**КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**
**БУДІВНИЦТВА І АРХІТЕКТУРИ**

**автоматизації і інформаційних технологій**

_____

(факультет)

**інформаційних технологій**

_____

(кафедра)

# ПОЯСНЮВАЛЬНА ЗАПИСКА
## ДО КВАЛІФІКАЦІЙНОЇ ВИПУСКНОЇ РОБОТИ
## НА ЗДОБУТТЯ ОСВІТНЬОГО РІВНЯ «БАКАЛАВР»

на тему: «Розробка веб-додатку для обміну навчальним матеріалом і виконанням індивідуальних завдань студентами»

**КОРСУН ІЛЛЯ РОМАНОВИЧ**

(прізвище, ім’я та по батькові студента повністю)

Київ, 2024 р.

**КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БУДІВНИЦТВА І АРХІТЕКТУРИ**

**автоматизації і інформаційних технологій**

_____

(факультет)

**інформаційних технологій**

_____

(кафедра)

**ЗАТВЕРДЖУЮ**

Завідувач кафедри ІТ

к.т.н., доцент Гончаренко Т.А.

_____

„__” _____2024 року

# ПОЯСНЮВАЛЬНА ЗАПИСКА
## ДО КВАЛІФІКАЦІЙНОЇ ВИПУСКНОЇ РОБОТИ НА ЗДОБУТТЯ ОСВІТНЬОГО РІВНЯ «БАКАЛАВР»

на тему: «Розробка веб-додатку для обміну навчальним матеріалом і виконанням індивідуальних завдань студентами»

Виконав: студент 4–го курсу, групи КН–20-1

Спеціальності: 122 «Комп`ютерні науки»

Освітня програма: «Інформаційні управляючі системи і технології»

(шифр і назва напряму підготовки, спеціальності)

Корсун І.Р.

(прізвище та ініціали)

Керівник к.т.н., доц. Горда О.В.

(прізвище та ініціали)

Рецензент к.т.н., доц. Шабала Є.Є.

(прізвище та ініціали)

Київ, 2024 р.

**КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**
**БУДІВНИЦТВА І АРХІТЕКТУРИ**

Факультет: автоматизації і інформаційних технологій
Кафедра: інформаційних технологій
Освітній рівень: «бакалавр» за ОП
Спеціальність: 122 «Комп`ютерні науки»
Освітня програма: Інформаційні управляючі системи і технології

**ЗАТВЕРДЖУЮ**

Завідувач кафедри ІТ

к.т.н., доцент Гончаренко Т.А.

_____

„__” _____2024 року

**ЗАВДАННЯ**
**ДО ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ ВИПУСКНОЇ РОБОТИ**
**НА ЗДОБУТТЯ ОСВІТНЬОГО РІВНЯ «БАКАЛАВР»**

Корсун Ілля Романович

1. Тема роботи: Розробка веб-додатку для обміну навчальним матеріалом і виконанням індивідуальних завдань студентами
затверджена наказом ректора КНУБА № 433/2 від 29.02.2024 р.
2. Керівник роботи: к.т.н, доцент Горда Олена Володимирівна
кафедри інформаційних технологій
3. Строк подання студентом роботи до захисту:
4. Зміст пояснювальної записки за розділами:
Р.1. Аналітичний огляд
Р.2. Методологія та проектування веб-додатку
Р.3. Програмна реалізація
Р.4. Ергономіка ІТ або техніко-економічне обґрунтування розробки
5. Інформаційні слайди:
С.1. Класифікація об’єкта дослідження
С.2. Дерево цілей
С.3. Аналіз існуючих платформ та порівняльні характеристики
С.4. Аналіз вимог для отримання кінцевого результату
С.5. Процес розробки веб-додатку
С.6. Функціональні та інтерфейсні особливості веб-додатку
С.7. Діаграми SADT

С.8. Проектування бази даних веб-додатку

С.9. Проектування модульної структури додатку

С.10. Техніко-економічне обґрунтування

6. Календарний план виконання атестаційної випускної роботи

| Види робіт та їх зміст | Дата виконання |
|---|---|
| Р. 1. Аналітичний огляд | Лютий 2024 р. |
| Р. 2. Методологія та проектування веб-додатку | Березень 2024 р. |
| Р. 3. Програмна реалізація | Квітень 2024 р. |
| Р. 4. Ергономіка ІТ або техніко-економічне обґрунтування розробки | Квітень 2024 р. |
| Остаточне оформлення роботи | Травень 2024 р. |
| Направлення роботи на рецензування | Червень 2024 р. |
| Попередній захист роботи на кафедрі | Червень 2024 р. |

7. Консультанти розділів атестаційної випускної роботи

| Розділ | Прізвище, ініціали та посада консультанта, представника комісії | дата | підпис |
|---|---|---|---|
| Ергономіка інформаційних технологій | д.т.н. проф. Рябчун Ю.В. | | |
| Прийом програмного продукту | к.т.н. доц., Шабала Є.Є. | | |

8. Дата видачі завдання: 18.11.2023 р.

Завідувач Гончаренко Т.А.

(підпис) (прізвище та ініціали)

Керівник Горда О.В.

(підпис) (прізвище та ініціали)

Студент Корсун І.Р.

(підпис) (прізвище та ініціали)

# АНОТАЦІЯ

**Корсун І.Р. Розробка веб-додатку для обміну навчальним матеріалом і виконанням індивідуальних завдань студентами.**

Дипломна бакалаврська робота за спеціальністю – «Комп'ютерні науки» – Київський національний університет будівництва та архітектури, Київ, 2024 рік.

Бакалаврську роботу присвячено дослідженню основних тенденцій, принципів та реалізацій веб-додатків для обміну навчальними матеріалами та виконання індивідуальних завдань студентами, їх інформаційного забезпечення та баз даних.

У роботі досліджено основні принципи та технології розробки інформаційного забезпечення для веб-додатків у сфері освіти, розроблено базу даних для зберігання та обробки інформації, що отримується від студентів та викладачів.

*Ключовими словами є: веб-додаток, системний аналіз, база даних, модель, захист даних.*

# ABSTRACT

**Korsun I.R. Development of a web application for the exchange of educational material and the implementation of individual tasks by students.**

Bachelor's thesis in Computer Science – Kyiv National University of Construction and Architecture, Kyiv, 2024.

The bachelor's thesis is devoted to the study of the main trends, principles and implementations of web applications for the exchange of educational materials and the performance of individual tasks by students, their information support and databases.

The paper investigates the basic principles and technologies of developing information support for web applications in the field of education, and develops a database for storing and processing information received from students and teachers.

Key words are: web application, system analysis, database, model, data protection.

Зміст

# ВСТУП

Сучасні інформаційні технології надають значні можливості для удосконалення освітнього процесу, зокрема шляхом впровадження інтерактивних інструментів та платформ. Однією з актуальних проблем є оптимізація обміну навчальними матеріалами та організація виконання індивідуальних завдань студентами. Вирішення цієї проблеми сприятиме підвищенню ефективності навчання, активізації пізнавальної діяльності студентів та формуванню у них навичок самостійної роботи.

**Мета роботи** полягає у розробці веб-додатку який забезпечить зручний та ефективний обмін навчальними матеріалами між викладачами та студентами, а також добавить інструменти для створення, виконання та перевірки індивідуальних завдань.

Обґрунтування необхідності розробки зумовлене невідповідністю сучасних платформ потребам освіти, потенційному підвищенню ефективності навчання та важливості розвитку навичок самостійної роботи.

У роботі планується використання HTML5, CSS3, та TypeScript для забезпечення кросплатформовості та інтеграції з іншими веб-сервісами. Додаток матиме багаторівневу структуру з моделями даних, API та механізмами кешування. Інтерфейс буде розроблений шляхом аналізу потреб користувачів. Функціонал включатиме обмін матеріалами, створення завдань, відстеження прогресу та автоматичну перевірку завдань з адаптивним навчанням.

Результати роботи можуть знайти застосування в освітній сфері на різних рівнях. Вищі навчальні заклади та школи зможуть використовувати веб-додаток для обміну матеріалами та контролю знань. У корпоративному навчанні інструмент допоможе створювати індивідуальні плати на проводити онлайн заняття. Для самоосвіти він надасть доступ до навчальних ресурсів та контролю прогресу. У науковій сфері забезпечить спільну роботу та онлайн-конференції.

**Актуальність та доцільність роботи для розвитку відповідної галузі науки або промисловості, особливо на користь України**

Існуючі платформи для дистанційного навчання, як Moodle, Google Classroom, Microsoft Teams, часто мають обмежений функціонал, недостатню гнучкість і не враховують специфіку української освітньої системи. Багато з них є комерційними, що створює фінансові бар'єри для їх використанні в Україні.

Розробка вітчизняного веб-додатку для обміну навчальними матеріалами та виконання завдань є актуальною через такі причини:

1. Підвищення якості освіти: використання сучасних методик та ефективного зворотного зв'язку;
2. Зменшення витрат: зниження витрат на друк матеріалів та організацію занять;
3. Підвищення доступності: освіта стане доступною для людей з обмеженими можливостями та мешканців віддалених регіонів;
4. Розвиток IT-галузі: створення нових робочих місць для підвищення конкурентоспроможності українських розробників.

Даний веб-додаток відповідає потребам української освітньої системи, враховуючи навчальні плани та методики викладання. Він має потенціал для комерціалізації на внутрішньому та зовнішньому ринку. Це сприятиме науковим дослідженням і впровадженню інноваційних підходів.

Впровадження веб-додатку підвищить рівень освіти в Україні, зробить її доступнішою і якіснішою, допоможе модернізувати освітню систему та підготувати фахівців, здатних конкурувати на світовому ринку праці.

**Об'єктом дослідження** даної дипломної роботи є процес обміну навчальними матеріалами та організації виконання індивідуальних завдань студентами в сучасному освітньому процесі.

**Предметом дослідження** є веб-система, що використовується для оптимізації цих процесів та сприяння розвитку самостійності та заохочення активності студентів.

У дослідженні будуть застосовані такі основні методи:

1. Теоретичні методи, які включають аналіз наукової літератури та нормативно-правових документів, системний аналіз для розуміння об’єкта дослідження як складної системи, а також порівняльних аналіз існуючих рішень;
2. Емпіричні методи, які полягатимуть у спостереженні за процесом використання наявних платформ для навчання, що дозволить виявити їхні переваги та недоліки;
3. Методи проектування та розробки, які передбачатимуть проектування архітектури та інтерфейсу веб-додатку, розробку програмного коду, а також тестування та налагодження програмного забезпечення для його якості та надійності.

# 1. АНАЛІТИЧНИЙ ОГЛЯД

## 1.1 Класифікація та основні характеристика об’єкта дослідження

У контексті цільової аудиторії веб-додаток для обміну навчальними матеріалами та виконання індивідуальних завдань можна класифікувати за різними характеристиками, що враховують потреби різних сегментів користувачів. Нижче, більш детально розглянемо цю класифікацію.

### *1.1.1 За цільовою аудиторією*

Перший тип орієнтований на вищі навчальні заклади, такі як університети та інститути, і має розширений функціонал, включаючи оцінювання студентів, керування курсами та сприяння взаємодії між викладачами та студентами.

Другий тип призначений для середніх загальноосвітніх шкіл, де інтерфейс і функціонал спрощені й адаптовані до потреб викладачів та учнів.

Третій тип цілеспрямований на корпоративне навчання, включаючи створення та проведення навчальних курсів для співробітників, внутрішню комунікацію та відстеження прогресу.

Четвертий тип призначений для дистанційного навчання, де можна отримати доступ до навчального матеріалу та виконання завдання з будь-якого місця з доступом до Інтернету.

Нарешті, п’ятий тип призначений для самоосвіти, де користувачі можуть самостійно вивчати новий матеріал або вдосконалювати свої навички за допомогою навчальних курсів, вправ, тестів та інших ресурсів, доступних в будь-який час.

Ця класифікація враховує різноманітні потреби користувачів та специфічні особливості навчального процесу в різних контекстах.

### *1.1.2 За функціональними можливостями*

Обмін навчальними матеріалами включає передачу текстових документів, презентацій, відео та аудіо матеріалів між викладачем та студентами. Це може бути як загальний матеріал для усіх учасників групи, так і індивідуальні матеріали для кожного студента залежно від його потреб та інтересів.

Створення та виконання індивідуальних завдань включає тести, завдання з відкритою відповіддю та практичні завдання, які студент виконує самостійно. Це дозволяє персоналізувати навчання, враховуючи потреби та рівень кожного конкретного студента.

Оцінювання та зворотний зв'язок включає систему оцінювання робіт студентів та надання зворотного зв'язку щодо їхньої продуктивності. Коментарі викладача допоможуть студентам зрозуміти свої помилки та покращити свої навички.

Спілкування включає можливості для спілкування між викладачем та студентами, такі як коментування, чати та відеоконференції. Вони дозволяють студентам отримувати необхідну підтримку та відповіді на питання, а також спілкування між собою для обміну думками та досвідом.

### *1.1.3 За типом доступу*

Відкритий доступ може бути використаний для забезпечення учням безкоштовного доступу до навчальних ресурсів, включаючи відкриті підручники, відеоуроки, наукові статті та інші матеріали. Це дозволяє студентам розширювати свої знання та досліджувати нові теми за допомогою доступних онлайн-ресурсів.

Доступ з обмеженням може бути використаний для створення індивідуальних освітніх просторів для педагогів та учнів, де вони зможуть обмінюватися конфіденційною інформацією, співпрацювати над проектами та

мати доступ до додаткових ресурсів, які доступні лише обмежуваному колу осіб.

Оплата за доступ відкриває можливість скористатися преміальними освітніми ресурсами, що дозволяє отримати додаткові переваги, такі як інтерактивні курси, особисте керівництво викладача та ексклюзивний навчальний матеріал. Це може стимулювати студентів активно залучатися до навчального процесу та мотивувати їх до самовдосконалення.


Рисунок 1.1.1 Класифікація об’єкта дослідження

### *1.1.4 Основні характеристики об’єкта дослідження*

Інтерактивність є однією з основних характеристик цього веб-додатка. Він створений для стимулювання активної взаємодії між учасниками освітнього процесу, забезпечуючи зручний обмін інформацією, можливістю обговорення навчальних тем та швидкого отримання зворотного зв’язку.

Адаптивність є ще однією важливою характеристикою веб-додатку. Враховуючи різноманітність навчальних закладів та індивідуальні особливості користувачів, додаток розроблений з урахування гнучкості та можливості налаштування, щоб відповідати потребам різних користувачів.

Мультимедійність веб-додатку допомагає зробити навчання більш цікавим та ефективним. Завдяки підтримці різних форматів навчальних матеріалів – тексту, зображень, аудіо та відео, користувачі можуть отримувати доступ до різноманітного контенту, що в свою чергу сприяє кращому засвоєнню матеріалу.

Мобільність є ще однією ключовою рисою цього веб-додатка. Доступність з різних пристроїв, таких як комп’ютери, планшети та смартфони, дозволяє

користувачам навчатися у будь-який час та в будь-якому місці, що робить процес навчання більш гнучким та доступним.

Безпека є невід’ємною частиною веб-додатка. Захист персональних даних користувачів та конфіденційність навчальних матеріалів – це пріоритетні завдання, які враховуються при розробці та впровадженні додатка.

Інтеграція з іншими інформаційними системами навчального закладу є важливою перевагою даного веб-додатка. Можливість автоматизувати деякі процеси та підвищити ефективність управління навчальним процесом шляхом інтеграції з іншими системами робить її цінним інструментом для освітніх установ.

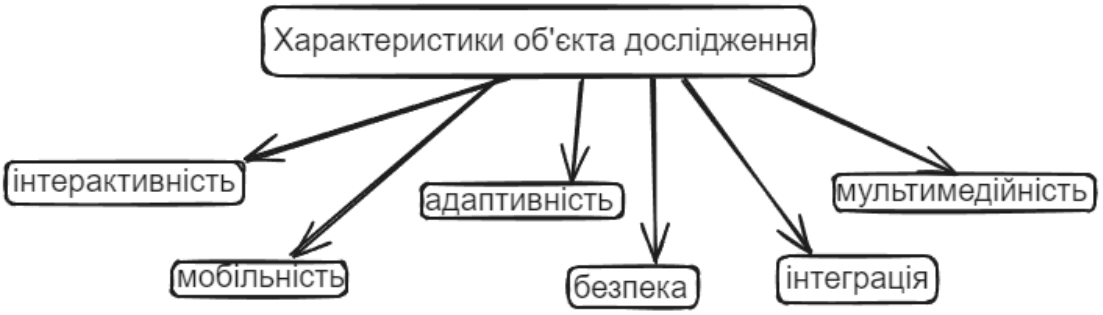



Рисунок 1.1.2 Характеристики об’єкта дослідження

## 1.2 Характеристика предметної області та обґрунтування актуальності розробки

### *1.2.2 Характеристика предметної області*

Предметна область дослідження, що охоплює сферу освітніх технологій, веб-розробки та інформаційних систем, виявляється динамічною і різноманітною. Динаміка полягає у постійних змінах і покращеннях як самого процесу навчання, так і технічних засобів, що використовуються для нього. Наприклад, нові методи навчання, які базуються на інтерактивних технологіях або адаптивних системах, регулярно виникають та розвиваються, створюючи постійних попит на нові технології та підходи до їх імплементації.[10]

Що стосується широкого спектру застосувань, освітні веб-додатки знаходять своє застосування в найрізноманітніших сферах. Вони використовуються не лише в загальноосвітніх та вищих навчальних закладах, а й у сфері корпоративного навчання, онлайн-курсах, тренінгах та самоосвіті. Така широка адаптивність відображається в різноманітності функціональних можливостей освітніх веб-застосунків, які можуть включати в себе від інтерактивних уроків та вправ до систем управління навчальним процесом та оцінювання.

Високі вимоги до якості стають невід'ємною частиною розробки освітніх веб-систем. Користувачі очікують, що такі системи будуть не лише зручними та ефективними, а й безпечними для використання. Це означає, що розробники повинні приділяти особливу увагу аспектам якості програмного забезпечення, включаючи відповідність сучасним стандартам безпеки та захисту даних.[13]

Необхідність індивідуалізації стає ключовим аспектом у розробці освітніх веб-додатків, оскільки потреби користувачів можуть значно відрізнятися. Індивідуалізація може виявлятися у можливості налаштування програмного забезпечення під конкретні потреби користувачів, включаючи різні методи навчання, темпи та стилі викладання, а також можливості адаптації під особисті особливості та потреби начальних програм.

### *1.2.3 Обґрунтування актуальності розробки*

Актуальність розробки веб-додатку для обміну навчальними матеріалами та виконання індивідуальних завдань обумовлена наступними чинниками: зростання популярності дистанційного навчання в контексті пандемії COVID-19 та зростаючою необхідністю в ефективних та зручних інструментах для його здійснення; потреба у підвищенні ефективності навчального процесу через застосування інтерактивних інструментів та автоматизації рутинних завдань, що дозволить викладачам приділити більше уваги індивідуальному супроводу студентів та підвищити загальну якість навчання; сприяння розвитку цифрової

грамотності студентів через використання веб-додатків у навчальному процесі, що відіграє важливу роль у їхній подальшій професійній кар’єрі.

### *1.2.4 Постановка проблеми*

Існуючі платформи для дистанційного навчання часто не відповідають вимогам української освітньої системи, характеризуються обмеженим функціоналом та недостатньою увагою до індивідуальних потреб користувачів. Таким чином, постає актуальна проблема у розробці веб-додатку, що має наступні ключові вимоги:

- Адаптація до специфіки української освітньої системи;
- Забезпечення широкого спектру інструментів для обміну навчальними матеріалами, виконання індивідуальних завдань та контролю над рівнем знань;
- Зручність та інтуїтивна зрозумілість інтерфейсу;
- Забезпечення високого рівня безпеки та конфіденційності даних;
- Сприяння розвитку самостійності та пізнавальної активності студентів.

## 1.3 Аналіз ступеня розробленості об’єкта проектування, досягнутого на даний момент часу

На даному етапі виконання роботи проведено попереднє дослідження, яке включає аналіз існуючих рішень, визначення вимог до веб-додатку, вибір технологічного стеку, проектування архітектури та розробку прототипу інтерфейсу.

Проведений аналіз існуючих рішень охоплює вивчення різноманітних платформ для дистанційного навчання та обміну навчальними матеріалами, як вітчизняних, так і зарубіжних. Виявлено їх сильні та слабкі сторони, визначено основні тенденції розвитку. На основі цього аналізу та потреб користувачів

сформульовано перелік функціональних та нефункціональних вимог до майбутнього веб-додатку.

Вибір технологічного стеку визначає основні технології, які будуть використані для розробки додатку: фронтенд (HTML, CSS, TypeScript, React) і бекенд (Node.js, база даних – PostgreSQL). Проектування архітектури включає розробку попередньої архітектури додатку, визначення основних модулів та їх взаємодію.

Розробка передбачає створення прототипу інтерфейсу користувача для основних функцій платформи. Таким чином, на даному етапі виконано підготовчу роботу, яка є основною для подальшої розробки веб-додатку. Проте, необхідно враховувати, що розробка програмного забезпечення є ітеративним процесом, тому в ході роботи можуть виникнути нові ідеї та вимоги які потребуватимуть коригування початкового плану.[2]

## 1.4 Визначення проблемних питань та мети роботи

У ході попереднього дослідження було виявлено ряд проблемних питань, що потребують вирішенню.

По-перше, обмежений функціонал існуючих платформ: більшість існуючих платформ для дистанційного навчання не забезпечують повного спектру інструментів, необхідних для ефективної організації навчального процесу, обміну матеріалами, виконання та перевірки індивідуальних завдань.

По-друге, низька адаптивність: існуючі платформи часто не враховують специфіку різних навчальних дисциплін та рівнів підготовки студентів, що ускладнює їх використання у різних освітніх контекстах.

По-третє, складність використання: інтерфейси деяких платформ можуть бути складними для освоєння як викладачами, так і студентами, що знижує ефективність їх використання.

По-четверте, недостатня інтеграція: відсутність інтеграції з іншими інформаційними системами навчальних закладів створює додаткові труднощі при організації навчального процесу.

По-п’яте, проблеми з безпекою даних: не всі платформи забезпечують належний рівень захисту персональних даних та конфіденційності навчальних матеріалів.

**Метою роботи** є розробка веб-додатку, який дозволить вирішити виявлені проблемні питання та забезпечити широкий функціонал, високу адаптивність, зручність використання, інтеграцію з іншими системами та безпеку даних. Додаток надаватиме викладачам та студентам усі необідні інструменти для ефективної організації навчального процесу, включаючи обмін матеріалами, створення та виконання індивідуальних завдань, оцінювання, спілкування та аналітику. Він буде гнучким та регульованим під потреби різних навчальних дисциплін та рівнів підготовки студентів.

Інтерфейс платформи буде інтуїтивно зрозумілим та простим у використанні як для викладачів, так і для студентів. Додаток буде інтегрований з іншими інформаційними системами навчальних закладів, що дозволить автоматизувати деякі процеси та підвищити ефективність управління навчальним процесом даних та конфіденційності навчальних матеріалів.

Отже, мета роботи полягає у створенні сучасного, зручного та ефективного веб-додатку, який зможе задовольнити потреби української освітньої системи та сприятиме підвищенню якості навчання.

## 1.5 Очікувані результати дипломного проекту

У ході виконання дипломного проекту очікується отримати такі результати:

- Розроблений веб-додаток. Це включає створення функціонального прототипу, який демонструватиме основні функції та можливості системи.

Прототип дозволить оцінити зручність використання, виявити потенційні проблеми та внести необхідні корективи на ранніх етапах розробки. Повноцінний додаток буде готовий для використання у реальних умовах. Він матиме інтуїтивно зрозумілий інтерфейс і забезпечуватиме широкий спектр функцій для обміну навчальними матеріалами, створення та виконання індивідуальних завдань, оцінювання, спілкування та аналітики.

- Технічна документація. Вона включає технічне завдання з детальним описом вимог до системи, включаючи функціональні та нефункціональні вимоги, вимоги до інтерфейсу, архітектури та безпеки. Проектна документація описує архітектуру додатку, структуру бази даних, API, алгоритми роботи основних модулів. Посібник користувача міститиме інструкції з використанням платформи для викладачів та студентів, а інструкція з адміністрування забезпечить вказівки з налаштування та підтримки додатку для адміністраторів.
- Наукові результати. Сюди входить аналіз існуючих рішень: систематизований огляд платформ для дистанційного навчання та обміну навчальними матеріалами, виявлення їхніх переваг та недоліків. Визначення основних вимог до веб-додатку для обміну навчальними матеріалами та виконання індивідуальних завдань з урахуванням потреб української освітньої системи. Розробка методології проектування та розробки веб-додатків для освітніх цілей. Оцінка ефективності розробленої веб-платформи на основі експериментального дослідження або аналізу даних про його використання.[9]
- Практичні результати. Впровадження розробленого додатку у навчальний процес одного або декількох навчальних закладів України. Застосування цієї платформи має призвести до підвищення якості навчання, збільшення зацікавленості студентів та покращення їхніх навчальних матеріалів, організацію очних занять та перевірку робіт студентів.

Очікується, що результати дипломного проекту матимуть наукову та практичну цінність, сприятимуть розвитку освітніх технологій в Україні та підвищенню якості освіти.

### 1.6 З метою досягнення основних цілей комп'ютерного проектування

Для зниження складності отримання проектних рішень та планування завдань веб-система має спростити процес створення та розповсюдження навчальних матеріалів, автоматизувати створення та перевірку завдань, а також полегшити планування навчального процесу для викладачів. Це досягається за рахунок інтуїтивно зрозумілого інтерфейсу, використання шаблонів та готових елементів, а також інструментів для автоматизації рутинних операцій.

Для скорочення часу, необхідного для вирішення завдань, платформа має прискорити процес обміну інформацією між викладачами та студентами, автоматизувати перевірку завдань, надати студентам швидкий доступ до навчальних матеріалів та зворотного зв'язку. Це дозволить зекономити час як викладачів, так і студентів, що можна буде використати для більш глибокого вивчення матеріалу та розвитку індивідуальних здібностей.

