобами. А, отже, відбувся великий наплив людей у автошколи. Останнні були переповнені, а їхні працівники – перевантажені. Виправленню даної ситуації посприяв нещодавно прийнятий закон в Україні, згідно якому учню дозволено самостійно вивчати правила дорожнього руху (ПДР) і потім складати теоретичний іспит без обов'язкової умови відвідування автошколи. Особливо це зацікавало людей у віддалених районах. Найближча автошкола може знаходитися за багато кілометрів, а відвідувати її кожного дня можливості немає. Але також така зміна правил щодо отримання водійського посвідчення стала доречною для зайнятих людей, які можуть виділяти тільки певний час на навчання. Тому, ураховуючи все вищезазначене, можна зробити висновок, що наразі є потреба у програмного продукті, який би дозволяв самостійно вивчати ПДР, проходити тести для самоперевірки, відслідковувати прогрес підготовки та визначити момент, коли людина готова дл складання істипу з теорії. Отже, було прийняте рішення щодо створення відповідного мобільного додатку. Дана розробка дійсно є актуальною сьогодні, оскільки забезпечить автоматизацію процесу підготовлення до отримання водійського посвідчення.

Тема. Мобільний додаток автоматизації процесу підготовлення до отримання водійського посвідчення

Мета. Розробка мобільного додатку автоматизації процесу підготовлення до отримання водійського посвідчення.

Для досягнення мети проєкту необхідно виконати наступні задачі:

- здійснити детальний аналіз предметної області та огляд сучасних публікацій, визначити актуальність роботи та цільову аудиторію мобільного додатку;
- провести аналіз існуючих продуктів-аналогів;
- визначити технології для розробки даного мобільного додатку автоматизацію процесу підготовлення до отримання водійського посвідчення;
- виконати структурно-функціональне моделювання;
- реалізувати структуру та функціональні можливості мобільного додатку автоматизацію процесу підготовлення до отримання водійського посвідчення;
- провести тестування розробленого мобільного додатку.

Об’єкт дослідження. Процес підготовлення до отримання водійського посвідчення.

Предмет дослідження. Мобільний додаток автоматизації процесу підготовлення до отримання водійського посвідчення.

Практична новизна. Зменшення витрат часу та зусиль на вивчення правил дорожнього руху за рахунок належної організації процесу самостійного підготовлення до отримання водійського посвідчення. Розвантаження автошкіл і сервісних центрів завдяки спрощенню та врегулюванню процедури запису для складання іспиту по теорії.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд останніх досліджень і публікацій

Для сучасного суспільства технології стали невід'ємною його частиною. Людям зараз важко уявити повсякденне життя без їх використання. Саме тому початковим етапом виконання даного проєкту стало проведення аналізу останніх досліджень у обраній галузі з метою кращого розуміння проблеми.

Мобільні додатки, це те що стало не від'ємною частиною життя сучасної людини. Вони використовуються у багатьох сферах, серед них можна виділити такі сфери, як: бізнес, інформаційні технології (IT) та навчання.

Будь-який бізнес має багато процесів, які можна перенести у мобільний додаток. Наприклад, ними можуть бути замовлення їжі або таксі, бронювання місця в лікарні, онлайн платежі тощо. У свою чергу це приводить до великого попиту на створення мобільних додатків для організації бізнесу, оскільки може збільшити престиж компанії [1]. А це забезпечить збільшення продажів. IT галузь у свою чергу може використовувати мобільні додатки безпосередньо під час робочого процесу. Як приклад, для виконання функцій редактора коду, планувальника задач або засоба для графічного дизайну [2]. На останок варто відзначити роль мобільних додатків у процесі навчання. Зазвичай вони подають матеріал із використанням інтерактивних елементів із метою утримання уваги користувача на дисципліні. Також варто зазначити мобільність такого підходу до навчання. Це досить зручно в сучасних умовах. Він дозволяє будь-кому та де завгодно навчатися у зручний для людини час і мати легкий механізм зворотнього зв'язку з вчителем [3].

У [4] описані переваги та недоліки введення в експлуатацію автоматизованої системи тестування водіїв. У результаті цього загальна безпека на дорогах погіршилася. При тестуванні дана система не може перевірити, чи було проведено усі необхідні регуляційні дії для початку безпечного водіння, або готовність учня. Тому питання створення програмного продукту, який би забезпечував якісне вивчення ПДР, є актуальним.

Важливість запобігання зловживанню недоліками подібних систем було розглянуто у [5]. Сучасні такі розробки мають велику кількість недоліків. У свою чергу їх можна використати для отримання незаконної вигоди. Тому перед введенням системи в експлуатацію необхідно нівелювати як умога більшу кількість недоліків. Наприклад, вони забезпечують запис на іспит за допомогою онлайн черги. А цей спосіб має свої слабкі місця. Будь-який охочий може зарезервувати місце та необхідну дату для здачі іспиту. Але в той же час система не вимагає ніяких особистих даних. І шахраї використовують цю можливість, бронюючи значну кількість місць і далі перепродають їх через мережу Інтернет. У результаті люди постають перед вибором: заплатити за місце або довго очікувати.

Тому тема автоматизації процесу підготовлення до отримання водійського посвідчення є актуальною. А саме рішення забезпечити це шляхом розробки відповідного мобільного додатку є доречним.

1.2 Аналіз програмних продуктів-аналогів

У сучасних реаліях не можливо уявити людину, яка б не використовувала смартфон у повсякденному житті. Тому застосування мобільних технологій стає все

більш популярним. Наприклад, у сфері освіти. Мобільні технології пропонують більш гнучкий підхід до навчання. Це через можливість виконувати завдання з будь-якої точки світу та у будь-який час. Ще одним фактором є зменшення витрат на виготовлення мобільних пристроїв. Це відіграє важливу роль. Оскільки призводить до поширення використання мобільних технологій у навчанні. Їхні переваги є такими:

– гнучкість і зручність: мобільні додатки дозволяють студентам навчатися у будь-якому місці та в будь-який час;

– персоналізований досвід: мобільні додатки можуть адаптуватися до індивідуальних потреб і стилів вивчення;

– ріст залученості: інтерактивні функції, відеоуроки та ігрові елементи можуть збільшити зацікавленість студентів у навчальному процесі;

– постійний доступ до ресурсів: студенти мають постійний доступ до навчальних матеріалів, завдяки чому можуть повторювати та вдосконалювати свої знання в будь-який час;

– економія ресурсів: використання мобільних додатків може зменшити необхідність у фізичних підручниках та інших друкованих матеріалах.

Але застосування мобільних технологій у навчанні має свої наступні недоліки:

– відсутність особистого взаємодії: брак фізичного контакту та спілкування може призвести до втрати особистого взаємодії між викладачем та студентом;

– залежність від технологій: не всі студенти можуть мати доступ до сучасних мобільних пристроїв, що може призвести до нерівності в доступі до освіти;

– відсутність моніторингу: брак фізичної присутності викладача може ускладнити моніторинг прогресу студентів та їхню активність;

– можливість відволікання: мобільні пристрої можуть відволікати студентів своїми іншими функціями, що може негативно вплинути на їхню увагу до навчання;

– необхідність значної самодисципліни: навчання за допомогою мобільного додатку вимагає від студентів великих зусиль для організації самостійного ефективного вивчення матеріалів [6].

Інтеграція мобільних додатків також все частіше стає необхідністю для надання послуг у державних установах. Однією з них є сервісний центр. За його допомогою можна зареєструвати машину, виготовити номерні знаки тощо. Також сервісний центр надає можливість отримати водійське посвідчення. Для цього треба скласти відповідні іспити, теоретичний і практичний. Зараз оновили правила отримання водійське посвідчення. Усі бажаючі можуть самостійно вивчити ПДР і скласти теоретичний іспит без необхідної умови відвідування автошколи. Це є досить зручною процедурою для великої кількості українців, особливо при сучасних обставинах у нашій країні.Тому використання такого мобільного додатку, який би надав можливість вивчати правила дорожнього руху дистанційно, проходити тести для самоперевірки, відслідковувати власний прогрес та записатися на здачу теоретичного іспиту є нагальним. Його застосування зменшить витрату часу на навчання та зробить процес підготовлення до отримання водійського посвідчення зручним і приємнішим.

Для проведення порівняльного аналізу було відібрано два мобільні додатки. Вони містять правила дорожнього руху та тести до них. А саме «Тести ПДР» [7] та «Driving Licence Practice Tests» [8].

До основної структури мобільного додатку «Тести ПДР» входять такі компоненти, як теорія (правила дорожнього руху, дорожня розмітка та знаки), тести та підрахунок алкоголю в крові на основі введених даних.

Його основні функції наступні:

– вивчення теорії (ПДР, дорожня розмітка та знаки);

– різноманітні тести перевірки знань правил дорожнього руху;

– підрахунок алкоголю в крові на основі введених даних.

Інтерфейс розділу «Теорія» мобільного додатку «Тести ПДР» представлено на рисинку 1.1.

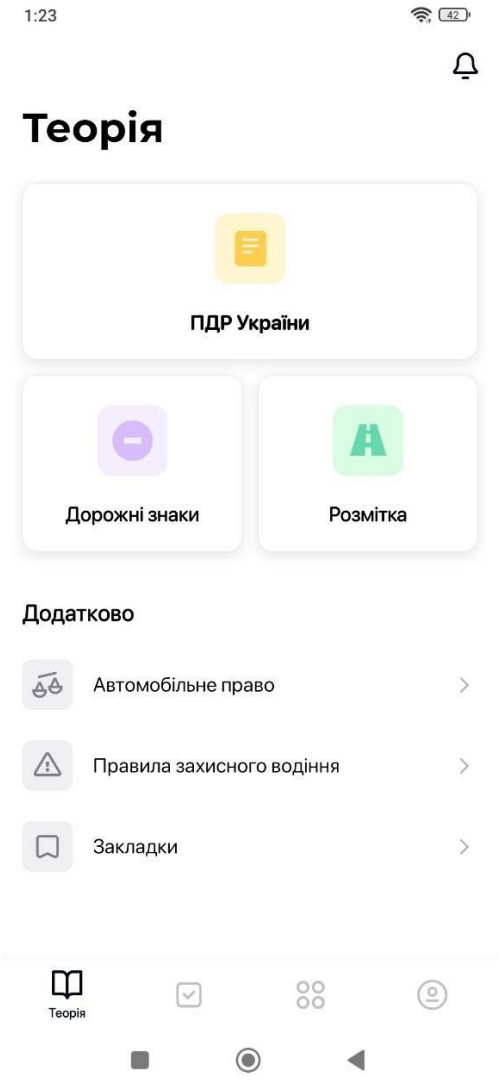


Рисунок 1.1 – Головне вікно мобільного додатку «Тести ПДР»

Джерело: [7]

Даний програмний продукт має інтуїтивно зрозуміле розміщення елементів на екрані (рис. 1.2-1.3).

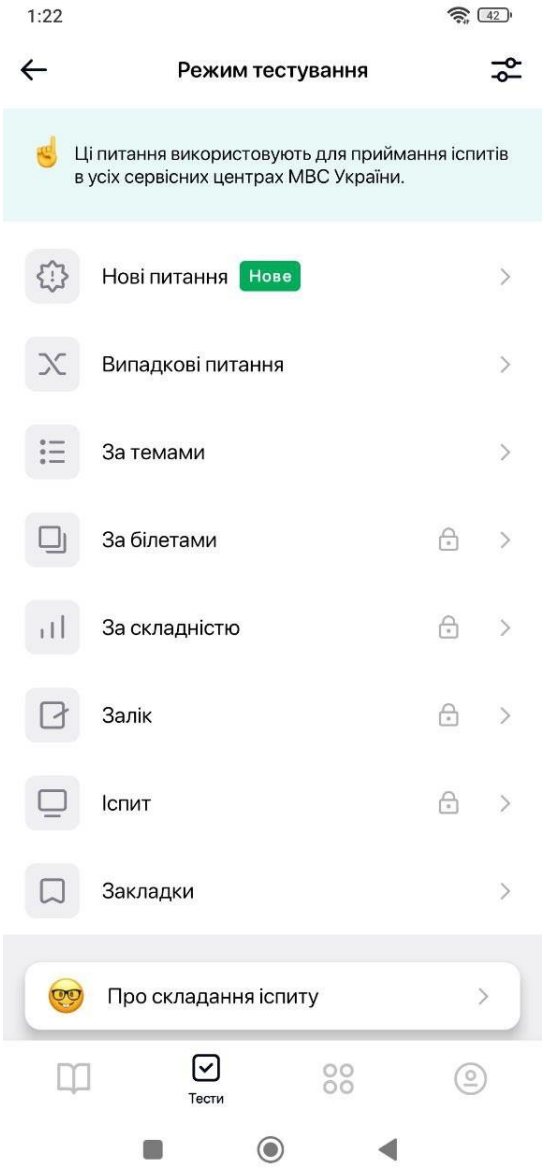


Рисунок 1.2 – Вікно режиму тестування мобільного додатку «Тести ПДР»

Джерело: [7]

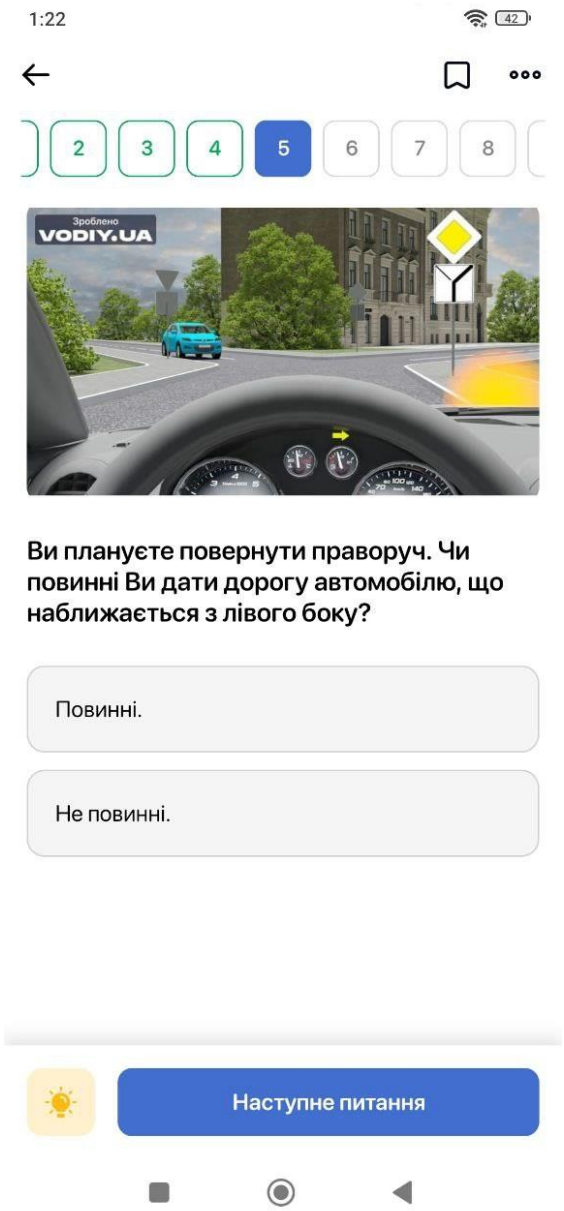


Рисунок 1.3 – Вікно режиму «Іспит» мобільного додатку «Тести ПДР»

Джерело: [7]

Після проведення дослідження вищезазначеного мобільного додатку «Тести ПДР», можна зробити висновок, що він має багато різноманітних функцій і виконує більшість із поставлених завдань щодо вивчення ПДР, але не надає можливості взаємодії з сервісним центром.

Далі було розглянуто мобільний додаток «Driving Licence Practice Tests». Він має на меті навчити користувачів звертати увагу на ситуації на дорозі. Його інтерфейс представлено на рисунках 1.4-1.5.

Рисунок 1.4 – Головне вікно мобільного додатку «Driving Licence Practice Tests»

Джерело: [8]

Рисунок 1.5 – Вікно тесту мобільного додатку «Driving Licence Practice Tests»

Джерело: [8]

Поміж присутнього функціоналу мобільного додатку «Driving Licence Practice Tests» варто виділити такі:

– вивчення правил дорожнього руху;
– проходження різноманітних видів тестування;
– вивід статистики (рис. 1.6).

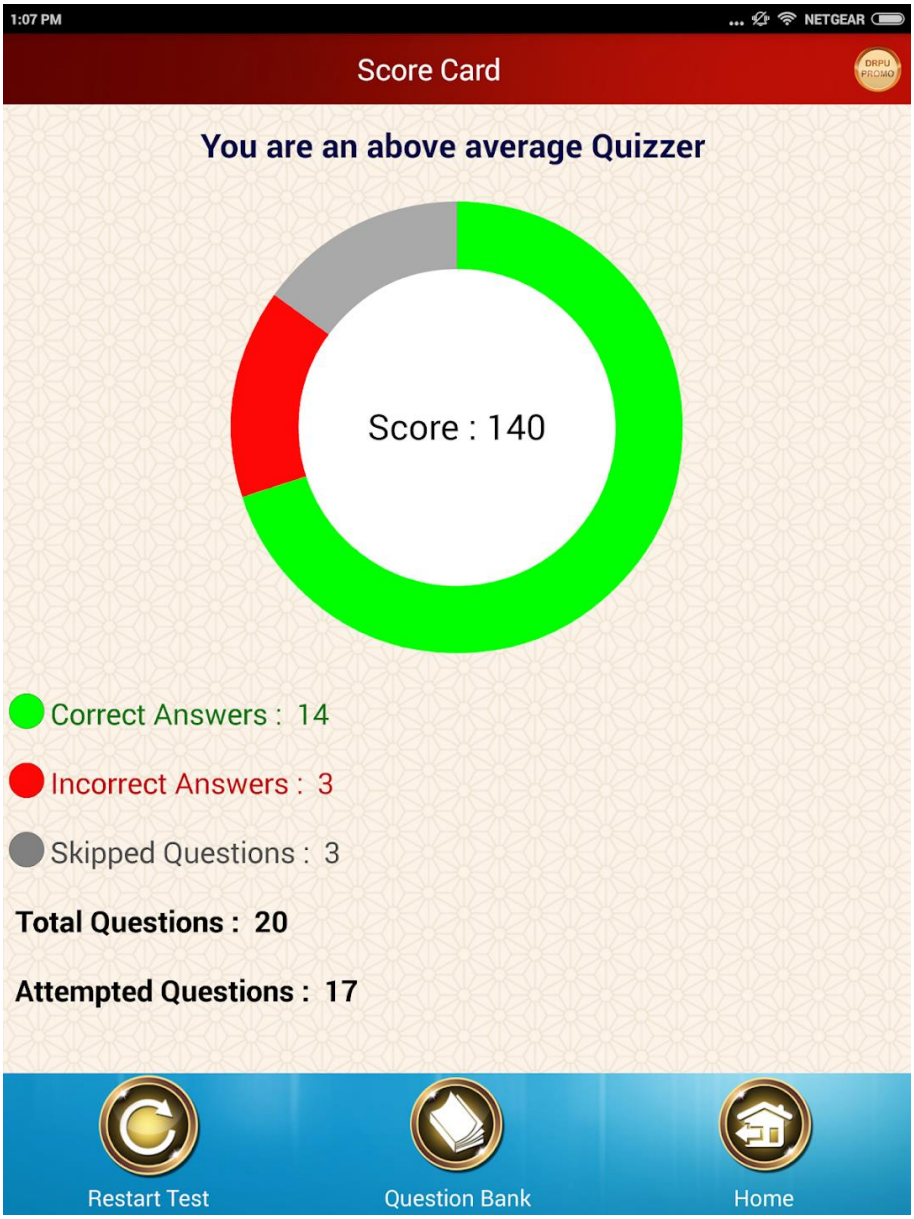


Рисунок 1.6 – Вікно «статистика» мобільного додатку «Driving Licence Practice Tests»

Джерело: [8]

Серед недоліків даного мобільного додатку варто зазначити застаріле оформлення, яке не відповідає сучасним вимогам UI/UX дизайну [9].

