**КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**
**ІМЕНІ ТАРАСА ШЕВЧЕНКА**

Факультет Комп'ютерних наук та кібернетики
Кафедра Теорії та технології програмування

# Кваліфікаційна робота
## на здобуття ступеня бакалавра

за спеціальністю 122 Комп'ютерні науки
на тему:
**СЕРВІС ДЛЯ СТВОРЕННЯ ТА ПРОВЕДЕННЯ ОПИТУВАНЬ**

Виконав студент 4-го курсу
Андрій ХРИСТОФОР _____

Науковий керівник:
професор, доктор фіз. - мат. наук
Микола НІКІТЧЕНКО _____

Засвідчую, що в цій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
_

Роботу розглянуто й допущено до захисту на засіданні кафедри теорії та технології програмування
«___» _____ 2023 р.,

протокол № __

Завідувач кафедри

Микола
НІКІТЧЕНКО _____

Київ – 2023 рік

# РЕФЕРАТ

Обсяг роботи 41 сторінка, 28 ілюстрацій, 14 використаних джерел, 7 таблиць, 2 додатки.

Об’єктом розробки є сервіс, який дозволяє створювати опитувальники, ділитись ними, проходити та збирати дані.

Метою роботи є розробка вебсервісу, який надаватиме інструмент для створення та проведення тестувань і опитувань, а також можливість поділитися опитувальниками з різними групами користувачів, такими як колеги, клієнти або спільноти. Це дозволить здійснювати дослідження і збирати дані від різних аудиторій.

Після проведення опитувань, сервіс також надаватиме можливість аналізувати результати. Користувачі матимуть можливість переглядати статистику, графіки та звіти, що допоможе їм зрозуміти відповіді та тенденції, які виявилися під час опитування. Це дозволить отримувати корисну інформацію для прийняття рішень, покращення продуктів або проведення подальших досліджень.

Таким чином, розробка цього вебсервісу спрямована на надання інструменту для створення, проведення та аналізу опитувань і тестувань, що дозволить користувачам ефективно збирати інформацію та отримувати цінні результати. Інструменти розроблення: MVC - фреймворк Angular, програмна платформа NodeJS, нереляційна база даних MongoDb, середовище програмування Intellij IDEA.

Розроблений сервіс має широкі можливості використання, особливо в університетському середовищі. Він може бути інструментом для університетів у створенні та проведенні різних видів тестів, оцінці знань студентів та здійсненні загальних опитувань та голосувань.

Завдяки цьому сервісу університети матимуть змогу автоматизувати та поліпшити процес тестування студентів. Використовуючи сервіс, вони зможуть створювати тестові завдання з різними типами питань, такими як

однозначний вибір, багатозначний вибір, заповнення пропуску, зіставлення, а також відкриті питання.

Додатковою перевагою сервісу є його здатність автоматично перевіряти тести. Після завершення тестування, система може швидко проаналізувати відповіді студентів і надати оцінки з урахуванням заданих критеріїв оцінювання. Це спрощує процес оцінювання та забезпечує об'єктивність результатів.

У результаті роботи був успішно розроблений вебсервіс, який відповідає потребам університету щодо створення та проведення тестувань. Цей сервіс надає інструмент для викладачів університету, щоб вони могли створювати тести з різними типами питань і варіантів відповідей.

Рекомендується подальший розвиток сервісу з метою забезпечення інтеграції з наявними системами університету. Це дозволить зручно обмінюватися даними, використовувати базу студентів та результатів, що існує, а також забезпечити автоматичне оновлення інформації.

Також варто розглянути можливість розширення функціональності сервісу. Наприклад, додавання функції нагадування про надходження тесту або розкладу тестувань, можливість встановлення обмежень у часі на проходження тестів, аналіз питань та відповідей з метою виявлення найпоширеніших помилок та покращення якості тестів.

Крім того, для оцінки ефективності сервісу та його впливу на навчальний процес та результати студентів рекомендується провести дослідження. Це дозволить зрозуміти, наскільки успішно впровадження сервісу впливає на покращення оцінок студентів, збільшення їх мотивації та зростання активності в навчанні.

# ЗМІСТ

# ВСТУП

Університети постійно шукають способи поліпшити навчальний процес та оцінювання знань своїх студентів. Одним із важливих аспектів є ефективне проведення оцінювання знань, яке вимагає зручних інструментів для створення тестів, аналізу результатів та надання зворотного зв'язку. У цьому контексті розробка сервісу для створення та проведення тестувань для університету набуває великого значення. Особливої актуальності дана тема набула з поширенням дистанційного навчання у зв’язку з карантинними обмеженнями через пандемію, а також введенням воєнного стану в Україні. В таких умовах досить важко використовувати традиційні методи оцінки знань, такі як паперові тести тощо.  Найзручнішим для викладачів та студентів способом провести її є проходження останніми онлайн тестів.

**Мета й завдання роботи**. Метою роботи є розробка вебсервісу, який надаватиме інструмент для створення та проведення тестувань і опитувань, що дозволить користувачам легко створювати свої власні опитувальники з використанням різних типів питань і варіантів відповідей.

**Оцінка сучасного стану об’єкта дослідження або розробки.** На сьогодні існують обмеження та недоліки в підходах до опитування студентів в університетському середовищі. Традиційні методи опитування, що включають паперові анкети або усні опитування, можуть бути часом кривими, малоефективними та вимагати значних ресурсів у вигляді людських і технічних зусиль.

У деяких випадках використання комп'ютерних програм для створення та оцінювання тестів може бути обмеженим та вимагати спеціалізованих знань. Не всі програми надають широкий спектр функціональності, яка може бути потрібна в університетському середовищі, таку як різноманітні типи питань, автоматична перевірка тестів та аналіз результатів.

**Актуальність роботи та підстави для її виконання.** Відсутність зручних та потужних інструментів для створення та проведення тестів може

обмежувати можливості університетів у вимірюванні знань студентів, оцінці їх успішності та підготовці до іспитів. Виникає потреба у більш сучасному та комплексному підході до опитування, який забезпечує зручність, ефективність та надійність процесу.

**Об’єкт, методи й засоби дослідження або розроблення.** Розроблений вебсервіс, який надає інструмент для створення та проведення тестів, є відповіддю на ці виклики. Він пропонує широкий набір функціональності, яка враховує потреби університетського середовища, включаючи різноманітні типи питань, автоматичну перевірку тестів та аналіз результатів. Цей сервіс допомагає університетам поліпшити процес створення, проведення та оцінювання тестів, сприяючи автоматизації та ефективності. Він надає можливість використовувати сучасні технології для опитування студентів, що полегшує і збільшує точність збору та обробки даних.

**Можливі сфери застосування.** Додатково, сервіс може бути використаний для оцінки знань студентів, дозволяючи викладачам миттєво отримувати результати тестів та швидко аналізувати здобуті студентами знання. Це допомагає виявляти слабкі місця та недоліки в навчанні та надавати індивідуальну підтримку студентам для подальшого вдосконалення. Однак, сучасні технології розробки вебзастосунків, баз даних та аналітики надають нові можливості для розробки сервісу для створення та проведення тестувань. Використання вебсервісу дозволить викладачам зручно створювати різноманітні тести з різними типами питань, а студентам - зручно проходити тести та отримувати детальний аналіз своїх результатів. Такий сервіс може стати важливим інструментом для університетів у поліпшенні процесу навчання та оцінювання знань.

В ході розробки даного продукту було проаналізовано наявні системи тестування, їх переваги та недоліки, потреби викладачів та студентів щодо даної проблеми, визначено функціональні вимоги до проєкту, засновуючись на отриманих даних, розробка програмного продукту, тестування та валідація

сервісу залежно від вимог та сценаріїв використання, включаючи тестування функціональності та продуктивності.

У майбутньому сервіс для створення та проведення опитувань може мати взаємозв’язки з іншими проєктами в галузі освіти, такі як: інтеграція з системами управління навчальним процесом(наприклад Triton Student), інтеграція з системами аналізу даних та використання сучасних технологій, таких як штучний інтелект, розподілені системи або блокчейн. Це дозволяє автоматизувати обмін даними та їх аналіз, сприятиме покращенню безпеки, ефективності та якості навчального процесу.

# РОЗДІЛ 1. ТЕОРЕТИЧНІ ВІДОМОСТІ

## 1.1. Огляд наявних на ринку систем

На сьогодні існує кілька систем для створення та проведення онлайн опитувань, серед яких:

- Google Forms – є популярним інструментом для створення опитувань та тестувань. Він має простий інтерфейс, легкий у використанні та безплатний. Однак, має і недоліки, наприклад: обмежена кількість налаштувань(Google Forms пропонує обмежений набір шаблонів та опцій), відсутність поглибленого функціонала(Google Forms не має деяких поглиблених функцій, які можуть бути потрібні для складних тестів або оцінювання, наприклад, він може бути обмежений у можливості обробки складних формул або автоматичного перетворення відповідей у числові оцінки), обмеженість аналітики(вбудовані аналітичні інструменти Google Forms є обмеженими та не надають докладних статистичних даних або розширених звітів), обмежена інтеграція з іншими системами(Google Forms може мати обмежену можливість інтеграції з іншими системами або платформами, що може ускладнити автоматизацію та обробку даних з форм;
- Blackboard – популярна платформа електронного навчання, яка включає функціонал для створення тестів та оцінювання студентів. Вона надає широкий спектр можливостей для налаштування тестів та збору результатів. Однак, система може бути складною у використанні та налаштуванні для користувачів без технічних навичок, крім того, має обмежені можливості настроювання та розширення, порівняно з деякими іншими системами та вимагає платну ліцензію;
- ClassTime – ще одна популярна платформа для проведення тестувань та опитувань. Має інтуїтивний та простий інтерфейс, гнучкі налаштування та розширені аналітичні можливості. Однак, система має

обмежені можливості інтеграції з іншими платформами, а також для багатьох функцій вимагається платний тарифний план.

Огляд наявних систем показує, що хоча є деякі рішення для створення та проведення тестувань, вони часто мають обмеження у функціональності, складні у використанні або потребують платних ліцензій. Тому, розробка власного сервісу, який враховує специфічні потреби університету, може бути актуальною та цінною задачею, яка вирішить ці обмеження та надасть ширший функціонал, який відповідатиме конкретним потребам університетського середовища.

## 1.2. Призначення системи

Призначенням розроблюваного сервісу є створення зручного та безпечного інструменту для викладачів та студентів для створення опитувальників з різними типами питань, включаючи багатовибіркові, відкриті, з вибором зі списку та інші та їх проходження. Система буде забезпечувати можливість переглядати результати опитувань та аналізувати об’єми даних. Сервіс повинен підтримувати можливість проходження тестів у електронному вигляді з будь-якого зручного пристрою, такого як комп'ютер, планшет або смартфон, що забезпечить зручність та гнучкість для користувачів. Крім того, необхідно забезпечити високий рівень захисту даних та приватності користувачів для мінімізації можливості недотримання академічної недоброчесності.

Загалом, система має стати необхідним інструментом для ефективного навчання, оцінювання рівня знань та отримання різних статистик в університеті, що сприятиме покращенню навчального процесу та забезпеченню зручності та ефективності для викладачів та студентів.