Для зниження вартості проектування, виробництва та експлуатаційних витрат розробка веб-додатку вимагає менших фінансових витрат порівняно зі створенням традиційних друкованих матеріалів та проведенням очних занять. Крім того, використання додатку дозволить зменшити витрати на папір, друк, оренду приміщень та інші витратні матеріали.

Для підвищення якості та техніко-економічного рівня проектних результатів застосування сучасних технологій веб-розробки дозволить створити застосунок з високим рівнем функціональності, надійності та безпеки. Використання інтерактивних елементів, мультимедійних матеріалів та адаптивних алгоритмів навчання сприятиме підвищенню якості освітнього процесу та кращому засвоєнню знань студентами.[18]

Для зниження вартості натурного моделювання та випробувань додаток дозволить проводити віртуальне моделювання та тестування навчальних сценаріїв, що значно зменшить потребу у проведенні дорогих натурних експериментів та випробувань.[15]

Отже, веб-додаток для обміну навчальними матеріалами та виконання індивідуальних завдань має потенціал для значного підвищення ефективності та якості освітнього процесу, зниження його вартості та сприяння розвитку цифрової грамотності студентів.

## 1.7 Дерево цілей для веб-додатку обміну навчальними матеріалами та виконання індивідуальних завдань

**Головна мета:** створення ефективного та зручного веб-застосунку для обміну навчальними матеріалами та використання індивідуальних завдань, що сприятиме підвищенню якості освіти в Україні.

**Цілі першого рівня** включають підвищення ефективності навчального процесу (1.1), покращення якості навчальних матеріалів (1.2), сприяння самостійній роботі студентів (1.3), забезпечення зручного та швидкого доступу до матеріалів (1.4), зниження витрат на освіту (1.5), та спрощення організації навчального процесу (1.6).

**Цілі другого рівня** деталізуються наступним чином:

Підвищення ефективності навчального процесу включає автоматизацію перевірки завдань, надання індивідуалізованих рекомендацій та можливість відстеження прогресу студентів (2.1.1-2.1.3).

Покращення якості навчальних матеріалів охоплює можливість використання різних форматів (текст, відео, аудіо, інтерактивні елементи), створення єдиної бази матеріалів та забезпечення їх актуальності (2.2.1-2.2.3).

Сприяння самостійній роботі студентів передбачає надання можливості самостійного вибору темпу та форми навчання, доступу до додаткових матеріалів та консультацій від викладачів (2.3.1-2.3.3).

Забезпечення зручного та швидкого доступу до матеріалів полягає у використанні інтуїтивно зрозумілого інтерфейсу, адаптивного дизайну для різних пристроїв та швидкого завантаження сторінок (2.4.1-2.4.3).

Зниження витрат на освіту включає зменшення витрат на друк матеріалів, економію часу викладачів на перевірку завдань та оптимізацію використання ресурсів (2.5.1-2.5.3).

Спрощення організації навчального процесу охоплює автоматизацію розкладу занять, можливість формування груп та розподілу завдань та зручний інструментарій для спілкування між учасниками процесу (2.6.1-2.6.3).

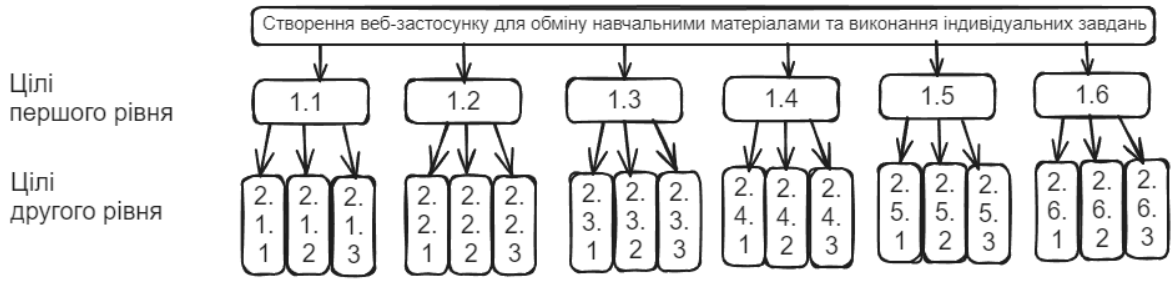



Рисунок 1.7 Дерево цілей

## 1.8 Аналіз проблем

**Основні проблеми,** з якими стикаються існуючі платформи, включають обмежений функціонал, недостатню адаптивність до потреб різних навчальних закладів, складність використання, недостатню інтеграцію з іншими інформаційними системами, та проблеми з безпекою даних.

**Деталізація проблем:**

Обмежений функціонал існуючих платформ є однією з ключових проблем, з якими стикаються викладачі та студенти під час навчання та викладання. Серед основних недоліків можна відокремити відсутність

необхідних інструментів, які б дозволяли створювати інтерактивні завдання, а також недостатню гнучкість у налаштуваннях. Це ускладнює процеси навчання та знижує ефективність взаємодії між студентами та викладачами.

Ще однією проблемою є недостатня адаптивність платформ до потреб різних начальних закладів. Часто платформи не враховують особливості різних навчальних дисциплін та не забезпечують достатню підтримку для різних форм навчання, таких як очна, дистанційна або змішана. Це ускладнює процес організації навчання та може призводити до зниження мотивації студентів.

Крім того, складність використання деяких платформ також є серйозною проблемою. Заплутаний інтерфейс та недостатня кількість інструкцій та підказок можуть призвести до того, що користувачі втрачають бажання використовувати платформу або не використовують її на повну потужність.

Не менш важливою є недостатня інтеграція з іншими інформаційними системами. Часто потрібно виконувати ручне перенесення даних з однієї системи в іншу, що призводить до зайвої втрати часу та можливих помилок. Відсутність єдиного входу ускладнює доступ до інформації та може породжувати проблеми з безпекою даних.

І, нарешті, проблеми з безпекою даних стають все більш актуальними в сучасному цифровому світі. Ризик несанкціонованого доступу до даних та недостатній захист від кібератак ставлять під загрозу конфіденційність та цілісність інформації, що може призвести до серйозних наслідків для користувачів та навчальних закладів.

Вирішення цих проблем та досягнення поставлених цілей дозволить створити веб-додаток, який стане ефективним інструментом для покращення якості освіти в Україні.

## 1.9 Аналіз існуючих досліджень та рішень на тему роботи

Аналіз існуючих досліджень та рішень на тему роботи дозволяє зробити висновок, що тема розробки веб-додатку для обміну навчальними матеріалами та виконанням індивідуальних завдань є актуальною та широко досліджуваною.

Існуючі дослідження демонструють наступне: у роботі досліджується вплив використання онлайн-платформ на академічну успішність студентів. Автори виявили, що студенти, які активно використовували онлайн-системи для навчання, демонстрували вищі результати, ніж ті, хто навчався традиційними методами.[5]

У статті розглядається різні підходи до проектування інтерфейсів користувача для освітніх веб-застосунків. Пропонується використовувати принципи універсального дизайну та враховувати потреби різних категорій користувачів.[20]

У книзі представлено огляд сучасних технологій веб-розробки, які можуть бути використані для створення освітніх застосунків, розглядаються різні фреймворки, бібліотеки та інструменти, а також надаються рекомендації щодо їх вибору та використання.[12]

У статті аналізуються різні моделі оцінювання знань студентів в онлайн-середовищі. Автори розглядають переваги та недоліки автоматизованого та ручного оцінювання, а також пропонують комбінований підхід.[27]

Щодо існуючих рішень, зазначено наступне: Moodle – популярна платформа з відкритими кодом, Google Classroom – безкоштовний сервіс від Google, Microsoft Teams – корпоративна платформа для спільної роботи.


Рисунок 1.9.1 Приклад Moodle платформи

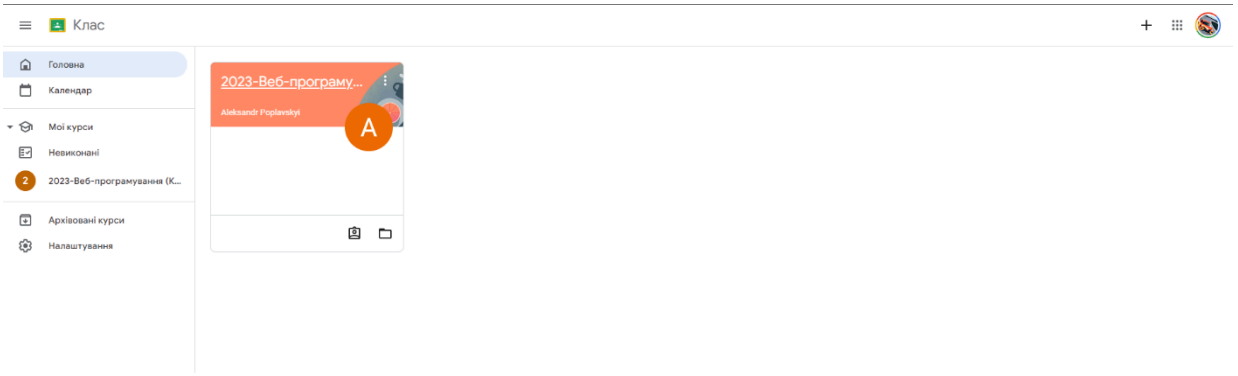



Рисунок 1.9.2 Приклад сервісу Google Classroom

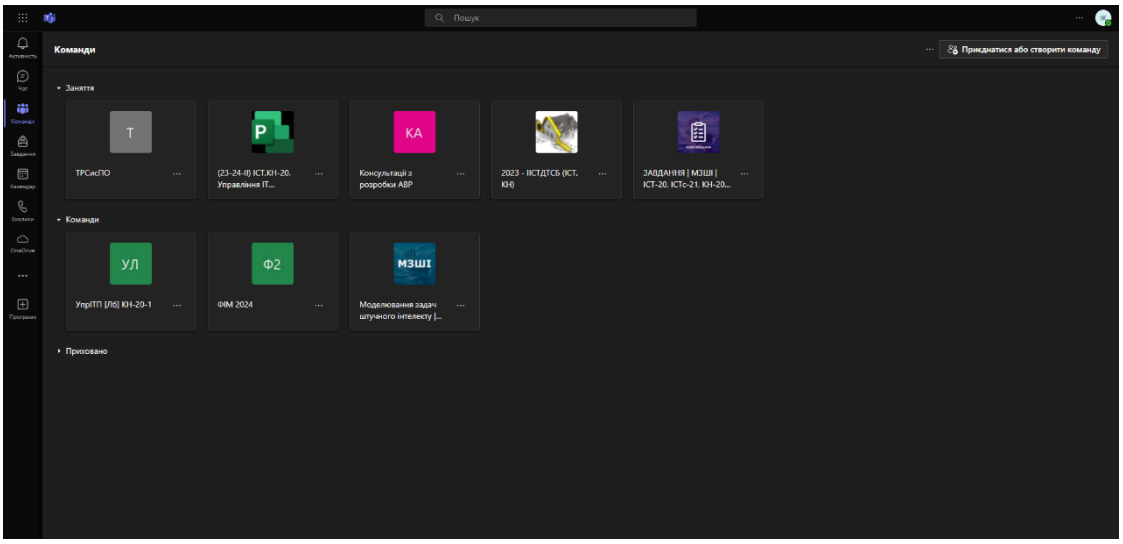



Рисунок 1.9.3 Приклад платформи Microsoft Teams, веб-версія додатку

Висновки аналізу показують, що розробка веб-додатку для обміну навчальними матеріалами для виконання індивідуальних завдань є актуальною та перспективною задачею. Незважаючи на наявність великої кількості існуючих платформ, вони не завжди повністю задовольняють потреби української освітньої системи та мають певні обмеження.

Тому розробка оригінального рішення, яке враховуватиме специфіку вітчизняної освіти та використовуватиме сучасні технології веб-розробки, є доцільною та необхідною. Деякі компоненти та ідеї з існуючих рішень можуть бути використані при розробці нового веб-додатку, проте необхідно враховувати, що вони повинні бути адаптовані до конкретних потреб та вимог української освітньої системи.

Таблиця 1.1

Порівняльні характеристики існуючих додатків

| **Функції/Аспекти** | **insync** | **Moodle** | **Google Classroom** | **Microsoft Teams** |
|---|---|---|---|---|
| Управління курсами | + | + | + | + |
| Надсилання завдань | + | + | + | + |
| Обмін матеріалами | + | + | + | + |

Продовження таблиці 1.1

| | | | | |
|---|---|---|---|---|
| Оцінювання та зворотний зв’язок | + | + | + | + |
| Спільна робота в реальному часі | + | - | - | + |
| Інтуїтивно зрозумілий і зручний інтерфейс | + | - | + | + |
| Дизайн інтерфейсу, зручний для | + | + | + | + |

| | | | | |
|---|---|---|---|---|
| мобільних пристроїв | | | | |

## 1.10 Постановка завдання

**Мета:** розробка веб-додатку для обміну навчальними матеріалами та виконання індивідуальних завдань студентами, що забезпечить ефективну організацію навчального процесу, покращення якості навчальних матеріалів, сприяння самостійній роботі студентів, зручний та швидкий доступ до матеріалів, зниження витрат на освіту, спрощення організації навчального процесу.

**Вхідна інформація:** навчальні матеріали (текстові документи, презентації, відео, аудіо, інтерактивні елементи, тести, завдання), дані про користувачів (ПІБ, посада, контактні дані, навчальні дисципліни), дані про студентів (ПІБ, група, контактні дані), розклад занять (інформація про розклад занять, групи, викладачів), результати виконання завдань (відповіді студентів на тести, виконані завдання, оцінки).

**Вихідна інформація:** інтерфейс користувача: відображення навчальних матеріалів у різних форматах, інструменти для створення та редагування матеріалів, форми для виконання завдань, інструменти для оцінювання та надання зворотного зв’язку, інструменти для спілкування (форуми, чати), звіти про прогрес навчання. Дані в базі даних: зберігання навчальних матеріалів, завдань, результатів виконання, інформації про користувачів та розкладу занять. Сповіщення: email-повідомлення просто нові завдання, оцінки, коментарі викладачів.

**Умови використання завдання:** використання сучасних веб-технологій (HTML, CSS, TypeScript, React, Node.js, PostgreSQL), кросплатформність, безпека, інтеграція з іншими інформаційними системами навчального закладу.

**Критерії успішності:** функціональність, зручність використання, надійність, безпека.

Виконання цього завдання дозволить створити веб-додаток, який стане ефективним інструментом для покращення якості освіти та сприятиме розвитку цифрової грамотності студентів та підвищенню його якості.

## 1.11 Аналіз пропозицій і вимог до вирішення, виконання, отримання кінцевого результату та вихідних даних

Пропозиції та вимоги до вирішення:

**Функціональні вимоги:**

- Система управління навчальним контентом (LMS): можливість створення, редагування та видалення курсів, лекцій, презентацій, тестів, завдань; підтримка різних форматів контенту (текст, зображення, аудіо, відео); організація контенту в структуровані модулі та розділи; можливість встановлення доступу до контенту для різних круп користувачів.
- Система виконання та оцінювання завдань: створення та призначення індивідуальних та групових завдань; різні типи завдань (тести, есе, файли для завантаження, форуми); автоматична та ручна перевірка завдань; зворотний зв’язок від викладача до студента.
- Система комунікації та співпраці: форуми для обговорення тем курсу; чати для особистого спілкування між студентами та викладачами; можливість проведення відеоконференцій.
- Система моніторингу та аналітики: відстеження прогресу студентів у виконанні завдань; збір статистики про активність студентів та використання матеріалів; генерація звітів для викладачів та адміністрації.
- Інші функції: календар подій; оголошення; рейтинг студентів; система рекомендацій навчальних матеріалів.

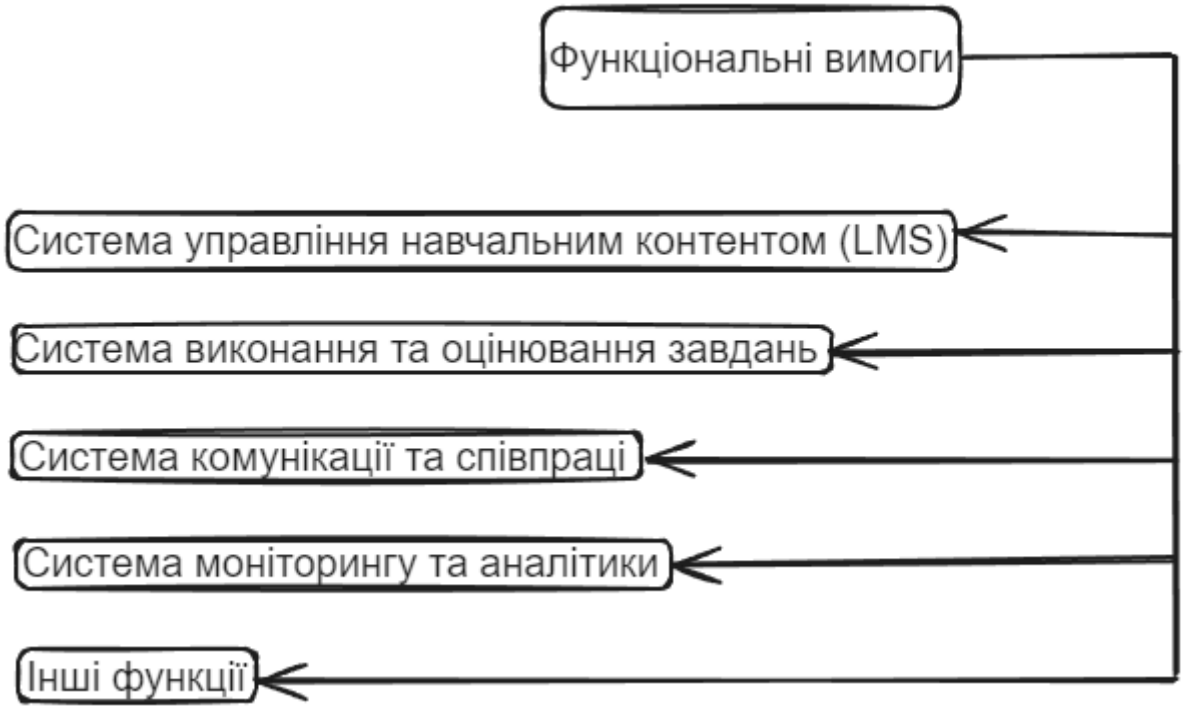



Рисунок 1.11.1 Схема функціональних вимог

**Нефункціональні вимоги:**

- Зручність використання: інтуїтивно зрозумілий інтерфейс, легка навігація, зручний пошук матеріалів та завдань;
- Надійність: мінімальна кількість збоїв, швидке відновлення після збоїв, резервне копіювання даних;
- Продуктивність: швидке завантаження сторінок, мінімальний час відгуку системи;
- Масштабованість: можливість розширення системи при збільшенні кількості користувачів та обсягу даних;
- Безпека: захист від несанкціонованого доступу, шифрування даних, захист від кібератак.


Рисунок 1.11.2 Схема нефункціональних вимог

**Технічні вимоги:**

- Кросплатформність: додаток повинен працювати на різних операційних системах (Windows, macOS, Linux) та браузерах (Chrome, Firefox, Safari, Edge);
- Відкритість: використання відкритих стандартів та технологій для забезпечення сумісності з іншими системами;
- Можливість інтеграції: додаток повинен мати API для інтеграції з іншими інформаційними системами навчального закладу (наприклад, системою обліку студентів).


Рисунок 1.11.3 Схема технічних вимог

**Очікуваний результат та вихідні дані:**

Готовий до використання веб-додаток, технічна документація (посібник користувача, інструкції з адміністрування, опис API), вихідні дані (дані про активність користувачів, результати виконання завдань, звіти про прогрес навчання, статистичні дані про використання застосунку).

# 2. МЕТОДОЛОГІЯ ТА ПРОЕКУВАННЯ ВЕБ-ДОДАТКУ

## 2.1 Загальна методологія розробки прикладного рішення

Для розробки веб-додатку спрямованого на обмін навчальними матеріалами та виконання індивідуальних завдань, використовується комбінована методологія, яка поєднує в собі підходи Agile та Scrum. Ця методологія забезпечує гнучкість у процесі розробки, швидку адаптація до змін вимог та ефективну співпрацю між членами команди.[3]

Процес розробки поділяється на кілька основних етапів:

**Планування:** на цьому етапі визначаються цілі та завдання проекту, проводиться аналіз вимог користувачів, розробляється концепція та прототип застосунку, а також створюється план проекту з визначенням термінів та ресурсів;

**Проектування:** у цьому етапі роботи розробляється інформаційна архітектура, проектується база даних, розробляється дизайн інтерфейсу користувача (UI/UX), а також створюється детальне технічне завдання;

**Розробка:** на даному етапі проводиться реалізація проекту. Розробляється серверна частина (backend) з використанням Node.js та фреймворку TRPC, клієнтська частина (frontend) з використанням React.js та Radix-UI, інтегрується з базою даних PostgreSQL, а також розробляється API для взаємодії між frontend та backend;

**Тестування:** на цьому етапі роботи проводиться комплексне тестування розробленої системи. Воно включає в себе модульне тестування окремих компонентів, інтеграційне тестування взаємодії між компонентами, системне тестування всього застосунку, навантажувальне тестування для перевірки продуктивності та юзабіліті-тестування залученими потенційними користувачами;

**Впровадження та підтримка:** на заключному етапі розгортається додаток на сервері, проводиться навчання користувачів, ведеться моніторинг

роботи платформи та збір відгуків користувачів. Також на цьому етапі проводиться виправлення помилок та внесення покращень до додатку.

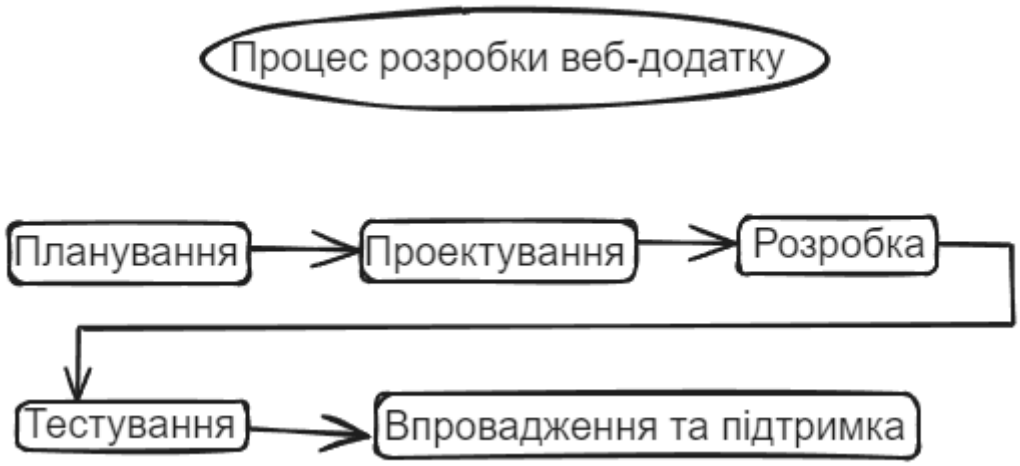



Рисунок 2.1 Схема процесу розробки веб-додатку

Під час розробки застосовуються такі методи та техніки:

- **Agile:** цей метод дозволяє швидко реагувати на зміни вимог та забезпечує гнучкість процесу розробки;
- **Scrum:** фреймворк Scrum використовує спринти (короткі ітерації) та щоденні зустрічі для синхронізації роботи команди;
- **Історії користувачів:** опис функціональності з точки зору користувача, що допомагає краще зрозуміти його потреби та спрощує розробку;
- **Шаблони та прототипи:** використання візуального представлення інтерфейсу додатку перед початком розробки для оцінки його зручності та функціональності;
- **Автоматизоване тестування:** використання інструментів для автоматизації процесу тестування, що дозволяє швидко виявляти та виправляти помилки;
- **Неперервна інтеграція/неперервна доставка:** практика автоматизації процесів збірки, тестування та розгортання додатку, що дозволяє швидко випускати нові версії та забезпечувати їх якість.

Ці методи та техніки допомагають ефективно організувати процес розробки веб-додатку, забезпечуючи його високу якість та відповідність потребам користувачів.

## 2.2 Функціональні та інтерфейсні особливості веб-додатку

### *2.2.1 Функціональні особливості*

**Система управління навчальним контентом (LMS):** ця система дозволяє адміністраторам та викладачам створити, організовувати та керувати навчальними курсами. Вона надає можливість гнучкого створення курсів з різною структурою, включаючи лекції, презентації, тести, завдання та інші матеріали у різних форматах, таких як текст, зображення, аудіо та відео. Крім того, вона має вбудований редактор, який спрощує процес редагування матеріалів та дозволяє використовувати різноманітні елементи форматування, включаючи формули, таблиці тощо. Також дана система забезпечує можливість класифікації доступу, дозволяючи призначати різні ролі користувачам (студентам, викладачам, адміністраторам) та налаштовувати права доступу до навчальних матеріалів.

**Система виконання та оцінювання завдань:** одна з ключових функцій цієї системи – це можливість виконання та оцінювання завдань студентами. Вона підтримує різноманітні типи завдань, такі як тести з різними типами питань (вибір однієї або декількох відповідей, встановлення відповідності, введення тексту), завдання з відкритою відповіддю, завантаження файлів тощо, Після виконання завдань система автоматично або за необхідності власноруч перевіряє їх і надає об’єктивну оцінку. Крім того, вона забезпечує можливість зворотного зв’язку, що дозволяє викладачам коментувати роботи студентів та надавати індивідуальні рекомендації.

**Система комунікації та співпраці:** для підтримки взаємодії між учасниками навчального процесу ця система надає різноманітні інструменти

комунікації. Зокрема, вона має можливість створення форумів для обговорення тем курсу, питань та відповідей, індивідуальні та групові чати для спілкування між студентами та викладачами, а також інтеграцію з сервісами відеоконференцій для проведення онлайн-занять та консультацій.