Результатом порівняльного аналізу вищевказаних мобільних додатків є таблиця 1.1.

Таблиця 1.1 – Порівняльна таблиця аналогів інформаційних систем

Характеристика/Інформаційна система	**«Тести ПДР»**	**«Driving Licence Practice Tests»**
Відповідність сучасним вимогам UI/UX дизайну	+	-
Зручний інтерфейс та навігація	+	-
Наявність можливості вивчення правил дорожнього руху	+	+
Наявність перегляду статистики	+	+
Можливість проходження тестів	+	+
Можливість коментування тестів	-	-
Можливість запису на здачу іспиту	-	-

Джерело: побудовано автором

Дані з таблиці 1.1 допомагають виділити основні переваги та недоліки вище розглянутих мобільних додатків, тим самим надаючи список функціональних рішень, які в подальшому можна реалізувати у власній розробці. А також слабкі сторони, які варто подолати. Підсумовуючи, можна зазначити, що створюваний мобільний додаток повинен вирішувати поставлені перед ним задачі. А саме, покращити процес підготовлення до отримання водійського посвідчення шляхом надання швидкого та зручного доступу як до вивчення ПДР, так і до тестів для самоперевірки, перегляду власного прогресу, та можливості запису на складання теоретичного іспиту у сервісному центрі.

2 ПОСТАНОВКА ЗАДАЧІ ТА МЕТОДИ ДОСЛІДЖЕННЯ

2.1 Мета та задачі дослідження

Метою кваліфікаційної роботи магістра є розробка мобільного додатку автоматизації процесу підготовлення до отримання водійського посвідчення.

Представлений програмний продукт матиме попит серед користувачів, які бажають керувати власним транспортним засобом. Його використання забезпечить покращення процесу підготовлення до отримання водійського посвідчення завдяки можливостям отримання необхідних знать шляхом вивчення правил дорожнього руху, проходження тестів для самоперевірки, перегляду власного прогресу та запису на складання теоретичного іспиту в сервісному центрі. Також застосування створюваного мобільного додатку дозволить зменшити навантаження на персонал даної державної установи. І допоможе покращити безпекову ситуацію на дорогах, а саме підвищить рівень знань правил дорожнього руху серед водіїв.

Розроблений програмний продукт повинен мати наступні функції:

- можливість вивчення правил дорожнього руху;
- можливість проходження тестів;
- можливість перегляду результату після проходження тесту;
- можливість запису на складання теоретичного іспиту в сервісному центрі.

Даний мобільний додаток повинен мати мінімалістичний, інтуїтивно зрозумілий інтерфейс, коректне відображення всіх зображень на екрані телефона та містити у собі всі вище зазначені функції.

Для досягнення мети даного проєкту необхідно виконати такі задачі:

– здійснити детальний аналіз предметної області та огляд сучасних публікацій, визначити актуальність роботи та цільову аудиторію даного мобільного додатку;

– провести аналіз існуючих продуктів-аналогів, виділити їх сильні та слабкі сторони;

– визначити технології для розробки мобільного додатку автоматизацію процесу підготовлення до отримання водійського посвідчення;

– виконати структурно-функціональне моделювання роботи запропонованого програмного продукту;

– реалізувати структуру та функціональні можливості мобільного додатку автоматизацію процесу підготовлення до отримання водійського посвідчення;

– провести тестування розробленого мобільного додатку.

Під час формалізації мети проєкту було проведено планування його робіт Воно детально описано у Додатку А.

2.2 Методи дослідження

Після формування мети проєкту, визначено функціональні та нефункціональні вимоги до розроблюваного мобільного додатку автоматизації процесу підготовлення до отримання водійського посвідчення.

Дане дослідження проводиться за допомогою наступних чотирьох основних методів:

– теоретичний метод;

– системно-функціональний метод;

– емпіричний метод;

– метод моделювання [10-12].

Проводячи роботу над проєктом «Мобільний додаток автоматизації процесу підготовлення до отримання водійського посвідчення», теоретичний метод було використано при детальному аналізі як предметної області, так і останніх публікацій, а також при виборі інструментів розробки. Він підходить для виокремлення переваг та недоліків досліджуваних об’єктів.

Суть емпіричного методу полягає у спостереженнях, опитувань та експериментів. Він проводиться для більш детального ознайомлення з предметною областю. Даний метод використовувався при аналізі потреб людей, які збираються отримати водійське посвідчення. Дані бралися з результатів опитування.

Метод моделювання має на меті створення різноманітних схем та діаграм. Вони описують процес розробки програмного продукту та візуально відображають те, яким чином буде проводитись взаємодія між користувачем та, у даному випадку, мобільним додатком. Протягом роботи над представленим проєктом цей метод було використано для побудови діаграм IDEF0 [13] та UseCase [14]. Їх застосування було корисним для представлення процесів, які забезпечує дана розробка.

Системно-функціональний допомагає отримати опис системи та її функціонал. Також відбувається дослідження її компонентів, їх залежностей та взаємозв’язок. Програмний модуль даного проєкту був створений за допомогою цього методу .

2.3 Вибір технологій

Із урахуванням високої популярності мобільних пристроїв із операційною системою Android, прийнято рішення розробляти програмний продукт саме для цієї платформи [15]. Для реалізації проєкту обрано мову програмування Kotlin [16]. Вона

є сучасною альтернативою Java й користується популярністю серед професіоналів у сфері створення мобільних додатків [17].

У якості середовища розробки обрано Android Studio [18]. Даний програмний засіб вважається одним із найкращих для створення Android продуктів. Android Studio підтримує розробку мобільних додатків мовою Kotlin, має емулятор для тестування та використовує систему контролю версій Git.

Вибір технологій здійснювався з урахуванням їхніх переваг і більшої популярності серед інших конкуруючих рішень.

3 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

Наступний етап – це проєктування даного мобільного додатку. При цьому було побудовано діаграму IDEF0, виконано її декомпозицію та змоделювано варіанти використання запропонованого програмного продукту.

3.1 Діаграми нотації IDEF0

Для наглядної демонстрації роботи функцій запропонованого мобільного додатку було використано діаграму в нотації IDEF0. Вона містить у собі графічний опис процесів, виконання яких забезпечує представлений програмний продукт, і відображає зв’язки між ними. Завдяки створенню діаграми IDEF0 можна досягти кращого розуміння функцій. Це робиться допомогою дослідження кожної з них, незалежно від об’єктів, для яких вона заготовлена.

Для створення контекстної діаграми для мобільного додатку містить наступні данні:

– вхідні дані: відображені стрілками, які входять зліва та представлені запитом для проходження тестів, тестами з правил дорожнього руху та запитом на запис на складання теоретичного іспиту;

– вихідні данні: відображені стрілками, виходять праворуч та представлені отриманими знаннями для участі у складанні теоретичного іспиту та результатами проходження тестування;

– управління: показано як стрілки, які входять зверху та містять у собі правила дорожнього руху та положення про порядок видачі водійського посвідчення;

– механізм: позначений стрілками, які входять знизу та представлені мобільним додатком, базою даних, користувачем та смартфоном.

На рисунку 3.1. представлена діаграма контекстна діаграма IDEF0.

Рисунок 3.1 – Контекстна діаграма IDEF0

Джерело: побудовано автором

На рисунку 3.2. представлена декомпозиція вищезазначеної контекстної діаграми.

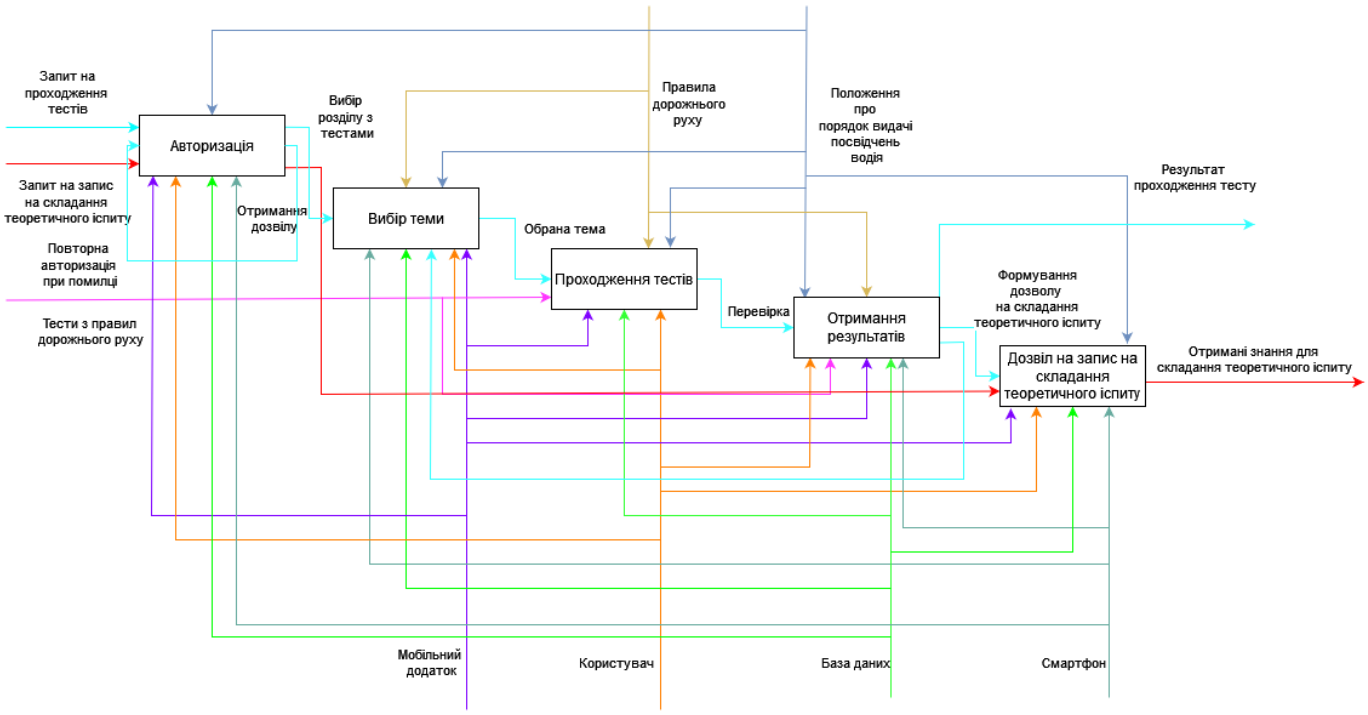


Рисунок 3.2 – Декомпозиція контекстної діаграми IDEF0

Джерело: побудовано автором

3.2 Діаграма Use Case

Далі було створено діаграму Use Case, яка відображає різноманітні варіанти використання даного мобільного додатку. Ключовими її складовими є актори, зв’язки, примітки та механізми розширення.

Для мобільного додатку автоматизації процесу підготовлення до отримання водійського посвідчення, було виділено такі сценарії:

– вивчення правил дорожнього руху;

– проходження тестів;

- перегляд результатів пройденого тесту
- можливість запису на складання теоретичний іспит.

Для діаграми Use Case були визначені такі актори:

- користувач (студент);
- інформаційний ресурс;
- база даних.

На рисунку 3.3 зображена діаграма Use Case.

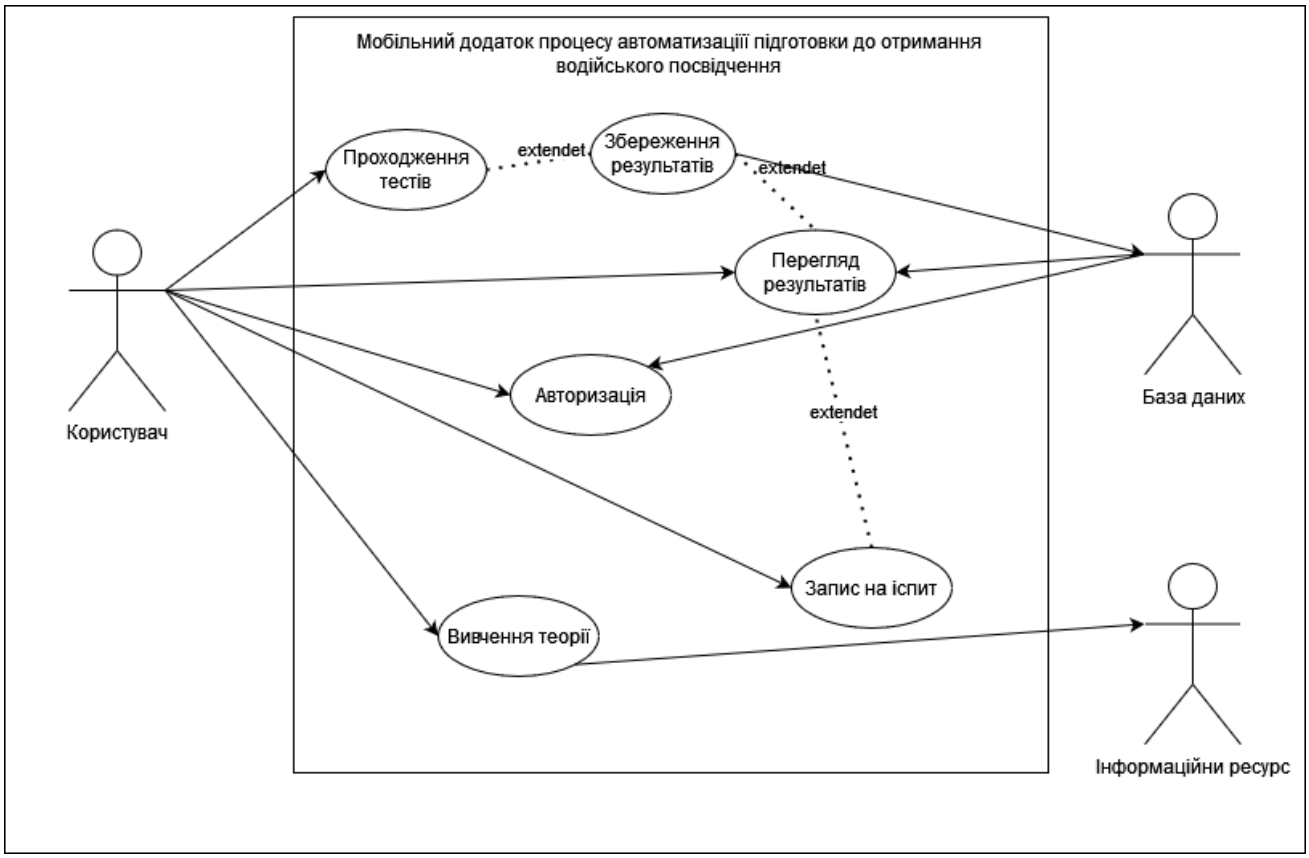


Рисунок 3.3 – Діаграма Use Case

Джерело: побудовано автором

3.3 Проєктування моделі бази даних

Модель даних буде використана для створення бази даних (БД) [19].

БД розроблюваного мобільного додатку має наступні таблиці:

– User – містить інформацію про користувача;
– Exam – містить інформацію іспит;
– Statistic – містить статистичну інформацію;
– Question – містить інформацію про питання, які присутні у тесті;
– Answer – містить дані про відповіді на питання.

На рисунку 3.3 представлена фізична модель бази даних.

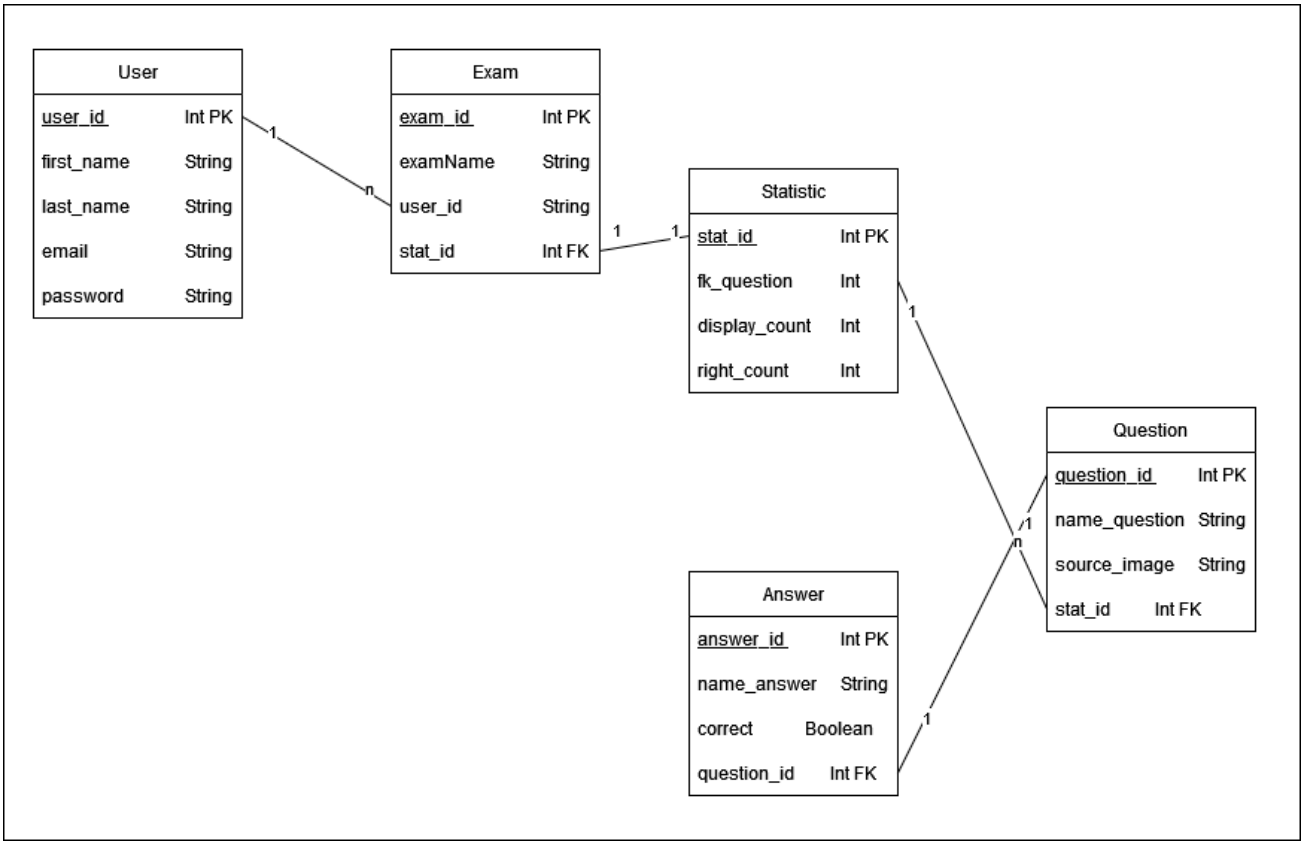


Рисунок 3.3 – Фізична модель бази даних

Джерело: побудовано автором

4 РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ АВТОМАТИЗАЦІЇ ПРОЦЕСУ ПІДГОТОВЛЕННЯ ДО ОТРИМАННЯ ВОДІЙСЬКОГО ПОСВІДЧЕННЯ

4.1 Архітектура мобільного додатку

Будь-який проєкт починається зі створення його архітектури. І даний мобільний додаток не є включенням. Він використовує MVVM (Model-View-ViewModel) [20] архітектуру, яка розділяє запропонований програмний продукт на такі компоненти, як модель, вид та модель представлення.