Робота передбачає:

- Аналіз наявних продуктів, їх переваг та недоліків;
- Аналіз потреб користувачів;
- Проєктування функціональності – на основі отриманих даних необхідно розробити дизайн та функціональні вимоги до системи. Наприклад, це охоплює визначення можливих типів запитань у формах, можливих дій користувача та інше;
- Розробка платформи – на основі визначених вимог, необхідно розробити сервіс та забезпечити його функціональність;
- Тестування та налагодження – після створення, необхідно провести тестування та налагодження, щоб переконатись, що всі функції працюють коректно та потреби користувачів були задоволені.

## 1.3. Цілі створення системи

Продукт створюється з метою:

- забезпечення зручного та ефективного процесу створення тестів;
- автоматизація процесу оцінювання;
- забезпечення збору та аналізу результатів;
- забезпечення безпеки та конфіденційності;
- покращення якості навчання та оцінювання.

## 1.4. Вимоги до системи

Функціональні вимоги:

- можливість створювати різні типи питань (однозначний вибір, багатозначний вибір, відкрита відповідь тощо);
- забезпечення можливості планування тестувань та встановлення термінів доступу до них;

- автоматичне оцінювання відповідей студентів та генерація результатів;
- зберігання результатів тестування та доступ до них для аналізу.

Інтерфейс та зручність використання:

- інтуїтивно зрозумілий та легкий у використанні інтерфейс для користувачів;
- забезпечення зручного проходження тестів студентами на будь-яких пристроях (комп'ютери, планшети, мобільні телефони);
- підтримка локалізації для різних користувачів.

Безпека та конфіденційність:

- захист особистих даних студентів та забезпечення конфіденційності їх результатів;
- механізми аутентифікації та авторизації, щоб забезпечити доступ лише авторизованим користувачам.

Масштабованість та надійність:

- здатність системи працювати зі кількістю викладачів та студентів, що зростає;
- забезпечення високої доступності та мінімізація можливих витоків даних;
- підтримка усіма сучасними браузерами, такими як Google Chrome – версія 79 або новіша, Mozilla Firefox – версія 72 або новіша, Apple Safari – версія 13 або новіша, Microsoft Edge – версія 79 або новіша, Opera – версія 66 або новіша.

Інтеграція та розширюваність:

- здатність інтегруватися з системами університету, що існують, такими як система управління навчанням Triton тощо;
- можливість розширення функціонала системи в майбутньому для вирішення нових вимог та потреб користувачів.

## 1.5. Огляд використаних технологій

Для реалізації сервісу обрано клієнт-серверну архітектуру та стек технологій MEAN(M – MongoDB, E – Express.js, A – Angular, N – Node.js). Цей стек технологій надає зручність, ефективність та розширюваність при розробці сучасних вебдодатків.

Клієнт-серверна архітектура – це модель розподіленої системи, в якій функції та обов'язки розділені між клієнтською та серверною частинами.

Клієнт – це кінцевий користувач або програмне забезпечення, яке взаємодіє з користувачем. Він звертається до сервера для отримання ресурсів, виконання операцій або отримання даних. Клієнт може бути веббраузером, мобільним додатком або іншою програмою, що виконується на пристрої користувача.

Сервер – це центральний компонент системи, який обробляє запити від клієнтів і надає їм необхідні ресурси або виконує необхідні операції. Сервер може зберігати та обробляти дані, виконувати бізнес-логіку, керувати доступом до ресурсів і забезпечувати інші функціональні можливості.

У процесі роботи було створено 2 окремі додатки. Серверна частина – програма, написана з використанням мови програмування JavaScript та платформи NodeJS, клієнтська – SPA(Single Page Application), написане з використанням MVC-фреймворку Angular.

SPA – це тип вебдодатка, який взаємодіє з користувачем, перезавантажуючи лише частину сторінки, а не всю сторінку цілком. У традиційному багатосторінковому додатку при кожній дії користувача відбувається перезавантаження всієї сторінки, що призводить до зайвої передачі даних та затримок в роботі.

SPA натомість побудовані на основі однієї сторінки, яка завантажується при початковому запуску додатка. Після цього взаємодія з користувачем відбувається шляхом динамічної зміни вмісту на сторінці без

перезавантаження. Це досягається за допомогою технологій, таких як JavaScript, AJAX і HTML5.

Основні переваги SPA включають:

- швидкі відгуки на дії користувача, оскільки перезавантаження сторінки не відбувається.
- зниження навантаження на сервер, оскільки відбувається передача лише необхідних даних.
- покращена взаємодія з користувачем через те, що додаток працює більш плавно та зручно, подібно до настільних додатків.
- можливість розробки кросс-платформеного додатку, оскільки код виконується на боці клієнта.

Проте, SPA мають і деякі недоліки, зокрема:

- завантаження великого обсягу JavaScript-коду на початку, що може тривати деякий час.

У контексті розробки сервісу для створення та проведення тестувань, використання SPA дозволить створити зручний та ефективний інтерфейс користувача, який буде працювати безперебійно та забезпечувати швидкість та зручність взаємодії.

Angular – фреймворк для розробки вебдодатків з високою продуктивністю і масштабованістю. Використання Angular дозволяє побудувати ефективний та інтерактивний інтерфейс користувача для системи. Він надає можливості для створення компонентів, маршрутизації, обробки подій та зв'язування даних[1].

Angular базується на мові програмування TypeScript. Також, в проєкті використовується препроцесор SCSS для стилізування.

TypeScript – це мова програмування, яка є розширенням мови JavaScript. Вона надає можливості статичної типізації, підтримку об'єктно-орієнтованого програмування та сучасні функції програмування, що допомагають розробникам писати більш безпечний і структурований код. Однією з ключових особливостей TypeScript є можливість визначати типи для змінних,

параметрів функцій, властивостей об'єктів та інших елементів коду. Це дозволяє виявляти помилки на етапі компіляції, що полегшує виявлення та виправлення проблем в коді перед його виконанням. TypeScript також підтримує класи, модулі, інтерфейси, спадкування та багато інших концепцій об'єктно-орієнтованого програмування. Це сприяє створенню більш структурованого та повторно використовуваного коду.

SCSS (Sassy CSS) є розширенням мови CSS, яке надає додаткові можливості і покращення в розробці стилів для вебсайтів і додатків. SCSS базується на синтаксисі CSS, але включає додаткові функції, змінні, міксіни та багато інших функціональних можливостей. Однією з основних переваг SCSS є можливість використовувати змінні. Змінні дозволяють зберігати значення, такі як кольори, розміри шрифту, відступи та інші, і використовувати їх у всьому коді. Це спрощує зміну стилів на всьому проєкті, оскільки потрібно змінити значення лише в одному місці.

Також, при розробці UI/UX додатку(інтерфейс користувача) було використано: Angular Material – це набір готових компонентів і стилів, які використовуються для розробки консистентного, стильного та сучасного інтерфейсу користувача, та Figma – інструмент для дизайну інтерфейсів. За допомогою Figma створено макети інтерфейсу, дизайн іконок та інші елементи, які згодом інтегруються в додаток.

Для серверної сторони застосунку було використано Node.js – середовище виконання JavaScript на стороні сервера, що дозволяє розробляти швидкі та масштабовані вебдодатки, а також Express.js – фреймворк для Node.js, який дозволяє легко створювати вебсервер та API.

Для забезпечення комунікації між клієнтською та серверною частинами системи використовується архітектурний стиль REST API. Він використовує HTTP-протокол для створення API(Application Programming Interface). В REST API існує набір ендпоінтів (URL-шляхів), які представляють різні ресурси та дозволяють виконувати операції з ними. Наприклад, /tests може бути ендпоінтом для отримання списку тестів, а /tests/{id} - для отримання

конкретного тесту за його ідентифікатором. Використання протоколу HTTP(а точніше його захищеної версії HTTPS) забезпечує шифрування та захист передачі даних між клієнтом і сервером у мережі Інтернет.

Для маніпулювання даними використовується MongoDB – документ-орієнтована база даних, яка дозволяє зберігати та опрацьовувати дані у форматі JSON-подібних документів. Використання MongoDB дозволяє ефективно зберігати та керувати даними системи.

Для тестування роботи сервера використовувався Postman. Postman – це популярний інструмент для розробки та тестування API. Він надає зручний інтерфейс для створення, відправки та перевірки HTTP-запитів до вебсерверів.

Розробка велась в IntelliJ IDEA – це інтегроване середовище розробки (IDE), яке надає розширені можливості для розробки вебдодатків з використанням Angular, Node.js та інших технологій. Використання IntelliJ IDEA дозволяє розробникам писати код, налагоджувати додатки та використовувати інші корисні функції розробки.

Для зберігання коду програм використовується система контролю версій Git та менеджер репозиторіїв GitHub, що дозволило ефективно контролювати етапи розробки проєкту.

Загалом, використання описаних вище технологій дозволило отримати такі переваги:

- гнучкість: MEAN стек дозволяє працювати з однією мовою програмування (JavaScript) на всіх рівнях застосунку – від клієнтської сторони (Angular) до серверної (Node.js). Це спрощує розробку, управління та розширення проєкту.
- швидкість розробки: MEAN стек забезпечує зручну інтеграцію між компонентами, що дозволяє розробникам швидко створювати та тестувати функціональність. Інструменти, такі як Angular CLI та Express.js, надають шаблони, автоматичну генерацію коду та багато інших функцій, що прискорюють розробку.

- розширюваність: MongoDB, як база даних MEAN стеку, є гнучкою, масштабованою та безсхемною. Вона дозволяє легко змінювати схему даних та додавати нові поля, що полегшує розширення функціональності застосунка.

# РОЗДІЛ 2. РЕАЛІЗАЦІЯ СИСТЕМИ

## 2.1 Опис організації інформаційної бази

У проєкті використовується система контролю базами даних MongoDb. Це одна із найпопулярніших нереляційних баз даних, яка легко інтегрується з Node.js за допомогою бібліотеки Mongoose.