**Система моніторингу та аналітики:** для оцінки ефективності навчання та прийняття обґрунтованих рішень щодо його покращення ця система забезпечує можливість відстеження прогресу студентів та викладачів у виконанні завдань та освоєнні матеріалу, збір статистики про активність користувачів, успішність виконання завдань, використання матеріалів, а також генерацію звітів для аналізу ефективності навчання.

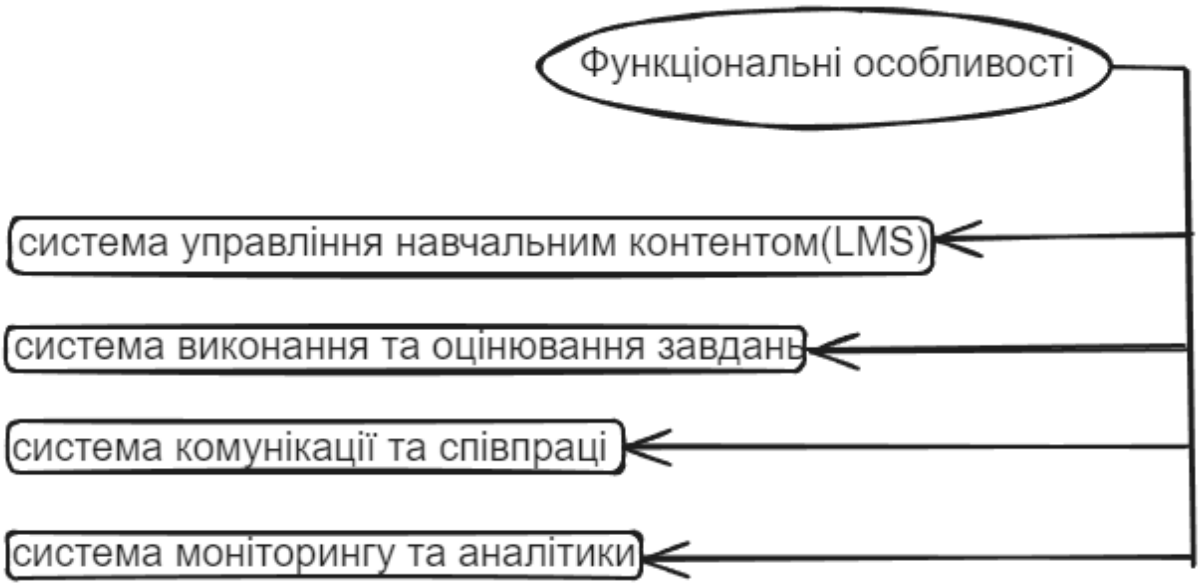



Рисунок 2.2.1 Схема функціональних особливостей

### *2.2.2 Інтерфейсні особливості*

Щоб забезпечити зручність користування та ефективність використання, розроблений веб-додаток має ряд інтерфейсних особливостей:

1. **Інтуїтивно зрозумілий інтерфейс:** дизайн інтерфейсу розробляється згідно сучасних стандартів та з урахуванням принципів мінімалізму та чіткої структури. Він має бути легко зрозумілим для користувачів будь-якого рівня підготовки, з чіткими інтерактивними

елементами, чіткою навігацією та інтуїтивно зрозумілими іконками та підказками.

2. **Адаптивний дизайн:** веб-платформа повинна правильно відображатися на різних пристроях з різними розмірами екранів, таких як комп'ютери, планшети та смартфони. Це забезпечує зручний доступ до системи з будь-якого пристрою та в будь-якому місці.

3. **Персоналізація:** користувачі мають можливість налаштовувати інтерфейс згідно своїх потреб та вподобань. Це може включати вибір теми оформлення, мови інтерфейсу, розташування та відображення елементів.

4. **Інтерактивність:** для підвищення зацікавленості та ефективності навчання, в інтерфейсі використовуються інтерактивні елементи, такі як графіки, діаграми та інтерактивні завдання. Це сприяє активній участі студентів у процесі навчання та полегшує їх розуміння складних концепцій.

5. **Доступність:** важливим аспектом розробки є забезпечення доступності веб-додатку для всіх користувачів, включаючи людей з обмеженими можливостями. Наприклад, система повинна підтримувати екранні диктори та інші технології, що полегшують використання для людей з різними видами інвалідності.

Загалом, комбінація цих функціональних та інтерфейсних особливостей робить веб-застосунок ефективним інструментом для організації навчального процесу, полегшує взаємодію між учасниками навчання та сприяє підвищенню якості освіти.


Рисунок 2.2.2 Схема інтерфейсу

## 2.3 Вибір методів, інструментів і технологій

**Структурний аналіз:** завдання розробки веб-додатку розбивається на наступні структурні компоненти: Frontend (клієнтська частина) – відповідає за відображення інтерфейсу користувача, взаємодію з користувачем та передачу даних на сервер; Backend (серверна частина) – відповідає за обробку запитів від клієнта, роботу з базою даних, логіку застосунку та формування відповідей клієнту; База даних – зберігає всі дані додатку (інформацію про користувачів, курси, матеріали, завдання, результати виконання тощо).

Діаграма SADT (Structured Analysis and Design Technique) допомагає зрозуміти та моделювати функціональну структуру системи. Вона включає різні рівні деталізації, зокрема контекстну діаграму та діаграми нижчих рівнів, які деталізують основні процеси.[31]

**Контекстна діаграма**

Контекстна діаграма показує систему як єдиний процес з її взаємодіями з зовнішніми циклами, приклад діаграми веб-додатку:

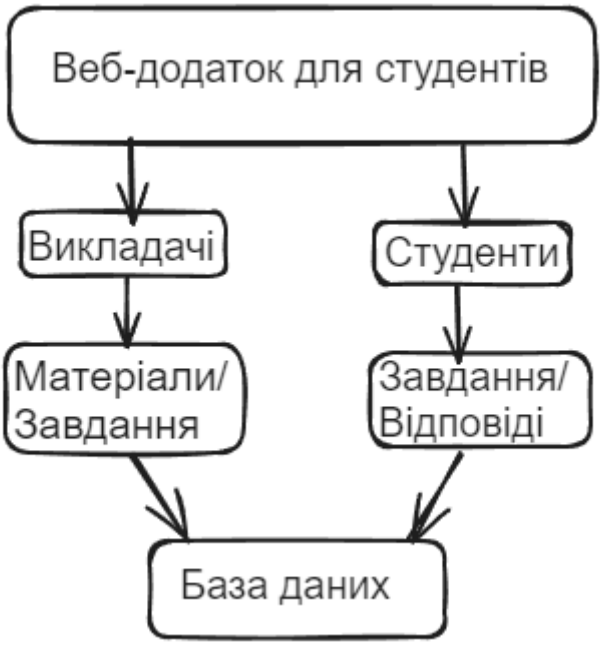



Рисунок 2.3.1 Контекстна діаграма

На діаграмі нижчого рівня ми розбиваємо основний процес на під процеси. Для системи веб-додатку можна виділити кілька основних під процесів, таких як управління користувачами, управління курсами, управління матеріалами та завданнями.

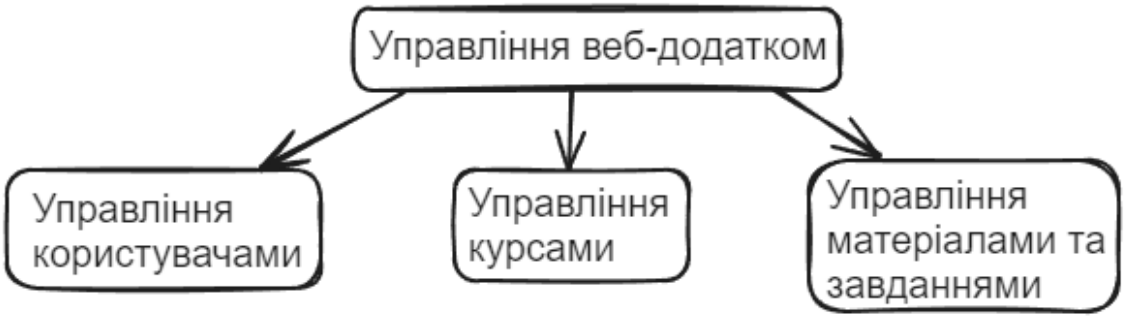



Рисунок 2.3.2 Діаграма нижчого рівня

Нижче буде представлена деталізація під процесів, кожен з під процесів зображений окремо.


Рисунок 2.3.3 Діаграма управління користувачами


Рисунок 2.3.4 Діаграма управління курсами


Рисунок 2.3.5 Діаграма управління матеріалами та завданнями

Ці діаграми SADT допомагають візуалізувати структуру та функціональні процеси веб-додатку, розбиваючи його на більш детальні під процеси.

**Функціональний аналіз:** завдання розбивається на функціональні блоки: авторизація та аутентифікація – реєстрація, вхід в систему, відновлення паролю,

розмежування прав доступу; управління курсами – створення, редагування, видалення курсів, додавання розділів та матеріалів; управління завданнями – створення, редагування, видалення завдань, призначення завдань студентам, встановлення термінів виконання; виконання завдань – інтерфейс для виконання завдань студентами, можливість завантаження файлів, перегляд результатів; оцінювання – автоматична та ручна перевірка завдань, виставлення оцінок, надання зворотного зв’язку; комунікація – форуми, чати, відеоконференції; аналітика – збір статистики про активність користувачів, успішність виконання завдань, формування звітів.


Рисунок 2.3.6 Схема функціонального аналізу

**Інформаційний аналіз:** вхідні дані – дані, які вводить користувач (логін, пароль, відповіді на запитання, файли), дані, які передаються з інших систем; вихідні дані – інформація, що відображається на екрані(сторінки курсів, завдання, результати), дані, що передаються до інших систем.

**Обмеження:** часові обмеження – розробка повинна бути завершена в рамках встановленого терміну; бюджетні обмеження – розробка повинна вкладатися у визначений бюджет; технічні обмеження – застосунок повинен працювати на різних платформах та пристроях, бути сумісним з іншими системами забезпечувати високий рівень безпеки.

**Методи:** Agile та Scrum – для гнучкого управління проектом та забезпечення швидкої адаптації до змін вимог; проектування UI/UX – для створення зручного та інтуїтивно зрозумілого інтерфейсу користувача; об’єктно-орієнтоване програмування (ООП) – для структурування коду та забезпечення його повторного використання; тестування – для забезпечення якості та надійності додатку.

**Інструменти та технології:** Frontend – React.js, Radix-UI; Backend – Node.js, TRPC; База даних – PostgreSQL; інші – Git (система контролю версій), Tyrbopack (збирач модулів).

Даний аналіз дозволяє визначити основні напрямки роботи над проектом та обрати оптимальні методи та інструменти для його реалізації.

## 2.4 Формалізація об’єкта дослідження

Веб-додаток для обміну навчальними матеріалами та виконання індивідуальних завдань можна формалізувати як систему, що складається з таких компонентів, як бази даних. Для зберігання даних веб-платформи буде використовуватися реляційна база даних (PostgreSQL).

**Структура бази даних:**

- Користувачі (User):
  1. Унікальний ідентифікатор користувача;
  2. Фото користувача;
  3. Ім’я;
  4. Прізвище;
  5. Електронна адреса;

6. Дата створення;
7. Дата редагування.

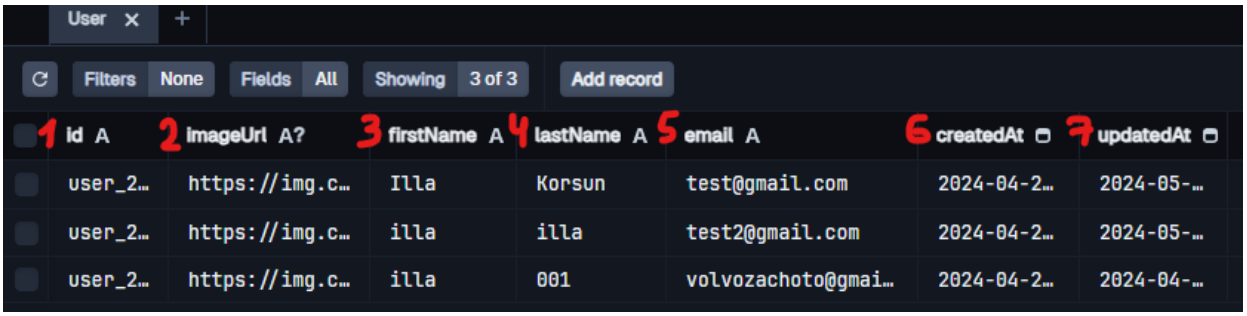



Рисунок 2.4.1 База даних «Користувачі (User)»

- Організації (Organization):
  1. Унікальний ідентифікатор організації;
  2. Назва організації;
  3. Власник організації;
  4. Запрошення;
  5. Дошки;
  6. Лог активності;
  7. Списки;
  8. Картки;
  9. Учасники.

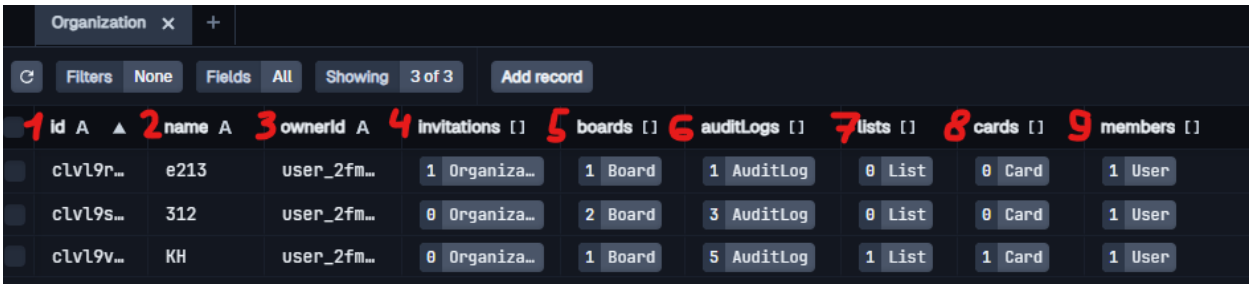



Рисунок 2.4.2 База даних «Організації (Organization)»

- Дошки (Boards):
  1. Унікальний ідентифікатор дошки;
  2. Назва дошки;
  3. Організація;

4. Списки;
5. Дата створення;
6. Дата редагування.

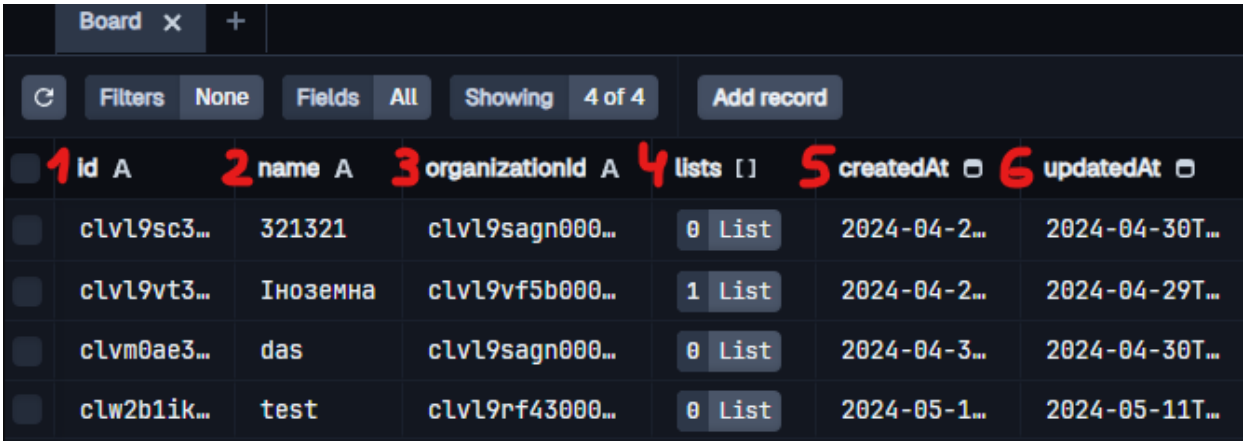



Рисунок 2.4.3 База даних «Дошки (Boards)»

- Списки (List):
  1. Унікальний ідентифікатор списків;
  2. Назва списку;
  3. Порядок;
  4. Організація;
  5. Дошка;
  6. Список карток;
  7. Дата створення;
  8. Дата редагування.

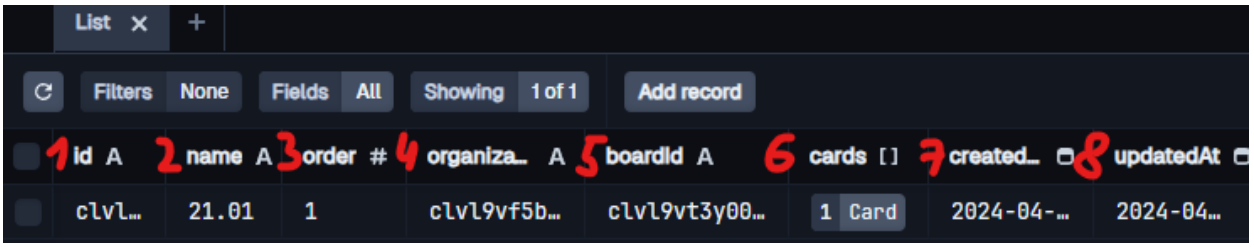



Рисунок 2.4.4 База даних «Списки (List)»

- Картки (Card):
  1. Унікальний ідентифікатор карток;

2. Назва картки;
3. Порядок;
4. Опис;
5. Організація;
6. Список;
7. Список коментарів;
8. Дата створення;
9. Дата редагування.

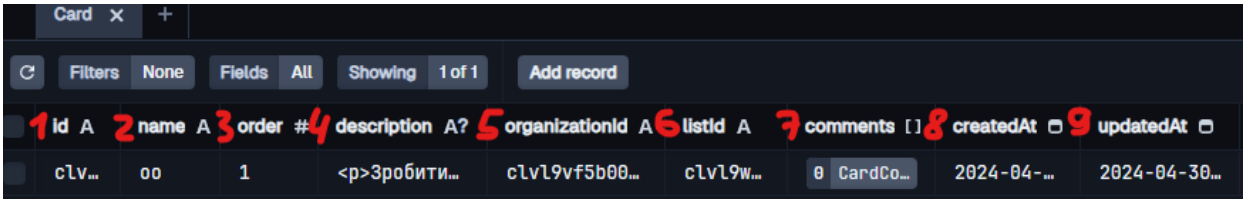



Рисунок 2.4.5 База даних «Картки (Card)»

## 2.5 Моделі та математична постановка задач

Моделі та методи для аналітики моніторингу включають в себе застосування теорії графів для представлення взаємозв'язків між користувачами, курсами, матеріалами та завданнями, а також використання різних алгоритмів обходу графів для аналізу цих зв'язків та виявлення закономірностей у поведінці користувачів. Теорія графів є потужним інструментом для ефективного моделювання складних систем взаємозв'язків та дозволяє застосувати різноманітні математичні алгоритми для аналізу даних, такі як визначення найпопулярніших курсів, матеріалів та завдань, а також виявлення груп користувачів зі схожими інтересами.[14]

Моделі та методи для рекомендаційної системи включають колаборативну фільтрацію, яка основана на оцінках та поведінці інших користувачів. Для групування користувачів за інтересами використовуються алгоритми кластеризації, такі як k-means, а для формування рекомендацій будуть застосовані алгоритми, що ґрунтуються на схожості між користувачами або

елементами. Колаборативна фільтрація є популярним методом рекомендацій, оскільки враховує думки та вподобання інших користувачів, що дозволяє підвищити точність рекомендацій. Задача полягає у побудові матриці користувач-елемент, заповненні відсутніх оцінок та формуванні рекомендацій на основі схожості з іншими користувачами або елементами.

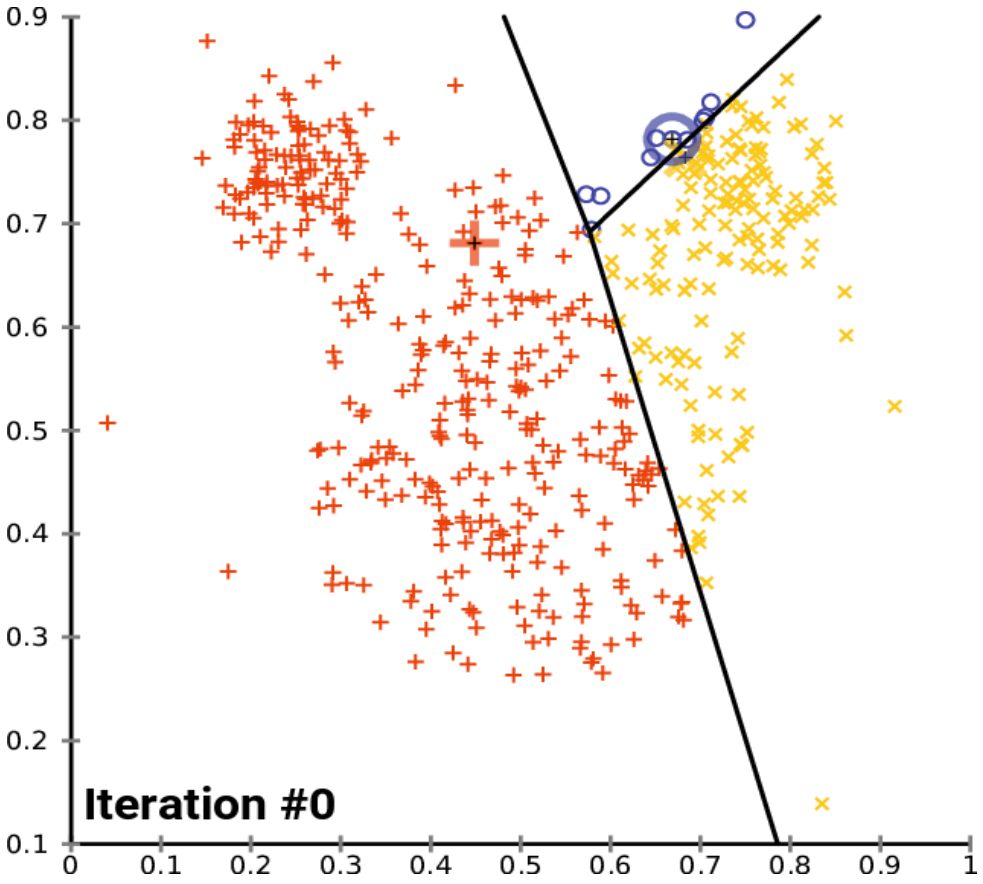



Рисунок 2.5.1 Приклад k-means групування[16]

Моделі та методи для автоматичної перевірки завдань включають застосування машинного навчання для класифікації відповідей на відкриті запитання або оцінювання якості текстів. Цей підхід передбачає використання алгоритмів класифікації, таких як Naive Bayes, або регресії, наприклад лінійної регресії, для навчання моделі на наборі розмічених даних, що складається з відповідей та їх оцінок. Машинне навчання відкриває можливості для автоматизації процесу перевірки завдань, що значно економить час викладачів та прискорює надання зворотного зв’язку студентам.

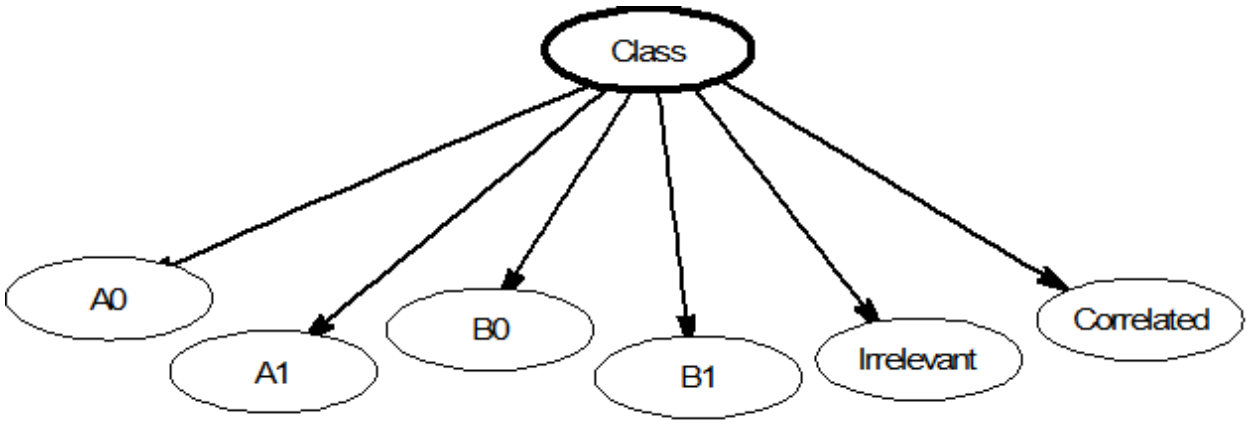



Рисунок 2.5.2 Приклад Naive Bayes алгоритму[19]

З математичної точки зору, завдання полягає у побудові моделі машинного навчання, яка, використовуючи вхідні дані може передбачати оцінку або класифікувати відповідь як правильну чи неправильну.

Моделі та методи для оптимізації розкладу занять використовують теорію розкладів для пошуку найкращого розподілу занять, викладачів та аудиторій. Генетичні алгоритми та інші методи евристичної оптимізації використовуються для пошуку оптимального рішення серед великої кількості варіантів. Теорія розкладів дозволяє враховувати різноманітні обмеження, такі як доступність викладачів та аудиторій, а також побажання студентів, що дозволяє знайти розклад, який найкращим чином задовольняє всім вимогам. Задача полягає у знаходженні такого розкладу, який мінімізує певну цільову функцію, наприклад, незадоволеність студентів.[1]

Застосування цих моделей та методів дозволить створити веб-додаток, який буде не лише зручним інструментом для обміну навчальними матеріалами та виконання завдань, але й інтелектуальною системою, здатною аналізувати дані, надавати персоналізовані рекомендації та автоматизувати деякі процеси.

**Математична постановка**

Щоб написати математичну постановку для веб-додатку, потрібно визначити основні компоненти системи, формалізувати їх у вигляді

математичних об’єктів і описати їх взаємодію. Для веб-додатку insync можна виділити такі компоненти: користувачі, організації, дошки, списки, картки.