Архітектура мобільного додатку зображена на рисунку 4.1.

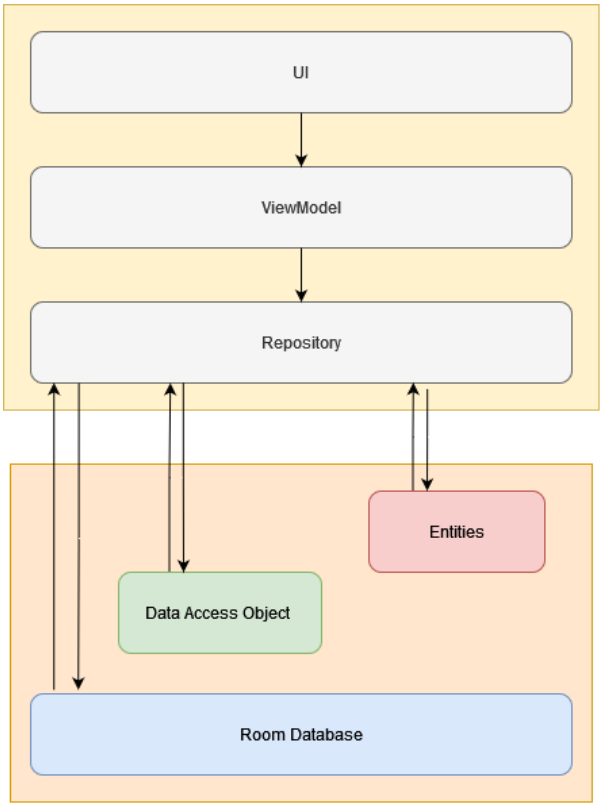


Рисунок 4.1 – Архітектура мобільного додатку

Джерело: побудовано автором

4.2 Програмна реалізація

Мова програмування Kotlin разом із середовищем розробки Android Studio були використані під час програмної реалізації запропонованого мобільного додатку. Для створення його зовнішнього вигляду та навігації між елементами застосовано Jetpack Compose [21]. Даний набір інструментів використовує мову програмування Kotlin. У свою чергу, це створює набагато більше можливостей для розробки сучасного та зручного інтерфейсу. Нижче надано код компонента, за допомогою якого проводиться навігація між елементами:

```
@Composable
fun Main(){
    Surface (
        modifier = Modifier.fillMaxSize(),
        ){
        Crossfade(targetState = LoginAppRoter.currentScreen)
{currentState ->
            when(currentState.value){
                is Screen.SignUpScreen ->{
                    SignUpScreen()
                }
                is Screen.LoginScreen ->{
                    LoginScreen()
                }
                is Screen.ChooseScreen ->{
                    ChooseScreen()
                }
                is Screen.Main->{

                    HomeScreen()
```

```
                    }
                }


            }
        }
}
```

Роботу бази даних у даному мобільному додатку було реалізовано за допомогою бібліотеки Room [22]. Вона використовується у якості оболонки для БД, створених за допомогою SQLite.

Бібліотека Room складається з таких основних компонентів:

– База даних – містить анотації щодо параметрів БД;

– Дао (Dao) – інтерфейс, який містить методи, за допомогою яких виконуються операції з БД;

– Сутність (Entity) – представляє таблицю, яка знаходиться у базі даних.

На рисунку 4.2 зображено схему взаємодії між вищезазначеними компонентами.

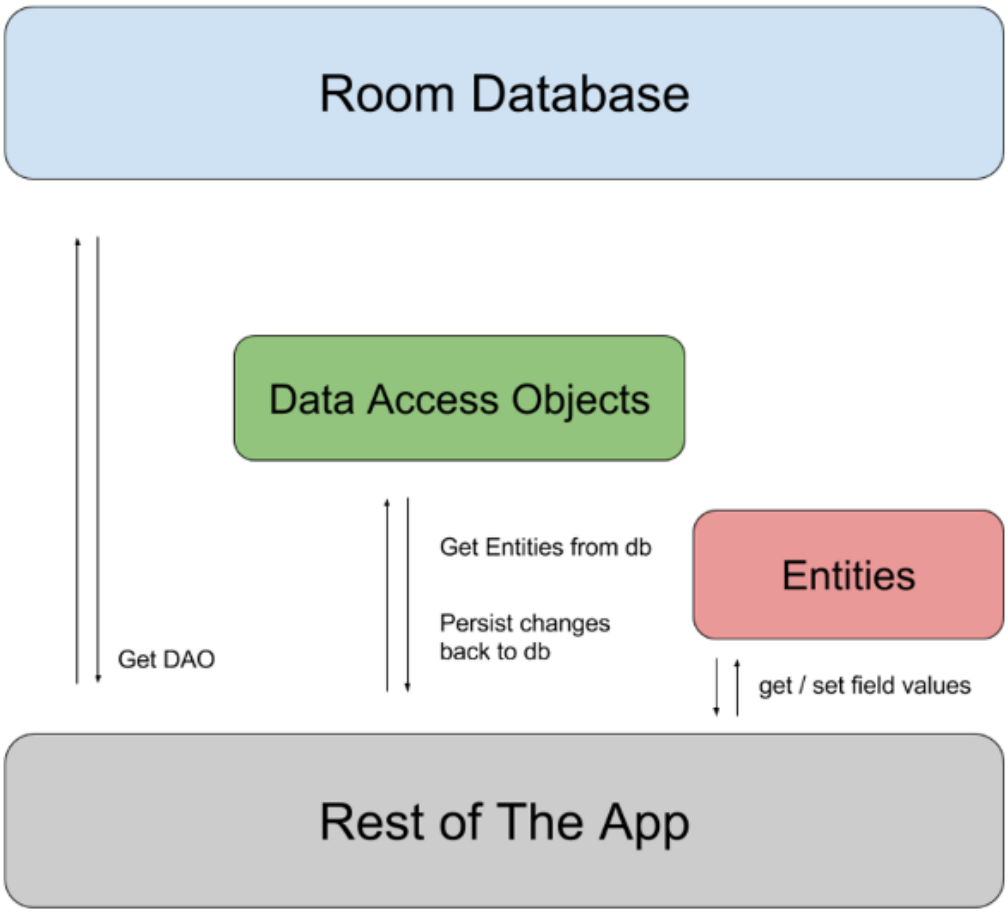


Рисунок 4.2 – Схема взаємодії компонентів бібліотеки Room

Джерело: [22]

Таблиці, які потім можна знайти у базі даних, генеруються за допомогою відповідного Entity класу, а виконання запитів реалізовано за допомогою Kotlin Coroutines [23].

Нижче надано приклади кодів класів Entity.

Код Entity клас QuestionEntity:

```
@Entity(tableName = "question")
class QuestionEntity constructor(
    @PrimaryKey
    @ColumnInfo(name = "question_id")
    var id: Int,
```

```
    @ColumnInfo(name = "name_question")
    var text: String,
    @ColumnInfo(name = "source_image")
    var image: String? = null,
    @ColumnInfo(name = "statd_id")
    var statId: Int?
)
```

Код Entity клас AnswerEntity:

```
@Entity(tableName = "answer")
class AnswerEntity constructor(
    @PrimaryKey
    @ColumnInfo(name = "answer_id")
    var id: Int,
    @ColumnInfo(name = "name_answer")
    var text: String,
    @ColumnInfo(name = "correct")
    var correct: Boolean,
    @ColumnInfo(name = "question_id")
    var questionId: Int
)
```

Інтерфейси типу Dao, використовуються для здійснення запису та зчитування даних з БД. У них описані методи, які містять у собі відповідні запити, можливі для виконання базою даних.

Приклад коду інтерфейсу типу Dao представлено нижче:

```
@Dao
interface UsersDao {

    @Insert
    suspend fun insert(register: Users)

    @Query("SELECT * FROM users ORDER BY user_id DESC")
    fun getAllUsers(): LiveData<List<Users>>

    @Query("DELETE FROM users")
```

```
    suspend fun deleteAll(): Int

    @Query("SELECT * FROM users WHERE email LIKE :userName")
    suspend fun getUsername(userName: String): Users?

}
```

Приклад коду омпоненту інтерфейсу, розробленого за допомогою Jetpack Compose, представлено нижче:

```
Column(modifier = Modifier.fillMaxSize()) {

                HeadingTextComponent(value = stringResource(id =
R.string.login_message))
                Spacer(modifier = Modifier.height(20.dp))
                MyTextField(
                    labelValue = stringResource(id = R.string.email),
onTextSelected = {

loginViewModal.onEvent(LoginUIEvent.EmailChanged(it))
                    },
                    errorStatus =
loginViewModal.loginUIState.value.emailError
                )
                PasswordTextField(
                    labelValue = stringResource(id =
R.string.password), onTextSelected = {

loginViewModal.onEvent(LoginUIEvent.PasswordChanged(it))
                    },
                    errorStatus =
loginViewModal.loginUIState.value.passwordError
                )
```

Лістинг коду основних модулів мобільного додатку надано в додатку Б.

4.3 Демонстрація роботи мобільного додатку

Після запуску мобільний додаток зустрічає користувача екраном завантаження. На ньому зображено його логотип (рис. 4.3).

Рисунок 4.3 – Екран завантаження мобільного додатку

Джерело: побудовано автором (знімок з екрану)

Далі користувач потрапляє на екран авторизації (рис. 4.4). Після чого йому необхідно надати інформацію свого облікового запису. Поля для введення даних будуть обведені червоним кольором до тих пір, поки користувач не внесе правильні адресу електронної скриньки та пароль, які задовольняють вимоги мобільного додатку. Якщо обліковий запис існує, необхідно натиснути кнопку «Увійти». Вона

стане активною у разі введення даних, які задовольняють усі вимоги. Після натискання цієї кнопки користувач перейде до основного екрану мобільного додатку. Якщо клієнт не має облікового запису, він має можлмвіть зареєструватися. Це може бути виконано шляхом переходу до екрану створення акаунту за допомогою натискання на посилання, яке виділене синім кольором і знаходиться нижче від кнопки «Увійти». Екран реєстрації (рис. 4.5) містить поля, які необхідно заповнити. Після цього потрібно натиснути кнопку «Зареєструватися». По завершеню процесу реєстрації користувач перейде до екрану авторизації.

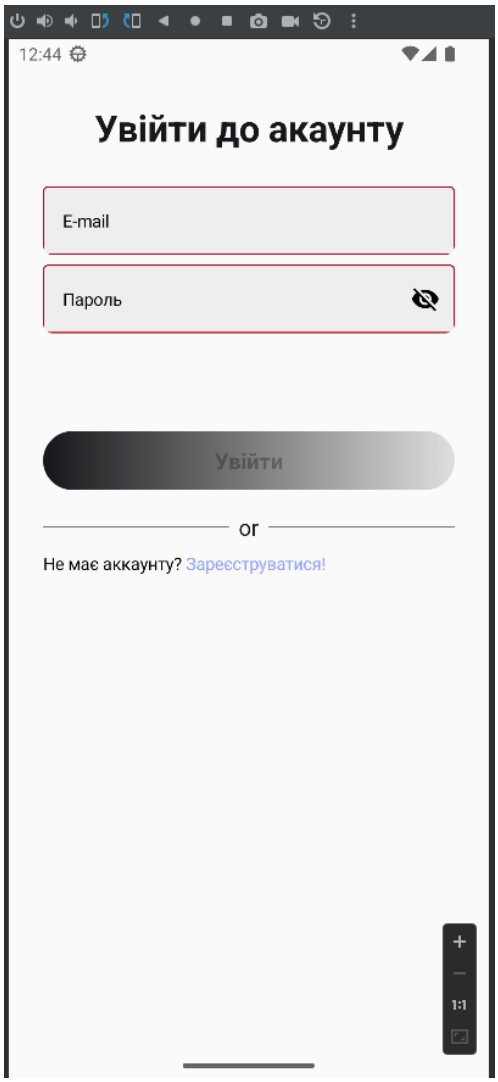


Рисунок 4.4 – Екран авторизації

Джерело: побудовано автором (знімок з екрану)

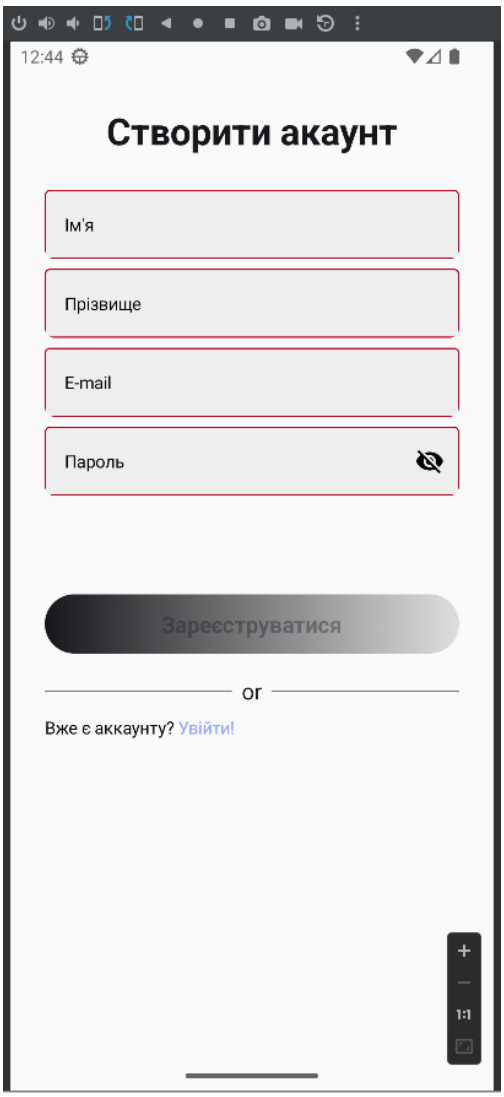


Рисунок 4.5 – Екран реєстрації

Джерело: побудовано автором (знімок з екрану)

Після успішної авторизації, відкривається головний екран (рис. 4.6). Він містить такі елементи, як назва додатку, кнопку з налаштуваннями та кнопку виходу з акаунту. У головній частині знаходиться посилання на екран з вибором теми для навчання та меню вибору типу тестування.

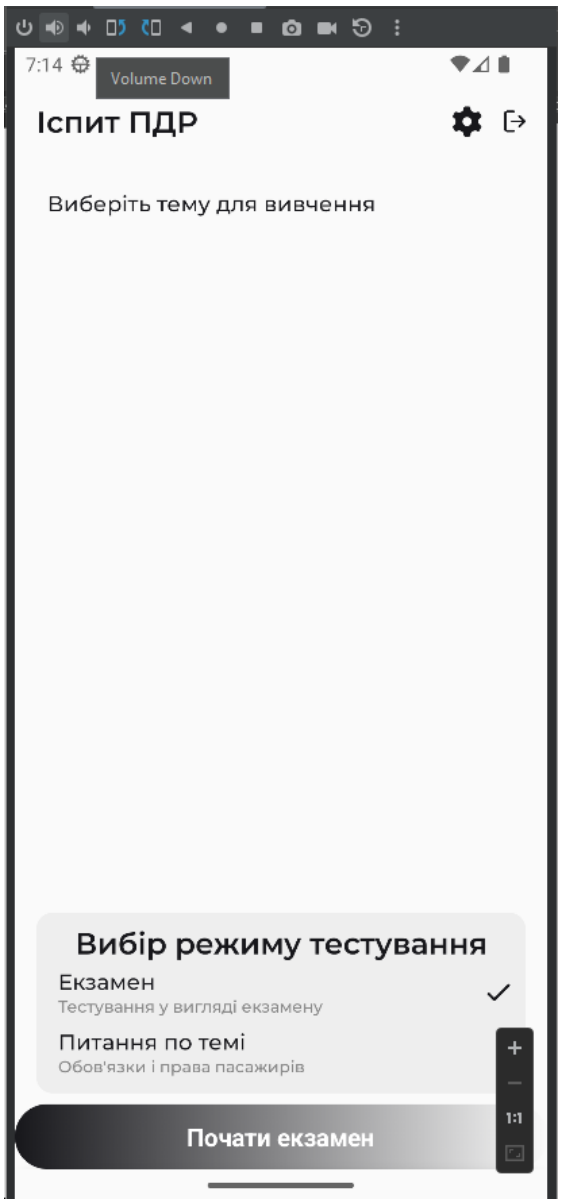


Рисунок 4.6 – Головний екран

Джерело: побудовано автором (знімок з екрану)

При натисканні на посилання «Виберіть тему для вивчення», користувач потрапляє на екран з вибором теми для вивчення ПДР (рис. 4.7). Далі необхідно обрати певний розділ. Далі користувач натискає на відповідну тему й переходить на сторонній ресурс. На ньому представлено правила дорожнього руху. Користувач має можливість передивитися всю інформацію, пов’язану з обраною ним темою.