### 2.1.1 Логічна структура бази даних

База даних містить такі сутності, як User, Form, Control, Validators, Answer, опис яких наведено нижче:

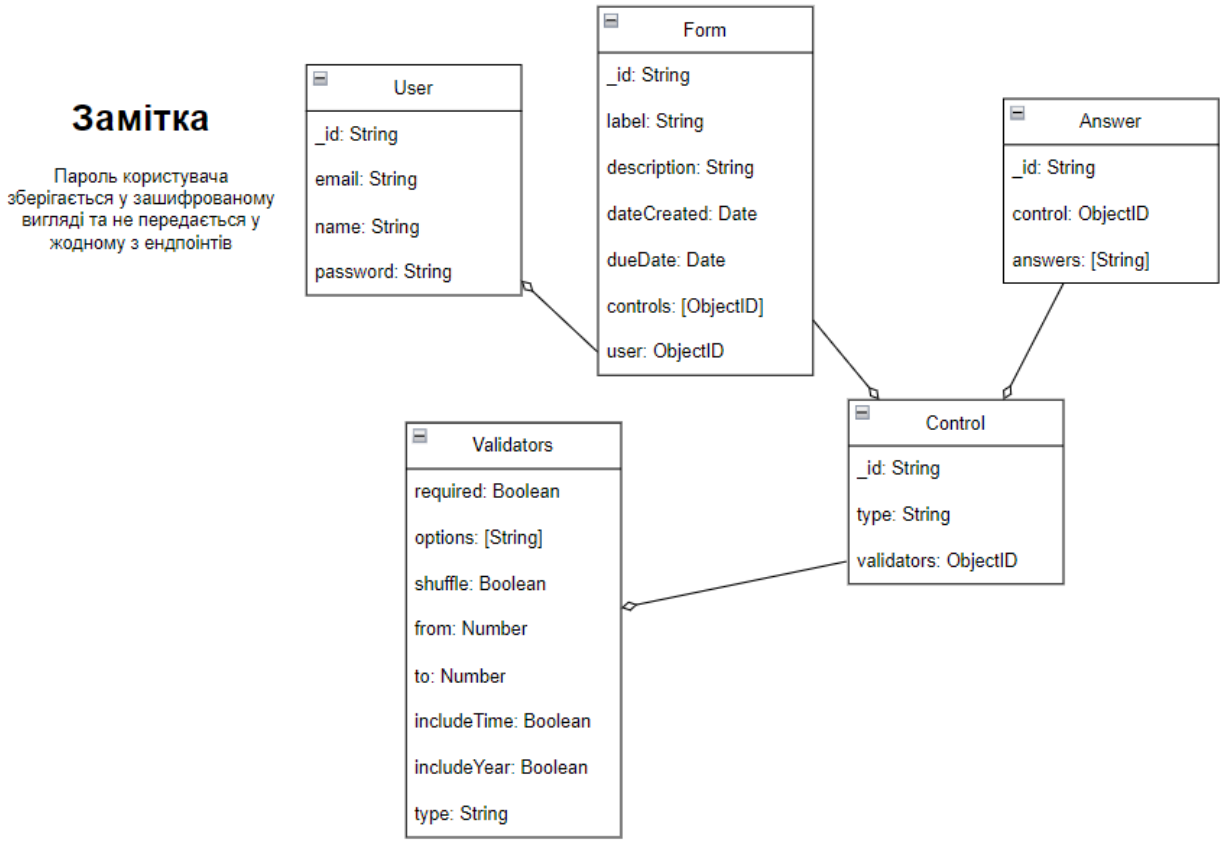



Рисунок 2.1 — Діаграма бази даних сервісу

| Номер | Таблиця | Опис |
|---|---|---|
| 1 | Users | Таблиця, в якій містяться всі зареєстровані користувачі та дані про них. |
| 2 | Forms | Таблиця, в якій містяться усі форми(опитування), які були створені у системі. |
| 3 | Controls | Таблиця, в якій містяться усі запитання, які були створені. |
| 4 | Answers | Таблиця відповідей на питання. |
| 5 | Validators | Допоміжна таблиця, які містить інформацію про питання. |

Таблиця 2.1 — Перелік таблиць бази даних

2.1.2 Опис таблиць

| Атрибут | Тип | Опис |
|---|---|---|
| _id | string | Ідентифікатор. |
| email | string | Email користувача. |
| name | string | Повне ім’я користувача. |
| password | password | Хеш пароля користувача. |

Таблиця 2.2 — Таблиця Users

| Атрибут | Тип | Опис |
|---|---|---|
| _id | string | Ідентифікатор. |
| label | string | Назва форми. |
| description | string | Опис форми. |
| dateCreated | Date | Дата створення форми. |
| dueDate | Date | Дата, до якої можна відповісти на цю форму. |
| controls | [ObjectID] | Список запитань до форми. |

| | | |
|---|---|---|
| user | ObjectID | Ідентифікатор користувача, який створив форму |

Таблиця 2.3 — Таблиця Forms

| **Атрибут** | **Тип** | **Опис** |
|---|---|---|
| _id | string | Ідентифікатор. |
| type | string | Тип запитання. Може мати одне із значень: 'text-input', 'paragraph-input', 'multiple-choice-input', 'checkboxes-input', 'dropdown-input', 'linear-scale-input', 'date-input', 'time-input' |
| validators | Object | Об’єкт параметрів запитання. |

Таблиця 2.4 — Таблиця Controls

| **Атрибут** | **Тип** | **Опис** |
|---|---|---|
| _id | string | Ідентифікатор. |
| control | ObjectID | Запитання, до якого прив’язана дана відповідь. |
| user | ObjectID | Ідентифікатор користувача, який відповів на зпитання. |
| answers | [string] | Відповіді на запитання. |

Таблиця 2.5 — Таблиця Answers

| **Атрибут** | **Тип** | **Опис** |
|---|---|---|
| required? | boolean | Чи є запитання обов’язковим. |
| options? | [string] | Список варіантів відповіді. |
| shuffle? | boolean | Чи перемішувати варіанти відповідей при проходженні. |
| from? | number | Початок проміжку. |
| to? | number | Кінець проміжку. |
| includeTime? | boolean | Чи включати час в дату. |

| includeYear? | boolean | Чи включати рік в дату. |
|---|---|---|
| type? | string | Тип запитання. |

Таблиця 2.6 — Таблиця Validators

## 2.2 Реалізація серверної частини застосунку

Для реалізації backend обрано технології Node.js, Express.js для написання програми, та MongoDB для зберігання даних[5].

Розглянемо деякі реалізації:

```
const express : e | () => core.Express  = require('express')
const mongoose :  = require('mongoose')
const bodyParser : BodyParser | bodyParser  = require('body-parser')
const passport : Authenticator | {...}  = require('passport')
const keys : {...} | {...}  = require('./config/keys')
const app : any | Express  = express()

const AuthRouter : Router | {...}  = require('./routes/auth')
const FormsRouter : Router | {...}  = require('./routes/forms')
const ControlRouter : Router | {...}  = require('./routes/controls')

app.use(passport.initialize())
require('./middleware/passport')(passport)

app.use(bodyParser.urlencoded( options: { extended: true }))
app.use(bodyParser.json())
app.use(require('cors')())
app.use(require('morgan')( format: 'dev'))

 Andrii
mongoose.connect(keys.mongoURI)
  .then(() => console.log('MongoDB connected.'))
  .catch(error => console.log(error))

app.use('/api/auth', AuthRouter)
app.use('/api/forms', FormsRouter)
app.use('/api/controls', ControlRouter)

module.exports = app
```

Рисунок 2.2 — Головний файл проєкту з підключеними бібліотеками

```
const app = require('./app')
const port : string | number  = process.env.PORT || 5000

 Andrii
app.listen(port,  callback: () => console.log(`Server has been started on ${port}`))

```

Рисунок 2.3 — Локальне розгортання проєкту для розробки та тестування

Для авторизації та захисту даних користувачів використовується бібліотека Passport JWT.

```
const JwtStrategy : function(any, any): void | {...}  = require('passport-jwt').Strategy
const ExtractJwt : {}  = require('passport-jwt').ExtractJwt
const mongoose :  = require('mongoose')
const User : Model  = mongoose.model( name: 'users')
const keys : {...} | {...}  = require('../config/keys')

const options : {jwtFromRequest: any, secretOrKey: string}  = {
  jwtFromRequest: ExtractJwt.fromAuthHeaderAsBearerToken(),
  secretOrKey: keys.jwt
}

module.exports = passport => {
  passport.use(
    new JwtStrategy(options,  verify: async (payload, done) : Promise<void>  => {
      try {
        const user : Query<any, any, unknown, any, "findOne">  = await User.findById(payload.userId).select( arg: 'email id')

        if (user) {
          done(null, user)
        } else {
          done(null, false)
        }
      } catch (e) {
        console.log(e)
      }

    })
  )
}
```

Рисунок 2.4 — Генерація секретних ключів користувачів

Для взаємодії з клієнтом використовується REST API, для підключення до бази даних – бібліотеку Mongoose.

```
const mongoose :  = require("mongoose");
const Schema = mongoose.Schema;

const formSchema : Schema<any, Model<...>, {...}, {...}, {...}, {...}, DefaultSchemaOptions,  = new Schema({
  label: {
    type: String,
    required: true,
  },
  description: {
    type: String,
  },
  dateCreated: {
    type: Date,
    default: Date.now,
  },
  dueDate: {
    type: Date,
  },
  public: {
    type: Boolean,
  },
  domains: [
    {
      type: String,
    },
  ],
  controls: [
    {
      type: Schema.Types.ObjectId,
      ref: "controls",
    },
```

Рисунок 2.5 — Схема форми з використанням Mongoose

```
module.exports.signup = async function (req, res) : Promise<void>  {
  const candidate = await User.findOne({ email: req.body.email });

  if (candidate) {
    res.status(409).json({
      message: "Email is already taken. Try to use another one.",
    });
  } else {
    const salt = bcrypt.genSaltSync( rounds: 10);
    const password = req.body.password;
    const user : HydratedDocument<InferSchemaType  = new User( doc: {
      email: req.body.email,
      name: req.body.name,
      password: bcrypt.hashSync(password, salt),
    });

    try {
      await user.save();
      const token = jwt.sign(
        payload: {
          email: user.email,
          userId: user._id,
          name: user.name,
        },
        keys.jwt,
        options: { expiresIn: 3600 }
      );
      res.status(201).json({ token });
    } catch (e) {
      // errorHandler(res, e)
      console.log(e);
```

Рисунок 2.6 — Endpoint для реєстрації нового користувача в системі

Для модуля авторизації наявна валідація та обробка помилок(email уже зайнятий, не знайдено зареєстрованого користувача, паролі не збігаються і т.д.). Пароль користувача передається та зберігається у захешованому вигляді, що унеможливлює витік даних про паролі, оскільки хеш-функції незворотні. На авторизації використано JWT(JSON Web Token) – відкритий стандарт RFC 7519, який використовується для безпечної передачі інформації у форматі JSON між сторонами. Токен доступу генерується з зашифрованими в собі даними про користувача, такими як ім’я, email та ідентифікатор, а також даними про те, коли цей токен буде вичерпаний, після чого клієнтський додаток робить запит на поновлення сесії[2].