Основні таблиці та змінні:

1. Користувачі (Users)

Нехай $U = \{u_1, u_2, \dots, u_n\}$ – множина користувачів. Кожен користувач має атрибути: ID, ім’я, роль (студент, викладач).

2. Організації (Organization)

Нехай $O = \{o_1, o_2, \dots, o_m\}$ – множина організацій. Кожна організація має атрибути: ID, назва, власник (відповідальний за організацію).

3. Дошки (Boards)

Нехай $B = \{b_1, b_2, \dots, b_k\}$ – множина дошок. Кожна дошка має атрибути: ID, назва, організація (до якої належить), дата створення.

4. Списки (List)

Нехай $L = \{l_1, l_2, \dots, l_a\}$ – множина списків. Кожний список має атрибути: ID, назва, організація, опис, дошка (до якої належить).

5. Картки (Card)

Нехай $C = \{c_1, c_2, \dots c_p\}$ – множина карток. Кожна картка має атрибути: ID, назва, опис, список (до якого належить), оцінка.

**Метою системи** є оптимізація процесу обміну навчальними матеріалами та виконання завдань, забезпечення зручного доступу до ресурсів і автоматизація оцінювання. Це можна формалізувати як задачу мінімізації часу взаємодії студентів з системою при максимальному збереженні функціональності та зручності користування.

$$min \sum_{u_i \in U} T(u_i)$$

Формула функції мети

де:

$T(u_i)$ – загальний час взаємодії користувача $u_i$ із системою.

U – множина всіх користувачів.

**Формулювання задачі**

Виходячи з вищенаведених описів, математичну постановку задачі можна сформулювати так:

1. Знайти оптимальні відповідності між користувачами, курсами, матеріалами та завданнями, що забезпечують ефективну роботу системи;
2. Мінімізувати час взаємодії користувачів із системою при виконанні основних функцій;
3. Забезпечити своєчасну відправку та оцінювання завдань відповідно до встановлених строків;
4. Максимізувати зручність та доступність навчальних матеріалів для студентів.

Ця математична постановка дає формальну основу для подальшої розробки алгоритмів та методів реалізації веб-додатку, спрямованого на обмін навчальними матеріалами та виконання завдань студентами.

## 2.6 Проектування рівнів проектних рішень

Архітектура додатку полягає в багаторівневому підході, що включає презентаційний рівень, рівень бізнес-логіки та рівень доступі до даних. Цей вибір обґрунтовується його здатністю до чіткого розділення відповідальності між компонентами системи.[4] Презентаційний рівень відповідає за відображення інтерфейсу та взаємодію з користувачем, рівень бізнес-логіки забезпечує обробку даних та реалізацію функціональності додатку, а рівень доступу до даних відповідає за зберігання та отримання даних з бази даних.

Презентаційний рівень системи здійснено з використанням React.js, що дозволяє створювати інтерактивні та динамічні інтерфейси. Використання Radix-UI забезпечує готовий набір компонентів та стилів, що прискорює розробку. Рівень бізнес-логіки реалізовано за допомогою Node.js та фреймворку TRPC. Він забезпечує обробку запитів від клієнта, валідацію даних, взаємодію з

базою даних та формування відповідей. Рівень доступу до даних реалізовано з використанням PostgreSQL, обумовлено його надійністю, масштабованістю та підтримкою складних запитів.

База даних обрана у вигляді PostgreSQL – це потужна реляційна база даних з відкритим вихідним кодом, яка підтримує широкий спектр типів даних, транзакції, тригери та інші можливості, необхідні для реалізації складного веб-додатку. База даних містить таблиці для зберігання інформації про користувачів, курси, розділи, матеріали, завдання, відповіді, оцінки тощо. Зв’язки між таблицями забезпечують цілісність даних та дозволяють ефективно отримувати необхідну інформацію.

Інтерфейс користувача побудований на основі React.js та Radix-UI. React.js використовується для створення компонентних інтерфейсів, що дозволяє розробляти складні та динамічні застосунки. Radix-UI, у свою чергу надає готовий набір елементів інтерфейсу, спрощуючи та прискорюючи розробку. Інтерфейс додатку є інтуїтивно зрозумілим та простим у використанні, з однорідним стилем оформлення та зручною навігацією, що забезпечується використанням Radix-UI. Для різних типів користувачів (викладачів, студентів) будуть передбачені різні набори функцій та можливостей.

Безпека веб-ресурсів є критичним аспектом, особливо коли йдеться про обробку особистої інформації користувачів. Використання HTTPS для шифрування трафіку, хешування паролів, валідація вхідних даних та обмеження прав доступу до даних на основі ролей користувачів є необхідними заходами для забезпечення безпеки. Ці заходи не лише запобігають несанкціонованому доступу до даних, а й забезпечують конфіденційність інформації, що зберігається в системі.

Інтеграція включає розробку API на основі RESTful архітектури. RESTful API вважається стандартом у сфері розробки веб-сервісів, оскільки він забезпечує сумісність з іншими системами та спрощує інтеграцію програмного забезпечення з іншими інформаційними системами навчального закладу.

Прийняті проектні рішення є обґрунтованими та відповідають сучасним вимогам до розробки веб-додатків. Вони дозволяють створити надійний, безпечний та зручний інструмент для обміну навчальними матеріалами та виконання індивідуальних завдань.

## 2.7 Альтернативні підходи до аналізу та проектування інформаційної системи: структурний та об’єктно-орієнтований підходи

Структурний підхід до аналізу проектування інформаційних систем (ІС) фокусується на розбитті системи на окремі функціональні модулі та визначенні взаємодії між ними. Цей підхід дозволяє чітко визначити функціональні вимоги до системи, полегшує розробку та тестування окремих модулів і забезпечує високий рівень контролю над процесом розробки. Однак він може бути складним для застосування у великих та складних системах, не завжди враховує можливі зміни вимог у майбутньому та може призвести до створення системи, яка важко модифікується то розширюється.

Об’єктно-орієнтований підхід (ООП) зосереджується на моделюванні системи у вигляді об’єктів, які мають свої властивості (атрибути) та поведінку (методи).[21] Його переваги включають можливість створення гнучких та масштабованих систем, полегшення повторного використання коду та компонентів, а також забезпечення більш природного моделювання реального світу. Однак, він може бути складним для вивчення та застосування, а також вимагає більш ретельного проектування на початкових етапах.

Для розробки веб-додатку для обміну навчальними матеріалами та виконання індивідуальних завдань доцільно використовувати об’єктно-орієнтований підхід (ООП). Це обумовлено наступними факторами. Спочатку, додаток має досить складну структуру та функціональність, що ускладнює його розробку за допомогою структурного підходу. Далі, важливою є необхідність гнучкості та масштабованості, оскільки додаток повинен бути легко розширюваним та адаптуватися до змін вимог у майбутньому, що краще

забезпечується ООП. Крім того, об’єктно-орієнтований підхід дозволяє створювати модульні компоненти, які можна повторно використовувати у різних частинах застосунку, що прискорює розробку та підвищує якість коду.

Структура класів веб-додатку, розроблена на основі об’єктно-орієнтованого підходу, має на меті ефективно моделювати різноманітні об’єкти та їх взаємозв’язки.

Починаючи з базового “User”, який відображає користувачів системи, включаються атрибути, такі як ідентифікатор, ім’я, прізвище, електронна адреса, пароль та роль (викладач чи студент). Методи цього класу дозволять користувачам реєструватися в системі, входити в неї та редагувати свої профілі.

Клас “Teacher”, що успадковує від “User”, розширюється методами для викладачів, такими як створення організації, додавання матеріалів, створення завдань та оцінювання відповідей.

Аналогічно, клас “Student”, також успадкований від “User”, володіє методами, що відображають функціональність студентів, такими як додавання до організацій, перегляд матеріалів, виконання завдань та перегляд оцінок.

Клас “Organization” відображає організації з атрибутами, такими як ідентифікатор, назва, опис, дата створення та викладач. Методи цього класу дозволяють додавати та видаляти дошки, матеріали та призначати завдання.

“Boards” та “List” відображають матеріали та завдання, відповідно з відповідними атрибутами, такими як ідентифікатор, назва, тип, вміст тощо. Методи цих класів дозволяють призначати їх студентам та оцінювати їх відповіді.

Клас “Card” відображає відповіді студентів на завдання з атрибутами, такими як, ідентифікатор, дата відправлення, вміст, оцінка тощо.

Ця базова структура класів дозволяє ефективно моделювати різноманітні об’єкти та їх взаємозв’язки в системі онлайн-навчання.

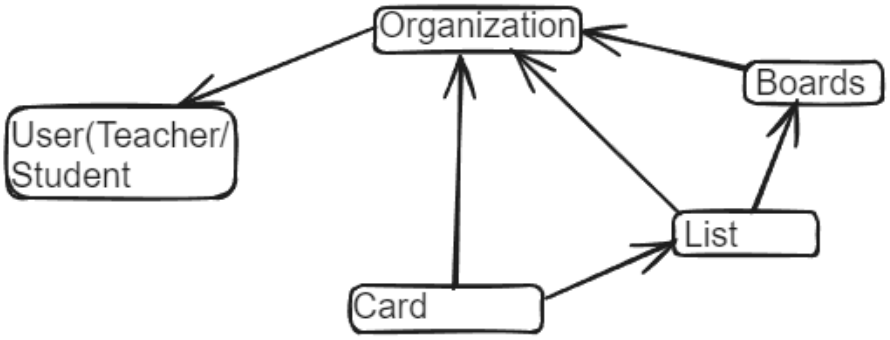



Рисунок 2.7 Схема структури класів для ООП

## 2.8 Проектування бази даних веб-додатку

### *2.8.1 Аналіз потоків даних*

Аналіз потоків даних у веб-додатку розділяється на кілька категорій: дані користувача, навчальні матеріали, дані про курси та завдання, а також дані про оцінки та зворотний зв’язок. До вхідних даних користувача відносяться дані реєстрації, вхід, редагування профілю та відповіді на завдання, а до вихідних – інформація профілю, список курсів, завдання та оцінки. Навчальні матеріали включають вхідні дані, такі як завантаження файлів та створення/редагування матеріалів, і вихідні дані – відображення матеріалів то доступ до завантажених файлів. Дані про курси та завдання містять вхідні дані про створення/редагування курсів та завдань, призначення завдань, та вихідні дані – список курсів, деталі курсу, список завдань та деталі завдання. Дані про оцінки та зворотний зв’язок включають в себе вхідні дані – оцінки викладачів та коментарі, і вихідні дані – відображення оцінок та коментарів.

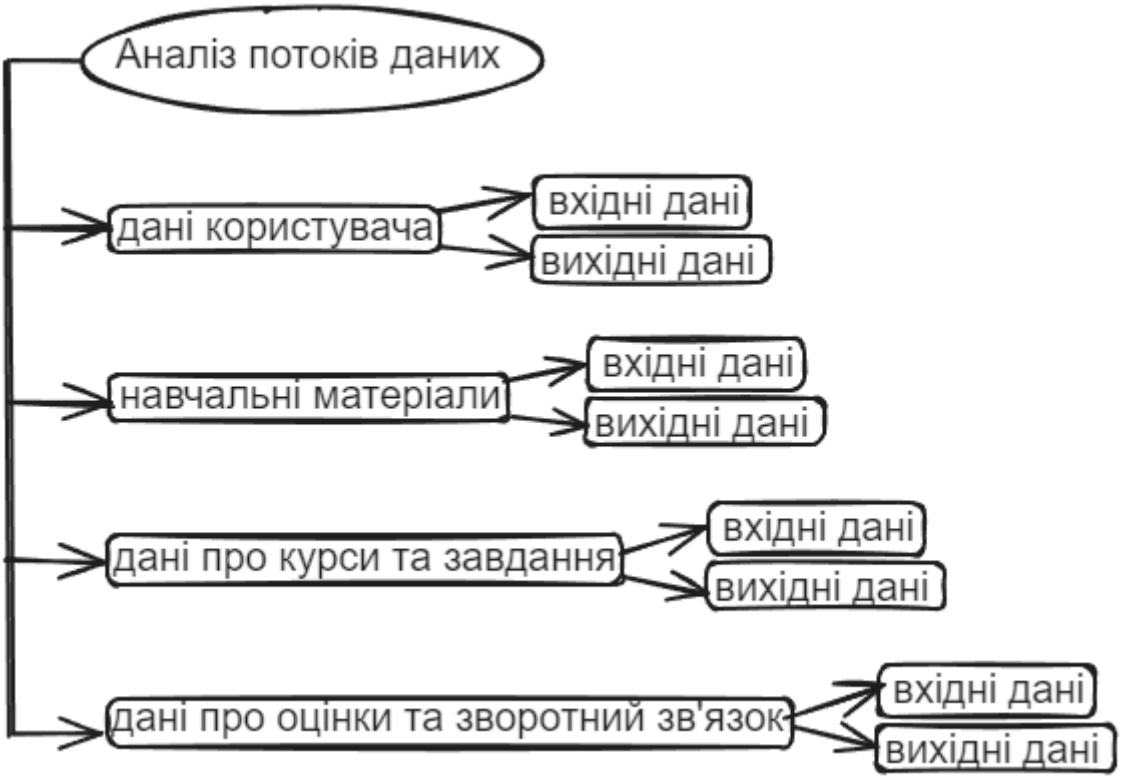



Рисунок 2.8.1 Схема аналізу потоків даних

### *2.8.2 Інфологічна (концептуальна) модель даних*

Концептуальна модель даних (КМД) є фундаментальним етапом у розробці веб-додатку для обміну навчальними матеріалами та виконання індивідуальних завдань. Вона відображає основний зміст предметної області (користувачі, курси, матеріали, завдання, відповіді) та зв’язки між ними, абстрагуючись від деталей реалізації. Як зазначають Чепмен та Маклафлін у своїй праці “Designing the Data Model” КМД є «основою для розуміння структури даних та їх взаємозв’язків, що є критично важливим для успішності розробки інформаційних систем».[6] У контексті даного проекту, КМД дозволяє визначити ключові елементи системи та їх взаємодію, що є необхідним для подальшого проектування бази даних та інтерфейсу користувача.

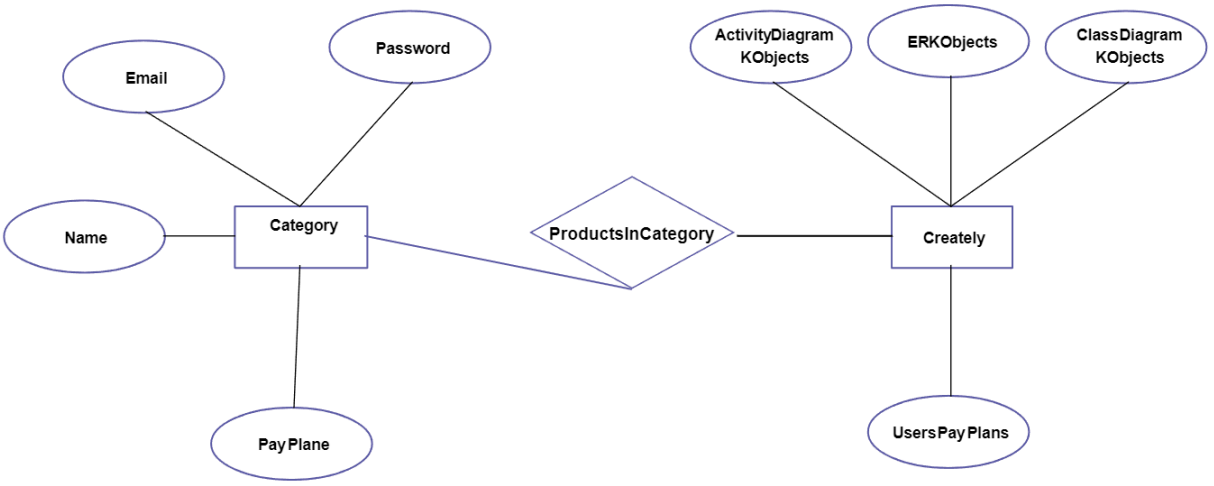



Рисунок 2.8.2 Концептуальна модель даних[7]

### *2.8.3 Логічна модель даних*

Логічна модель даних (ЛМД) є наступним етапом після концептуальної моделі та слугує основою для створення бази даних веб-додатку. ЛМД деталізує сутності, атрибути та зв'язки між ними, визначені в КМД, та перетворює їх на конкретні таблиці, стовпці та обмеження, враховуючи особливості обраної системи управління базами даних (СУБД). У своїй книзі "Database Modeling and Design: Logical Design" Теор та Лайтстоун підкреслюють важливість ЛМД, зазначаючи, що вона «дозволяє розробникам зрозуміти, як дані будуть зберігатися та оброблюватися в системі, що є ключовим для ефективної реалізації бізнес-логіки».[28] У контексті insync, ЛМД визначає структуру таблиць PostgreSQL, їх атрибути та типи даних, а також обмеження цілісності, що забезпечують коректність та узгодження даних. Це дозволяє перейти до фізичної реалізації бази даних та подальшої розробки додатку.

Зв'язки між таблицями:

1. Картки (Card) – Списки (List)

Один список може мати декілька карток (зв'язки через listId);

2. Списки (Card) – Дошки (Board)

Одна дошка може мати декілька списків (зв'язок через boardId);

3. Дошки (Board) – Організації (Organization)

Одна організація може мати декілька дошок (зв’язок через organizationId);

4. Організації (Organization) – Користувач і(User)

Один користувач може мати декілька організацій (зв’язок через ownerId);

5. Коментарі карток (CardComment) – Картки (Card)

Одна картка може мати декілька коментарів (зв'язок через cardId);

6. Коментарі карток (CardComment) – Користувачі (User)

Один користувач може мати декілька коментарів карток (зв'язок через authorId).

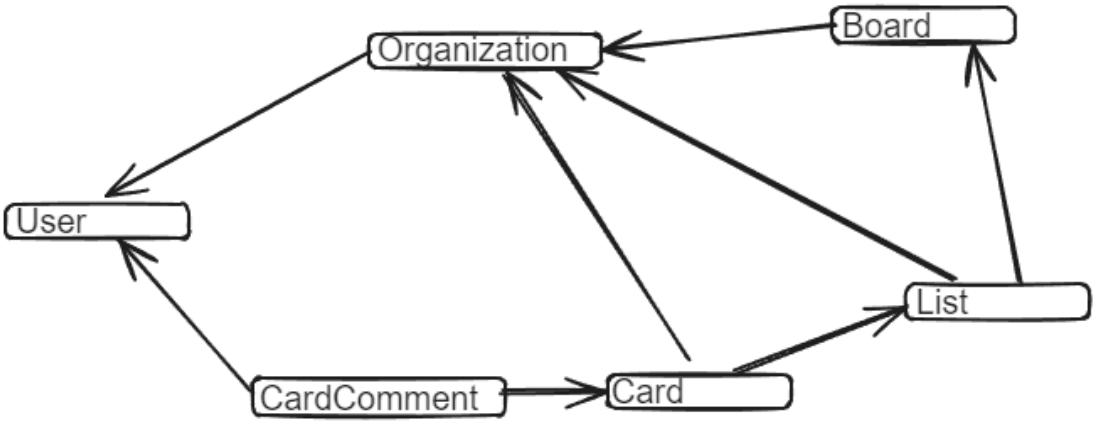



Рисунок 2.8.3 Логічна модель даних

### *2.8.4 Датологічна модель даних*

Датологічна модель даних (ДМД) у зв’язках веб-додатку для обміну навчальними матеріалами та виконання індивідуальних завдань визначає внутрішню структуру, формати та типи даних, необхідні для ефективного функціонування системи. Вона базується на логічній моделі даних, але враховує конкретні особливості реалізації бази даних та вимог до зберігання та обробки інформації. Як зазначає Дейт у свої книзі “An Introduction to Database System”, ДМД «описує, як дані фізично зберігаються в базі даних, включаючи типи даних, довжину полів, обмеження цілісності та індекси».[8] Для веб-платформи ДМД визначає типи даних для кожного атрибута в таблицях PostgreSQL,

забезпечуючи оптимальне використання ресурсів та ефективність запитів. До того ж, датологічна модель даних враховує обмеження цілісності, що гарантує коректність та узгодженість даних у системі.

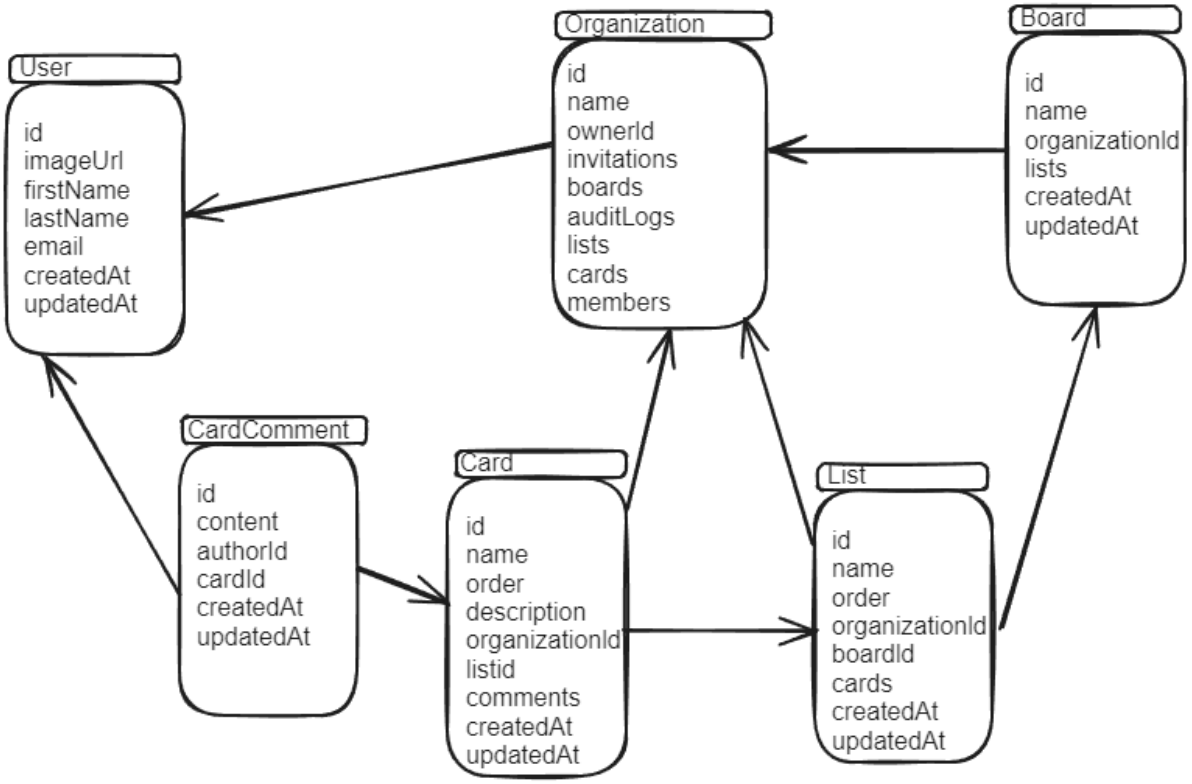



Рисунок 2.8.4 Датологічна модель даних

### *2.8.5 Обмеження та правила цілісності*

- Первинні ключі: кожна таблиця має первинний ключ унікальної ідентифікації записів;
- Зовнішні ключі: забезпечують зв’язки між таблицями та цілісність даних;
- Унікальність: деякі поля (наприклад, електронна адреса користувача) повинні бути унікальними.

Правила коригування в базі даних, такі як CASCADE та SET NULL, виконують критичну роль у забезпеченні цілісності та стабільності даних. Завдяки CASCADE, коли видаляється користувач, курс або розділ, всі пов’язані з ними записи автоматично видаляються з інших таблиць, що запобігає

виникненню суперечностей або залишків даних. SET NULL забезпечує аналогічний механізм, проте встановлює значення зовнішніх ключів у NULL, замість видалення пов’язаних записів.

Особливості адміністрування з використанням PostgreSQL дозволяють забезпечити надійність, продуктивність та безпеку бази даних. Механізми резервного копіювання та відновлення дозволяють швидко відновити базу даних у разі потреби, зменшуючи ризик втрати інформації в результаті збоїв або дефектів. Моніторинг та оптимізація продуктивності бази даних дозволяють ідентифікувати та вирішувати проблеми з продуктивністю, забезпечуючи стабільну та ефективну роботу системи. Засоби безпеки, такі як механізми аутентифікації, авторизації та шифрування даних, гарантують захищеність конфіденційної інформації та запобігають несанкціонованому доступу до бази даних.

Загальновизнані моделі даних, реалізовані з використанням PostgreSQL, становлять основу надійного та ефективного зберігання та обробки даних у веб-додатках, забезпечуючи їхню цілісність та доступність для користувачів.

## 2.9 Аналіз нормативно-довідкової інформації, розробка форм документів

При розробці веб-додатку для обміну навчальними матеріалами та виконання індивідуальних завдань важливо врахувати ряд нормативно-правових актів та стандартів. Зокрема, законодавство України, такі як «Про освіту», який визначає основні принципи освітньої діяльності та права учасників освітнього процесу, «Про захист персональних даних», що регулює обробку особистих даних, і «Про авторське право і суміжні права», що забезпечує захист авторських прав на навчальні матеріали. Крім того, важливо враховувати стандарти веб-доступності, такі як WCAG, які встановлюють вимоги до доступності веб-контенту для людей з обмеженими можливостями.