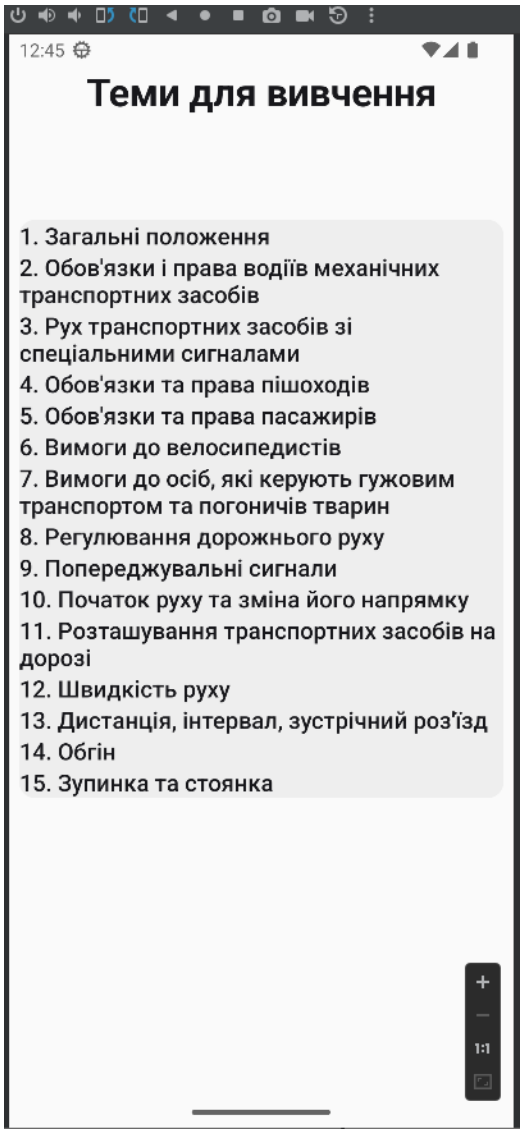


Рисунок 4.7 – Екран вибору тем для вивчення

Джерело: побудовано автором (знімок з екрану)

Вибір режиму тестування присутній у нижній частині головного екрану. При натисканні на будь-який із запропонованих його режимів, справа від нього стане доступним індикатор у вигляді «галочки». Вона надає користувачу певну інформацію. Це розуміння того, який режим тестування обраний. У мобільному додатку присутні два їхні варіанта. Це режим тестування «Екзамен» та «Питання по темам». Вибравши перший, користувач отримує можливість натиснути на кнопку

«Почати екзамен». Таким чином він перейде на екран, який міститиме 20 запитань та моделюватиме умови справжнього іспиту (обмеження по часу й допустиму кількість неправильних відповідей). Також допустим для вибору є тип тестування «Питання по темі». Обравши його відкриється випадаюче меню. Воно містить теми, за якими можна пройти тестування. Треба обрати одну з них і натиснути кнопку «Почати тестування» . Користувач перейде на екран із тестами по відповідній темі.

На рисунку 4.8 зображено екран із тестуванням.

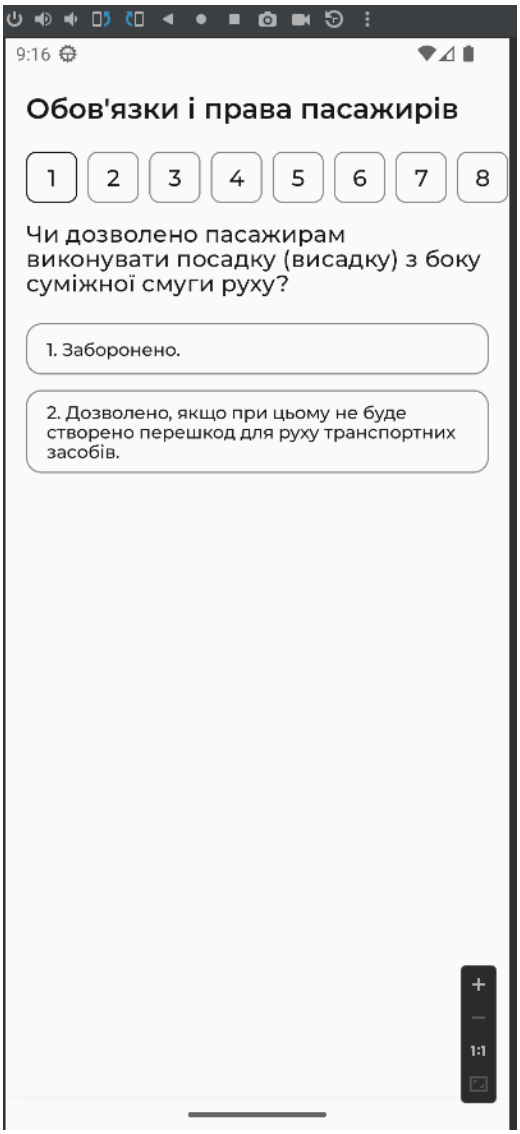


Рисунок 4.8 – Екран тестування

Джерело: побудовано автором (знімок з екрану)

Після того, як були дані відповіді на всі питання відкриється випадаюче меню. Воно містить статистику користувача за пройдений тест. На рисунку 4.9 зображене випадаюче меню з результатами для режиму тестування «Питання по темі».

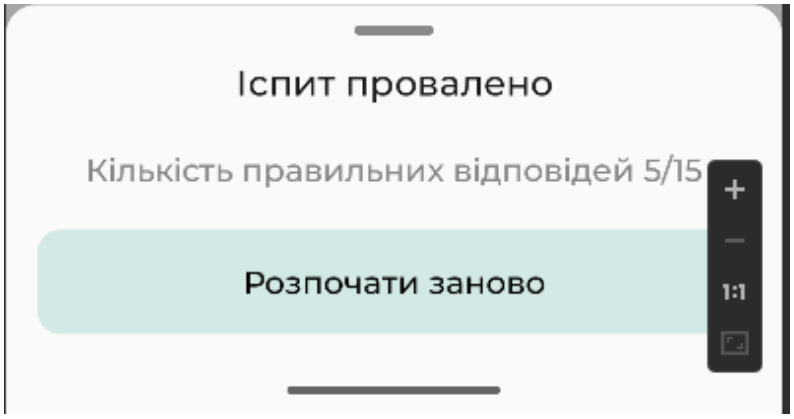


Рисунок 4.9 – Випадаюче меню з результатами пройденого тесту в режимі «Питання по темі»

Джерело: побудовано автором (знімок з екрану)

Для режиму тестування «Екзамен» передбачено видозмінене меню. Воно відображає стан іспиту. Це або провалений або зданий. При останньому варіанті буде надано можливість переходу на ресурс для запису на складання теоретичного іспиту у сервісному центрі. Інструкцію, як записатися на теоретичний іспит, описано далі.

На рисунку 4.10 зображене випадаюче меню з результатами для режиму тестування «Екзамен».

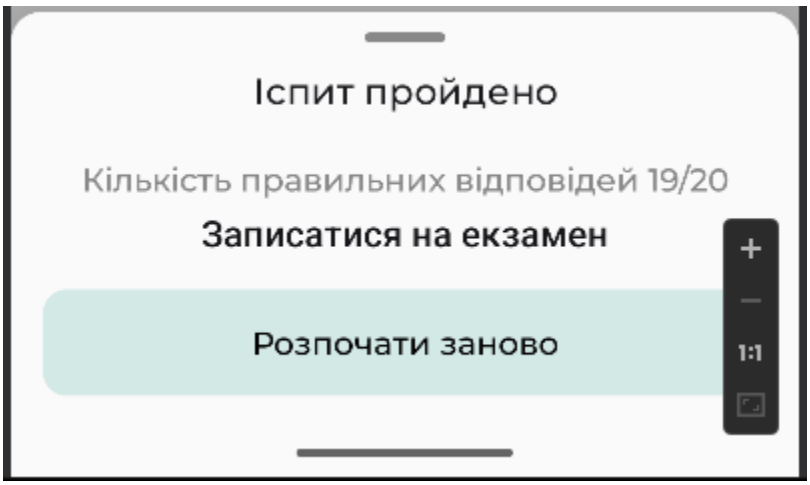


Рисунок 4.10 – Випадаюче меню з результатами режиму тестування «Екзамен»

Джерело: побудовано автором (знімок з екрану)

Далі користувач обрає записатися на екзамен. Його переводить на екран, де показана покрокова інструкція як це зробити.

Крок 1. Для того, щоб потрапити на сторінку запису для складання теоретичного іспиту, користувач повинен пройти тестування у режимі «Екзамен» за менше, ніж 20 хвилин та з 2 та менше помилками і натиснути на посилання «Записатися на екзамен» (рис. 4.11).

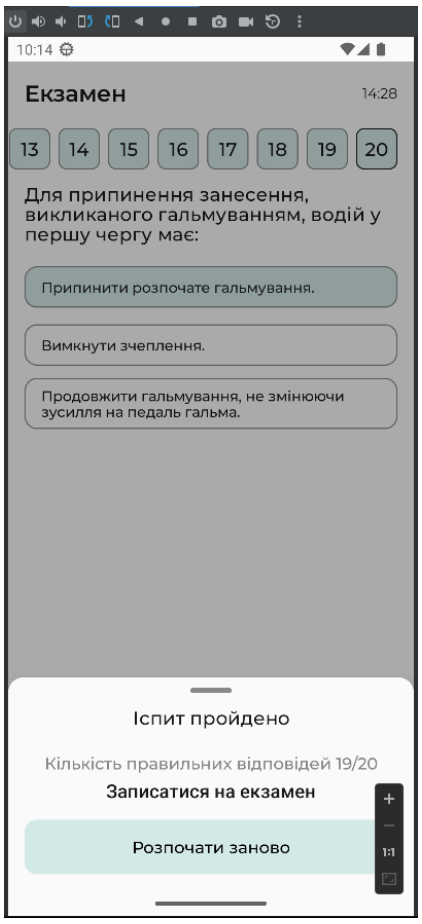


Рисунок 4.11 – Графічне відображення 1 кроку

Джерело: побудовано автором (знімок з екрану)

Крок 2. Після переходу на сторінку запису сайту сервісного центру, користувача зустрічає текст, який ознайомлює його з доступними функціями. Для продовження необхідно ознайомитися та погодитися з умовами надання послуг та натиснути кнопку «Е-запис» (рис. 4.12).

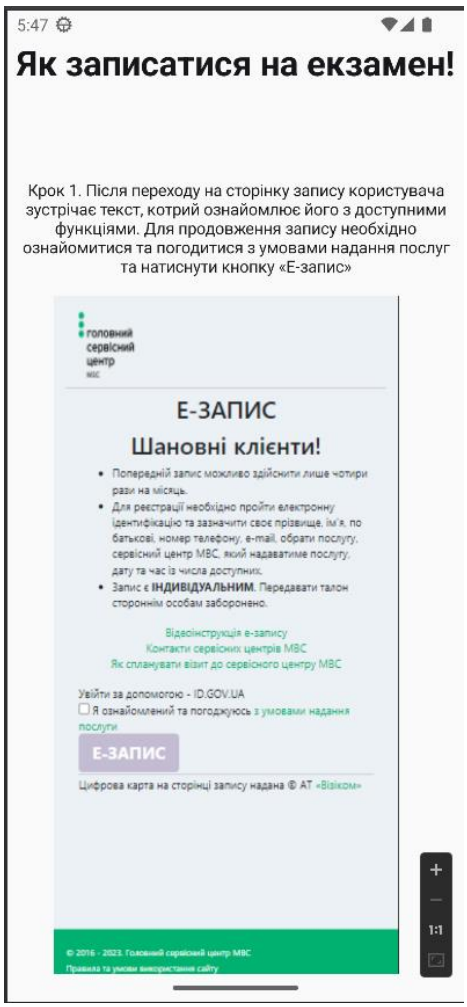


Рисунок 4.12 – Графічне відображення 2 кроку

Джерело: побудовано автором (знімок з екрану)

Крок 3. Після цього користувача переносить на сторінку авторизації за допомогою банку, ID картки та інших методів web-додатку «Дія» (рис. 4.13).

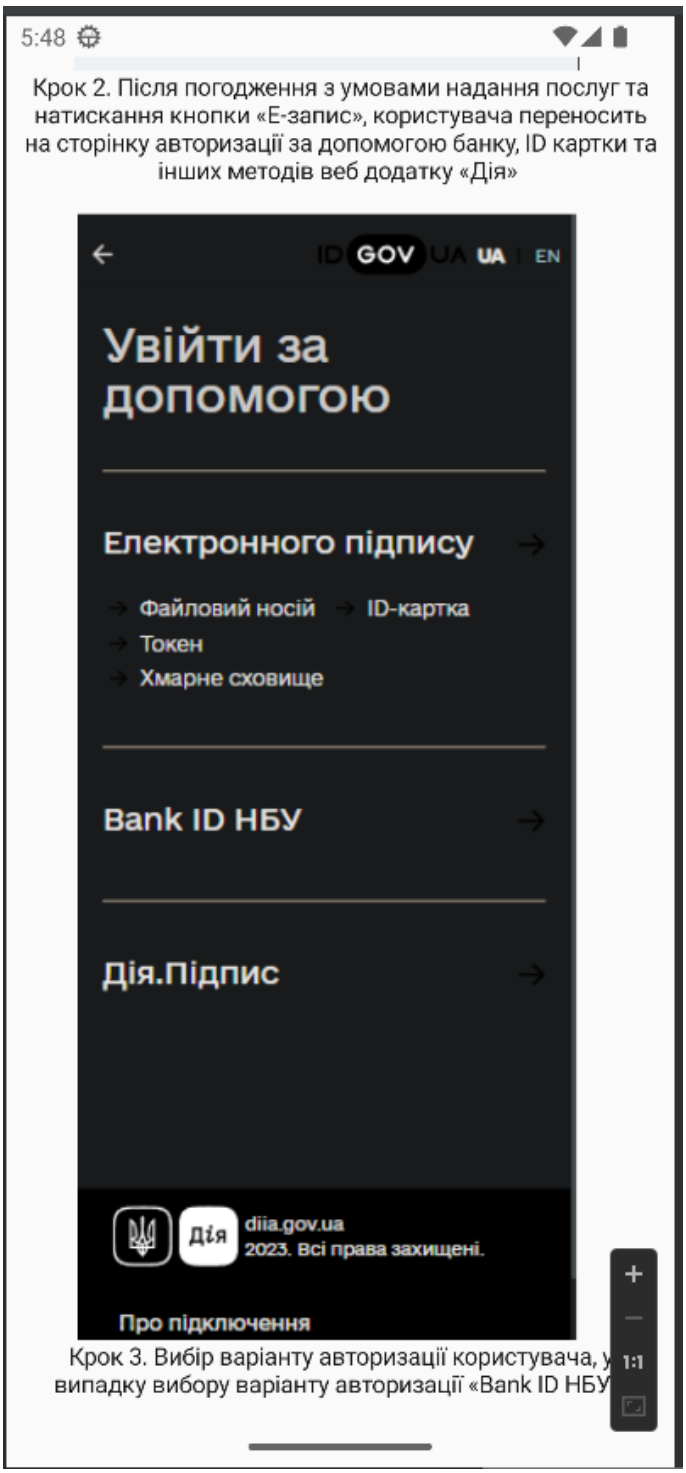


Рисунок 4.13 – Графічне відображення 3 кроку

Джерело: побудовано автором (знімок з екрану)

Крок 4. Далі, у випадку вибору варіанту авторизації «Bank ID НБУ» (рис. 4.14), користувача перенесе на сторінку обраного банку (рис. 4.15), де

треба авторизуватися через відповідний банківський акаунт, або, у випадку вибору варіанту авторизації «Електронного підпису», користувачу необхідно надати файл з електронним підписом (рекомендується використовувати авторизацію шляхом «Bank ID НБУ» через простоту та зрозумілість).

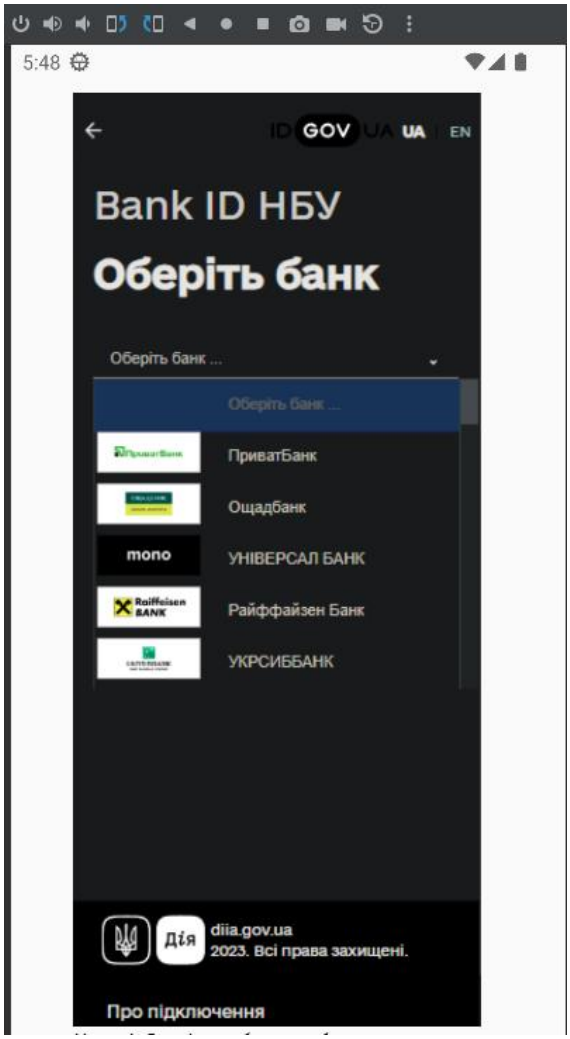


Рисунок 4.14 – Графічне відображення 4 кроку

Джерело: побудовано автором (знімок з екрану)

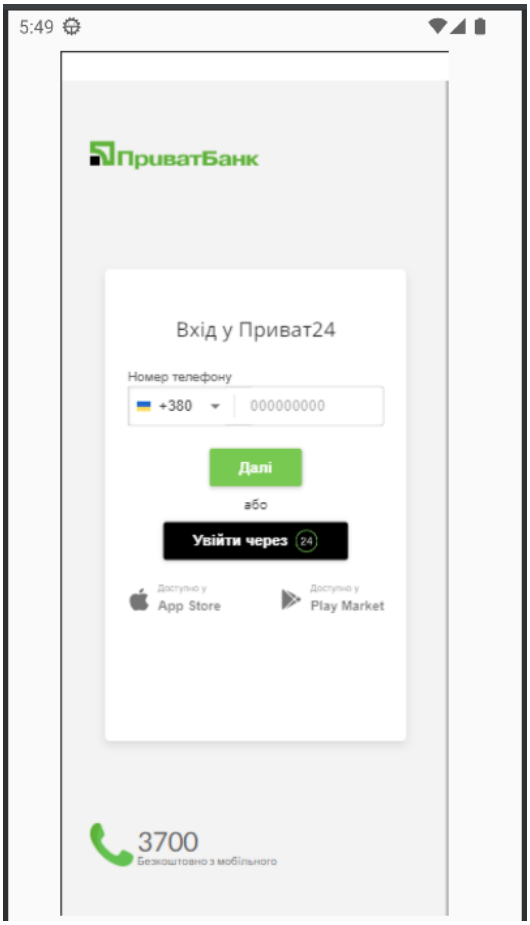


Рисунок 4.15 – Графічне відображення 4 кроку

Джерело: побудовано автором (знімок з екрану)

Крок 5. Для продовження запису необхідно авторизуватися у акаунт банку за допомогою мобільного додатку або шляхом введення даних користувача. Після користувача перенесе на сторінку погодження використання даних web-додатком «Bank ID» (рис. 4.16).