Подібний функціонал для CRUD(Create, Read, Update, Delete) операцій з використанням Mongoose реалізовано для усіх інших сутностей(Forms, Controls і тд).

```
module.exports.create = async function (req, res) : Promise<void>  {
    console.log(req.user)
    const form : HydratedDocument<InferSchemaType  = new Form( doc: {
        label: req.body.label,
        description: req.body.description,
        dueDate: req.body.dueDate,
        public: req.body.public,
        domains: req.body.domains,
        user: req.user._id
    })

    try {
        await form.save()
        res.status(201).json(form)
    } catch (e) {
        errorHandler(res, e)
    }
}

module.exports.get = async function (req, res) : Promise<void>  {
    try {
        const form : Query  = await Form.findById(req.params.id).populate('controls')
        res.status(200).json(form)
    } catch (e) {
        errorHandler(res, e)
    }
}

module.exports.update = async function (req, res) : Promise<void>  {
    const updatedForm : {description: any, domains: any, dueDate: any, label: any, public: any}  = {
        label: req.body.label,
```

Рисунок 2.7 — Приклад деяких CRUD операцій для сутності форм

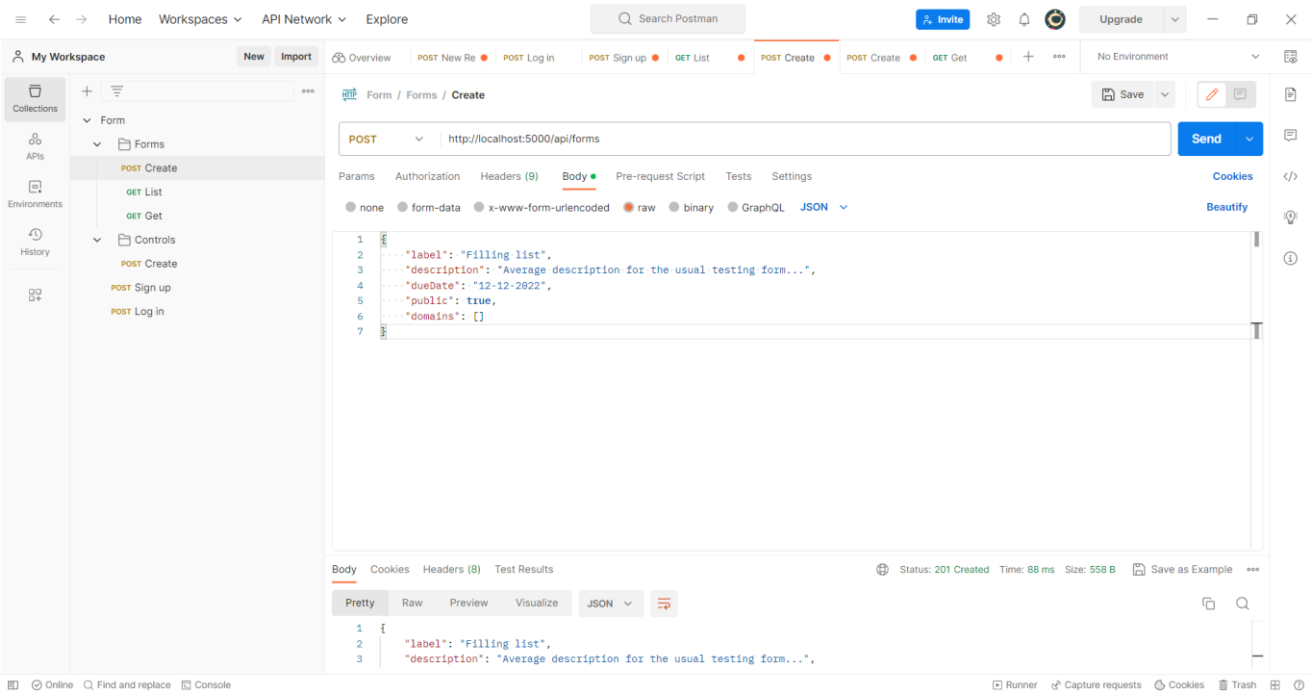



Рисунок 2.8 — Тестування роботи сервера у Postman

## 2.3 Реалізація клієнтської сторони застосунку

Angular – це фреймворк для розробки вебдодатків мовою програмування TypeScript, який забезпечує створення високопродуктивних і масштабованих додатків. Він містить багато компонентів та бібліотек, які допомагають розробнику створювати різні частини додатка, включаючи UI компоненти, маршрутизацію, форми та багато іншого.

Angular Material – це набір компонентів, які розроблені з використанням Material Design – дизайн-мови від Google. Angular Material дозволяє швидко створювати привабливі та функціональні користувацькі інтерфейси, які виглядають однаково на різних пристроях[4].

Angular містить в собі велику кількість сутностей, таких як Component(відповідає за відображення контенту на вебсторінці), Service(відповідає за роботу з даними, їх отримання та трансформування), Interceptor(перехоплює усі запити від клієнта до сервера та модифікує їх) тощо, які значно спрощують розробку додатків.

```scss
/* You can add global styles to this file, and also import other style files */
@import url("https://fonts.googleapis.com/css2?family=Poppins:wght@400;500;600;700&display=swap");
@import url("https://fonts.googleapis.com/css2?family=Open+Sans:wght@400;600&display=swap");


$base-dark-blue: #253A82;
$base-green: #32E0D5;
$base-light-blue: #3B6ECD;
$base-purple: #572E9E;

$base-background: #F0EBF8;


// $base-blue: #1053EC;
$light-blue: #2783FE;
$input-light-grey: #f5f7f9;
$input-grey: #9BA2AB;
$dashboard-light-grey: #E9EBF0;
$dashboard-grey: #7C828D;

$FILTER_TABS_SECTION_BOX_SHADOW: 0 0 9px rgb(32 32 32 / 18%);

body {
    margin: 0;
    font-family: -apple-system, BlinkMacSystemFont, Segoe UI, Roboto, Noto Sans, Ubuntu, Droid Sans, Helvetica Neue, sans-serif;
}
```

Рисунок 2.9 — Фрагмент глобальних стилів для проєкту

```
<div class="main-page" *ngIf="!isFetchingForms; else loading">
  <div class="main-page__header">
    <div class="main-page__new-form">
      <button mat-raised-button color="primary" (click)="navigateToNewForm()">
        {{ "newForm" | translate }}
      </button>
    </div>

    <mat-form-field appearance="outline" class="main-page__search">
      <mat-label>{{ "search" | translate }}</mat-label>
      <input matInput [(ngModel)]="searchKey"/>
      <mat-icon matSuffix>search</mat-icon>
    </mat-form-field>
  </div>

  <div class="main-page__forms forms">
    <div class="forms__form" *ngFor="let form of forms | searchForms : searchKey">
      <mat-card class="forms__card card" (click)="navigateToFormOverview(form._id)">
        <mat-card-header>
          <mat-card-title>{{ form.label }}</mat-card-title>
          <mat-card-subtitle>
            {{ form.dateCreated | date : format: "dd.MM.YYYY" }}
          </mat-card-subtitle>
        </mat-card-header>
        <mat-divider></mat-divider>
        <mat-card-content class="card__content">
          <p>{{ form.description }}</p>
        </mat-card-content>
        <div class="card__bottom-block">
```

Рисунок 2.10 — Приклад шаблону компонента з використанням Angular Material

Для реалізації маршрутизації в Angular використовується вбудований механізм маршрутизації, який дозволяє описати шляхи до різних компонентів у додатку. Для цього використовуються модулі RouterModule та Route, які є частинами фреймворку Angular.

Доступ до основної частини застосунку перевіряється на моменті маршрутизації. Якщо користувач авторизований в системі – він може користуватись додатком, інакше – його переводить на сторінку авторизації.

```typescript
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { AuthLayoutComponent } from './components/auth-layout/auth-layout.component';
import { LoginPageComponent } from './components/login-page/login-page.component';
import { SignupPageComponent } from './components/signup-page/signup-page.component';
import { WelcomePageComponent } from './components/welcome-page/welcome-page.component';
import { AuthGuard } from './core/guards/auth.guard';
const routes: Routes = [
  { path: '', component: WelcomePageComponent, pathMatch: 'full' },
  {
    path: '',
    component: AuthLayoutComponent,
    children: [
      { path: 'login', component: LoginPageComponent },
      { path: 'signup', component: SignupPageComponent },
    ],
  },
  {
    path: 'dashboard',
    canActivate: [AuthGuard],
    loadChildren: () =>
      import('./dashboard/dashboard.module').then((m) => m.DashboardModule),
  },
];
2 usages  Andrii
@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule],
})
export class AppRoutingModule {}
```

Рисунок 2.11 — Приклад роутер модуля

Для спілкування з сервером було створено сервіси з усіма необхідними функціями для Rest API.

```typescript
2 usages  Andrii
login(user: User): Observable<any> {
  return this.httpPost( url: 'auth/login', user);
}

2 usages  Andrii *
signup(user: User): Observable<any> {
  return this.httpPost( url: 'auth/signup', user);
}

2 usages  Andrii
private httpPost(
  url: string,
  payload?: any,
  options?: object
): Observable<any> {
  return this.http.post( url: `${this.apiUrl}/${url}`, payload, options);
}

no usages  Andrii
private httpPut(
  url: string,
  payload?: any,
  options?: object
): Observable<any> {
  return this.http.put( url: `${this.apiUrl}/${url}`, payload, options);
}

no usages  Andrii
private httpGet(url: string, params?: HttpParams): Observable<any> {
  return this.http.get( url: `${this.apiUrl}/${url}`,  options: { params });
```

Рисунок 2.12 — Приклад методів взаємодії з сервером

```
1 usage  Andrii
isAuthenticated(): boolean {
  return !!this.token;
}

2 usages  Andrii
private handleError(error: HttpErrorResponse) : void  {
  const { message } = error.error.error;

  switch (message) {
    case 'INVALID_EMAIL':
      this.error$.next( value: 'Invalid Email');
      break;
    case 'INVALID_PASSWORD':
      this.error$.next( value: 'Invalid Password');
      break;
    case 'EMAIL_NOT_FOUND':
      this.error$.next( value: 'Email Not Found');
      break;
  }

  throwError(error);
}
```

Рисунок 2.13 — Приклад обробки помилок

```
import { Injectable } from '@angular/core';
import {
  HttpRequest,
  HttpHandler,
  HttpEvent,
  HttpInterceptor,
} from '@angular/common/http';
import { Observable } from 'rxjs';
import { ConfigService } from '../services/config.service';

2 usages  new *
@Injectable()
export class ReqHeadersInterceptor implements HttpInterceptor {
  no usages  new *
  constructor(private configSrv: ConfigService) {}

  no usages  new *
  intercept(
    request: HttpRequest<unknown>,
    next: HttpHandler
  ): Observable<HttpEvent<unknown>> {
    const modifiedReq : HttpRequest<?>  = request.clone( update: {
      setHeaders: { Authorization: `Bearer ${this.configSrv.token}` },
    });
    return next.handle(modifiedReq);
  }
}
```

Рисунок 2.14 — Перехоплювач, який додає до запитів секретні параметри користувача для його розпізнання сервером

Додаток розділений на 2 модулі – авторизація та основне меню(dashboard).

Для реалізації усіх форм(авторизація, безпосередньо форми опитувань і т.д.) було використано Reactive forms в Angular, які дозволяють зручно та гнучко керувати як станом форми, так і її відображенням на вебінтерфейсі. Reactive forms дозволяють реагувати на зміни введених даних та проводити їх валідацію[3].

```
<app-checkbox-grid-input *ngIf="control.type === 'checkbox-grid'" [control]=control [form]="form"></app-checkbox-grid-input>
<app-checkboxes-input *ngIf="control.type === 'checkboxes'" [control]=control [form]="form"></app-checkboxes-input>
<app-date-input *ngIf="control.type === 'date'" [control]=control [form]="form"></app-date-input>
<app-dropdown-input *ngIf="control.type === 'dropdown'" [control]=control [form]="form"></app-dropdown-input>
<app-file-upload-input *ngIf="control.type === 'file-upload'" [control]=control [form]="form"></app-file-upload-input>
<app-linear-scale-input *ngIf="control.type === 'linear-scale'" [control]=control [form]="form"></app-linear-scale-input>
<app-multiple-choice-grid-input *ngIf="control.type === 'multiple-choice-grid'" [control]=control [form]="form"></app-multiple-choic
<app-multiple-choice-input *ngIf="control.type === 'multiple-choice'" [control]=control [form]="form"></app-multiple-choice-input>
<app-paragraph-input *ngIf="control.type === 'paragraph'" [control]=control [form]="form"></app-paragraph-input>
<app-text-input *ngIf="control.type === 'text'" [control]=control [form]="form"></app-text-input>
<app-time-input *ngIf="control.type === 'time'" [control]=control [form]="form"></app-time-input>
```

Рисунок 2.15 — Приклад основних компонентів обробки форми

# РОЗДІЛ 3. ОГЛЯД ГОТОВОГО ПРОДУКТУ

## 3.1 Користувацький інтерфейс

При переході на сайт за посиланням користувач потрапляє на початкову сторінку, з якої він може перейти до авторизації або переглянути основні інформаційні ресурси університету, такі як LinkedIn, Facebook, сайт університету.