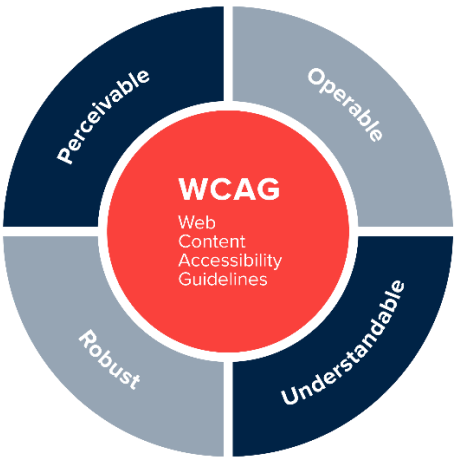



Рисунок 2.9 Приклад настанов веб-змісту WCAG[11]

У контексті веб-програми, яка спеціалізується на навчальних потребах, розглядається важливий аспект створення та обробки різноманітних документів. Серед них можна виділити наступні форми:

**Профіль користувача:** ця форма включає в себе поля для введення та редагування особистих даних, таких як повне ім’я, адреса електронної пошти, номер телефону тощо. Більше того, можуть бути передбачені можливості завантаження фотографії користувача або інших персоналізованих елементів.

**Форма створення організації:** для створення нової організації необхідно ввести її назву, опис, визначити програму, а також обрати викладача. Ця форма надає можливість детального опису організації та її структури для подальшого використання та аналізу.

**Форма додавання дошки:** для розширення навчального контенту веб-додаток має дошки, які дозволяють завантажувати різноманітні матеріали, такі як тести, зображення, аудіо або відео, а також вказувати їх назву та короткий опис.

**Форма створення завдання:** для організації навчальних завдань система пропонує форму, яка включає в себе поля для введення назви завдання, його опису, інструкцій, критеріїв оцінювання та встановлення терміну виконання.

**Форма виконання завдання:** користувачі мають можливість введення своїх відповідей на завдання, що може включати в себе текстові відповіді, завантаження файлів або виконання певних дій прямо на платформі.

**Форма оцінювання завдання:** для оцінювання виконаних завдань викладачі мають доступ до спеціальної форми, де вони можуть виставляти оцінки та писати коментарі робіт студентів, щоб надати зворотний зв’язок та сприяти їх подальшому розвитку.

Ці форми дозволяють створювати, організовувати та взаємодіяти з навчальним контентом, забезпечуючи ефективний процес навчання та викладання.

Для ефективної організації та пошуку навчальних матеріалів та завдань у веб-додатку доцільно розробити систему класифікації та кодування. Класифікація курсів може бути здійснена за навчальними дисциплінами, рівнем складності та цільовою аудиторією. Матеріали можуть бути класифіковані за типом (текст, зображення, аудіо, відео), тематикою та автором. Завдання, у свою чергу, можуть бути класифіковані за типом (тест, доповідь, завантаження файлу), рівнем складності та тематикою. Впровадження такої системи класифікації та кодування дозволить забезпечити зручну навігацію по додатку, швидкий пошук потрібних матеріалів та завдань, а також полегшить аналіз даних про використання платформи.

## 2.10 Результати аналізу та синтезу проектних рішень у вигляді стандартних схем та діаграм

**Діаграма прецедентів** – ця діаграма відображає основні функціональні можливості системи та взаємодію користувачів (викладач, студент) з системою.

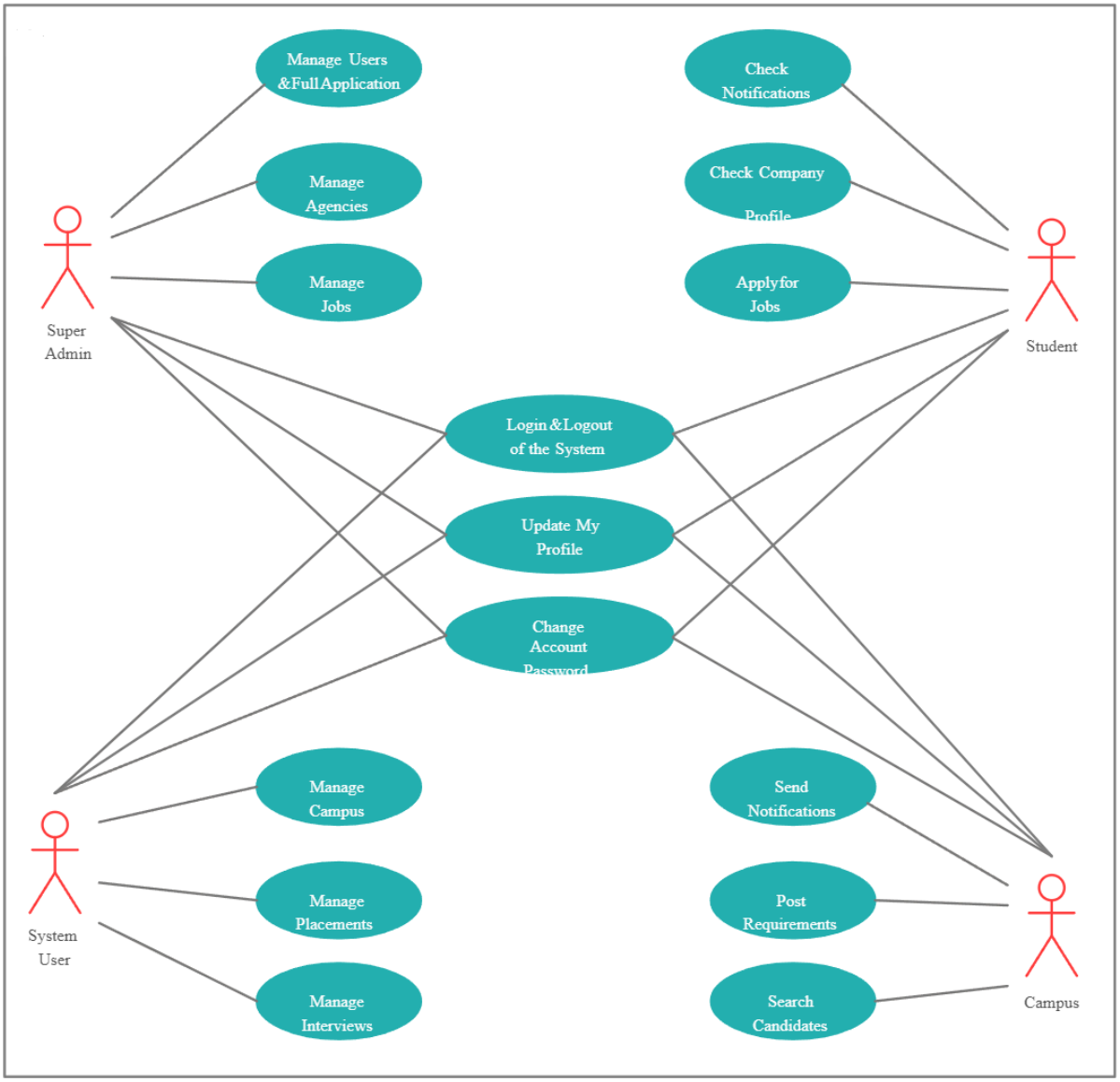



Рисунок 2.10.1 Діаграма прецедентів[25]

**Діаграма послідовності** – ця діаграма ілюструє послідовність взаємодії між об’єктами системи при виконанні конкретного сценарію (наприклад, реєстрації користувача або входу в систему).

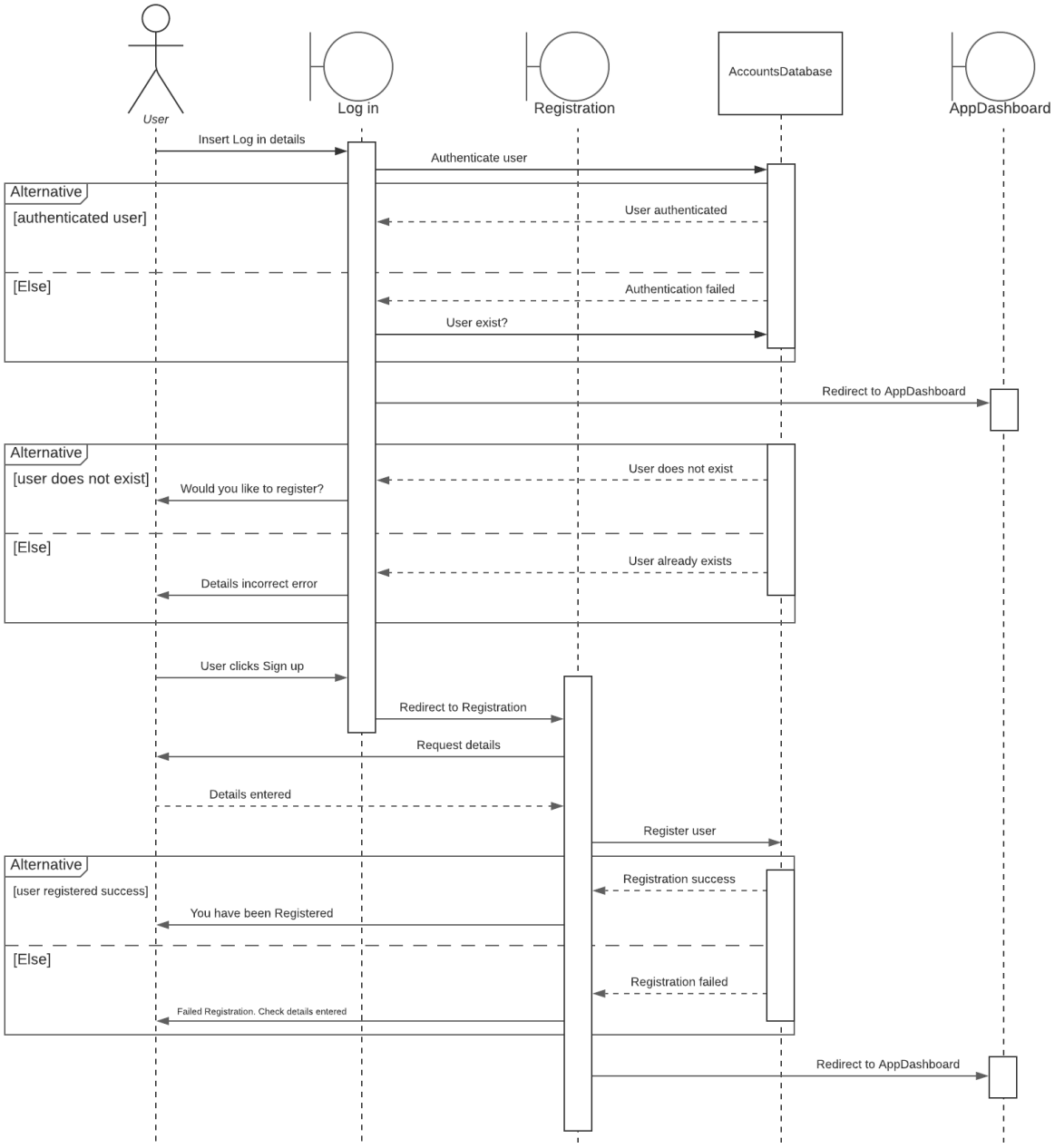



Рисунок 2.10.2 Діаграма послідовності[24]

**Діаграма компонентів** – ця діаграма показує, як система розбита на окремі компоненти та як вони взаємодіють між собою.

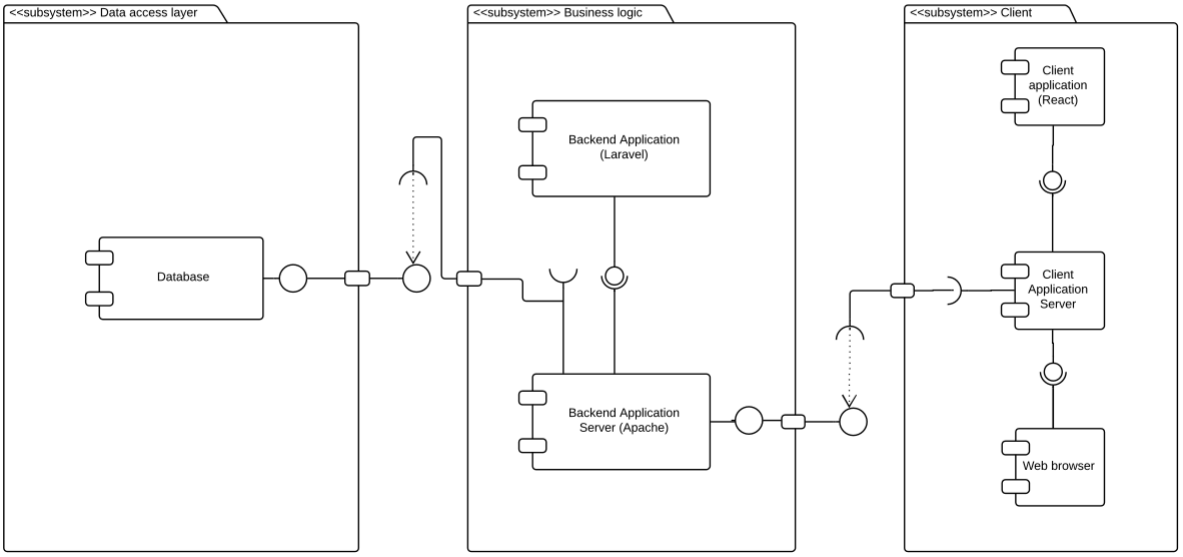



Рисунок 2.10.3 Діаграма компонентів[30]

**Діаграма розгортання** – ця діаграма показує фізичну архітектуру системи, тобто як компоненти додатку розгорнуті на серверах та як вони взаємодіють з мережею та клієнтськими пристроями.

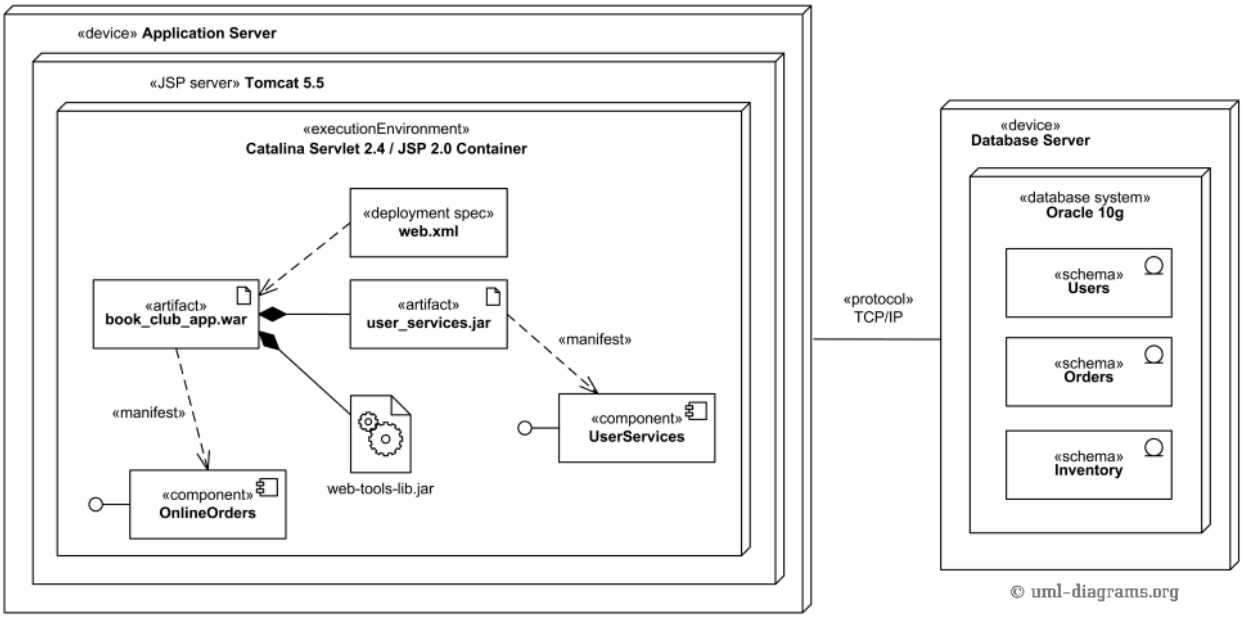



Рисунок 2.10.4 Діаграма розгортання[32]

**Діаграма відношень** – ця діаграма відображає структуру бази даних, сутності (таблиці), їх атрибути (поля) та зв’язки між ними.

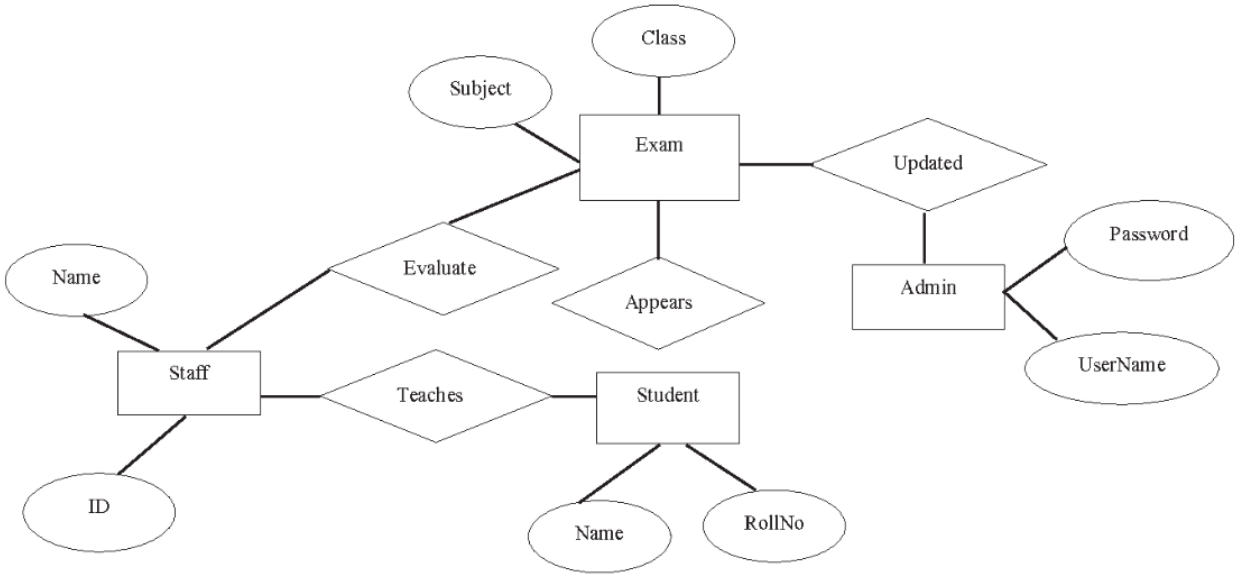



Рисунок 2.10.5 Діаграма відношень[23]

Використання цих діаграм та схем дозволяє візуалізувати структуру та функціональність системи, що дозволяє краще зрозуміти, як працює додаток та як його компоненти взаємодіють між собою. Не менш важливою частиною є їхня роль у забезпеченні документації проекту, що може бути використана для подальшого розвитку та підтримки додатку.

## 2.11 Розробка зовнішнього інтерфейсу веб-додатку

Зовнішній інтерфейс веб-додатку – це сукупність екранів, елементів управління та візуальних компонентів, з якими безпосередньо взаємодіє користувач. Його дизайн має бути інтуїтивно зрозумілим, та зручним та відповідати принципам універсального дизайну, щоб забезпечити доступність для всіх категорій користувачів, включаючи людей з обмеженими можливостями.

1. Головна сторінка веб-додатку включає в себе:

- Створення нових організацій;
- Створення/редагування дошок;
- Список членів організації.

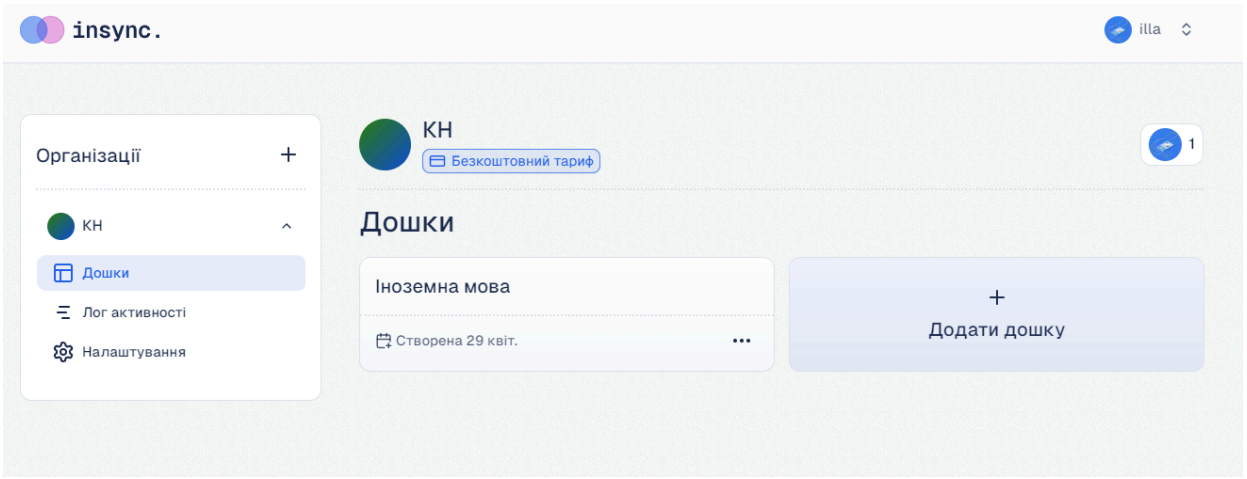



Рисунок 2.11.1 Головне меню веб-додатку

2. Приклад завдання в дошці «ООП»:

- Додавання/редагування списків, карток;
- Перегляд завдань;
- Завантаження робіт.

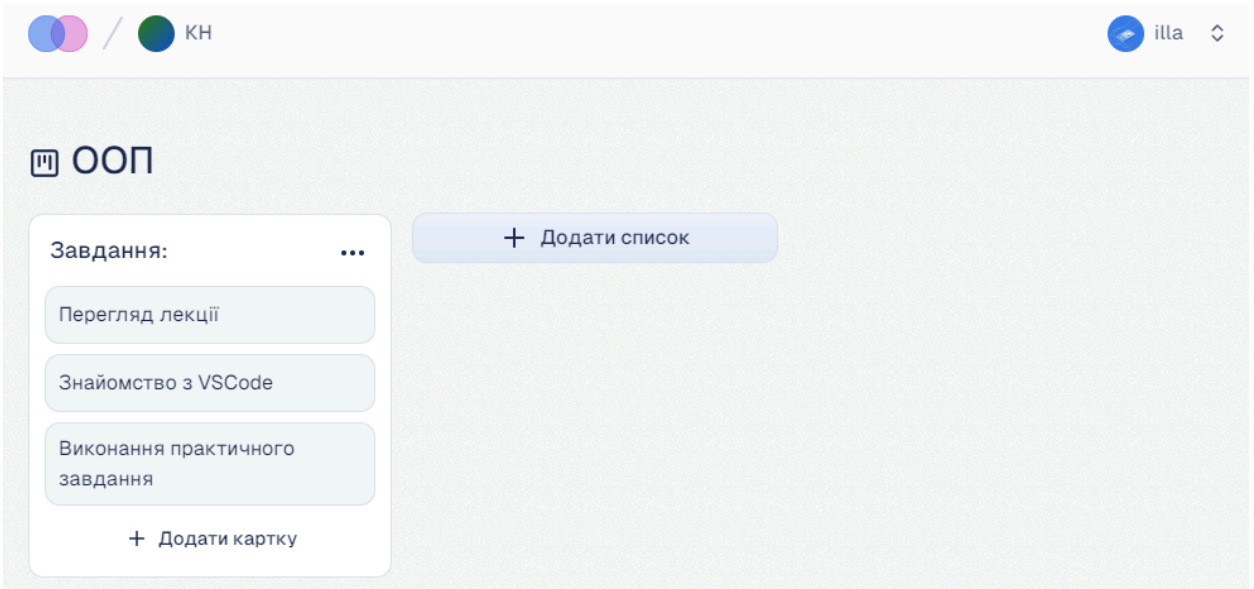



Рисунок 2.11.2 Дошка «ООП»

# 3. ПРОГРАМНА РЕАЛІЗАЦІЯ

## 3.1 Обґрунтування вибору програмних засобів та визначення архітектури системи

### *3.1.1 Обґрунтування вибору програмних засобів буде представлено у виді таблиці*

Таблиця 3.1

Порівняльні характеристики існуючих технологій

| **Технологія** | **Обґрунтування** |
|---|---|
| **Frontend:** | |
| React.js | Компонентний підхід, віртуальний DOM, велика спільнота та екосистема, висока продуктивність, підтримка TypeScript, популярність та широке використання в індустрії |
| Radix-UI | Готовий набір компонентів, що включає WAI-ARIA від W3C, швидкість розробки, гарний зовнішній вигляд, адаптивний дизайн, підтримка тем та кастомізації, інтеграція з React.js та іншими бібліотеками |
| **Backend:** | |
| Node.js | Висока продуктивність завдяки одночасному виконанню коду, велика спільнота та екосистема, можливість використання TypeScript як для frontend, так і для backend, що |

| | |
|---|---|
| | спрощує розробку та підтримку, широке використання в індустрії |

Продовження таблиці 3.1

| | |
|---|---|
| TRPC | Мінімалістичний фреймворк, що надає необхідний функціонал для створення веб-серверів та API, гнучкість та можливість розширення, легкість вивчення та використання |
| **База даних:** | |
| PostgreSQL | Відкритий вихідний код, висока надійність та масштабованість, підтримка ACID транзакцій, широкий спектр типів даних (включаючи JSON), потужна мова запитів SQL, підтримка індексів, тригерів, можливість розгортання на різних платформах (Linux, Windows, macOS) |

### *3.1.2 Визначення технології розробки*

Для даного проекту буде використана **спіральна модель** розробки. Обґрунтування спіральної моделі полягає у кількох ключових аспектах. По-перше, її ітеративний характер сприяє постійному уточненню вимог, проектуванню, розробці та тестуванню протягом кожної ітерації. Це забезпечує можливість отримання робочих версій продукту на ранніх етапах розробки та вносить зміни у сірі необхідності, що дозволяє уникнути непорозумінь та забезпечує відповідність вимогам замовника. По-друге, спіральна модель

передбачає поетапний аналіз та управління ризиками на кожній ітерації, що дозволяє мінімізувати їх вплив на проект та забезпечує вчасне виявлення та вирішення потенційних проблем. Крім того, гнучкість цієї моделі дозволяє адаптуватися до змін вимог та умов проекту, забезпечуючи ефективну реакцію на зміни в процесі розробки. Такий поетапний підхід до уточнення вимог, архітектури та дизайну дозволяє уникнути помилок та недоліків на ранніх етапах розробки та забезпечує якісний результат у кінцевому продукті.[26]

Архітектура системи побудована на основі багаторівневої архітектури, яка включає наступні рівні: презентаційний рівень (Frontend), що відповідає за відображення інтерфейсу користувача та взаємодію з ним; рівень бізнес-логіки (Backend), який обробляє запити від клієнта, виконує бізнес-логіку та взаємодіє з базою даних; та рівень доступу до даних (Database), що відповідає за зберігання та отримання даних. Такий підхід дозволяє чітко розділити відповідальність між компонентами системи, забезпечити її масштабованість та гнучкість, а також спростити розробку та підтримку.