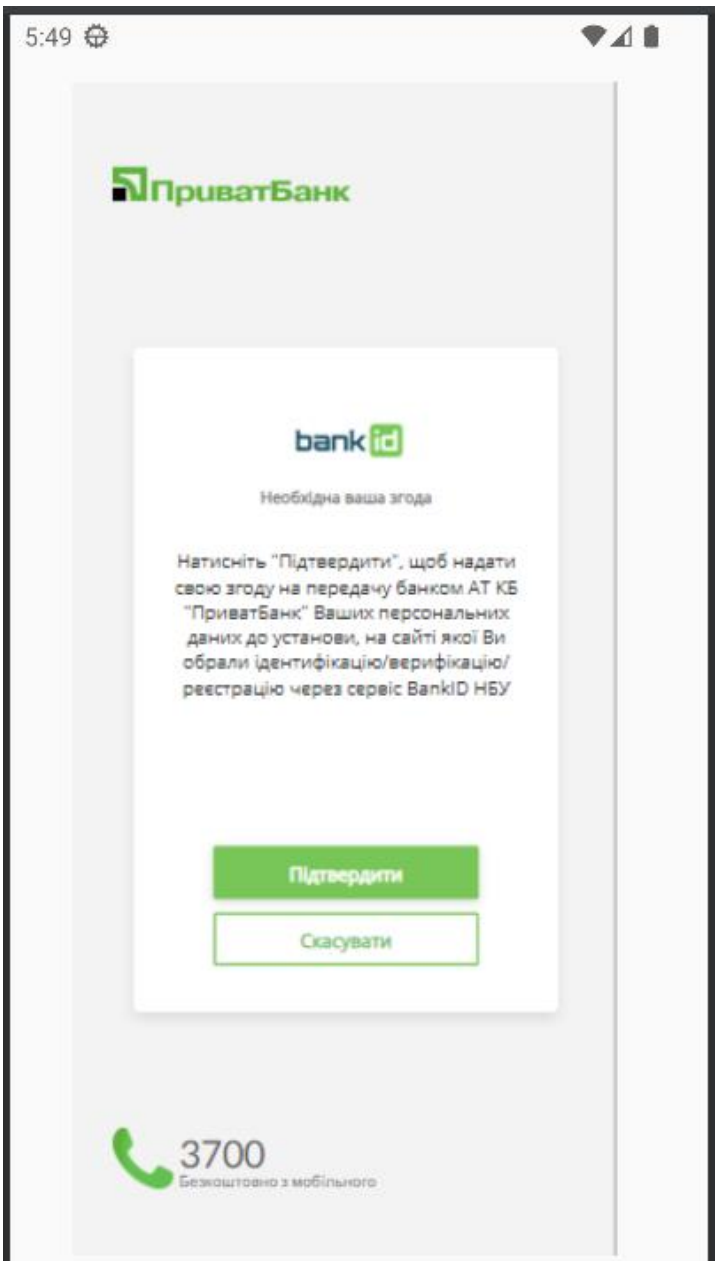


Рисунок 4.16 – Графічне відображення 5 кроку

Джерело: побудовано автором (знімок з екрану)

Крок 6. Після погодження користувача перенесе на сторінку web-додатку «Дія», де будуть відображені його персональні дані. Для продовження процесу необхідно погодитися з обробкою персональних даних та натиснути кнопку «Продовжити» (рис. 4.17).

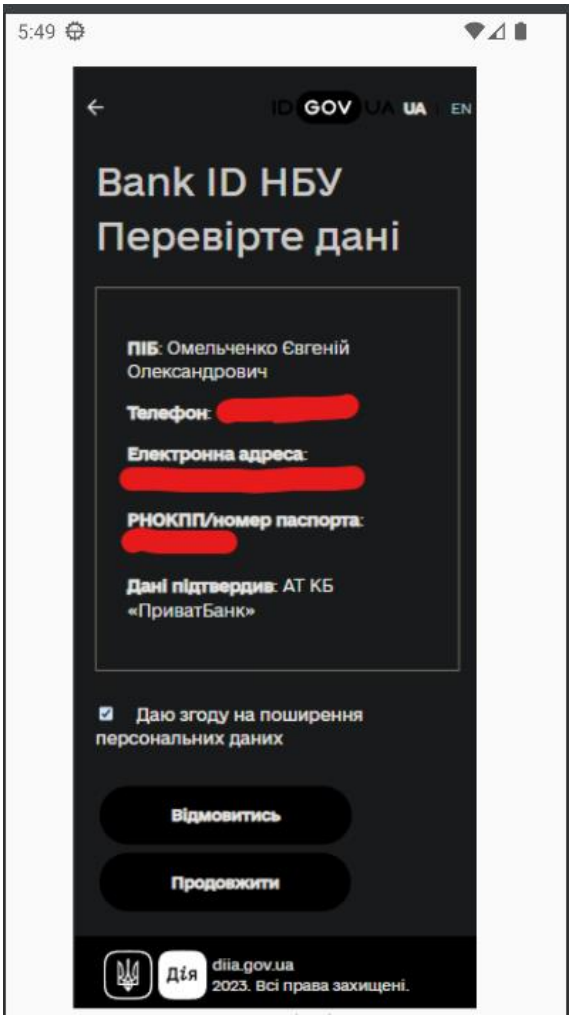


Рисунок 4.17 – Графічне відображення 6 кроку

Джерело: побудовано автором (знімок з екрану)

Крок 7. Погодившись на обробку даних, користувач потрапляє на сторінку з відображенням записів (якщо такі існують) та можливістю записатися. Для створення нового запису, необхідно натиснути кнопку «Записатися» та вибрати пункт «Теоретичний іспит». Після користувача перенесе на сторінку вибору дати складання іспиту. Здійснивши це, необхідно обрати найближчий сервісний центр і зручний час. На рисунках 4.18-4.21 зображені вище зазначені кроки.

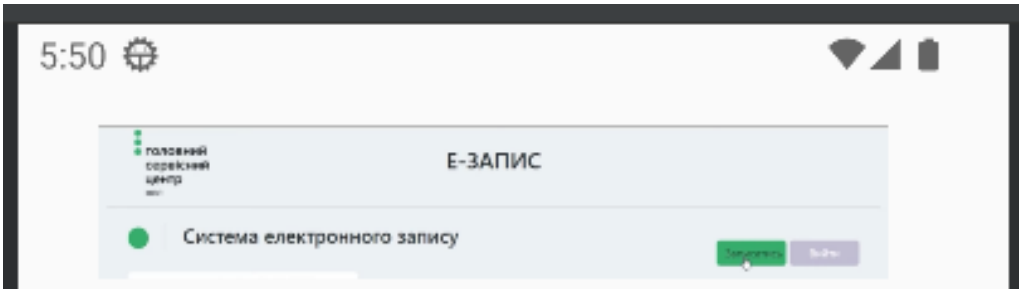


Рисунок 4.18 – Графічне відображення 7 кроку

Джерело: побудовано автором (знімок з екрану)

Рисунок 4.19 – Графічне відображення 7 кроку

Джерело: побудовано автором (знімок з екрану)

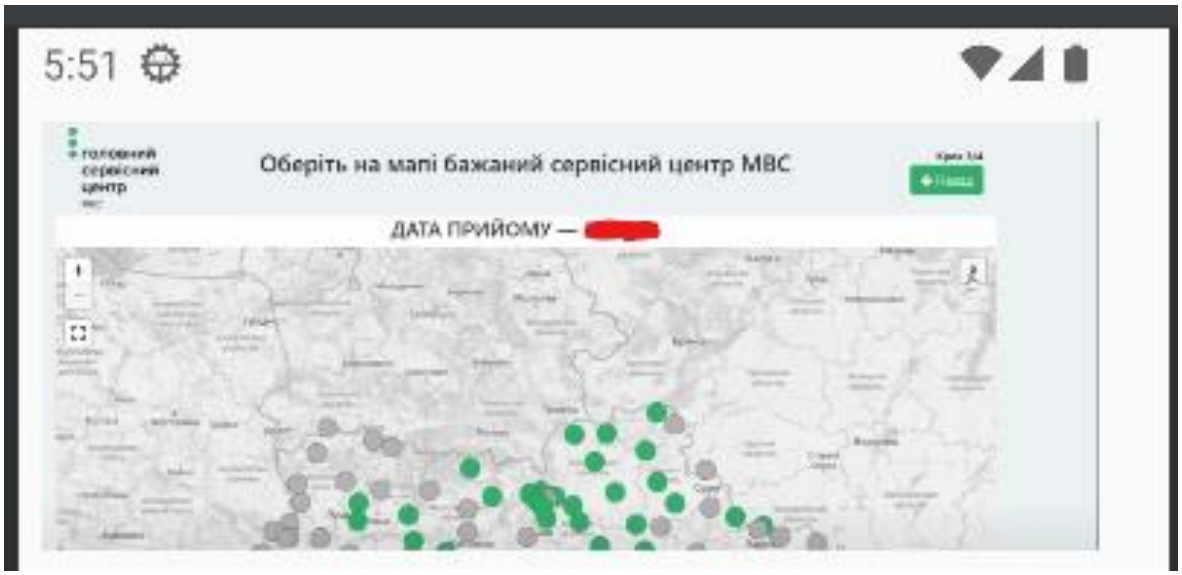


Рисунок 4.20 – Графічне відображення 7 кроку

Джерело: побудовано автором (знімок з екрану)

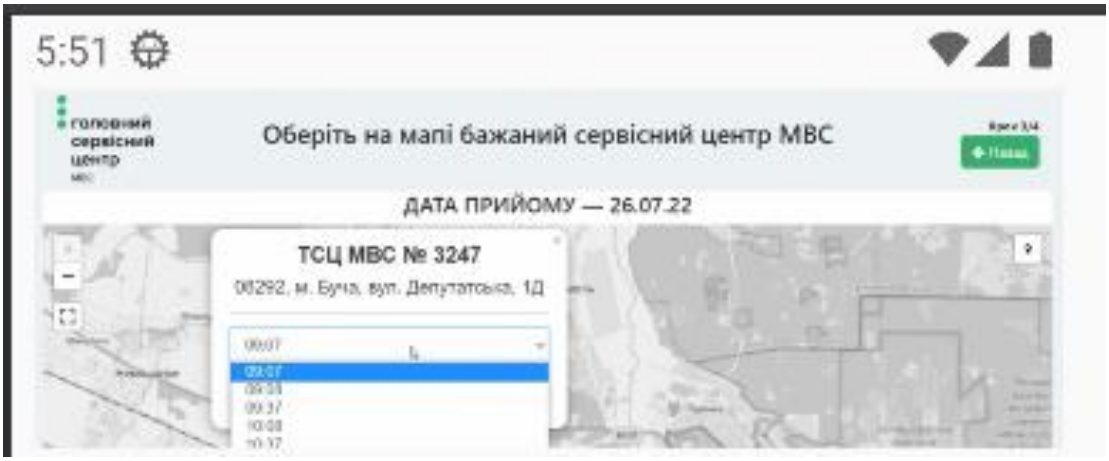


Рисунок 4.21 – Графічне відображення 7 кроку

Джерело: побудовано автором (знімок з екрану)

Крок 8. Далі необхідно підтвердити запис та за необхідності відмінити його або завантажити талон. На рисунках 4.22-4.23 зображені вище зазначені кроки.

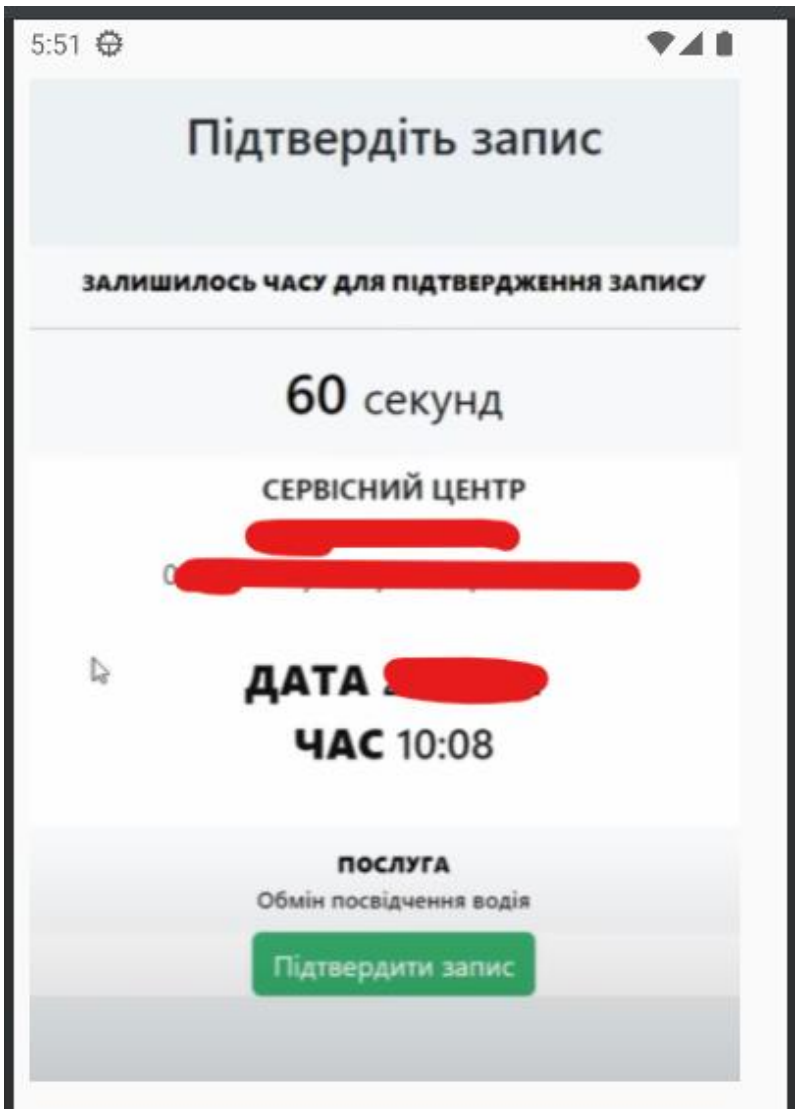


Рисунок 4.22 – Графічне відображення 8 кроку

Джерело: побудовано автором (знімок з екрану)

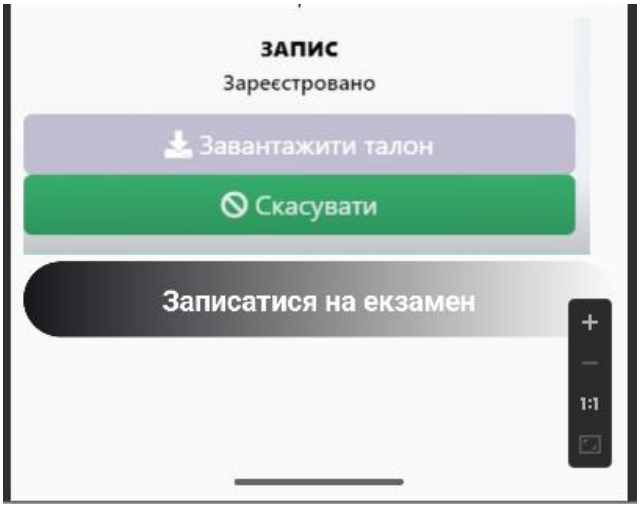


Рисунок 4.23 – Графічне відображення 8 кроку

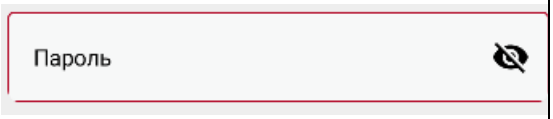
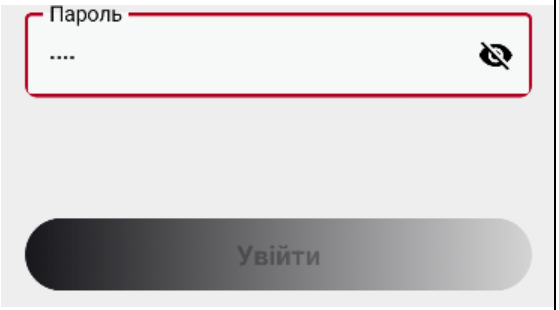
Джерело: побудовано автором (знімок з екрану)

4.4 Тестування мобільного додатку

Для перевірки коректності виконання описаного функціоналу мобільного додатку, необхідно провести функціональне тестування [24]. Воно включає верифікацію введених користувачем даних, коректність роботи модулів та посилань.

Спочатку необхідно перевірити працездатність мобільного додатку при введені невалідних значень. Тому треба дослідити роботу модуля реєстрації та авторизації. Результати тестування показали, що не дозволяється проводити дії з невалідними даними. Надалі було перевірено посилання на сторонні ресурси. Результати тестування показали, що користувача переводить до необхідних сторінок. На останок було проведено перевірку режимів тестування на коректність роботи. Вона показала коректну їхню роботу. Тестування мобільного додатку було проведено за допомогою методу «black box» [25]. Це означає перевірку працездатності модулів без прямого доступу до коду. Результати проведеного тестування надано в таблиці 4.1.

Таблиця 4.1 – Результати проведеного тестування мобільного додатку

№	Назва тесту	Очікуваний результат	Фактичний результат	0/1
1	Перевірка форми реєстрації шляхом введення коректних даних	Користувач успішно створений	Створений користувач відображається у базі даних.	1
2	Перевірка форми реєстрації шляхом введення некоректних даних	Некоректно введені поля підкреслені червоним та недоступна кнопка «Зареєструватися».		1
3	Перевірка форми авторизації шляхом введенням коректних даних	Успішна авторизація.	Відображення головного екрану мобільного додатку свідчити про успішну авторизацію.	1
4	Перевірка форми авторизації шляхом введенням некоректних даних	Після введення некоректних даних та натискання кнопки «Увійти», кнопка «Увійти» стає неактивною та поля з інформацією виділяються червоним, що свідчить про провалену авторизацію.		1
5	Перевірка посилань на сторонні ресурси	Успішний перехід на сторонні ресурси.	Відображення сторінки стороннього ресурсу.	1
6	Перевірка режиму тестування «Екзамен»	Перехід на екран тестування з випадковими питаннями	Відображення екрану режиму тестування «Екзамен».	1

Продовження табл. 4.1

№	Назва тесту	Очікуваний результат	Фактичний результат	0/1
7	Перевірка режиму тестування «Питання по темам»	Перехід на екран тестування з питаннями, які відносяться до обраної теми.	Відображення екрану режиму тестування «Питання по темам».	1
8	Перевірка переходу на сторінку запису для складання теоретичного іспиту	Успішний перехід на сторонній ресурс запису на складання теоретичного іспиту.	Відображення сторінки запису на складання теоретичного іспиту.	1

Джерело: побудовано автором

У результаті проведеного тестування не виявлено блокуючих дефектів. Увесь функціонал працює коректно.

ВИСНОВКИ

Пересування за допомогою автомобілів стало невід’ємною частиною життя сучасної людини, але далеко не кожен зараз має час на вивчення правил дорожнього руху, відвідуючи автошколу. Тому обрана тема кваліфікаційної роботи є актуальною.

Під час виконання дипломного проєкту було детально розглянуто предметну область. Визначено актуальність даного дослідження. Також проведено порівняльний аналіз мобільних додатків-аналогів та виділено їхні переваги та недоліки. Сформавано мету та задачі проєкту. Установлено функціональні та нефункціональні вимоги до запропонованого мобільного додатку.

До функціональних вимог можна віднести наступні:

- можливість вивчення правил дорожнього руху;
- можливість проходження тестів;
- можливість перегляду результату після проходження тесту;
- можливість запису на складання теоретичного іспиту в сервісному центрі.

Далі було створено діаграму IDEF0 для даного проєкту та її декомпозицію. Також надано можливі варіанти взаємодії з представленим мобільним додатком автоматизації процесу підготовлення до отримання водійського посвідення Це графічно відображено у Use Case діаграмі.