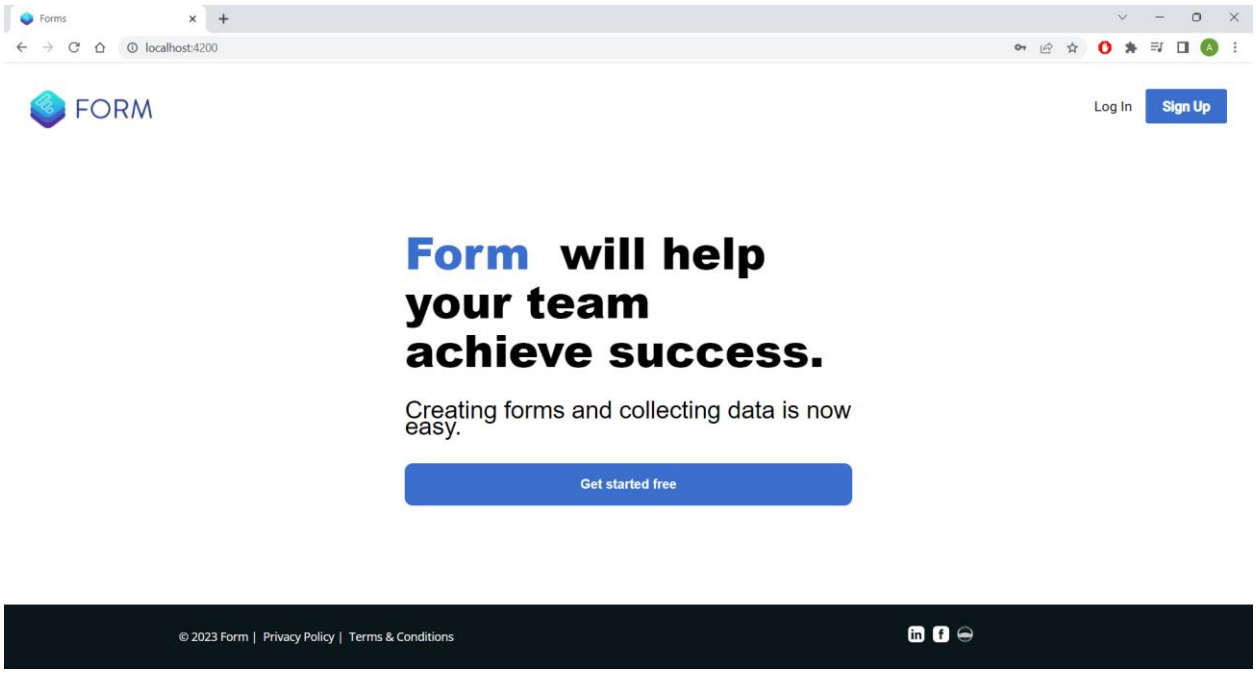



Рисунок 3.1 — Початкова сторінка

Користувач має можливість обрати мову інтерфейсу між українською та англійською.

При виборі Вхід користувач потрапляє на сторінку логіну, де може увійти до системи як зареєстрований користувач. При виборі Реєстрація – на сторінку, де може зареєструвати нового користувача.

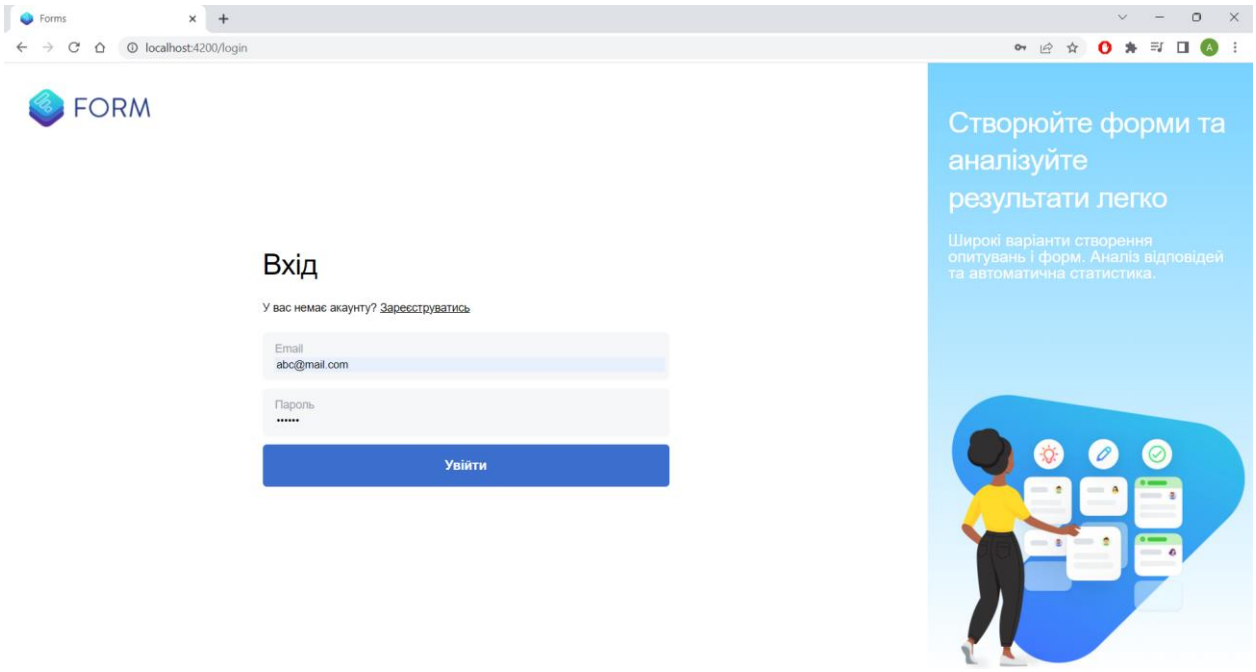



Рисунок 3.2 — сторінка входу

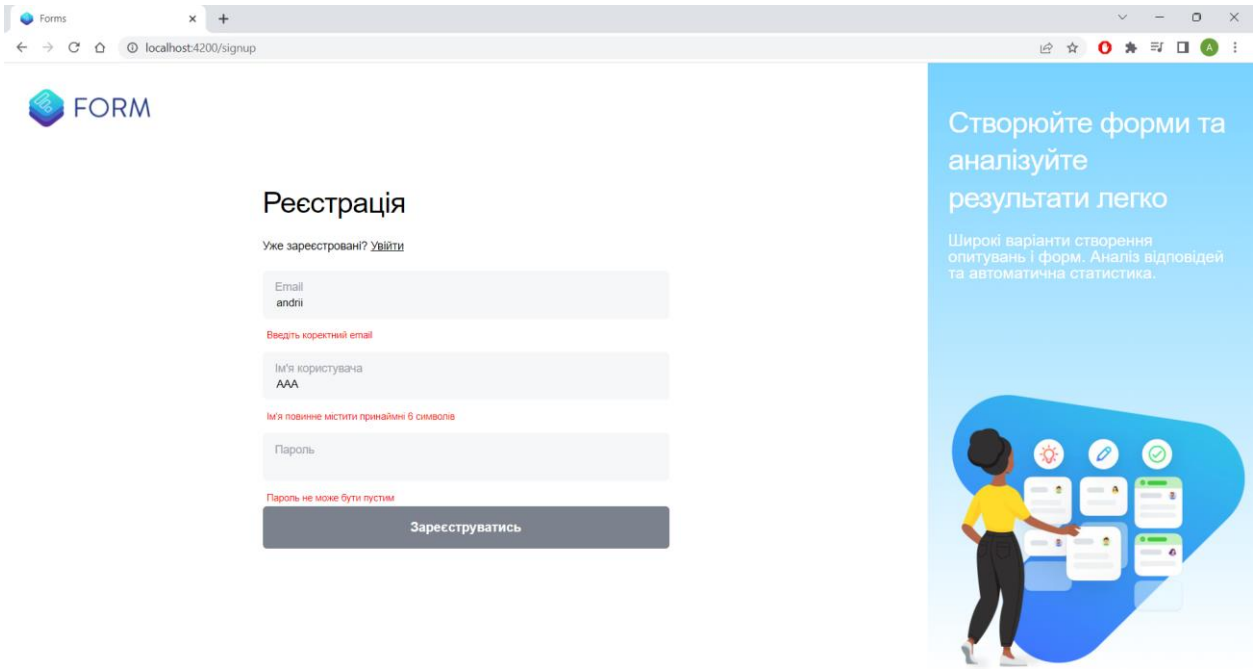



Рисунок 3.3 — сторінка реєстрації з деякими валідаційними полями

Після успішної авторизації користувач потрапляє у головне меню додатка, де він може переглянути усі створені ним форми, редагувати або видалити кожну з них, відфільтрувати форми за ключовими словами Назва та Опис, перейти до створення нової форми і т.д .