## 3.2 Архітектура та модульна структура веб-додатку

### *3.2.1 Загальна архітектура (логічна структура)*

Веб-додаток буде побудований на основі багаторівневої архітектури, яка складається з наступних рівнів:

- На клієнтському рівні (Frontend) забезпечує відображення інтерфейсу користувача та його взаємодія. Використовується React.js та Radix-UI, дозволяючи користувачам взаємодіяти через веб-браузер, надсилаючи запити на сервер та отримуючи HTML, CSS та TypeScript код у відповідь;
- На рівні бізнес-логіки (Backend) відбувається обробка клієнтських

запитів, виконання бізнес-логіки, взаємодія з базою даних та формування відповідей. Для цього використовуються Node.js та фреймворк TRPC. Тут також можуть включатися різні сервіси, такі як аутентифікація, авторизація, обробка платежів тощо;

- На рівні доступу до даних (Database) забезпечується зберігання та отримання даних за допомогою реляційної бази даних PostgreSQL.

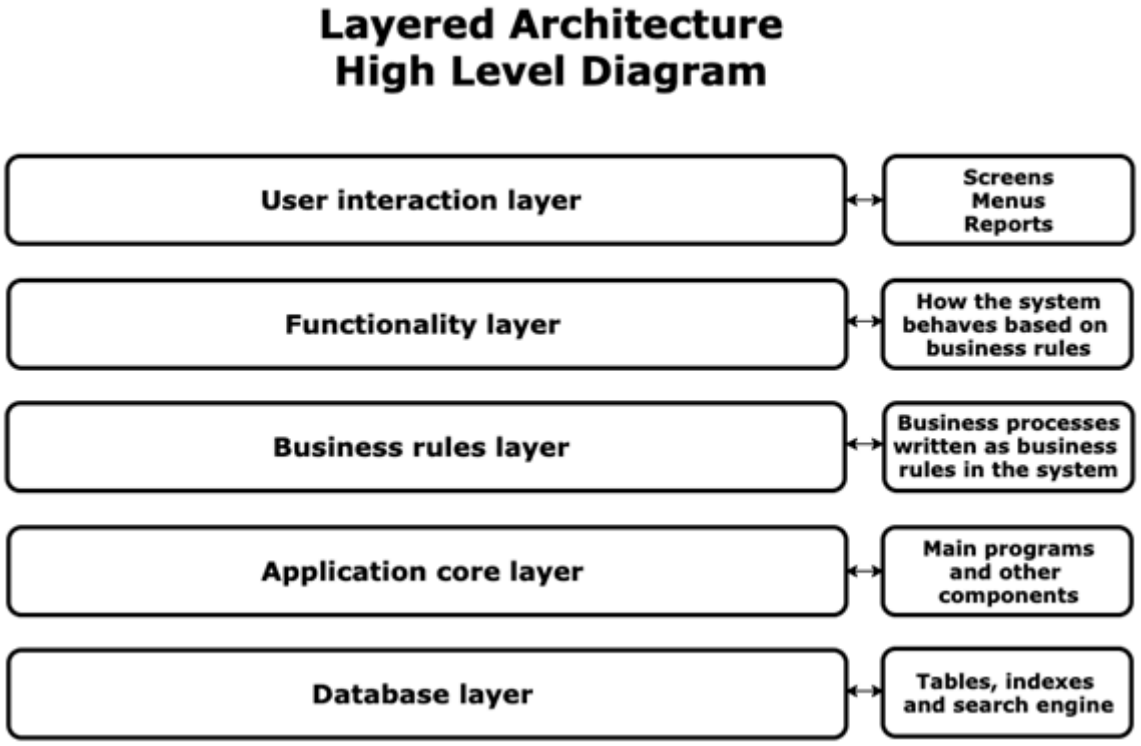



Рисунок 3.2.1 Логічна структура[17]

### *3.2.2 Модульна структура (фізична структура)*

Додаток розбивається на окремі модулі, кожен з яких відповідає за певну функціональність:

- Модуль автоматизації та аутентифікації (Модуль 1) реалізує

функціональність реєстрації, входу в систему, відновлення паролю та управління сесіями користувачів;

- Модуль управління курсами (Модуль 2) дозволяє створювати, редагувати та видаляти курси, додавати до них розділі та матеріали;
- Модуль управління завданнями (Модуль 3) дозволяє створювати, редагувати та видаляти завдання, призначати їх студентам, встановлювати терміни виконання та оцінювати відповіді;
- Модуль виконання завдань (Модуль 4) надає студентам інтерфейс для перегляду та виконання завдань, завантаження файлів та отримання зворотного зв’язку від викладача;
- Модуль комунікації (Модуль 5) забезпечує можливість спілкування між викладачами та студентами через форуми, чати та відеоконференції;
- Модуль аналітики (Модуль 6) збирає та аналізує дані про активність користувачів, успішність виконання завдань, використання матеріалів та формує звіти для викладачів та адміністраторів.

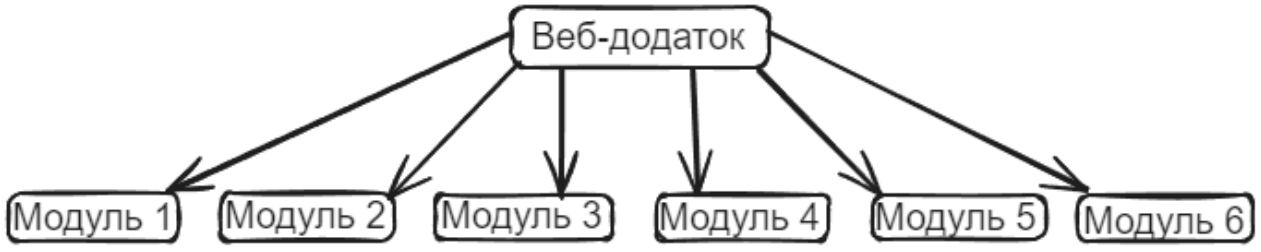



Рисунок 3.2.2 Модульна структура

Кожен модуль може бути розроблений та протестований окремо, що спрощує розробку та підтримку веб-додатка. Модулі взаємодіють між собою через API, що забезпечує їх незалежність та гнучкість.

Переваги такої структури включають модульність, гнучкість, масштабованість і можливість повторного використання. Кожен модуль має чутко визначену відповідальність, що полегшує розробку, тестування та підтримку. Модулі можна легко змінювати або замінювати, не впливаючи на роботу інших частин системи. Систему можна легко розширювати, додаючи нові модулі або збільшуючи потужність існуючих. Це також сприяє повторному використанню модулів у інших проектах, що економить час та витрати на розробку. Вибрана архітектура та модульна структура дозволять створити масштабований, гнучкий та легко підтримуваний веб-додаток, який буде відповідати вимогам сучасної освіти.

### 3.3 Структура та функціонування веб-додатку

Програмна система складається з трьох основних рівнів:

- Frontend (клієнтський рівень), який реалізований за допомогою

React.js та Radix-UI, відповідає за відображення інтерфейсу користувача, обробку дій користувача та відправлення запитів до сервера;

- Backend (серверний рівень), що реалізований за допомогою Node.js та

TPRC, відповідає за обробку запитів від клієнта, виконання бізнес-логіки, взаємодію з базою даних та відправлення відповідей клієнту;

- Бази даних (PostgreSQL), яка є реляційною базою даних і зберігає всю

інформацію про користувачів, курси, матеріали, завдання, відповіді та інше.

Процес функціонування безперервний: користувач взаємодіє з інтерфейсом додатку у веб-браузері, спонукаючи браузер відправити запит на сервер. Сервер, що працює на Node.js + TRPC, приймає цей запит та визначає відповідний контролер для обробки. Після отримання даних з бази даних

PostgreSQL, контролер виконує необхідну бізнес-логіку та формує відповідь у форматі JSON. Ця відповідь надсилається клієнту, який працює на React.js і відображається на інтерфейсі користувача, оновлюючи його згідно отриманих даних.

**Приклад сценарію**

Розглянемо сценарій перегляду студентом матеріалів курсу:

1. **Студент відкриває сторінку курсу** – браузер відправляє запит на сервер з ідентифікатором курсу;
2. **Сервер отримує запит** – контролер курсів отримує запит, витягує дані про курс з бази даних (назва, опис, розділи, матеріали) та формує відповідь у форматі JSON;
3. **Frontend отримує відповідь** – компонент сторінки курсу отримує дані про курс та відображає їх на екрані (назва опис, список розділів);
4. **Студент відкриває розділ** – браузер відправляє запит на сервер з ідентифікатором розділу;
5. **Сервер отримує запит** – контролер розділів отримує запит, витягує дані про розділ з бази даних (назва, опис, матеріали) та формує відповідь у форматі JSON;
6. **Frontend отримує відповідь** – компонент сторінки розділу отримує дані про розділ та відображає їх на екрані (назва, опис, список матеріалів);
7. **Студент відкриває матеріал** – браузер відправляє запит на сервер

з ідентифікатором матеріалу;

8. **Сервер отримує запит** – контролер матеріалів отримує запит, витягує дані про матеріал з бази даних (назва, тип, вміст) та формує відповідь у форматі JSON;

9. **Frontend отримує відповідь** – компонент перегляду матеріалу отримує дані та відображає їх на екрані (назва, вміст).

Таким чином, архітектура програмної системи забезпечує чіткий поділ відповідальності між frontend та backend, що спрощує розробку та підтримку додатку. Використання PostgreSQL як бази даних забезпечує надійне та ефективне зберігання даних. Процес функціонування додатку базується на обміні даними між клієнтом та сервером у форматі JSON, що дозволяє створювати динамічні та інтерактивні веб-сторінки.

**Структурна схема програмної системи**

Структурна схема веб-додатку insync складається з Frontend (React.js, Radix-UI), Backend (Node.js, TRPC) та бази даних (PostgreSQL), що взаємодіють між собою для забезпечення функціональності платформи. Користувачі взаємодіють із системою через веб-браузер, який надсилає запити до сервера, а той, у свою чергу, обробляє їх та надає відповідь.

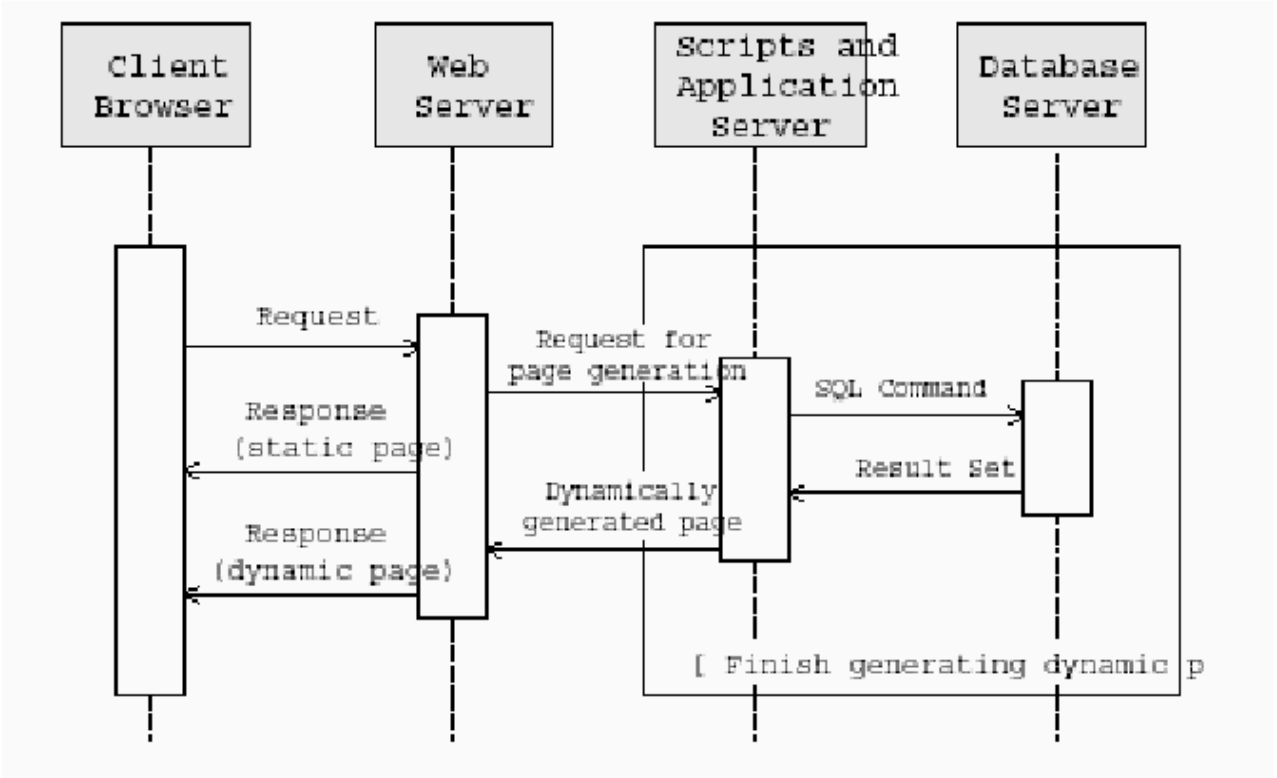



Рисунок 3.3 Структурна схема веб-додатку[33]

**Опис компонентів:**

- **Веб-браузер:** клієнтська програма, яка використовується для відображення веб-сторінок та взаємодії з користувачем. Відправляє запити на сервер та отримує від нього відповіді;
- **Веб-сервер:** програмне забезпечення, що приймає запити від веб-браузерів та відправляє їм у відповідь. У нашому випадку веб-сервером буде Node.js з використанням фреймворку TRPC;
- **Маршрутизатор:** компонент веб-сервера, який відповідає за визначення, який контролер має обробляти конкретний запит від клієнта;
- **Контролери:** компоненти, що обробляють запити від клієнта,

отримують дані з моделі, виконують необхідну бізнес-логіку та формують відповідь для клієнта. Наприклад, контролер курсів відповідає за обробку запитів, пов’язаних зі створенням, редагуванням та переглядом курсів;

- **Модель:** компонент, що представляє дані додатку ті логіку роботи з ними. Взаємодіє з базою даних для отримання та збереження даних;
- **База даних:** PostgreSQL це реляційна база даних, що зберігає всю інформацію про користувачів, курси, матеріали, завдання, відповіді тощо.

**Інформаційні зв’язки:**

- **Веб-браузер <–> Веб-сервер:** обмін даними відбувається за протоколом HTTP (або HTTPS для шифрування). Браузер відправляє запити (GET, POST, PUT, DELETE) та отримує відповіді у вигляді HTML, CSS та TypeScript;
- **Веб-сервер <–> Маршрутизатор:** веб-сервер передає отриманні запити маршрутизатору для визначення відповідного контролера;
- **Маршрутизатор <–> Контролери:** маршрутизатор викликає відповідний контролер для обробки запиту;
- **Контролери <–> Модель:** контролери взаємодіють з моделлю для отримання або збереження даних у базі;
- **Модель <–> База даних:** модель виконує запити до бази даних (SELECT, INSERT, UPDATE, DELETE) для отримання або збереження даних.

Дата структурна схема відображає загальну логіку роботи веб-додатку та взаємодію його основних компонентів. Це дозволяє краще зрозуміти принципи функціювання системи та спрощує процес її розробки та підтримки.

## 3.4 Структурна схема програмної системи (веб-додатку) з описом призначення елементів та інформаційних зв'язків

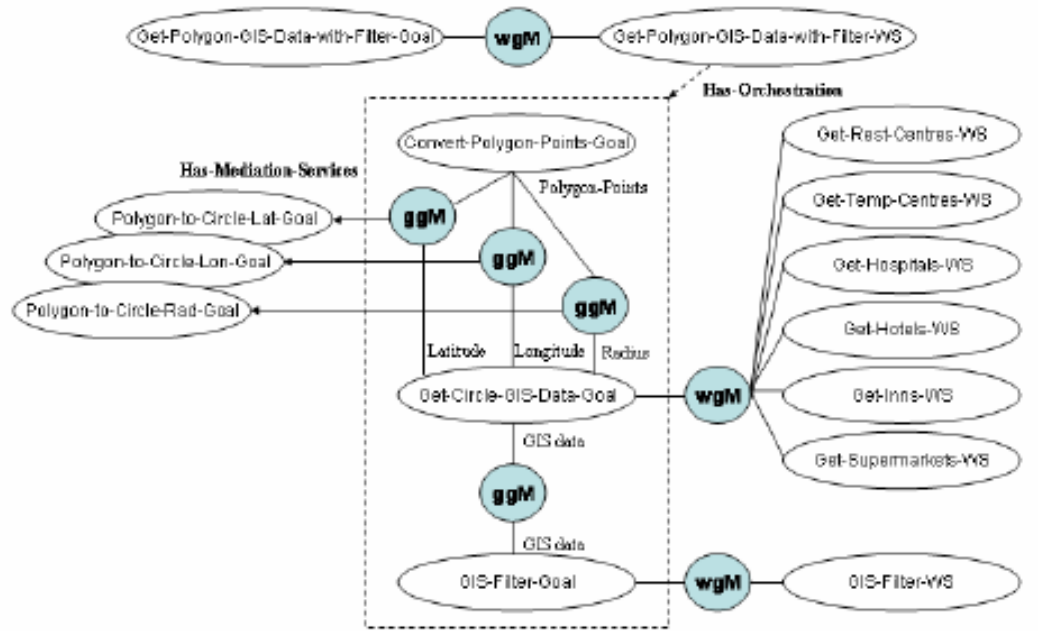



Рисунок 3.4 Структурна схема програмної системи[22]

- **Користувач:** призначення – викладач або студент, який взаємодіє з

веб-додатком. Взаємодія – користувач відправляє запити до веб-сервера через веб-браузер, отримує відповіді від сервера у вигляді веб-сторінок, виконує дії у відповідності до інтерфейсу застосунку;

- **Веб-браузер:** призначення – програмне забезпечення, що

відображає веб-сторінки та забезпечує взаємодію користувача з веб-додатком. Взаємодія – приймає вхідні дані від користувача, відправляє запити на веб-сервер, отримує відповіді від сервера у вигляді HTML, CSS та TypeScript коду, відображає отриманий контент у вигляді веб-сторінок, виконує TypeScript код для забезпечення інтерактивності;

- **Веб-сервер:** призначення – програмне забезпечення, що приймає

запити від веб-браузерів, обробляє їх та відправляє відповіді. Взаємодія – отримує запити від веб-браузера, передає їх маршрутизатору, той отримує дані від контролерів та відправляє відповіді веб-браузеру;

- **Маршрутизатор:** призначення – компонент веб-сервера, який

визначає контролер має обробляти конкретний запит від клієнта. Взаємодія – отримує запити від веб-сервера, аналізує URL запиту та HTTP метод, викликає відповідний контролер, передає йому параметри запиту;

- **Контролери:** призначення – обробляють запити від клієнта,

отримують дані з моделі, виконують бізнес-логіку та формують відповідь від клієнта. Взаємодія – отримують дані від маршрутизатора, взаємодіють з моделлю для отримання або збереження даних у базі даних, формують відповідь у вигляді JSON або HTML та передають її веб-серверу;

- **Модель:** призначення – представляє дані додатку та логіку роботи з

ними. Взаємодія – отримує запити від контролерів на отримання або збереження даних, виконує запити до бази даних, повертає результати контролерам;

- **База даних:** призначення – зберігає дані додатку (інформацію про

користувачів, курси, матеріали, завдання, відповіді тощо). Взаємодія – отримує запити від моделі на виконання операцій з даними (SELECT, INSERT, UPDATE, DELETE) та повертає результати запитів.

**Інформаційні зв’язки:** всі компоненти системи взаємодіють між собою за допомогою передачі даних. Основними типами даних, що передаються між компонентами, є:

- **Запити від клієнта:** URL, HTTP метод, параметри запиту, дані форм;

- **Відповіді сервера:** HTML, CSS, TypeScript код , JSON дані;
- **Дані між контролерами та моделлю:** параметри запитів, результати запитів до бази даних, об’єкти, що представляють дані додатку;
- **Запити до бази даних:** SQL запити;
- **Результати запитів до бази даних:** набори даних.

Дана структурна схема з описом призначення елементів та інформаційних зв’язків надає повне уявлення про роботу веб-додатку та дозволяє ефективно реалізувати його функціональність.

### 3.5 Розробка програмних компонентів

**Загальна інформація про програму:**

- **Назва:** insync (робоча назва);
- **Призначення:** веб-додаток для обміну навчальними матеріалами та виконання індивідуальних завдань студентами;
- **Застосування:** вищі навчальні заклади, школи, корпоративне навчання, самоосвіта;
- **Мова програмування:** TypeScript (frontend – React.js, Radix-UI; backend – Node.js, TRPC).
- **База даних:** PostgreSQL.

Функціональне призначення програми полягає у забезпеченні всіх необхідних інструментів для якісної навчальної взаємодії. Для викладачів це включає можливість створення та управління курсами, додавання та редагування навчальних матеріалів, створення та призначення завдань, а також проведення перевірки завдань та надання зворотного зв’язку. Крім того, вони можуть спілкуватися зі студентами та проводити аналіз успішності останніх. Для студентів це надає можливість перегляду навчальних матеріалів, виконання

завдань, отримання оцінок та зворотного зв’язку, спілкування з викладачами та іншими студентами, зокрема відстеження свого прогресу.

Обмеження на технічні засоби включають в себе вимоги до клієнтської та серверної частини системи. Для клієнтської частини потрібний сучасний веб-браузер, такий як Chrome, Firefox, Safari або Edge. Щодо серверної частини, необхідно мати Node.js версії 18 та базу даних PostgreSQL версії 12 або вище.

Опис інформації включає вхідні дані, такі як дані користувача (ім’я, електронна адреса, пароль), навчальні матеріали (текст, зображення, аудіо, відео) і дані про курси та завдання (назва, опис, терміни оцінки), а також вихідні дані, які включають веб-сторінки з навчальними матеріалами, завданнями та результатами виконання, звіти про успішність студентів.

**Опис логіки програми**

**Графічно:**


Рисунок 3.5 Опис логіки програми

**Текстово:**

- Користувач відправляє запит до сервера;
- Сервер обробляє запит, отримує дані з бази даних та формує відповідь;
- Відповідь відправляється клієнту, який відображає її у браузері.

Особливості встановлення включають клієнтську частину, яка збирається за допомогою Turbopack та розгортається на веб-сервері, серверну частину, що розгортається на Node.js сервері, та базу даних, яка встановлюється та налаштовується на PostgreSQL. Додатково, проект буде розроблятися з використанням Agile та Scrum, код буде покритий юніт-тестами та інтеграційними тестами, слід зазначити, що додаток буде регулярно оновлюватися та підтримуватися. Цей детальний опис програмних компонентів надає повну інформацію про розроблюваний веб-додаток, його функціональність, технічні вимоги та процес розробки.

### 3.6 Опис текстових випадків та методи тестування веб-додатку

Текстові випадки описують конкретні сценарії використання додатку та очікувані результати. Вони допомагають перевірити коректність роботи всіх функцій та виявити можливі помилки.

Таблиця 3.2

Авторизація та аутентифікація

| № | Текстовий випадок | Вхідні дані | Очікуваний результат |
|---|---|---|---|
| 1 | Реєстрація нового користувача (студента) | Валідні дані | Успішна реєстрація, створення облікового запису студента, перехід на сторінку входу |

Продовження таблиці 3.2

| | | | |
|---|---|---|---|
| 2 | Реєстрація нового користувача (викладача) | Валідні дані | Успішна реєстрація, створення облікового запису викладача, перехід на сторінку входу |

| | | | |
|---|---|---|---|
| 3 | Реєстрація з невалідними даними (наприклад, порожнє поле) | Невалідні дані | Повідомлення про помилку, підказка щодо виправлення |
| 4 | Вхід з валідними даними | Валідний логін та пароль | Успішний вхід, перехід на головну сторінку |
| 5 | Вхід з невалідними даними | Невалідний логіт або пароль | Повідомлення про помилку |
| 6 | Відновлення паролю | Валідна електронна адреса | Відправлення листа з інструкціями щодо відновлення паролю |

Таблиця 3.3

Управління курсами

| № | Текстовий випадок | Валідні дані | Очікуваний результат |
|---|---|---|---|
| 1 | Створення нового курсу викладачем | Назва, опис | Успішне створення курсу, перехід на сторінку курсу |
| 2 | Редагування курсу викладачем | Змінені дані курсу | Успішне оновлення даних |
| 3 | Видалення курсу викладачем | Підтвердження видалення | Успішне видалення курсу |
| 4 | Перегляд списку курсів студентом | - | Відображення списку доступних курсів |

Продовження таблиці 3.3

| | | | |
|---|---|---|---|
| 5 | Перегляд детальної інформації про курс студентом | - | Відображення назви, опису програми та матеріалів курсу |

Методи тестування включають модульне, інтеграційне, системне, навантажувальне, юзабіліті, регресійне та тестування безпеки. Модульне тестування перевіряє окремі функції та компоненти додатку, в той час як інтеграційне тестування оцінює взаємодію між цими компонентами. Системне тестування зосереджується на роботі додатку в цілому, включаючи всі функції та сценарії використання. Навантажувальне тестування перевіряє, як додаток працює під великим навантаженням. Юзабіліті-тестування оцінює зручність використання додатку для реальних користувачів. Регресійне тестування вивчає нові помилки чи проблеми, що можуть виникнути вже після внесених змін. Тестування безпеки вивчає вразливості додатку до різних видів атак.[29]

На даному етапі розробки тексти та результати тестування ще не доступні, оскільки проект знаходиться на стадії проектування. Проте, після завершення розробки буде проведено повне тестування додатку згідно з описаними вище методами та тестовими випадками.