Наступним кроком було реалізовано мобільний додаток за допомогою мови програмування Kotlin з використанням фреймворку Jetpack Compose у середовищі програмування Android Studio. Створений програмний продукт повністю задовольняє поставлені під час виконання дипломного проєкту задачі.

Проведено тестування мобільного додатку. Його результати показали відсутність багів.

Отже, використання даного мобільного додатку дозволить вдосконалити процес підготовлення до отримання водійського посвідчення. Це можна досягти завдяки можливостям отримання необхідних знать шляхом вивчення правил дорожнього руху, проходженням тестів для самоперевірки, перегляду власного прогресу та запису на складання теоретичного іспиту в сервісному центрі. Також використання створеного програмного продукту дозволить зменшити навантаження на персонал даної державної установи. А також допоможе покращити безпекову ситуацію на дорогах, а саме підвищить рівень знань ПДР серед водіїв.

.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Мобільний додаток для бізнесу: за, проти та кому необхідно [Електронний ресурс] – Режим доступу до ресурсу:https://indigo.co.ua/ua/blog/mobilnoe-prilozhenie-dlya-biznesa-za-protiv-i-komu-neobxodimo

2. 9 кращих мобільних застосунків для програмістів [Електронний ресурс] – Режим доступу до ресурсу:https://highload.today/uk/9-krashhih-mobilnih-zastosunkiv-dlya-programistiv-i-ne-tilki/

3. How Is Mobile Learning More Beneficial Than Classroom Education? [Електронний ресурс] – Режим доступу до ресурсу: https://elearningindustry.com/how-is-mobile-learning-more-beneficial-than-classroom-education

4. Advantages & disadvantages of automated driving license tests in India [Електронний ресурс] – Режим доступу до ресурсу: https://www.team-bhp.com/news/advantages-disadvantages-automated-driving-license-tests-india

5. How we’re dealing with bots and the reselling of driving tests [Електронний ресурс] – Режим доступу до ресурсу: https://despatch.blog.gov.uk/2023/06/29/how-were-dealing-with-bots-and-the-reselling-of-driving-tests/

6. Mobile Learning Benefits: The Advantages Of Learning On-The-Go [Електронний ресурс] – Режим доступу до ресурсу: https://elearningindustry.com/mobile-learning-benefits-the-advantages-of-learning-on-the-go

7. Тести ПДР для Android та iOS [Електронний ресурс] – Режим доступу до ресурсу: https://vodiy.ua/mobile/

8. Driving Licence Practice Tests [Електронний ресурс] – Режим доступу до ресурсу:

https://play.google.com/store/apps/details?id=com.ProDataDoctor.DrivingLicencePracticeTests&hl=en_US

9. 9 Rules For UX Design [Електронний ресурс] – Режим доступу до ресурсу: https://usabilitygeek.com/rules-for-ux-principles/.

10. Томас Х. Кормен, Чарльз Е. Лейзерсон , Рональд Л. Рівест , Кліффорд Стайн, Вступ до алгоритмів, fourth edition, 2022 – 1312 с.

11. Олександр Косякофф, Стівен М. Бімер, Принципи та практика системної інженерії, 2020 – 688 с.

12. Ехтірам Раза Хан , Хума Анвар, Методи дослідження інформатики, Laxmi Publications, 2022 – 95 с.

13. Створення схем IDEF0 [Електронний ресурс] – Режим доступу до ресурсу: https://support.microsoft.com/uk-ua/office/%D1%81%D1%82%D0%B2%D0%BE%D1%80%D0%B5%D0%BD%D0%BD%D1%8F-%D1%81%D1%85%D0%B5%D0%BC-idef0-ea7a9289-96e0-4df8-bb26-a62ea86417fc.

14. Use Case Diagram Tutorial (Guide with Examples) [Електронний ресурс] – Режим доступу до ресурсу: https://creately.com/blog/diagrams/use-case-diagram-tutorial/.

15. StatCounter. Mobile Operating System Market Share Worldwide [Електронний ресурс] – Режим доступу до ресурсу: https://gs.statcounter.com/os-market-share/mobile/worldwide

16. Kotlin [Електронний ресурс] – Режим доступу до ресурсу: https://kotlinlang.org/

17. Which is better, Kotlin vs. java? [Електронний ресурс] – Режим доступу до ресурсу: https://dac.digital/kotlin-vs-java/#kotlin-vs-java-which-is-better

18. Android Studio [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/studio

19. Організація баз даних та знань [Електронний ресурс] – Режим доступу до ресурсу: https://elearning.sumdu.edu.ua/free_content/lectured:89b3d175c06a6b137e410cb14821d0e94549ad5a/20151030211833/44150/index.html

20. What Is MVVM Architecture? [Електронний ресурс] – Режим доступу до ресурсу: https://builtin.com/software-engineering-perspectives/mvvm-architecture

21. Build better apps faster with Jectpack Compose [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/jetpack/compose

22. Room Database with Kotlin [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/huawei-developers/room-database-with-kotlin-mvvm-architecture-477c3ad3c264

23. Coroutines [Електронний ресурс] – Режим доступу до ресурсу: https://kotlinlang.org/docs/coroutines-overview.html

24. Функціональне тестування [Електронний ресурс] – Режим доступу до ресурсу: https://qalight.ua/baza-znaniy/funktsionalne-testuvannya/

25. White/Black/Grey Box-тестування [Електронний ресурс] – Режим доступу до ресурсу: https://qalight.ua/baza-znaniy/white-black-grey-box-testirovanie/

26. SMART-постановка цілей, або як зробити будь-яке завдання здійсненним? [Електронний ресурс] – Режим доступу до ресурсу: https://cybershark.pro/uk/smart-postanovka-cilej/

27. What is a Work Breakdown Structure? [Електронний ресурс] – Режим доступу до ресурсу: https://www.workbreakdownstructure.com/

28. Organization Breakdown Structure (OBS) [Електронний ресурс] – Режим доступу до ресурсу: https://uplandsoftware.com/psa/resources/glossary/organization-breakdown-structure-obs/

29. Що таке діаграма Ганта та як нею правильно користуватися? [Електронний ресурс] – Режим доступу до ресурсу: https://nachasi.com/creative/2020/09/03/gantt-chart/

30. Ризики проєкту: аналіз, оцінка та стратегії управління [Електронний ресурс] – Режим доступу до ресурсу: https://skillsetter.io/blog/risk-management-ua

ДОДАТОК А

Планування робіт

Потреба в автоматизації процесу підготовлення до отримання водійського посвідчення збільшується з кожним днем через безперервний потік бажаючих володіти й керувати власним транспортним засобом. Наразі це питання не лише спрощення перебігу навчання та складання іспиту, а й поліпшення безпекової ситуації на дорогах. Тому завдяки впровадженню запропонованого мобільного додатку можна відчутно зменшити витрати часу на процес підготовлення до отримання водійського посвідчення завдяки можливості самостійного вивчення правил дорожнього руху, можливості проходження тестів для самоперевірки та подальшого запису на складання теоретичного іспиту.

Деталізація мети методом SMART[26]**.** Для успішної реалізації даного проєкту, необхідно правильно сформулювати його мету за допомогою SMART-методу. А саме: «Розробка мобільного додатку автоматизації процесу підготовлення до отримання водійського посвідчення на основі затвердженого технічного завдання для зменшення витрат часу та зусиль на вивчення правил дорожнього руху та покращення безпекову ситуацію на дорогах шляхом підвищення рівня знань ПДР серед водіїв до 15 грудня 2023 року».

Результати деталізації мети проєкту методом SMART представлено у таблиці А.1

Таблиця А.1 – Деталізація мети проєкту методом SMART

Specific (Конкретна)	Розробити мобільний додаток автоматизації процесу підготовлення до отримання водійського посвідчення, який містить такі функції, як можливість вивчення правил дорожнього руху, проходження тестів, перегляду результатів після проходження тестів та можливість запису на складання теоретичного іспиту в сервісному центрі.
Measurable (Вимірювана)	Результатом роботи над проєктом стане мобільний додаток автоматизації процесу підготовлення до отримання водійського посвідчення.
Achievable (Досяжна)	Для реалізації проєкту даного мобільного додатку було обрано середовище розробки Android Studio, в якості мови програмування – Java(Kotlin).
Relevant (Реалістична)	Зменшення витрат часу та зусиль на вивчення правил дорожнього руху за рахунок належної організації процесу самостійного підготовлення до отримання водійського посвідчення та допоможе покращити безпекову ситуацію на дорогах шляхом підвищення рівня знань ПДР серед водіїв.
Time-framed (Обмежена у часі)	Термін виконання робіт було визначено за допомогою двосторонньої домовленості між замовником та виконавцем – до 15 грудня 2023 року.

Джерело: побудовано автором

Планування змісту робіт. WBS (Work Breakdown Structure) [27] – це графічне відображення структури робіт проєкту. Вони поєднуються у єдине ціле з його продуктом за допомогою ієрархічного порядку. Розробка такої моделі спрямована на організацію командної роботи над проєктом. А також на ведення контролю та встановлення термінів виконання кожної частини.

Є перший, найвищий, рівень такої ієрархічної моделі. На ньому знаходиться кінцевий продукт проєкту. На другому рівні декомпозиції, описані певні необхідні заходи. Вони забезпечують досягнення мети даного проєкту. Декомпозиція поставлених робіт виконуються до того моменту, поки вказані елементи не стануть елементарними, такими, яким можна призначити відповідальну особу, термін реалізації і яка має чіткий результат. На рисунку А.1 представлено WBS на створення даного мобільного додатку

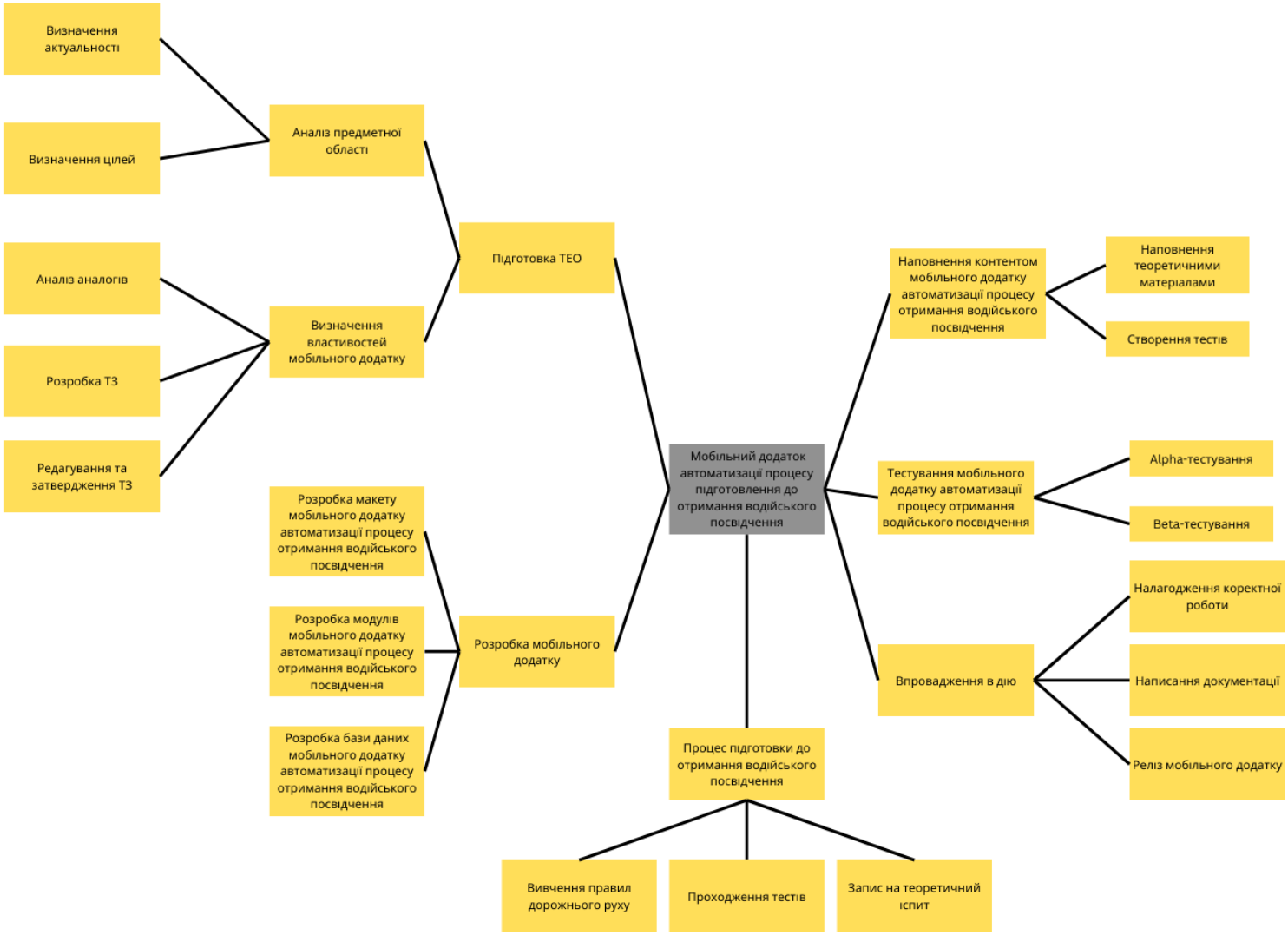


Рисунок А.1 – WBS-структура робіт проєкту

Джерело: побудовано автором

Планування структури виконавців. Після того, як було проведено декомпозицію робіт необхідно розробити організаційну структуру виконавців або OBS [28]. Organizational Breakdown Structure (OBS) – це графічно представлена ієрархічна модель, на якій зазначені відповідальні особи, яку беруть участь у реалізації проєкту, та призначені на виконання елементарних робіт. На рисунку А.2 представлено організаційну структуру на реалізацію даної розробки. Учасників проєкту описано у таблиці А.2.

Таблиця А.2 – Виконавці проєкту

Роль	Ім’я	Проєктна роль
Розробник	Омельченко Є.О.	Виконує розробку описаного функціоналу.
Проєктувальник	Омельченко Є.О.	Виконує проєктування бази даних та розробляє структуру мобільного додатку.
Тестувальник	Омельченко Є.О.	Відповідає за тестування кінцевої розробки.
Керівник проєкту	Антипенко В.П.	Формує завдання на розробку проєкту.
Менеджер проєкту	Антипенко В.П.	Відповідає за дотримання термінів, розподіл ресурсів та завдань між учасниками проєкту.

Джерело: побудовано автором

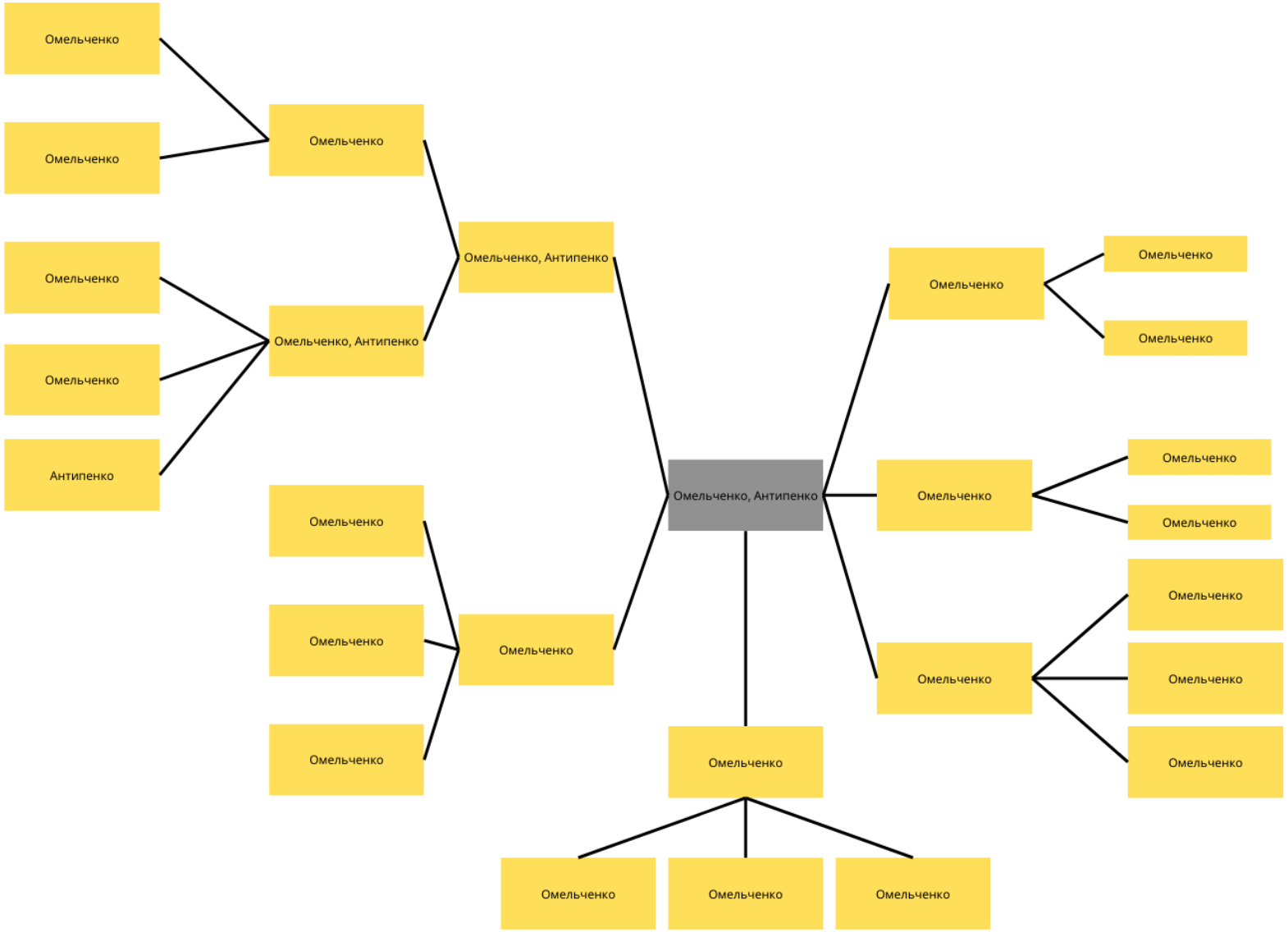


Рисунок А.2 – OBS-структура робіт проєкту

Джерело: побудовано автором

Діаграма Ганта. Наступним важливим етапом роботи є побудова діаграми Ганта [29]. Це розклад робіт із реальним розподілом дат. За допомогою цього графіка можна отримати уявлення про тривалість виконання різних завдань проєкту. Зазначаються обмеження в ресурсах і враховуються святи та вихідні дні.

Календарний графік даного проєкту представлено на рисунку А.3.

Рисунок А.3 – Календарний графік виконання робіт

Джерело: побудовано автором

Управління ризиками проєкту. Під час виконання проєкту необхідно, визначити ризики [30], які можуть виникнути, та відповідні заходи реагування на них. Далі провести їхню оцінку. Перелік можливих ризиків даного проєкту надано

в таблиці А.3 Результати їхньої оцінки представлено в таблиці А.4. Таблиця А.5 демонструє шкалу для класифікації ризиків за величиною впливу на проєкт та ймовірністю їх виникнення.