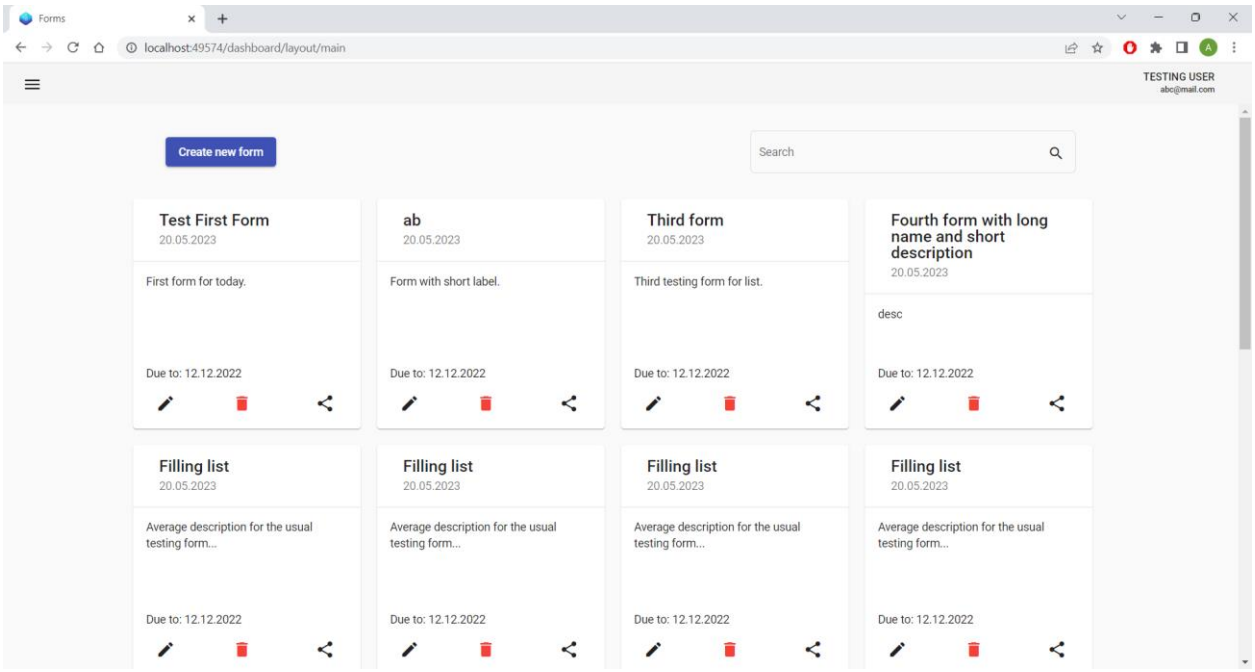



Рисунок 3.4 — головна сторінка додатка

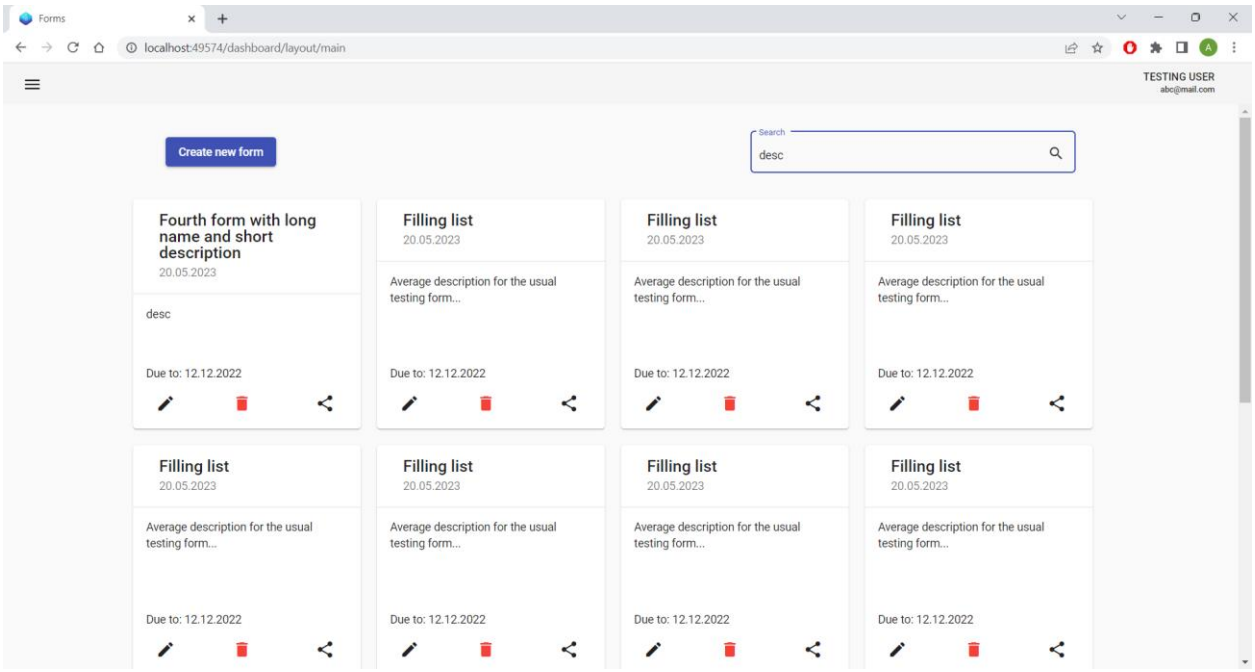



Рисунок 3.5 — фільтрація форм за полями Назва та Опис і ключовим словом desc

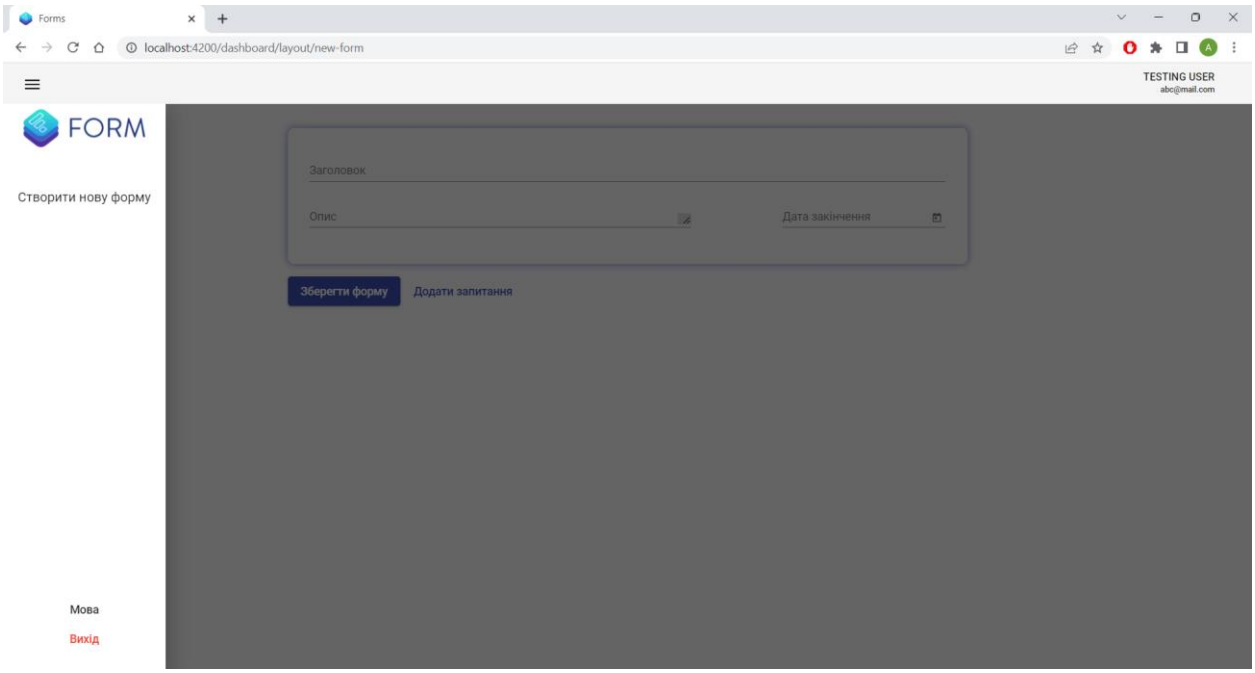



Рисунок 3.6 — панель з можливістю переходу в основне меню, короткими командами, вибором мови інтерфейсу та можливістю вийти з системи

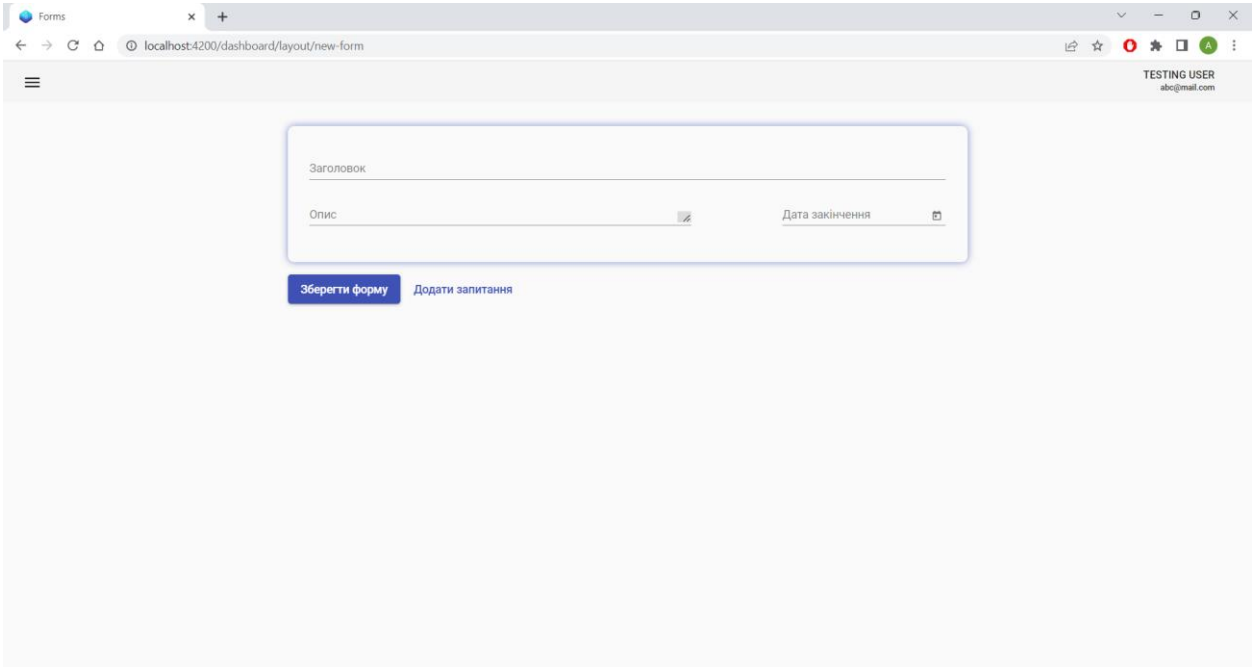



Рисунок 3.7 — сторінка створення нової форми

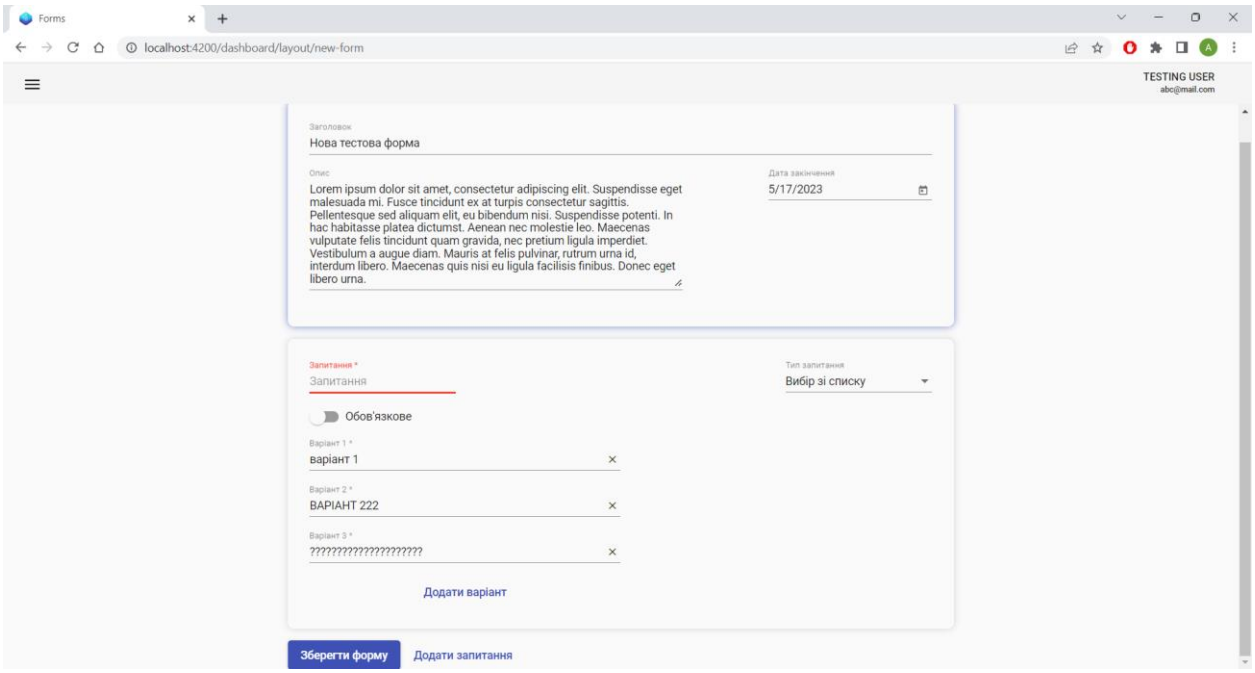



Рисунок 3.8 — сторінка створення нової форми

У цій же сторінці відкривається форма для редагування.