# 4. ЕРГОНОМІКА ІТ АБО ТЕХНІКО-ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ РОЗРОБКИ

## 4.1 IT ергономіка

### *4.1.1 Вимоги до програмного забезпечення та основні підходи до його проектування з точки зору користувача*

Розробка вед-додатку для обміну навчальними матеріалами та виконанням індивідуальних завдань вимагає особливої уваги до IT-ергономіки, тобто врахування потреб та особливостей користувачів (як викладачів, так і студентів) для забезпечення комфортної та ефективної взаємодії з системою

Основні вимоги до програмного забезпечення з точки зору користувача охоплюють ряд ключових аспектів. Перш за все, важлива зручність використання, що передбачає інтуїтивно зрозумілий інтерфейс без складних інструкцій, легку навігацію та ефективний пошук потрібної інформації. Також важлива зрозуміла структура та мінімальне навантаження на користувача.

По-друге, функціональність включає в себе повний спектр необхідних функцій для викладачів, так і для студентів, а також гнучкість налаштувань та інтеграцію з іншими системами для максимальної адаптивності та зручності використання.

Надійність та безпека – ще один важливий аспект. Користувачі очікують стабільної роботи програми без збоїв, а також захисту їх персональних даних та навчальних матеріалів.

Естетичність також має значення, оскільки привабливий дизайн сприяє позитивному користувацькому досвіду, а доступність забезпечує, що програма буде корисною для всіх, включаючи людей з обмеженими можливостями (наприклад, підтримка екранних дикторів, збільшення шрифту, контрастні кольори).

Основні підходи до проектування з точки зору користувача включають у себе інклюзивний дизайн, прототипування та тестування, а також ітеративний

підхід. Усі ці методи спрямовані на створення продукту, який буде максимально зручним та приємним у використанні для кожного користувача, незалежно від їхніх потреб і можливостей. Врахування цих вимог та підходів дозволить створити веб-додаток, який буде не тільки функціональним, але й зручним та приємним у використанні, що сприятиме підвищенню ефективності навчального процесу.

### *4.1.2 Параметри, що враховують при розробці інтерфейсу користувача*

Розробка інтерфейсу користувача (UI) веб-додатку для обміну навчальними матеріалами та виконання індивідуальних завдань вимагає комплексного підходу з урахуванням різних параметрів. Ергономічні цілі та показники якості програмного продукту: ергономічні цілі – зниження когнітивного навантаження на користувача, забезпечення зручності та легкості використання, забезпечення позитивного користувацького досвіду. Показники якості: ефективність – час виконання завдань, кількість помилок, швидкість навчання користувачів. Продуктивність – час відгуку системи, швидкість завантаження сторінок. Задоволеність користувача – суб’єктивні оцінки зручності, привабливості та корисності інтерфейсу.

**Основні характеристики, що враховуються при розробці інтерфейсу користувача:**

- Простота та зрозумілість – інтерфейс повинен бути інтуїтивно зрозумілим, логічно структурованим та не містити зайвих елементів;
- Послідовність – використання єдиного стилю оформлення, однакових елементів управління та навігації на всіх сторінках додатку;
- Інформативність – надання користувачу необхідної та достатньої

інформації для виконання завдань, використанні підказок та повідомлень про помилки;

- Зворотній зв’язок – інформування користувача про результат його дій (наприклад повідомлення про успішне збереження даних);
- Гнучкість – можливість налаштування інтерфейсу під індивідуальні потреби користувача (наприклад, вибір теми оформлення, розміру шрифту).

**Вимоги до зручності та комфорту інтерфейсу включають в себе наступні аспекти:** візуальна привабливість, що передбачає використання приємної кольорової гами, якісних шрифтів та графічних елементів; ергономічність, що полягає у розміщенні елементів управління у зручних для доступу місцях та використанні звичайних жестів та комбінацій клавіш; та мінімальне навантаження на зір, що вимагає використання достатнього контрасту між текстом та фоном, уникнення мерехтіння та дрібного шрифту.

**Проблеми та особливості розробки прототипу інтерфейсу:** визначення основних сценаріїв використання необхідно визначити, як користувачі будуть взаємодіяти з додатком, які завдання вони будуть виконувати та які функції їм будуть потрібні. Створення макетів передбачає створення схематичних макетів екранів додатку для визначення розташування елементів та структури сторінок. Розробка прототипу передбачає створення інтерактивного прототипу, який дозволить користувачам протестувати основні функції та оцінити зручність використання. Тестування включає проведення юзабіліті-тестування з залученням представників цільової аудиторії для виявлення та усунення недоліків інтерфейсу.

**Принципи реалізації інтерфейсу користувача:**

**Принцип найменшого подиву** – цей принцип передбачає, що інтерфейс користувача повинен бути інтуїтивно зрозумілим і передбачуваним для

користувачів. Він орієнтований на те, щоб уникнути зайвих складних або непередбачуваних дій з боку користувача. Наприклад, розміщення елементів управління на екрані повинно відповідати загальним очікуванням користувачів, щоб вони легко знаходити те, що потрібно.

**Принцип зворотного зв'язку** – цей принцип передбачає, що користувач повинен отримувати зворотний зв'язок про результат своїх дій. Це може бути відображенням повідомлень про успішне виконання операцій, попереджень про помилки або підтвердження вибору користувача. Зворотній зв'язок допомагає користувачеві зрозуміти, що відбувається в системі під час їхніх дій і сприяє покращенню користувацького досвіду.

**Принцип простоти** – цей принцип передбачає, що інтерфейс користувача повинен бути простим та зрозумілим. Він відсилає до уникнення зайвих складнощів, заплутаності або непотрібних елементів у дизайні інтерфейсу. Простота сприяє швидкому освоєнню системи користувачем і підвищує загальний рівень зручності використання.

**Принцип послідовності** – цей принцип передбачає використання однакових або схожих елементів управління та навігації на всіх сторінках або екранах додатку. Це допомагає користувачам швидше орієнтуватися в системі, оскільки вони вже знайомі з певними елементами і можуть використовувати їх без зайвих зусиль.

**Принцип доступності** – цей принцип передбачає, що інтерфейс користувача повинен бути доступним для людей з обмеженими можливостями. Це означає, що дизайн інтерфейсу повинен бути розроблений з урахуванням потреб користувачів з різними типами обмежень, такими як візуальні, слухові або фізичні обмеження. Досягнення доступності в інтерфейсі допомагає зробити його придатним до використання всіма користувачами, що сприяє рівному доступу до інформації та послуг.

### *4.1.3 Вимоги до інтерфейсних процесів та проектування і реалізація його компонентів*

Для забезпечення ефективних взаємодії користувачів з веб-додатком необхідно враховувати вимоги до інтерфейсних процесів та ретельно проектувати кожен компонент інтерфейсу.

Вимоги до інтерфейсних процесів в дипломній роботі визначають ключові аспекти, які впливають на спосіб, яким користувачі взаємодіють з системою.

**Розглянемо їх детальніше:**

**Інтуїтивність та передбачуваність** – це означає, що користувач повинен легко розуміти, як виконати певну дію та який результат очікувати. Інтерфейс повинен бути послідовним та логічним, щоб користувач міг передбачити, як працює система. Це важливо для забезпечення приємного і безпроблемного користувацького досвіду.

**Ефективність** – інтерфейсні процеси мають бути оптимізовані для швидкого та зручного виконання завдань. Це означає, що кількість необхідних кліків та переходів між екранами має бути мінімальною. Важливо, щоб користувачі могли швидко досягти своїх цілей без зайвого зусилля.

**Надання зворотного зв’язку** – система повинна інформувати користувача про результат його дій, незалежно від того, чи вони успішні, чи сталися помилки. Це допомагає користувачам розуміти, що відбувається в системі та як вони можуть продовжувати роботу.

**Гнучкість** – інтерфейс повинен адаптуватися до різних рівнів досвіду користувачів та надати можливість налаштування під індивідуальні потреби. Наприклад, довідні користувачі можуть бажати швидких шляхів взаємодії, тоді як новачки можуть потребувати більшої підтримки та настанов.

**Відповідність стандартам** – інтерфейс повинен відповідати загальноприйнятим стандартам веб-дизайну та юзабіліті. Це включає в себе правила та рекомендації щодо розміщення елементів інтерфейсу, використання кольорів та шрифтів, а також доступність для користувачів з різними потребами.

Забезпечення відповідності цим вимогам допоможе створити інтерфейс, який буде ефективним, зручним та приємним у використанні для всіх користувачів.

Проектування та реалізація компонентів інтерфейсу вимагає уважного підходу до кожного аспекту:

**Навігація** повинна бути зручною та інтуїтивно зрозумілою, включаючи головне меню з основними розділами, зручного пошуку та «хлібні крихти»;

**Форми введення** даних мають бути простими, зрозумілими та надійними, з використанням підказок і обмежень на введення некоректних даних, а також автоматичним збереженням;

**Елементи відображення** інформації повинні забезпечувати читабельність та видимість, включаючи чіткі шрифти, якісні зображення та наочне представлення даних у таблицях та графіках;

**Елементи управління** повинні бути зрозумілими та легкими і використанні, з яскравими кнопками та іншими елементами для зміни параметрів;

**Зворотний зв’язок** важливий для користувачів, тому слід забезпечити повідомлення про дії, індикатори прогресу та можливість скасування.

**Технології:**

- React.js – для створення компонентів інтерфейсу та управління їх станом;

- Radix-UI – для забезпечення готового набору компонентів та стилів, що відповідають WAI-ARIA;
- HTML, CSS – для розмітки та оформлення сторінок;
- TypeScript – для додавання інтерактивності та динаміки.

**Принципи реалізації:**

- **Компонентний підхід:** розбиття інтерфейсу на окремі компоненти, що полегшує його розробку, тестування та підтримку;
- **Тестування:** проведення модульного, інтеграційного та юзабіліті-тестування для забезпечення якості та зручності використання інтерфейсу.

Врахування цих вимог та принципів дозволить створити інтуїтивно зрозумілий, ефективний та зручний інтерфейс користувача, який відповідатиме потреба цільової аудиторії та забезпечить позитивний досвід взаємодії з веб-додатком.

### *4.1.4 Проектування та реалізація інтерфейсу*

Цей розділ детально описує, як будуть спроектовані та реалізовані основні компоненти інтерфейсу користувача веб-додатку, враховуючи принципи ергономіки та користувацького досвіду (UX).

**Навігація:**

**Головне меню:** розташоване у верхній частині екрану, містить логотип додатку, посилання на основні розділи, а також елементи керування обліковим записом (вхід/реєстрація, вихід). Використання “sticky” меню для забезпечення постійної доступності навігації.

**Бічне меню:** використовується на сторінках курсів та завдань для швидкого доступу до розділів та матеріалів.

**«Хлібні крихти»:** розташовані під головним меню, відображають поточне місцезнаходження користувача в ієрархії сторінок та дозволяють швидко переходити на вищі рівні.

**Форми введення даних** мають відповідати кільком принципам, щоб забезпечити зручність користування та надійність зберігання інформації. Перш за все, вони повинні мати мінімальну кількість полів, щоб не перевантажувати користувача зайвими деталями, кожне поле має бути чітко підписане, щоб користувач міг легко розібратися у введеній інформації.

Для забезпечення правильного форматування даних можна використовувати маски введення. Наприклад, для номеру телефону або дати можна встановити відповідні маски, які автоматично форматують введені дані в потрібний стиль.

Важливим аспектом є також валідація даних у реальному часі з повідомленнями про помилки. Це дозволяє уникнути введення некоректних або недостовірних даних та надає користувачеві можливість виправити помилки без зайвих зусиль.

На прикладі форм включають форму реєстрації та входу, яка містить поля для введення імені, прізвища, електронної пошти та паролю. Також можуть бути форми створення курсу, які містять поля для введення назви, опису, вибору категорії, або форми створення завдання, що включають поля для назви, опису, інструкцій, вибору типу завдання та дати завершення.

**Елементи відображення інформації** повинні відповідати певним принципам для забезпечення зрозумілості та зручності користування. Потрібно мати чітку структуру контенту, щоб користувач міг легко орієнтуватися на сторінці. Важлива інформація має бути виділена, щоб користувач міг швидко її знайти.

Використання різних рівнів заголовків допомагає структурувати інформацію та робить її більш зрозумілою. Списки таблиці також сприяють кращій організації даних, дозволяючи швидко знайти необхідну інформацію.

Наприклад, на сторінці курсу повинні бути представлені назва курсу, його опис, програма, список розділів, список завдань та календар подій. Сторінка матеріалу може містити заголовок, текст, зображення, відео та посилання на додаткові матеріали. На сторінці завдання повинні бути зазначені назва, опис, інструкції, критерії оцінювання, статус виконання та оцінка.

**Елементи управління** повинні відповідати певним принципам, щоб забезпечити зручність використання та ефективність взаємодії користувачів з додатком. По-перше, важливо використовувати зрозумілі іконки та підписи, щоб користувачі могли легко розпізнавати призначення кожного елемента. По-друге, елементи управління повинні бути розміщені у зручних для доступу місцях, щоб користувачі могли швидко їх знайти і використовувати.

Приклади таких елементів управління включають кнопки для виконання різних дій, таки як «Створити курс», «Додати матеріал», «Відправити завдання» та «Написати коментар». Також до елементів управління належать посилання для переходу на інші сторінки або завантаження файлів.

**Зворотній зв’язок** має забезпечувати користувачів зрозумілою та корисною інформацією про їхні дії в системі. Принципи надання зворотного зв’язку включають повідомлення про успішне виконання дій, такі як «Завдання успішно відправлено». Для довгих операцій, таких як завантаження файлу, необхідно використовувати індикатори прогресу, щоб бачити хід виконання завдання.

Прикладом зворотного зв’язку можуть бути повідомлення про успішну реєстрацію або вхід в систему. Якщо користувач введе некоректні дані у форму, система повинна надати повідомлення про помилку з відповідними інструкціями щодо виправлення.

**Технології реалізації:**

- React.js – для створення компонентної структури інтерфейсу та управлінням станом;
- Material-UI – для використання готових компонентів та стилів;
- CSS Modules – для локалізації стилів та уникнення конфліктів;
- React Router – для організації навігації між сторінками.

Ретельне проектування та реалізація компонентів інтерфейсу з урахуванням принципів ергономіки та користувацького досвіду дозволить створити веб-додаток, який буде не лише функціональним, але й зручним та приємним у використанні.

## 4.2 Техніко-економічне обґрунтування

**Назва проекту:** insync – веб-додаток для обміну навчальними матеріалами та виконання індивідуальних завдань.

**Локація:** Україна (орієнтовано на український ринок, з перспективою виходу на міжнародний).

**Мета:** підвищення ефективності та якості освітнього процесу шляхом надання зручної та функціональної платформи для взаємодії між викладачами та студентами.

**Зміст:** розробка та впровадження веб-додатку, який дозволить викладачам створювати та керувати навчальними курсами, завантажувати матеріали різних форматів, створювати та призначати завдання, оцінювати роботи студентів та надавати зворотний зв’язок. Студенти зможуть отримувати доступ до навчальних матеріалів, виконувати завдання, спілкуватися з викладачами та одногрупниками, відстежувати свій прогрес.

insync – це веб-додаток, що надає наступні можливості для викладачів: створення та налаштування курсів з гнучкою структурою, завантаження та організація навчальних матеріалів (тексти, презентації, відео, аудіо, інтерактивні елементи), створення та призначення різноманітних завдань (тести, есе, файли для завантаження, форуми), автоматична та ручна перевірка завдань, надання індивідуального зворотного зв'язку студентам, спілкування зі студентами через форуми та чати, моніторинг прогресу студентів та формування звітів. Для студентів платформа надає можливість перегляду начальних матеріалів у зручному форматі, виконання завдань та відстеження термінів, отримання оцінок та коментарів від викладачів, спілкування з викладачами та одногрупниками, а також моніторинг власного прогресу у навчанні.

**Оцінка ринку:**

**Цільова аудиторія:** вищі навчальні заклади (університети, коледжі, технікуми), середні загальноосвітні школи, центри підвищення кваліфікації та професійного розвитку, компанії, що проводять корпоративне навчання.

**Потенціал ринку:** згідно з даними Міністерства освіти і науки України, в Україні налічується понад 300 вищих навчальних закладів та близько 12000 загальноосвітніх шкіл. Попит на онлайн-освіту та дистанційне навчання стрімко зростає після пандемії COVID-19. insync може стати ефективним інструментом для покращення якості освіти та підвищення конкурентоспроможності українських навчальних закладів.

**Аналіз конкуренції:**

- **Основні конкуренти:**

- Moodle – платформа з відкритим кодом;
- Google Classroom – безкоштовний сервіс від Google;
- Microsoft Teams – платформа для спільної роботи.

- **Конкурентні переваги insync:**

- Адаптація до української освітньої системи;
- Ширший функціонал, що включає в себе: автоматичну перевірку завдань, індивідуальні рекомендації, розширені можливості аналітики;
- Інтуїтивно зрозумілий інтерфейс, розроблений з урахуванням потреб українських користувачів;
- Можливість інтеграції з іншими інформаційними системами навчальних закладів.

**Маркетингова стратегія:**

Ціноутворення: пропонування різних тарифних планів для навчальних закладів та індивідуальних користувачів (безкоштовний план з обмеженим функціоналом, платні плани з розширеними можливостями).

Просування: контекстна реклама в пошукових системах, реклама у соціальних мережах, участь у освітніх виставках та конференціях, партнерство з навчальними закладами та освітніми організаціями.

**Виробничий план:**

- **Етапи розробки:**

- Аналіз вимог та проектування – 1 місяць;
- Розробка frontend backend – 3 місяці;
- Тестування та налагодження – 1 місяць;
- Впровадження та підтримка – постійно.

- **Ресурси:**

- Команда розробників;
- Дизайнер інтерфейсу;
- Менеджер проекту;
- Серверне обладнання та програмне забезпечення.

**Організаційний та правовий план:**

- **Організаційна структура:**

- ТОВ або ФОП.

- **Правові аспекти:**

- Реєстрація підприємства;
- Отримання необхідних ліцензій та дозволів;
- Укладення договорів з користувачами та партнерами;
- Захист інтелектуальної власності.

**Фінансовий план та стратегія фінансування:**

- **Джерела фінансування:**

- Власні кошти;
- Інвестиції;
- Гранти та конкурси.

- **Основні витрати:**

- Заробітна плата співробітників;
- Маркетинг та реклама;
- Оренда офісу та обладнання;
- Ліцензії на програмне забезпечення.

- **Прогнозовані доходи:**

- Передплата від навчальних закладів та індивідуальних користувачів;
- Продаж додаткових послуг (наприклад, розробка індивідуальних курсів);
- Реклама на платформі.

**Оцінка ризиків та страхування:**

- **Основні ризики:**

- Технічні ризики (збої в роботі системи, кібератаки);
- Фінансові ризики (недостатність фінансування, низький попит на продукт);

- Правові ризики (порушення авторських прав, невиконання договірних зобов’язань).

- **Заходи щодо зниження ризиків:**

- Ретельне тестування та налагодження програмного забезпечення;
- Забезпечення резервного копіювання даних;
- Розробка політики безпеки та захисту даних;
- Страхування відповідальності;
- Укладення договорів з надійними партнерами.

Запропонований проект insync має значний потенціал для розвитку освітньої галузі України. Він дозволить підвищити якість та ефективність навчання, зробити освіту більш доступною та сприятиме розвитку цифрової грамотності.

# ВИСНОВКИ

Розробка веб-додатку insync є ваговим внеском у модернізацію освітнього процесу в Україні. Проект демонструє комплексний підхід до вирішення актуальних проблем, пов’язаних з обміном навчальними матеріалами, організацію індивідуальних завдань та підвищенням ефективності навчання.

У першому розділі  класифіковано об’єкт дослідження, а саме, за цільовою аудиторією, функціональними можливостями, типом доступу, основними характеристиками та обґрунтовано актуальність даної розробки.

Проведено аналіз проблем та побудова дерева цілей. Крім того, було проаналізовано конкурентів, що дозволило сформувати функціональні, нефункціональні та технічні вимоги до нового веб-додатку.

У другому розділі було досліджено функціональні та інтерфейсні особливості веб-додатку.

Розроблена база даних, запроектовані концептуальна, датологічна та фізична моделі БД. Результати аналізу проектних рішень були представлені у вигляді діаграм: прецедентів, послідовності, компонентів, розгортання та відношень.

У третьому розділі розроблено програмну реалізацію веб-додатку в рамках якого було описано структуру та функціонування веб-додатку. Створена структурна схема веб-додатку та проведено тестування.

Четвертий розділ визначає вимоги до програмного забезпечення, параметри, принципи та вимоги, що враховують при розробці інтерфейсу користувача. Слід зауважити, що було здійснено техніко-економічне обґрунтування веб-додатку.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. A genetic algorithm analysis towards optimization solutions [Electronic resource]. – Mode of access: https://bit.ly/3ReXLt8
2. About face: the essentials of interaction design [Electronic resource] / David Cronin [et al.]. – [S. l.] : Wiley, 2014. – 720 p. 425-446 – Mode of access: https://bit.ly/3x59nYW
3. Agile, scrum, kanban: у чому різниця і навіщо використовувати? [Електронний ресурс]. – Режим доступу: https://it-skills.in.ua/agile-scrum-kanban-u-chomu-riznytsia-i-navishcho-vykorystovuvaty/
4. C M. R. Clean architecture: A craftsman's guide to software structure and design (robert C. martin series) / Martin Robert C. – [S. l.] : Prentice Hall, 2017. – 432 p. 281-286
5. Chen, B., Lambert, A. D., & Guidry, K. R. (2010). Engaging online learners: The impact of Web-based learning technology on college student engagement. Computers & Education, 54(4), 1222-1232
6. Chapman, R., & McLaughlin, B. (2006). Designing the data model. Oracle Magazine, 21(1), 40-46
7. Charles Lai. ER diagram for website [classic] [Electronic resource] / Charles Lai. – Mode of access: https://creately.com/diagram/example/hubct0481/er-diagram-for-website-classic
8. Date, C. J. (2004). An Introduction to Database Systems (8th ed.). Pearson Education
9. Davidson-Shivers G. V. Web-Based learning: design, implementation, and evaluation / Gayle V. Davidson-Shivers, Karen L. Rasmussen. – [S. l.] : Prentice Hall, 2006. – 405 p. 120-140
10. Designing effective instruction [Electronic resource] / Howard Kalman [et al.]. – [S. l.] : Wiley & Sons, Incorporated, John, 2010. – Mode of access:

https://books.google.com.tj/books?id=ygIbaClN3KMC&printsec=frontcover#v=onepage&q&f=false

11. Foleon and web accessibility [Electronic resource]. – Mode of access: https://www.foleon.com/foleon-and-web-accessibility
12. Freeman, E., & Robson, E. (2018). Head First HTML and CSS: A learner's guide to creating standards-based web pages. O'Reilly Media
13. Gollmann D. Computer security - ESORICS 2005: 10th european symposium on research in computer security, milan, italy, september 12-14, 2005, proceedings / Dieter Gollmann, Paul Syverson, Sabrina De Capitani di Vimercati. – [S. l.] : Springer London, Limited, 2005. – 516 p. 150-160
14. Graph theory: a comprehensive survey about graph theory applications in computer science and social networks [Electronic resource]. – Mode of access: https://www.mdpi.com/2411-5134/5/1/10
15. Intelligent systems design and applications: 21st international conference on intelligent systems design and applications held december 13-15 2021 [Electronic resource] / Tatiane Nogueira Rios [et al.]. – [S. l.] : Springer International Publishing AG, 2022. 379-382 – Mode of access: https://bit.ly/3V65sTq
16. K-means clustering [Electronic resource]. – Mode of access: https://en.wikipedia.org/wiki/K-means_clustering
17. Layered architecture [Electronic resource]. – Mode of access: https://openclassrooms.com/en/courses/6397806-design-your-software-architecture-using-industry-standard-patterns/6896176-layered-architecture
18. Mayer R. Multimedia learning / Richard Mayer. – [S. l.] : Cambridge University Press, 2020. – 320 p. 145-170
19. Naive Bayes classifier [Electronic resource]. – Mode of access: https://en.wikipedia.org/wiki/Naive_Bayes_classifier
20. Norman, D. A. (2013). The design of everyday things: Revised and expanded edition. Basic Books

21. Object-Oriented analysis and design with applications (3rd edition) [Electronic resource] / Robert A. Maksimchuk [et al.]. – [S. l.] : Addison-Wesley Professional, 2007. – 720 p. – Mode of access: https://bit.ly/4bRCgqA
22. Rob Davies. Structure of the WSMO description of the EMS prototype [Electronic resource] / Rob Davies. – Mode of access: https://bit.ly/4bPM4l1
23. S. Begum, C. Indumathi. ER diagram based web application testing [Electronic resource] / S. Begum, C. Indumathi. – Mode of access: https://bit.ly/3yRqjme
24. Sequence diagram including Registration and Login [Electronic resource]. – Mode of access: https://stackoverflow.com/questions/66265720/sequence-diagram-including-registration-and-login
25. Shalin Siriwardhana. Use case diagram for web application for recruiting [Electronic resource] / Shalin Siriwardhana. – Mode of access: https://creately.com/diagram/example/gm8kr28v1/use-case-diagram-for-web-application-for-recruiting
26. Spiral model [Electronic resource]. – Mode of access: https://qalight.ua/baza-znaniy/cpiralna-model-spiral-model/
27. Suskie, L. (2018). Assessing student learning: A common sense guide. John Wiley & Sons
28. Teorey, T. J., & Lightstone, S. S. (2006). Database Modeling and Design: Logical Design. Morgan Kaufmann
29. Types of software testing strategies with examples [Electronic resource]. – Mode of access: https://www.testrail.com/blog/software-testing-strategies/
30. UML component diagram [Electronic resource]. – Mode of access: https://bit.ly/45lAOdK
31. Using the structred analysis and design technique (SADT) in simulation conceptual modeling [Electronic resource]. – Mode of access: https://ieeexplore.ieee.org/abstract/document/7019963
32. Web application UML deployment diagram example [Electronic resource]. – Mode of access:

https://www.uml-diagrams.org/web-application-uml-deployment-diagram-example.html

33. Zuhaimy Ismail. Typical web application structure [Electronic resource] / Zuhaimy Ismail. – Mode of access: https://www.researchgate.net/figure/Typical-Web-Application-Structure-2_fig1_224366357