Таблиця А.3 – Ризики проєкту

№ ризику	**Назва (опис) ризику**
1	Неналагоджена комунікація між замовником та розробником.
2	Неправильно поставлена задача замовником.
3	Неякісно проведений менеджмент часу.
4	Використання невідповідних технологій .
5	Відсутність резервних копій.
6	Присутність серйозних багів, важких для виправлення.
7	Збої у роботі при використанні користувачем.
8	Недостатній моніторинг виконання проєкту.
9	Відсутній доступ до мережі Інтернет
10	Відсутність світла

Джерело: побудовано автором

Таблиця А.4 – Результати визначення ймовірності, впливу та рангу ризиків проєкту

№ ризику	**Назва (опис) ризику**	**Ймовірність (0,1-0,9)**	**Вплив (0,05-0,8)**	**Ранг**
1	Непорозуміння між замовником та розробниками	0,2	0,03	0,006
2	Погано поставлена задача замовником	0,4	0,06	0,024
3	Поганий менеджмент часу	0,3	0,2	0,06

Продовження табл. А.4

№ ризику	**Назва (опис) ризику**	**Ймовірність (0,1-0,9)**	**Вплив (0,05-0,8)**	**Ранг**
4	Використання непідходящих технологій	0,3	0,2	0,06
5	Відсутність резервних копій	0,4	0,2	0,08
6	Присутність багів	0,7	0,04	0,028
7	Збої у роботі при використанні користувачем	0,4	0,1	0,04
8	Поганий моніторинг виконання проєкту	0,5	0,03	0,015
9	Відсутній доступ до мережі Інтернет	0,3	0,08	0,024
10	Відсутність світла	0,2	0,05	0,01

Джерело: побудовано автором

Таблиця А.5 – Шкала оцінювання ризиків за ймовірністю виникнення на величиною впливу

Оцінка	Ймовірність виникнення	Вплив ризику	Тип ризику
1	Низька	Низький	Прийнятні
2	Середня	Середній	Виправдані
3	Висока	Високий	Недопустимі

Джерело: побудовано автором

Для зниження негативного впливу ризиків на роботу мобільного додатку, необхідно створити план реагування на них. Аналіз проводиться з даними, які знаходяться у таблиці А.4. На рисунку А.4 зображено кінцеву матрицю ризиків. Вона стала основою планування заходів реагування на ризики проєкту. Зеленим кольором на матриці позначають прийнятні ризики, жовтим – виправдані, а червоним – недопустимі.

Ймовірність виникнення ризику	**Вплив ризику**				
	0,05	**0,1**	**0,2**	**0,4**	**0,8**
0,9	0,045	0,09	0,18	0,36	0,72
0,7	0,035	0,07	0,14	0,28	0,56
0,5	0,025 **R2, R6**	0,05 **R10**	0,10	0,20	0,40
0,3	0,015 **R8**	0,03	0,06 **R3,R4**	0,12	0,24
0,1	0,005 **R1**	0,01	0,02	0,04 **R7**	0,08 **R5, R9**

Рисунок А.4 – Матриця ймовірності

Джерело: побудовано автором

У таблиці А.6 представлена класифікація ризиків за отриманим рангом. Ризики та стратегії реагування на них описані у таблиці А.7.

Таблиця А.6 – Шкала оцінювання за рівнем ризику

№	Назва	Межі	Ризики, які входять (номера)
1	Прийнятні	0,005≤R≤0,05	1,2,6,7,8,10
2	Виправдані	0,05≤R≤0,14	3,4,5,9
3	Недопустимі	0,14≤R≤0,72	

Джерело: побудовано автором

Таблиця А.7 – Ризики та стратегії реагування

ID	Статус ризику	Опис ризику	Ймовірність виникнення	Вплив ризику	Ранг ризику	Тип стратегії реагування	План А	План Б
RS_1	Відкритий	Погана комунікація між замовником та розробниками	Низька	Низький	0.006	Зменшення	1.Встановити позитивні взаємини між розробником і керівником. 2.Строго дотримуватися правил ділового етикету під час спілкування. 3.Створити сприятливі умови для ефективної співпраці.	Коли виникає непорозуміння, важливо розібратися в його причинах, обговорити їх та сприяти створенню позитивної атмосфери в колективі.
RS_2	Відкритий	Погано поставлена задача замовником	Низька	Низький	0.024	Зменшення	1.Чітко і зрозуміло обговорити всі вимоги з замовником. 2.Розробити глосарій для уникнення непорозумінь у використанні термінів та визначень. 3.Регулярний контроль з боку замовника на різних етапах виконання робіт.	При виявленні розбіжностей у характеристиках продукту порівняно з вимогами, важливо уважно і точно вказати, в чому полягає невідповідність, та внести необхідні корективи.

Продовження табл. А.7

ID	Статус ризику	Опис ризику	Ймовірність виникнення	Вплив ризику	Ранг ризику	Тип стратегії реагування	План А	План Б
RS_3	Відкритий	Поганий менеджмент часу	Низька	Високий	0.06	Зменшення	Здійснити оцінку актуальності ключових процесів та завдань, приділяючи особливу увагу ефективному управлінню часом. Визначити пріоритети в роботі та точно дотримуватися графіку виконання завдань у календарному плані.	Переставити порядок пріоритетів завдань. Розглянути можливі варіанти оптимізації роботи в рамках існуючого графіку. Оговорити з замовником можливі варіанти корекцій у терміни виконання.
RS_4	Відкритий	Використання непідходящих технологій	Низька	Середній	0.06	Зменшення	1.Провести аналіз методів та засобів для реалізації проєкту. 2.Вибрати технологію розробки, яка є зрозумілою та легкою у використанні.	Відвести час і ресурси на пошуки можливостей поліпшення використаної технології та скористатися допоміжними ресурсами.
RS_5	Відкритий	Відсутність резервних копій	Низька	Середній	0.08	Зменшення	1.Налаштувати автоматичне збереження даних. 2.Зберігати дані на різних носіях інформації.	Робити копію даних після кожного виконаного етапу.

Продовження табл. А.7

ID	Статус ризику	Опис ризику	Ймовірність виникнення	Вплив ризику	Ранг ризику	Тип стратегії реагування	План А	План Б
RS_6	Відкритий	Присутність багів	Низька	Низький	0.028	Зменшення	1.Провести тестування перед запуском 2. Звернутися до фахівця для виправлення несправностей	Замінити програмне забезпечення.
RS_7	Відкритий	Збої у роботі при використанні користувачем	Низька	Низький	0.04	Зменшення	1.Розробка проєкту з урахуванням вимог до програмного забезпечення користувачів проєкту.	Модифікація проєкту з урахуванням різних версій програмного забезпечення, яке буде застосовуватися.
RS_8	Відкритий	Поганий моніторинг виконання проєкту	Низька	Низький	0.015	Зменшення	Здійснювати проміжний контроль результатів в ході виконання проєкту. Здійснювати моніторинг проєкту працівниками.	Здійснювати моніторинг проєкту замовником. Надання проміжних результатів виконання проєкту після кожного етапу.

Продовження табл. А.7

ID	Статус ризику	Опис ризику	Ймовірність виникнення	Вплив ризику	Ранг ризику	Тип стратегії реагування	План А	План Б
RS_9	Відкритий	Відсутній доступ до мережі Інтернет	Низька	Середній	0,08	Зменшення	Мати сім-карти для підключення мобільного Інтернету.	Перейти у місце з доступом до мережі Інтернет.
RS_10	Відкритий	Відсутність світла	Низька	Низький	0,05	Ухилення	Мати пристрій для автономного живлення (паувербанк, генератор тощо).	Знайти місце де є світло.

Джерело: побудовано автором

ДОДАТОК Б

Лістинг основних модулів

Файл Main.kt

```
import androidx.compose.animation.Crossfade
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.material.Surface
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import com.testdai.core.navigation.DriverLicenseExamNavGraph
import com.testdai.core.navigation.LoginAppRoter
import com.testdai.core.navigation.Screen
import com.testdai.core.theme.Theme
import com.testdai.core.theme.ThemePreferences
import com.testdai.ui.activity.DriverLicenseExamApp
import com.testdai.ui.activity.MainActivity
import com.testdai.ui.screen.chose.ChooseScreen
import com.testdai.ui.screen.exam.ExamScreen
import com.testdai.ui.screen.home.HomeScreen
import com.testdai.ui.screen.login.LoginScreen
import com.testdai.ui.screen.login.SignUpScreen
```

```
@Composable
fun Main(){
    Surface (
        modifier = Modifier.fillMaxSize(),
        ){
        Crossfade(targetState      =      LoginAppRoter.currentScreen)
{currentState ->
            when(currentState.value){
                is Screen.SignUpScreen ->{
                    SignUpScreen()
                }
                is Screen.LoginScreen ->{
                    LoginScreen()
                }
                is Screen.ChooseScreen ->{
                    ChooseScreen()
                }
                is Screen.ChooseScreen ->{
                    ExamScreen()
                }
                is Screen.Main->{
                    HomeScreen()
                }
            }
```

```
        }

    }

}
```

Файл Screen.kt

```
import androidx.compose.runtime.MutableState
import androidx.compose.runtime.mutableStateOf

sealed class Screen (){
    object SignUpScreen : Screen()
    object LoginScreen: Screen()
    object Main: Screen()
    object ChooseScreen: Screen()
    object ExamScreen: Screen()
}


object LoginAppRoter{
    var currentScreen: MutableState<Screen> =
mutableStateOf(Screen.SignUpScreen)

    fun navigateTo(destination: Screen){
        currentScreen.value = destination
    }
}
```

Файл UserDao.kt

```
import androidx.lifecycle.LiveData
import androidx.room.Dao
import androidx.room.Delete
import androidx.room.Insert
import androidx.room.Query


@Dao
interface UsersDao {

    @Insert
    suspend fun insert(register: Users)
```

```
    @Query("SELECT * FROM users ORDER BY user_id DESC")
    fun getAllUsers(): LiveData<List<Users>>

    @Query("DELETE FROM users")
    suspend fun deleteAll(): Int

    @Query("SELECT * FROM users WHERE email LIKE :userName")
    suspend fun getUsername(userName: String): Users?

}
```

Файл UserEntity.kt

```
import androidx.room.ColumnInfo
import androidx.room.Entity
import androidx.room.PrimaryKey

@Entity(tableName = "users")
data class Users(

    @PrimaryKey(autoGenerate = true)
    var user_id: Int = 0,

    @ColumnInfo(name = "first_name")
    var firstName: String,

    @ColumnInfo(name = "last_name")
    var lastName: String,

    @ColumnInfo(name = "email")
    var email: String,

    @ColumnInfo(name = "password_text")
    var passwrd: String
)
```

Файл SignUpViewModel.kt

```
import android.util.Log
import android.widget.Toast
import androidx.compose.runtime.mutableStateOf
import androidx.lifecycle.ViewModel
```

```
import com.testdai.core.navigation.LoginAppRoter
import com.testdai.core.navigation.Screen
import com.testdai.core.users.DataBaseHelper
import com.testdai.rules.Validator


class SignUpViewModel : ViewModel() {


    private val TAG = SignUpViewModel::class.simpleName
var userDetailsLiveData = MutableLiveData<Array<UserEntity>>()

    var registrationUIState = mutableStateOf(RegistrationUIState())

    var allValidationsPassed = mutableStateOf(false)

    var signUpInProgress = mutableStateOf(false)

    fun onEvent (event:SignUpUIEvent){
        //db = DataBaseHelper(this)
        validateDataWithRules()
        when(event){
            is SignUpUIEvent.FirstNameChanged -> {
                registrationUIState.value =
registrationUIState.value.copy(
                    firstName = event.firstName
                )

                printState()
            }
            is SignUpUIEvent.LastNameChanged -> {
                registrationUIState.value =
registrationUIState.value.copy(
                    lastName = event.lastName
                )
                printState()
            }
            is SignUpUIEvent.EmailChanged -> {
                registrationUIState.value =
registrationUIState.value.copy(
                    email = event.email
                )
                printState()
            }
            is SignUpUIEvent.PasswordChanged -> {
                registrationUIState.value =
registrationUIState.value.copy(
```

```
                password = event.password
            )
            printState()
        }
        is SignUpUIEvent.RegisterButtonClicked ->{
            signUP()
        }

    }
}

private fun signUP() {
    Log.d(TAG,"Inside_signup")
    printState()


    createUser(
        email = registrationUIState.value.email,
        password = registrationUIState.value.password
    )
}

private fun validateDataWithRules() {
    val fNameResult = Validator.validateFirstName(
        fName = registrationUIState.value.firstName
    )

    val lNameResult = Validator.validateLastName(
        lName = registrationUIState.value.lastName
    )

    val emailResult = Validator.validateEmail(
        email = registrationUIState.value.email
    )

    val passwordResult = Validator.validatePassword(
        password = registrationUIState.value.password
    )


    registrationUIState.value = registrationUIState.value.copy(
        firstNameError = fNameResult.status,
        lastNameError = lNameResult.status,
        emailError = emailResult.status,
        passwordError = passwordResult.status,
    )
```

```
        allValidationsPassed.value = fNameResult.status &&
lNameResult.status && emailResult.status && passwordResult.status

    }

    private fun printState(){
        Log.d(TAG,"Inside_printstate")
        Log.d(TAG,registrationUIState.value.toString())
    }

    fun createUser(first_name:String, last_name:String, email:String,
password:String){
        Log.i("MYTAG", userDetailsLiveData.value.toString() +
"ASDFASDFASDFASDF")
                    Log.i("MYTAG", "OK im in")
                    val firstName = inputFirstName.value!!
                    val lastName = inputLastName.value!!
                    val email = inputUsername.value!!
                    val password = inputPassword.value!!
                    Log.i("MYTAG", "insidi Sumbit")
                    insert(RegisterEntity(0, firstName, lastName,
email, password))
                    inputFirstName.value = null
                    inputLastName.value = null
                    inputUsername.value = null
                    inputPassword.value = null
                    _navigateto.value = true
    }

}
```

Файл LoginScreen.kt

```
import androidx.compose.foundation.layout.*
import androidx.compose.material.Surface
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.res.stringResource
import androidx.compose.ui.tooling.preview.Preview
import androidx.compose.ui.unit.dp
import androidx.lifecycle.viewmodel.compose.viewModel
import com.testdai.R
import com.testdai.core.navigation.AppRoter
import com.testdai.core.navigation.Screen
```

```
import com.testdai.data.LoginUIEvent
import com.testdai.data.LoginViewModal
import com.testdai.data.SignUpViewModel
import com.testdai.data.SignUpUIEvent
import com.testdai.widget.*

@Composable

fun LoginScreen (loginViewModal: LoginViewModal = viewModel()) {
    Surface(
        modifier = Modifier
            .fillMaxSize()
            .padding(28.dp)
    )
    {
        Column(modifier = Modifier.fillMaxSize()) {

            HeadingTextComponent(value = stringResource(id =
R.string.login_message))
            Spacer(modifier = Modifier.height(20.dp))
            MyTextField(labelValue = stringResource(id =
R.string.email),onTextSelected = {
                loginViewModal.onEvent(LoginUIEvent.EmailChanged(it))
            },
                errorStatus =
loginViewModal.loginUIState.value.emailError)
            PasswordTextField(labelValue = stringResource(id =
R.string.password),onTextSelected = {

loginViewModal.onEvent(LoginUIEvent.PasswordChanged(it))
            },
                errorStatus
=loginViewModal.loginUIState.value.passwordError)
            Spacer(modifier = Modifier.height(80.dp))
            ButtonComponent(value = stringResource(id =
R.string.login_btn), onButtonClicked = {



loginViewModal.onEvent(LoginUIEvent.LoginButtonClicked)
            },
            isEnabled = loginViewModal.allValidationsPassed.value)
            Spacer(modifier = Modifier.height(10.dp))
            DividerComponent()
            ClickableLoginTextComponent(tryingToLogin = false,
onTextSelected = {
                LoginAppRoter.navigateTo(Screen.SignUpScreen)
```

```
            })
        }


    }
}

@Preview
@Composable
fun DefaultPreviewOfLoginScreen(){
    LoginScreen()
}
```

Файл SignUpScreen.kt

```
import androidx.compose.foundation.layout.*
import androidx.compose.material.CircularProgressIndicator
import androidx.compose.material.Surface
import androidx.compose.runtime.Composable
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.res.stringResource
import androidx.compose.ui.tooling.preview.Preview
import androidx.compose.ui.unit.dp
import androidx.lifecycle.viewmodel.compose.viewModel
import com.testdai.R
import com.testdai.core.navigation.LoginAppRoter
import com.testdai.core.navigation.Screen
import com.testdai.data.SignUpViewModel
import com.testdai.data.SignUpUIEvent
import com.testdai.widget.*

@Composable
fun SignUpScreen (signUpViewModel: SignUpViewModel = viewModel()){
    Box(modifier = Modifier.fillMaxSize(), contentAlignment =
Alignment.Center) {
        Surface(
            modifier = Modifier
                .fillMaxSize()
                .padding(28.dp)
        ) {
            Column(modifier = Modifier.fillMaxSize()) {
                //NormalTextComponent(value = "Hello there")
                HeadingTextComponent(value = stringResource(id =
R.string.signup_message))
```

```
                Spacer(modifier = Modifier.height(20.dp))
                MyTextField(
                    labelValue = stringResource(id =
R.string.firstName), onTextSelected = {

signUpViewModel.onEvent(SignUpUIEvent.FirstNameChanged(it))
                    },
                    errorStatus =
signUpViewModel.registrationUIState.value.firstNameError
                )
                MyTextField(
                    labelValue = stringResource(id =
R.string.lastName), onTextSelected = {

signUpViewModel.onEvent(SignUpUIEvent.LastNameChanged(it))
                    },
                    errorStatus =
signUpViewModel.registrationUIState.value.lastNameError
                )
                MyTextField(
                    labelValue = stringResource(id = R.string.email),
onTextSelected = {

signUpViewModel.onEvent(SignUpUIEvent.EmailChanged(it))
                    },
                    errorStatus =
signUpViewModel.registrationUIState.value.emailError
                )
                PasswordTextField(
                    labelValue = stringResource(id =
R.string.password), onTextSelected = {

signUpViewModel.onEvent(SignUpUIEvent.PasswordChanged(it))
                    },
                    errorStatus =
signUpViewModel.registrationUIState.value.passwordError
                )
                /*CheckboxComponent(value = "You aggre to proced
further", onCheckedChange = {

signUpViewModel.onEvent(SignUpUIEvent.agreeBoxClicked(it))
            })*/
                Spacer(modifier = Modifier.height(80.dp))
                ButtonComponent(
                    value = stringResource(id = R.string.signup_btn),
onButtonClicked = {
```

```
signUpViewModel.onEvent(SignUpUIEvent.RegisterButtonClicked)
                    },
                    isEnabled =
signUpViewModel.allValidationsPassed.value
                )

                Spacer(modifier = Modifier.height(10.dp))
                DividerComponent()
                ClickableLoginTextComponent(tryingToLogin = true,
onTextSelected = {
                    LoginAppRoter.navigateTo(Screen.LoginScreen)
                })
            }

        }
        if(signUpViewModel.signUpInProgress.value) {
            CircularProgressIndicator()
        }
    }
}

@Preview
@Composable
fun DefaultPreviewOfSignUpScreen(){
    SignUpScreen()
}
```