Після збереження користувач може обрати необхідну йому форму та натиснути «Поділитись», після чого з’явиться спливаюча підказка з посиланням, яке користувач копіює та передає іншим користувачам, яким потрібно пройти дане опитування.

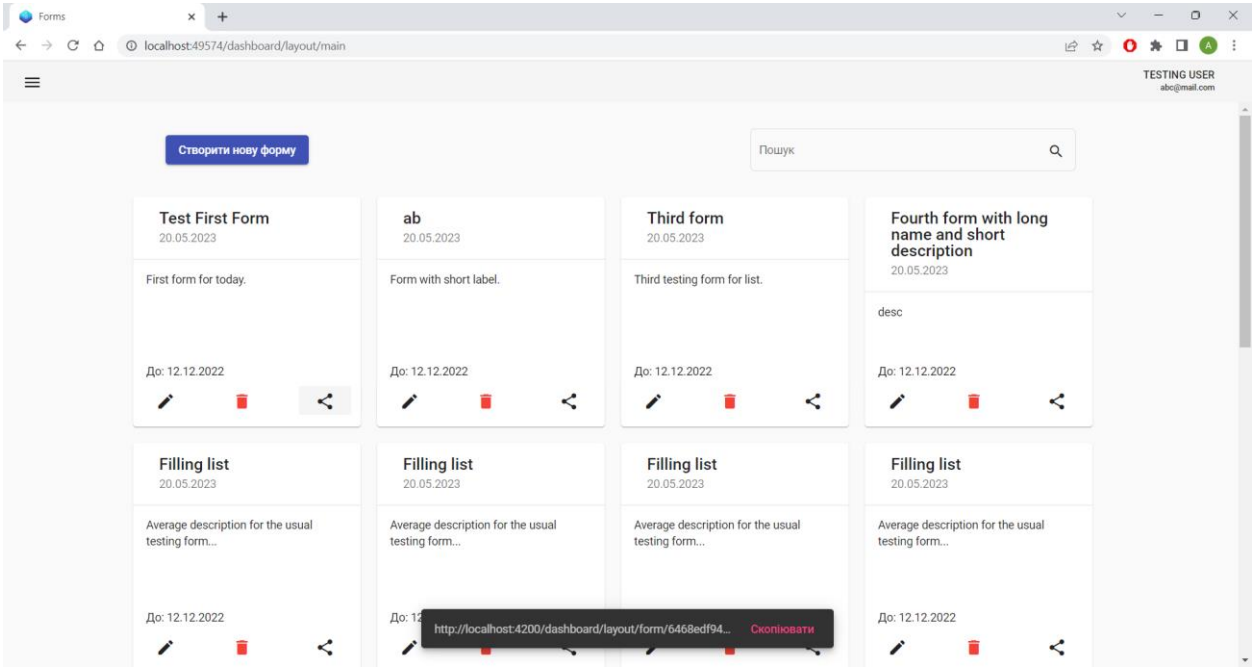



Рисунок 3.9 — можливість поділитись формою

Перейшовши за даним посиланням, користувач потрапляє у меню проходження форми.

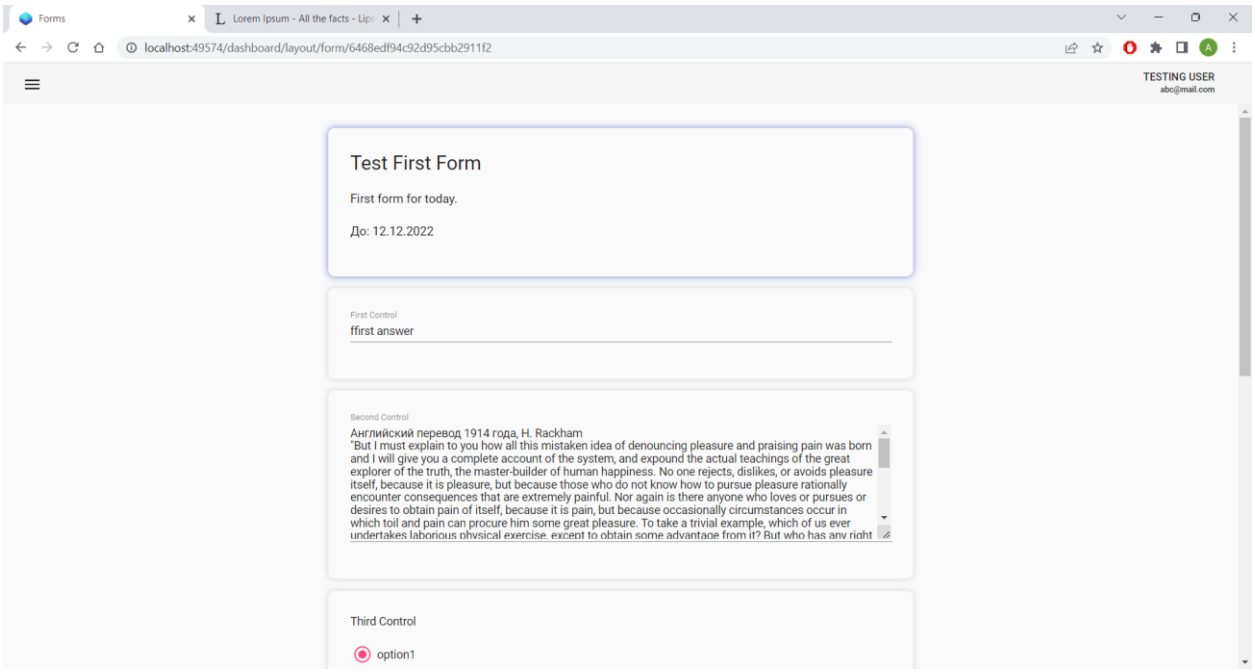



Рисунок 3.10 — сторінка проходження форми

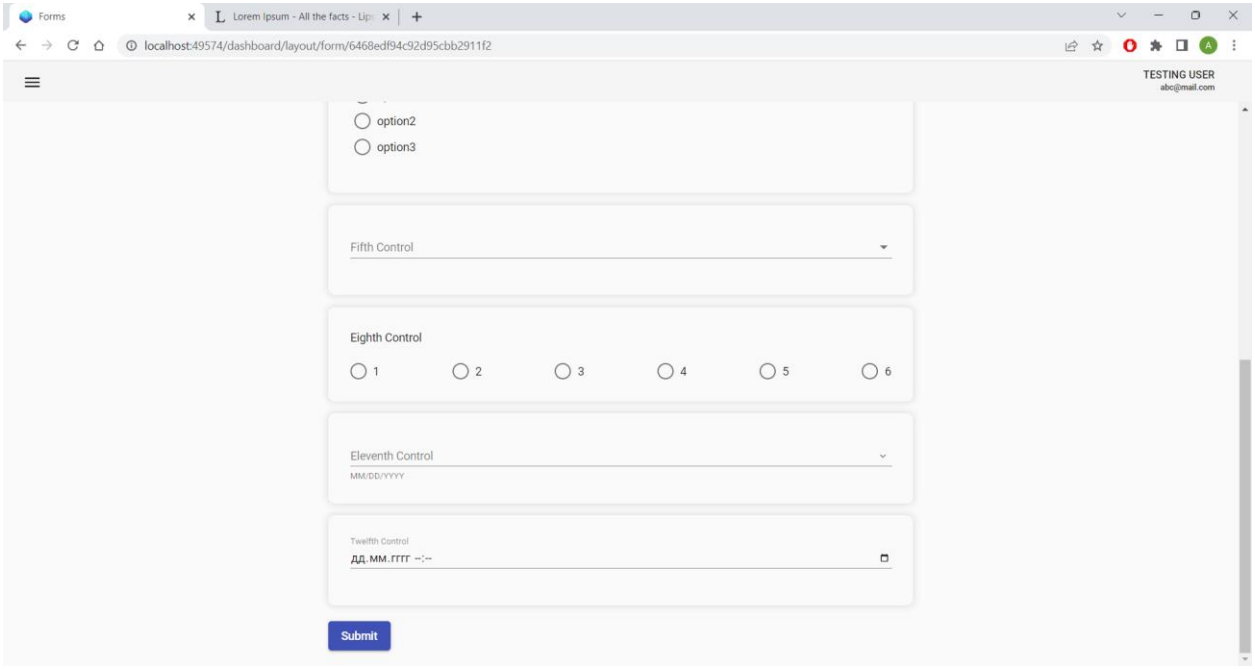



Рисунок 3.11 — приклади типів запитань та можливість відповісти на форму

Після натискання «Відповісти» відповідь записується у базу даних і в разі успіху користувач потрапляє у головне меню.

Власник форми, натиснувши в основному меню на форму, може перейти до перегляду усіх відповідей, які на неї надавали.

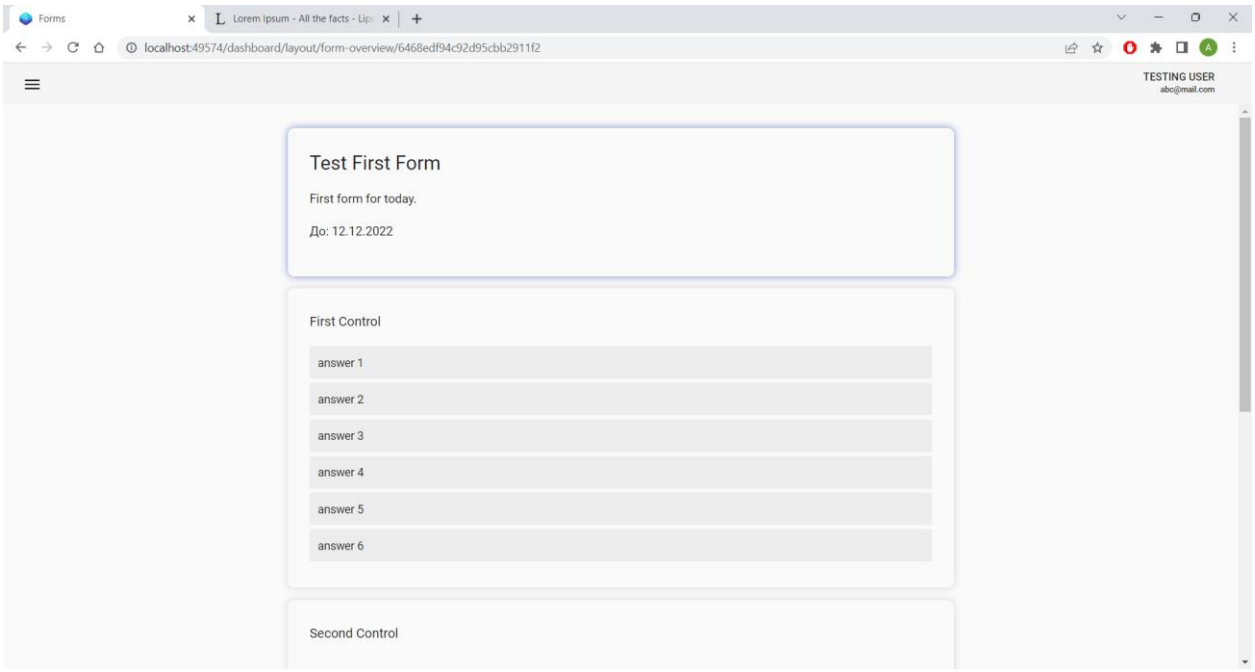



Рисунок 3.12 — приклади перегляду відповідей на форму

## 3.2 Можливості використання

Даний сервіс може використовуватись в різних сферах, таких як освіта(проведення модульного, залікового, підсумкового контролю тощо), ресурсний менеджмент(дані про робочі навички, задоволеність роботою, потреби тощо), маркетинг(збір даних про споживачів) і т.д. Користувачів даного продукту можна умовно розділити на 2 категорії: інтерв’юер та респондент. Завдяки простому та інтуїтивному інтерфейсу кожен з них може зручно користуватись продуктом. Так, відповідно до потреб кожної з категорій, для інтерв’юерів є можливість створювати опитування, редагувати їх, переглядати та аналізувати відповіді, а для респондентів – проходити опитування, отримувати результати або переглядати загальну статистику наданих відповідей.

Завдяки обраним технологіям, які є досить оптимізованими, час виконання усіх згаданих запитів не залежить від розміру вхідних даних та конкретних операцій, а лише від швидкості інтернет-з’єднання, що буде дуже зручним для користувачів.

## 3.3 Порівняння з наявними продуктами на ринку

У додатку Б наведено порівняння розробленого сервісу з іншими наявними на ринку продуктами, які було розглянуто раніше. З нього бачимо, що застосунок має весь необхідний функціонал, що дозволить використати його в університетському середовищі для створення закритої екосистеми, яка не залежить від зовнішніх сервісів. Так бачимо, що застосунок має деякі переваги порівняно з іншими схожими сервісами, такі як: можливість налаштування та розширення функціонала завдяки відкритому API та вихідному коду застосунку, можливість інтеграції з іншими платформами(як отримання даних про результати та статистику тестувань, так і вбудовування безпосередньо клієнтської частини в інший сервіс), а також не вимагає платних підписок.

# ВИСНОВКИ

В даній роботі був розроблений сервіс для створення та проведення опитувань для університету. В процесі дослідження та розробки було оцінено сучасний стан об'єкта дослідження та проведений огляд наявних на ринку систем. На основі цього було виявлено недоліки популярних систем, таких як Google Forms, ClassTime та Blackboard, що стимулювало розробку власного сервісу.

Метою даної роботи було створення сервісу, який забезпечує зручне створення та проведення тестувань. Завдання включали аналіз вимог до системи, вибір технологій, розробку архітектури та реалізацію функціональності.

У роботі було використано стек технологій MEAN (MongoDB, Express, Angular, Node.js), який дозволяє створити потужний та масштабований вебзастосунок. Також було використано Angular Material для розробки зручного та привабливого користувацького інтерфейсу.

У результаті було створено функціональний сервіс, який дозволяє створювати та проводити тестування та отримувати результати. Реалізовані функції системи відповідають сучасному рівню наукових і технічних знань і технологій.

Ступінь впровадження та можливі галузі або сфери використання результатів роботи є доволі широкими. Розроблений сервіс може бути успішно використаний в освітніх установах, зокрема університетах, коледжах та школах, для організації електронного тестування та оцінювання знань студентів.

Крім освітніх установ, система також може бути використана в інших галузях, де необхідно проводити тести, опитування або оцінювання. Наприклад, в компаніях під час підбору персоналу, для оцінки навичок співробітників або для проведення сертифікаційних екзаменів. Також система може бути використана в організаціях, які здійснюють підготовку та навчання фахівців у різних сферах.

Враховуючи швидкий розвиток технологій, зростання вимог до ефективності та зручності використання систем для створення та проведення тестувань, подальші дослідження та розробки в цій області є вкрай важливими. Продовження розробок може зосередитися на розширенні функціональності системи, вдосконаленні її інтерфейсу та користувацького досвіду, впровадженні нових методик оцінювання та аналізу результатів, підтримці різних типів питань та завдань, а також на розробці мобільних додатків для зручного доступу до системи з різних пристроїв.

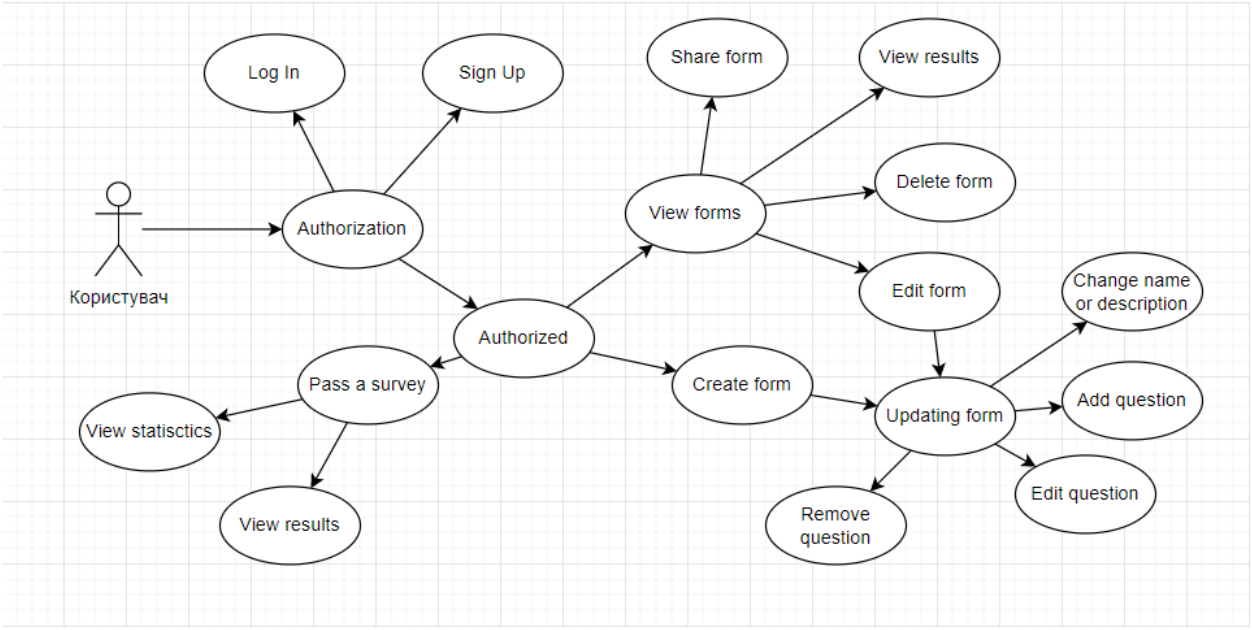
Також можливим напрямком розширення є інтеграція з іншими системами, наприклад, системами управління навчальним процесом(наприклад Triton Student), що дозволить використовувати розроблену систему в комплексі з наявними платформами для навчання та курсового матеріалу. Дослідження та розробки в галузі автоматизації тестувань та оцінювання знань мають потенціал для подальшого росту і розширення своїх можливостей. Враховуючи значущість цієї області та постійну потребу в удосконаленні процесів тестування, продовження розробок за відповідною тематикою є доцільним та привабливим з практичної та наукової точок зору.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Angular. Документація. URL: https://angular.io. (Дата звернення: 29.04.2023).
2. Authentication in Angular & JWT. URL: http://surl.li/heftf. (Дата звернення: 29.04.2023).
3. Angular Reactive Forms: The Ultimate Guide to FormArray. URL: http://surl.li/heftl. (Дата звернення: 29.04.2023).
4. Angular. Angular Material. Angular Material. URL: https://material.angular.io/. (Дата звернення: 29.04.2023).
5. Team M. D. MongoDB Documentation. MongoDB: The Developer Data Platform | MongoDB. URL: https://www.mongodb.com/docs. (Дата звернення: 29.04.2023).
6. Official Git Documentation. URL: https://git-scm.com/doc/. (Дата звернення: 29.04.2023).
7. Фрімен Е. Патерни проектування. - Вильямс, 2019.
8. Ньюкірк А. Інтернет програмування. - Вильямс, 2017.
9. Нілсон Б. Велика книга HTML5, CSS3 та JavaScript. - ДМК Прес, 2015.
10. CryptoJS: growing collection of standard and secure cryptographic algorithms implemented in JavaScript using best practices and patterns. URL:https://code.google.com/archive/p/crypto-js/ (Дата звернення: 29.04.2023).
11. D. Flanagan, “JavaScript: The Definitive Guide, 7th Edition” – 2020.
12. G. Lim, “Beginning Node.js, Express & MongoDB Development” – 2019.
13. NPM: library and registry for JavaScript software packages. URL:https://www.npmjs.com/ (Дата звернення: 29.04.2023).
14. EJS: Embedded JavaScript templating. URL: https://ejs.co/ (Дата звернення: 29.04.2023).

# ДОДАТОК А

## Use-Case діаграма для застосунку



# ДОДАТОК Б

## Порівняння з наявними продуктами на ринку

| Критерій | Розроблений сервіс | Google Forms | ClassTime | Blackboard |
|---|---|---|---|---|
| Функціонал | Створення тестів, проведення тестувань, аналітика результатів | Створення опитувань та форм, збір відповідей, аналітика результатів | Створення різних типів тестів, опитувань та завдань, оцінка тестів | Курси, завдання, форуми, відеолекції, оцінювання |
| Користувацький інтерфейс | Зручний та інтуїтивний | Простий та зрозумілий | Простий та зрозумілий | Складний, комплексний інтерфейс |
| Розширюваність | Можливості налаштування та розширення функціонала завдяки відкритому API та вихідному коду | Обмежені можливості налаштування та розширення функціонала | Розширюється за допомогою додаткових модулів та плагінів | Не передбачено |
| Можливості інтеграції | Можлива інтеграція завдяки відкритому API | Обмежена можливість інтеграції | Обмежена інтеграція | Обмежена інтеграція |
| Вартість | Безплатний | Безплатна для основного функціонала, платні пакети для додаткових функцій | Має різні тарифні плани, включаючи безплатні та платні пакети з розширеними можливостями | Комерційний продукт, вартість залежить від обсягу та функціоналу |