Файл ExamScreen.kt

```
import androidx.activity.compose.BackHandler
import androidx.compose.foundation.*
import
androidx.compose.foundation.gestures.snapping.rememberSnapFlingBehavi
or
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.LazyRow
import androidx.compose.foundation.lazy.rememberLazyListState
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.material.*
import androidx.compose.material3.ButtonDefaults
import androidx.compose.material3.OutlinedButton
import androidx.compose.material3.Text
```

```
import androidx.compose.runtime.*
import androidx.compose.runtime.livedata.observeAsState
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.clip
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.platform.LocalConfiguration
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.res.colorResource
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.res.stringResource
import androidx.compose.ui.text.style.TextAlign
import androidx.compose.ui.tooling.preview.Preview
import androidx.compose.ui.unit.dp
import androidx.lifecycle.viewmodel.compose.viewModel
import coil.compose.AsyncImagePainter
import coil.compose.rememberAsyncImagePainter
import coil.request.ImageRequest
import coil.size.Size
import com.testdai.R
import com.testdai.model.AnswerModel
import com.testdai.model.QuestionModel
import com.testdai.model.State
import com.testdai.ui.bottom.result.ResultBottomSheet
import com.testdai.ui.screen.exam.data.ExamScreenState
import com.testdai.ui.screen.exam.data.Toolbar
import com.testdai.ui.theme.*
import com.testdai.utils.dpToPx
import com.testdai.widget.ButtonWidget
import com.testdai.widget.ToolbarWidget
import kotlinx.coroutines.launch

@OptIn(ExperimentalMaterialApi::class)
@Composable
fun ExamScreen(
    viewModel: ExamViewModel = viewModel(factory =
ExamViewModel.Factory)
) {

    val toolbar by viewModel.toolbar.observeAsState(Toolbar.Noop)
    val examState by viewModel.exam.observeAsState(State.Loading())
    val timer by
viewModel.timer.observeAsState(viewModel.initialTime)

    val sheetState = rememberModalBottomSheetState(
        initialValue = ModalBottomSheetValue.Hidden,
```

```
        skipHalfExpanded = true,
        confirmStateChange = { it !=
ModalBottomSheetValue.HalfExpanded }
    )
    val coroutineScope = rememberCoroutineScope()

    fun hideBottomSheet() {
        coroutineScope.launch {
            sheetState.hide()
        }
    }

    BackHandler(sheetState.isVisible) {
        hideBottomSheet()
    }

    val toolbarTitle = when (val t = toolbar) {
        Toolbar.Exam -> stringResource(id = R.string.toolbar_exam)
        Toolbar.Training -> stringResource(id =
R.string.toolbar_training)
        is Toolbar.Topic -> t.name
        Toolbar.Noop -> ""
    }

    ModalBottomSheetLayout(
        modifier = Modifier.fillMaxSize(),
        sheetState = sheetState,
        sheetContent = {
            ResultBottomSheet(viewModel) {
                viewModel.loadQuestions()
                hideBottomSheet()
            }
        },
        sheetBackgroundColor = MaterialTheme.colors.background,
        sheetShape = RoundedCornerShape(topStart = 15.dp, topEnd =
15.dp),
    ) {
        Column(
            modifier = Modifier
                .fillMaxSize()
                .background(MaterialTheme.colors.background)
        ) {
            ToolbarWidget(toolbarTitle = toolbarTitle) {
                if (timer.isNotBlank()) {
                    Text(
                        modifier = Modifier
                            .padding(16.dp)
```

```
                            .wrapContentHeight(align =
Alignment.CenterVertically),
                        style = MaterialTheme.typography.body2,
                        textAlign = TextAlign.Center,
                        text = timer,
                        color = MaterialTheme.colors.primary
                    )
                }
            }
            when (val exam = examState) {
                is State.Loading -> LoadingState()
                is State.Success -> {
                    if (exam.data.isActiveExam.not()) {
                        LaunchedEffect(Unit) {
                            coroutineScope.launch {
                                sheetState.show()
                            }
                        }
                    }
                    ExamState(
                        state = exam.data,
                        onQuestionClick = {
                            viewModel.onQuestionClick(it)
                        },
                        onAnswerClick = {
                            viewModel.onAnswerClick(it)
                        }
                    )
                }
                is State.Error -> exam.data?.let { state ->
                    ExamState(
                        state = state,
                        onQuestionClick = {
                            viewModel.onQuestionClick(it)
                        },
                        onAnswerClick = {
                            viewModel.onAnswerClick(it)
                        }
                    )
                } ?: ErrorState()
            }
        }
    }

}

@Composable
```

```
fun LoadingState() {
    Placeholder(stringResource(id = R.string.placeholder_loading)) {
        CircularProgressIndicator()
    }
}

@Composable
fun ErrorState() {
    Placeholder(stringResource(id = R.string.placeholder_error)) {
        Image(
            painter = painterResource(R.drawable.ic_warning),
            contentDescription = ""
        )
    }
}

@OptIn(ExperimentalFoundationApi::class)
@Composable
fun ExamState(
    state: ExamScreenState,
    onQuestionClick: (QuestionModel) -> Unit = {},
    onAnswerClick: (AnswerModel) -> Unit = {}
) {
    val configuration = LocalConfiguration.current

    val screenWidth = configuration.screenWidthDp.dp.dpToPx()

    val horizontalPadding = 16.dp
    val horizontalPaddingPx = horizontalPadding.dpToPx()
    val horizontalSpacing = 8.dp
    val horizontalSpacingPx = horizontalSpacing.dpToPx()

    val listState = rememberLazyListState()
    val flingBehavior = rememberSnapFlingBehavior(listState)

    val questions = state.questions
    val question = state.question
    val answers = question.answers

    LaunchedEffect(key1 = question) {
        questions.indexOf(question)
            .takeIf { it != 0 }
            ?.let {  index ->
                val offset = -screenWidth / 2 + horizontalPaddingPx *
2 + horizontalSpacingPx / 2
                listState.animateScrollToItem(index,  offset)
            }
```

```
    }

    val painter = rememberAsyncImagePainter(
        model = ImageRequest.Builder(LocalContext.current)
            .data(question.image)
            .size(Size.ORIGINAL)
            .build()
    )

    Column(
        modifier = Modifier
            .fillMaxSize(),
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        LazyRow(
            state = listState,
            modifier = Modifier
                .fillMaxWidth()
                .padding(vertical = 4.dp),
            horizontalArrangement =
Arrangement.spacedBy(horizontalSpacing),
            contentPadding = PaddingValues(horizontal =
horizontalPadding),
            flingBehavior = flingBehavior
        ) {
            items(count = questions.size, itemContent = { index ->
                Question(
                    question = questions[index],
                    position = index.plus(1),
                    onClick = {
                        onQuestionClick(questions[index])
                    }
                )
            })
        }
        Text(
            modifier = Modifier
                .padding(vertical = 8.dp, horizontal = 16.dp)
                .fillMaxWidth()
                .wrapContentHeight(),
            style = MaterialTheme.typography.h2,
            textAlign = TextAlign.Start,
            text = question.text,
            color = MaterialTheme.colors.primary
        )
        if (question.hasImage && painter.state is
AsyncImagePainter.State.Success) {
```

```
            val painterState = painter.state as
AsyncImagePainter.State.Success
            val ratio = painterState.result.drawable.let {
                it.intrinsicWidth.toFloat() / it.intrinsicHeight
            }
            Image(
                painter = painter,
                modifier = Modifier
                    .padding(horizontal = 16.dp, vertical = 4.dp)
                    .fillMaxWidth()
                    .wrapContentHeight()
                    .aspectRatio(ratio)
                    .clip(RoundedCornerShape(8.dp))
                    .heightIn(min = 150.dp, max = 300.dp),
                contentDescription = null,
                contentScale = ContentScale.Fit
            )
        }
        LazyColumn(
            modifier = Modifier
                .fillMaxWidth()
                .fillMaxHeight()
                .padding(horizontal = 8.dp),
            verticalArrangement = Arrangement.spacedBy(8.dp),
            contentPadding = PaddingValues(8.dp)
        ) {
            items(count = answers.size, itemContent = { index ->
                val answer = answers[index]
                Answer(
                    question = question,
                    answer = answer,
                    onClick = {
                        onAnswerClick(answer)
                    }
                )
            })
        }
    }
}


@Composable
fun Question(
    question: QuestionModel,
    position: Int,
    onClick: () -> Unit = {}
) {
```

```
    val textColor = when {
        question.answered -> Black
        else -> MaterialTheme.colors.primary
    }
    val borderColor = when {
        question.selected -> MaterialTheme.colors.primary
        else -> Color.Gray
    }
    val containerColor = when {
        question.answered && question.correct -> Skeptic
        question.answered -> LavenderBlush
        else -> MaterialTheme.colors.secondary
    }
    Row(
        verticalAlignment = Alignment.CenterVertically,
    ) {
        ButtonWidget(
            modifier = Modifier.size(40.dp),
            borderColor = borderColor,
            containerColor = containerColor,
            roundedCornerSize = 8.dp,
            contentPadding = PaddingValues(0.dp),
            style = MaterialTheme.typography.h2,
            text = position.toString(),
            textColor = textColor,
            onClick = onClick
        )
    }
}

@Composable
fun Answer(
    question: QuestionModel,
    answer: AnswerModel,
    onClick: () -> Unit = {}
) {
    val textColor = when {
        question.answered && answer.correct -> Shark
        answer.answered -> Shark
        else -> MaterialTheme.colors.primary
    }
    val borderColor = Color.Gray
    val containerColor = when {
        question.answered && answer.correct -> Skeptic
        answer.answered -> LavenderBlush
        else -> MaterialTheme.colors.secondary
    }
```

```
    OutlinedButton(
        modifier = Modifier
            .fillMaxWidth()
            .wrapContentHeight(),
        shape = RoundedCornerShape(12.dp),
        border = BorderStroke(1.dp, borderColor),
        colors = ButtonDefaults.outlinedButtonColors(
            containerColor = containerColor
        ),
        contentPadding = PaddingValues(horizontal = 8.dp, vertical =
4.dp),
        onClick = onClick
    ) {
        Text(
            modifier = Modifier
                .fillMaxSize()
                .padding(horizontal = 8.dp, vertical = 4.dp),
            style = MaterialTheme.typography.body1,
            textAlign = TextAlign.Start,
            text = answer.text,
            color = textColor
        )
    }
}

@Composable
fun Placeholder(
    message: String,
    content: @Composable ColumnScope.() -> Unit
) {
    Column(
        modifier = Modifier
            .fillMaxSize(),
        horizontalAlignment = Alignment.CenterHorizontally,
        verticalArrangement = Arrangement.Center
    ) {
        content()
        Text(
            modifier = Modifier
                .padding(16.dp)
                .fillMaxWidth()
                .wrapContentHeight(),
            style = MaterialTheme.typography.subtitle1,
            textAlign = TextAlign.Center,
            text = message,
            color = colorResource(id = R.color.alabaster)
        )
```

```
    }
}

@Preview
@Composable
fun LoadingStatePreview() {
    LoadingState()
}

@Preview
@Composable
fun ErrorStatePreview() {
    ErrorState()
}
```

Файл LoginViewModel.kt

```
import android.util.Log
import android.widget.Toast
import androidx.compose.runtime.mutableStateOf
import androidx.compose.ui.res.stringResource
import androidx.lifecycle.ViewModel
import com.google.firebase.auth.FirebaseAuth
import com.testdai.R
import com.testdai.core.navigation.LoginAppRoter
import com.testdai.core.navigation.Screen
import com.testdai.rules.Validator

class LoginViewModal : ViewModel(){
    private val TAG = SignUpViewModel::class.simpleName

    var loginUIState = mutableStateOf(LoginUIState())

    var allValidationsPassed = mutableStateOf(false)

    var loginInProgress = mutableStateOf(false)

    fun onEvent(event:LoginUIEvent){
        when(event){
            is LoginUIEvent.EmailChanged -> {
                loginUIState.value = loginUIState.value.copy(
                    email = event.email
                )
            }
```

```
            is LoginUIEvent.PasswordChanged -> {
                loginUIState.value = loginUIState.value.copy(
                    password = event.password
                )
            }

            is LoginUIEvent.LoginButtonClicked ->{

                logIn()
            }
        }
        validateLoginDataWithRules()
    }


    private fun logIn() {

        val email = loginUIState.value.email
        val password = loginUIState.value.password
                if (users != null) {
                    if(users.passwrd == password){
                        email = null
                        password = null
                        AppRoter.navigateTo(Screen.Main)
                   }else{
                           Log.d(TAG,"Inside_login_failure")
                    }
    }
    }

    private fun validateLoginDataWithRules (){
        val emailResult = Validator.validateEmail(
            email = loginUIState.value.email
        )

        val passwordResult = Validator.validatePassword(
            password = loginUIState.value.password
        )

        loginUIState.value = loginUIState.value.copy(
            emailError = emailResult.status,
            passwordError = passwordResult.status,
        )

        allValidationsPassed.value = emailResult.status &&
passwordResult.status
```

```
    }

}
```

Файл HomeScreen.kt

```
@OptIn(ExperimentalMaterialApi::class)
@Composable
fun HomeScreen(

    viewModel: HomeViewModel = viewModel(),
) {



    val examMode: ExamModeState by
viewModel.examMode.observeAsState(ExamModeState())

    val sheetState = rememberModalBottomSheetState(
        initialValue = ModalBottomSheetValue.Hidden,
        skipHalfExpanded = true,
        confirmStateChange = { it !=
ModalBottomSheetValue.HalfExpanded }
    )
    val coroutineScope = rememberCoroutineScope()

    var bottomSheet by remember {
mutableStateOf(BottomSheet.Category) }

    fun showTopicsSheet() {
        bottomSheet = BottomSheet.Topic
        coroutineScope.launch {
            sheetState.show()
        }
    }

    fun hideBottomSheet() {
        coroutineScope.launch {
            sheetState.hide()
        }
    }

    BackHandler(sheetState.isVisible) {
```

```
        hideBottomSheet()
    }

    ModalBottomSheetLayout(
        modifier = Modifier.fillMaxSize(),
        sheetState = sheetState,
        sheetContent = {
            when (bottomSheet) {
                BottomSheet.Category ->
CategoryBottomSheet(viewModel) {
                    hideBottomSheet()
                }
                else -> TopicsBottomSheet(viewModel) {
                    hideBottomSheet()
                }
            }

        },
        sheetBackgroundColor = MaterialTheme.colors.background,
        sheetShape = RoundedCornerShape(topStart = 15.dp, topEnd =
15.dp),
    ) {

        Column(
            modifier = Modifier
                .fillMaxSize()
                .background(MaterialTheme.colors.background)
        ) {
            ToolbarWidget(toolbarTitle = stringResource(id =
R.string.app_name)) {
                Image(
                    modifier = Modifier
                        .padding(end = 16.dp)
                        .clickable {
                            navigate(AppRoter.navigateToSettings)
                        },
                    painter = painterResource(id =
R.drawable.ic_settings),
                    colorFilter =
ColorFilter.tint(MaterialTheme.colors.primary),
                    contentDescription = ""
                )

                Image(
                    modifier = Modifier
                        .padding(end = 16.dp)
                        .width(16.dp)
```

```
                        .height(16.dp)
                        .clickable {
                            logout()
                        },
                    painter = painterResource(id =
R.drawable.logout),
                    colorFilter =
ColorFilter.tint(MaterialTheme.colors.primary),
                    contentDescription = ""
                )
            }
            Column(
                modifier = Modifier
                    .padding(horizontal = 8.dp, vertical = 8.dp),
            ) {
                ClickableThemeTextComponent()
            }



            Column(
                modifier = Modifier
                    .fillMaxWidth()
                    .weight(1f)
            ) {}



            if (examMode.mods.isNotEmpty()) {
                ExamModeSelector(
                    modifier = Modifier.padding(horizontal = 16.dp,
vertical = 8.dp),
                    mods = examMode.mods,
                    onClick = {
                        when {
                            it.mode is ExamMode.Topic && it.selected
-> showTopicsSheet()
                            else -> viewModel.changeExamMode(it.mode)
                        }
                    }
                )
            }
            ButtonComponent(value = "Почати екзамен", onButtonClicked
= { navigate(AppRoter.navigateTo.Exam) },isEnabled = true)
        }
    }
}
```

```
@Composable
fun ExamModeSelector(
    modifier: Modifier = Modifier,
    mods: List<ExamModeWrapper>,
    onClick: (ExamModeWrapper) -> Unit
) {

    Column(
        modifier = modifier
            .fillMaxWidth()
            .background(MaterialTheme.colors.surface,
RoundedCornerShape(12.dp))
            .padding(vertical = 8.dp, horizontal = 8.dp),
    ) {
        Text(
            modifier = Modifier.fillMaxWidth(),
            style = MaterialTheme.typography.h1,
            textAlign = TextAlign.Center,
            text = "Вибір режиму тестування",
            color = MaterialTheme.colors.primary
        )
        mods.forEach { modeWrapper ->
            ExamModeItem(
                mode = modeWrapper.mode,
                topicName = modeWrapper.topic?.name.orEmpty(),
                selected = modeWrapper.selected,
                onClick = {
                    onClick(modeWrapper)
                }
            )
        }
    }
}


@Composable
fun ExamModeItem(
    mode: ExamMode = ExamMode.Exam,
    topicName: String = "",
    selected: Boolean = true,
    onClick: () -> Unit
) {
    val subtitle = when (mode) {
        ExamMode.Exam -> stringResource(id =
R.string.mode_exam_description)
```

```
        //ExamMode.Training -> stringResource(id =
R.string.mode_training_description)
        is ExamMode.Topic -> if (selected)
            stringResource(id = R.string.mode_topic_description,
topicName)
        else topicName
    }
    Row(
        modifier = Modifier
            .fillMaxWidth()
            .clip(RoundedCornerShape(12.dp))
            .clickable {
                onClick()
            },
        verticalAlignment = Alignment.CenterVertically
    ) {
        Column(
            modifier = Modifier
                .weight(1f)
                .padding(vertical = 4.dp, horizontal = 8.dp)
        ) {
            Text(
                modifier = Modifier.fillMaxWidth(),
                style = MaterialTheme.typography.subtitle1,
                textAlign = TextAlign.Start,
                text = stringResource(id = mode.titleRes),
                color = MaterialTheme.colors.primary
            )
            Text(
                modifier = Modifier.fillMaxWidth(),
                style = MaterialTheme.typography.overline,
                textAlign = TextAlign.Start,
                text = subtitle,
                color = MaterialTheme.colors.primaryVariant
            )
        }
        if (selected) {
            Image(
                painter = painterResource(id = R.drawable.ic_check),
                colorFilter =
ColorFilter.tint(MaterialTheme.colors.primary),
                contentDescription = ""
            )
        }
    }
}
```

```
@Preview()
@Composable
fun HomeScreenPreview() {
    HomeScreen()
}
```