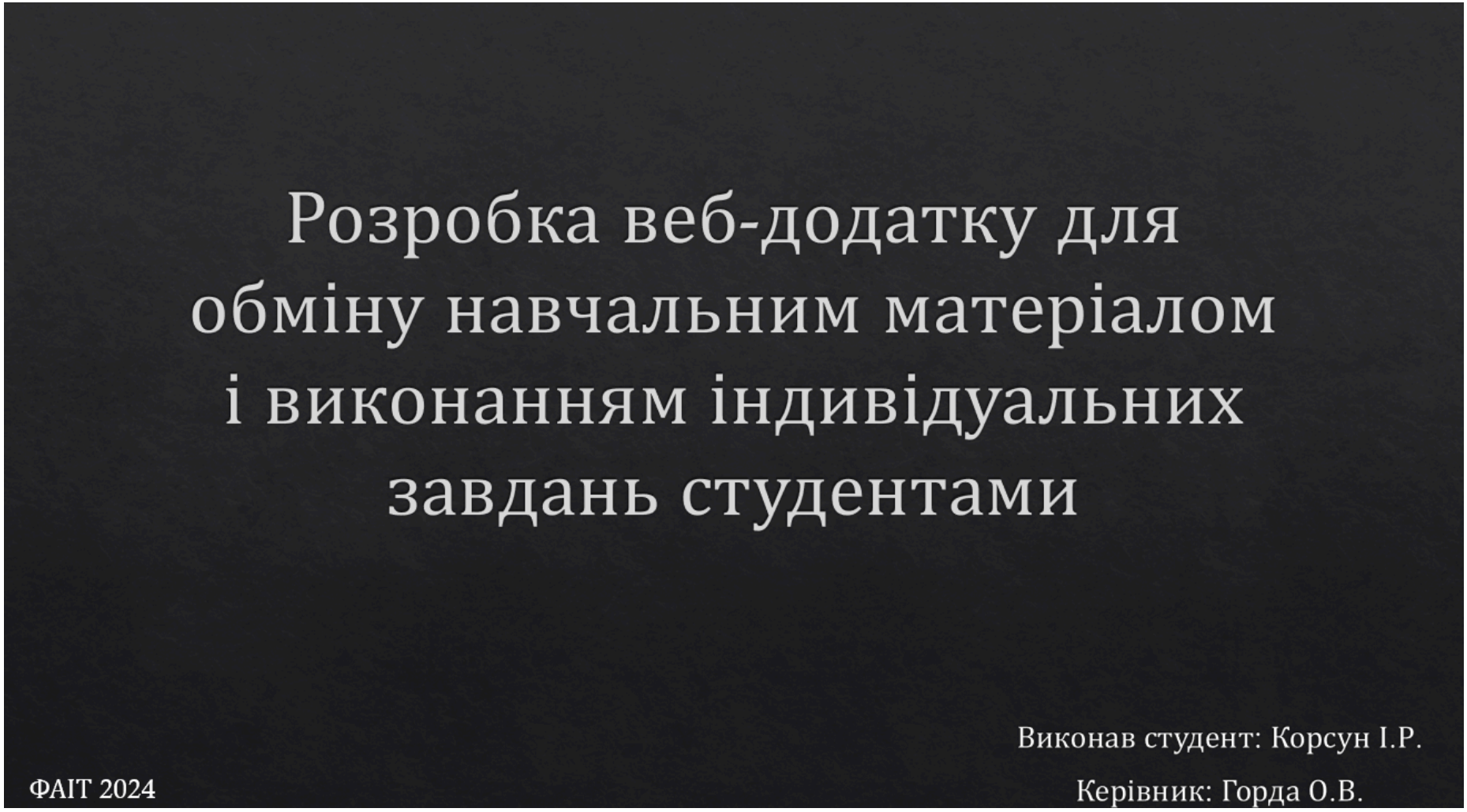
## Додаток А. Програмна реалізація

```
generator client {
    provider = "prisma-client-js"
}

datasource db {
    provider = "postgresql"
    url      = env("POSTGRES_URL")
}

model User {
    id        String  @id
    imageUrl  String?
    firstName String
    lastName  String
    email     String

    ownedOrganizations Organization[]
    organizations      Organization[] @relation("OrganizationMembers")

    auditLogs                       AuditLog[]
    cardComments                    CardComment[]
    sentOrganizationInvitations     OrganizationInvitation[]
@relation("InvitationSender")
    receivedOrganizationInvitations OrganizationInvitation[]
@relation("InvitedUser")

    createdAt DateTime @default(now())
    updatedAt DateTime @updatedAt
}

model Organization {
    id   String @id @default(cuid())
    name String

    auditLogs AuditLog[]
    boards    Board[]
    lists     List[]
    cards     Card[]
    members   User[]     @relation("OrganizationMembers")

    ownerId String
```

```
    owner   User   @relation(fields: [ownerId], references: [id], onDelete:
Cascade)

    createdAt DateTime @default(now())
    updatedAt DateTime @updatedAt

    color String

    invitations OrganizationInvitation[]

    @@index([ownerId])
}

model OrganizationInvitation {
    id String @id @default(cuid())

    token String @unique

    invitedUser      User?   @relation("InvitedUser", fields: [invitedUserId],
references: [id], onDelete: Cascade)
    invitedUserId    String?
    invitedUserEmail String? @unique

    sender   User   @relation("InvitationSender", fields: [senderId], references:
[id], onDelete: Cascade)
    senderId String

    organization   Organization @relation(fields: [organizationId], references:
[id], onDelete: Cascade)
    organizationId String

    createdAt DateTime @default(now())
    updatedAt DateTime @updatedAt

    @@index([invitedUserId])
    @@index([senderId])
    @@index([organizationId])
}

model Board {
    id   String @id @default(cuid())
    name String

    organization   Organization @relation(fields: [organizationId], references:
[id], onDelete: Cascade)
    organizationId String

    lists List[]

    createdAt DateTime @default(now())
    updatedAt DateTime @updatedAt

    @@index([organizationId])
}
```

```
model List {
    id    String @id @default(cuid())
    name  String
    order Int

    organization   Organization @relation(fields: [organizationId], references:
[id], onDelete: Cascade)
    organizationId String

    boardId String
    board   Board  @relation(fields: [boardId], references: [id], onDelete:
Cascade)

    cards Card[]

    createdAt DateTime @default(now())
    updatedAt DateTime @updatedAt

    @@index([boardId])
    @@index([organizationId])
}

model Card {
    id          String  @id @default(cuid())
    name        String
    order       Int
    description String? @db.Text

    organization   Organization @relation(fields: [organizationId], references:
[id], onDelete: Cascade)
    organizationId String

    listId String
    list   List   @relation(fields: [listId], references: [id], onDelete: Cascade)

    comments CardComment[]

    createdAt DateTime @default(now())
    updatedAt DateTime @updatedAt

    @@index([listId])
    @@index([organizationId])
}

model CardComment {
    id      String  @id @default(cuid())
    content String? @db.Text

    authorId String
    author   User   @relation(fields: [authorId], references: [id], onDelete:
Cascade)

    card   Card   @relation(fields: [cardId], references: [id], onDelete: Cascade)
```

```
    cardId String

    createdAt DateTime @default(now())
    updatedAt DateTime @updatedAt

    @@index([authorId])
    @@index([cardId])
}

enum ACTION {
    CREATE
    UPDATE
    MOVE
    DELETE
}

enum ENTITY_TYPE {
    BOARD
    LIST
    CARD
}

model AuditLog {
    id String @id @default(cuid())

    action                ACTION
    entityId              String
    entityType            ENTITY_TYPE
    entityName            String
    destinationEntityName String?
    sourceEntityName      String?

    organization   Organization @relation(fields: [organizationId], references:
[id], onDelete: Cascade)
    organizationId String

    user   User   @relation(fields: [userId], references: [id])
    userId String

    createdAt DateTime @default(now())
    updatedAt DateTime @updatedAt

    @@index([organizationId])
    @@index([userId])
}
```

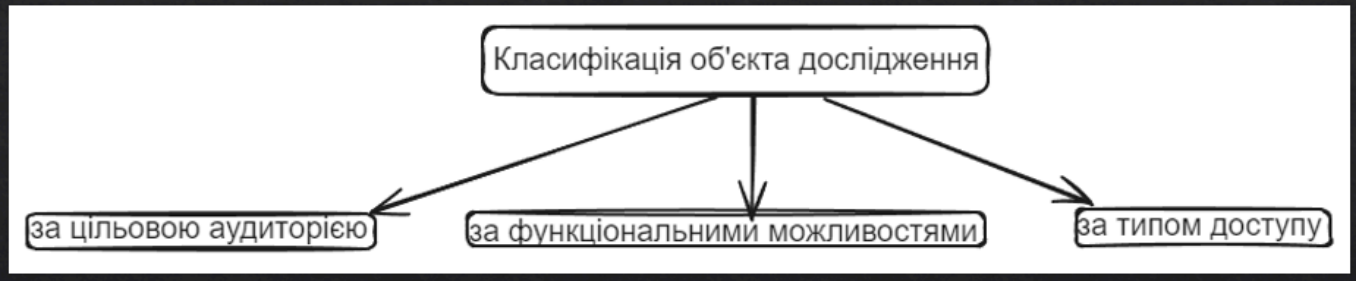



Додаток Б. Презентація

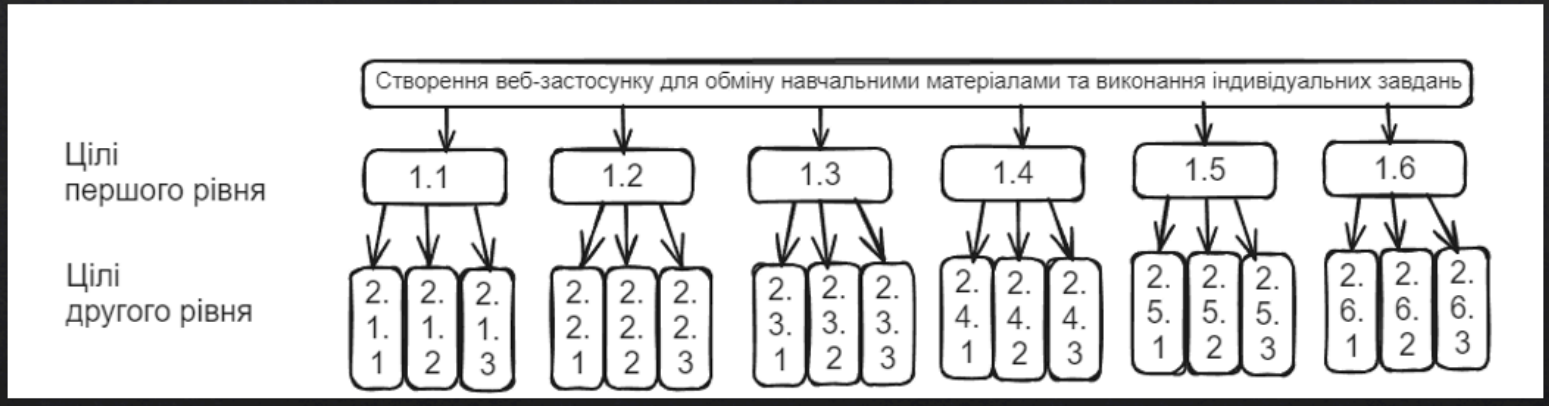
# Вступ

- Мета роботи
- Актуальність роботи
- Об’єкт дослідження
- Предмет дослідження

# Класифікація об’єкта дослідження



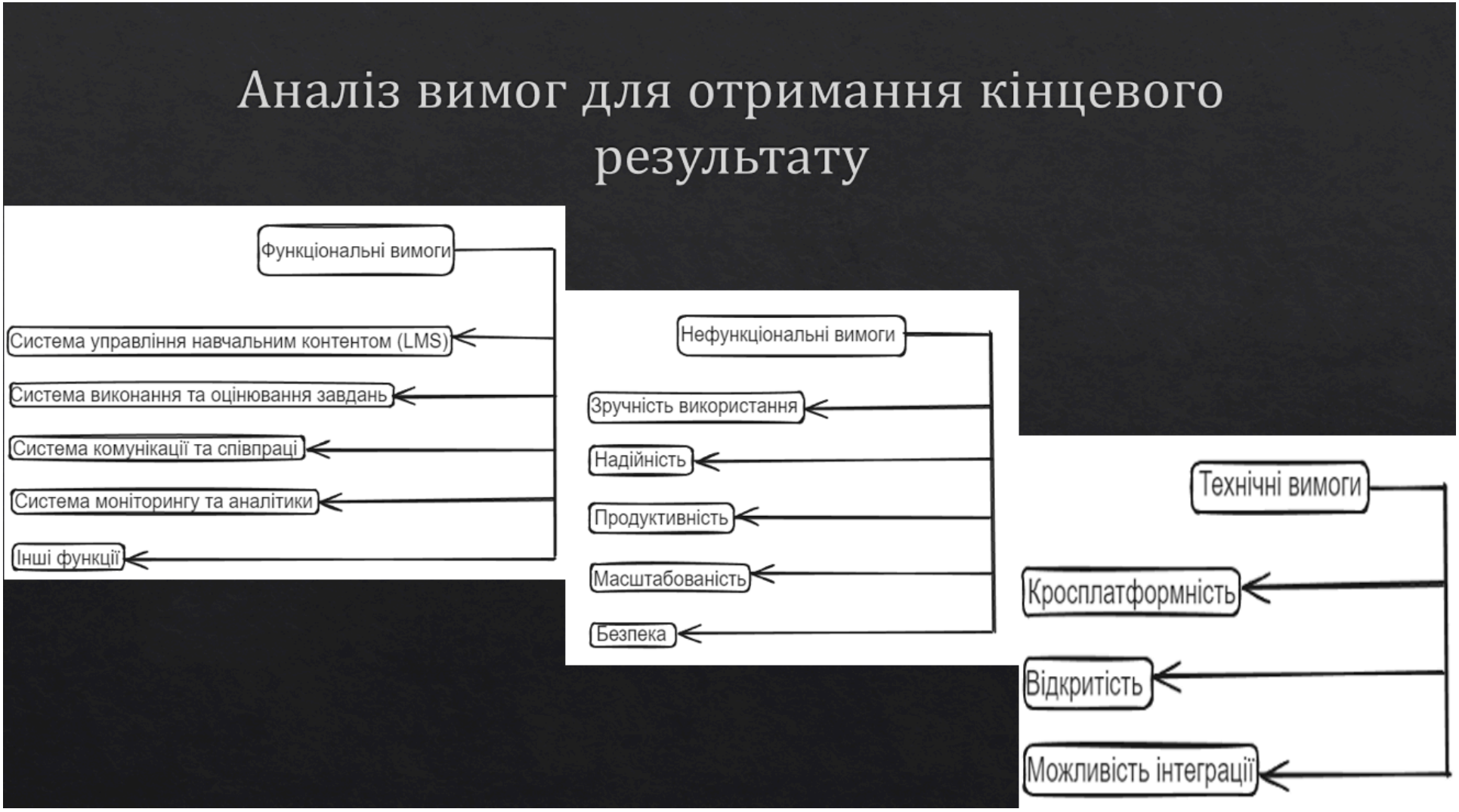
# Дерево цілей



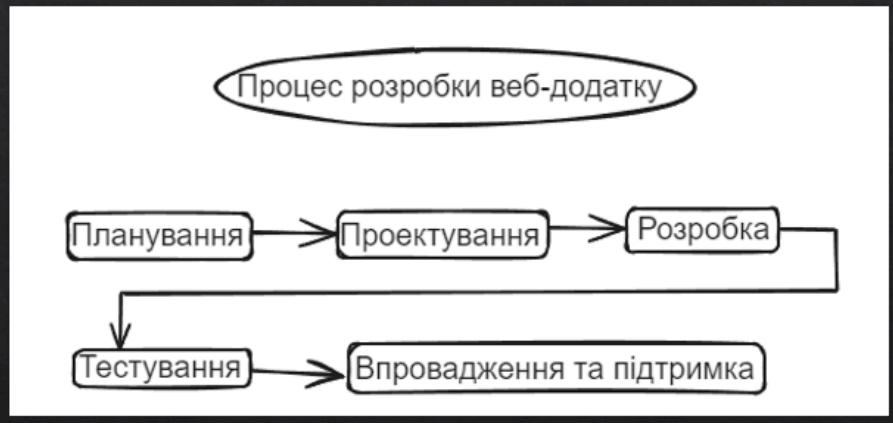
# Аналіз існуючих платформ та порівняльні характеристики

| **Функції/Аспекти** | **insync** | **Moodle** | **Google Classroom** | **Microsoft Teams** |
|---|---|---|---|---|
| Управління курсами | + | + | + | + |
| Надсилання завдань | + | + | + | + |
| Обмін матеріалами | + | + | + | + |
| Оцінювання та зворотний зв'язок | + | + | + | + |
| Спільна робота в реальному часі | + | - | - | + |
| Інтуїтивно зрозумілий і зручний інтерфейс | + | - | + | + |
| Дизайн інтерфейсу, зручний для мобільних пристроїв | + | + | + | + |

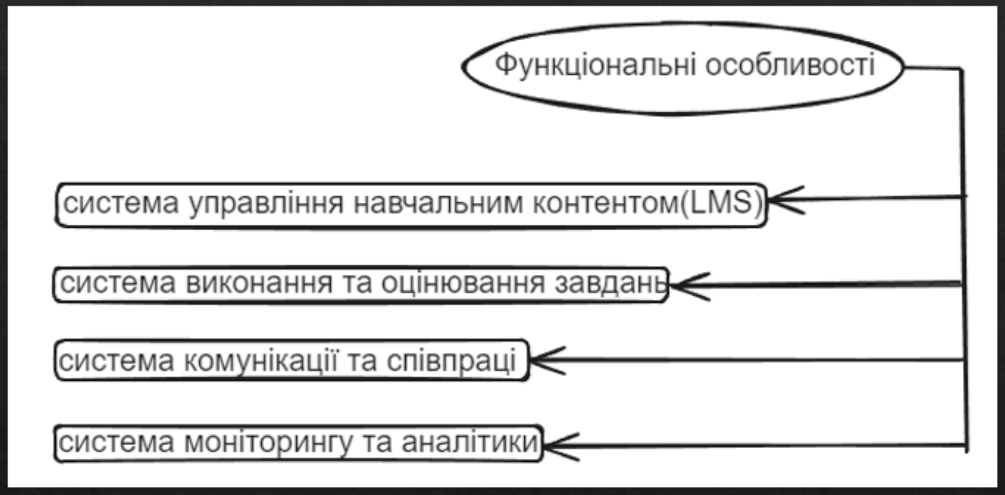
# Аналіз вимог для отримання кінцевого результату

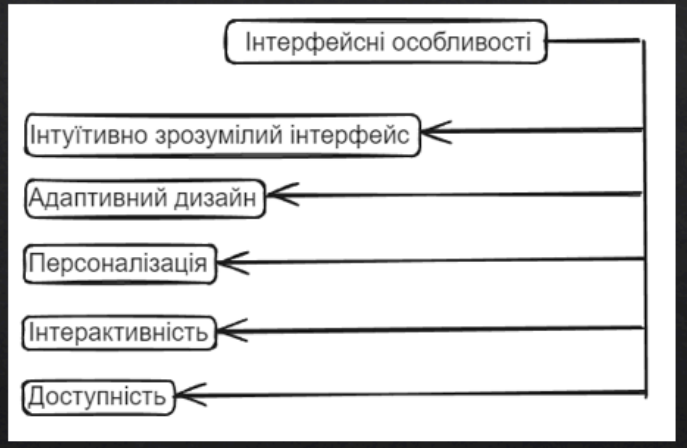


# Процес розробки веб-додатку

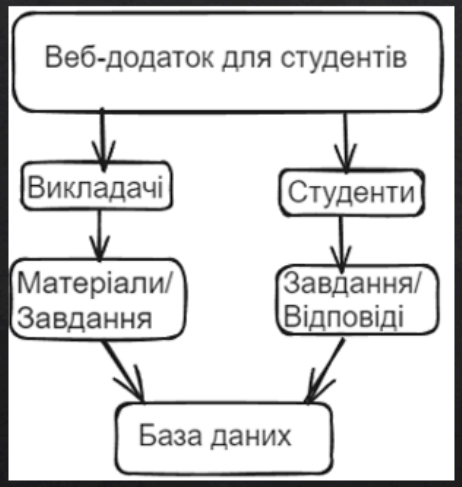


# Функціональні та інтерфейсні особливості веб-додатку

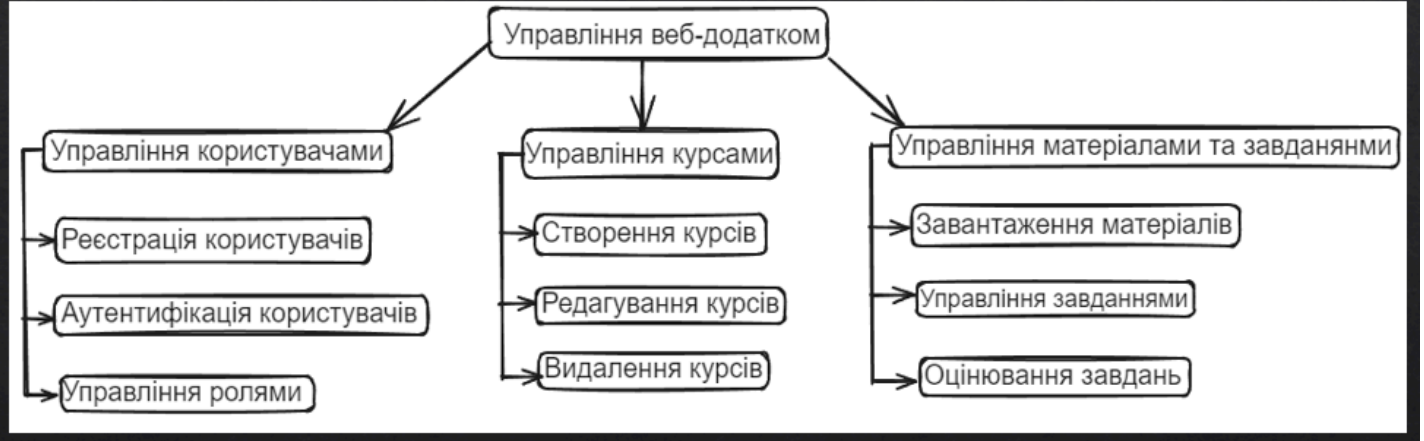


# Діаграми SADT

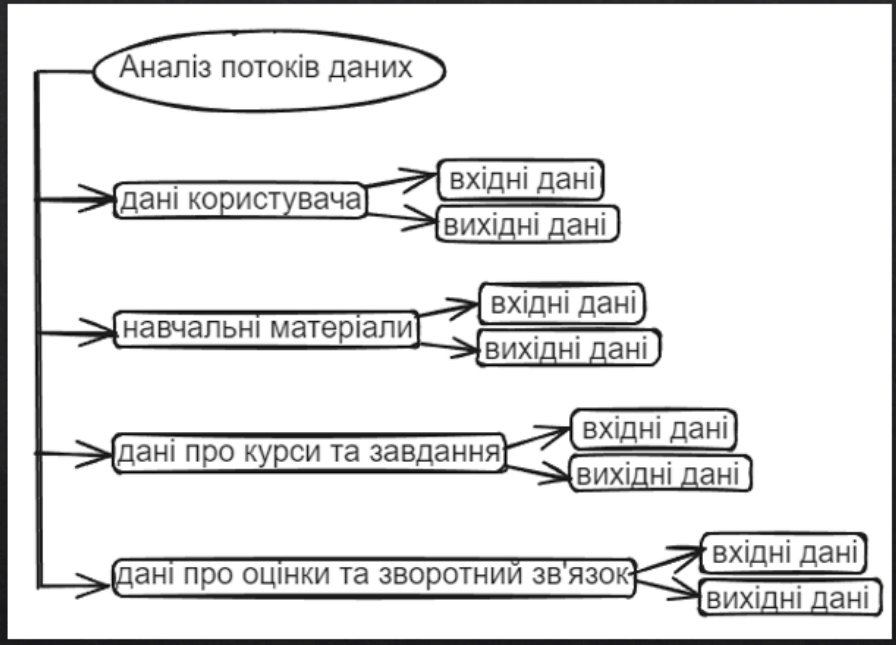



Контекстна діаграма

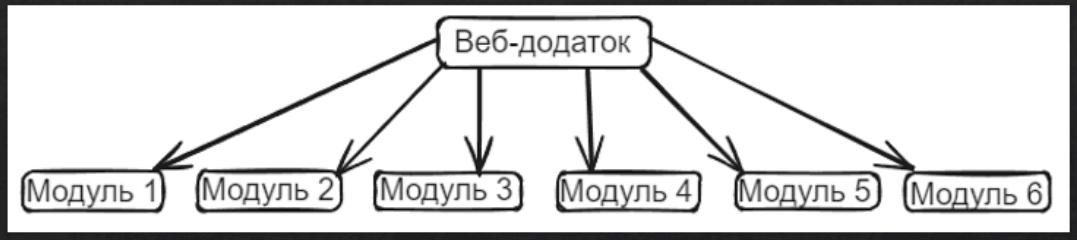



Діаграма нижчого рівня та
деталізація під процесів

# Проектування бази даних веб-додатку



# Проектування модульної структури інтерфейсу



# Техніко-економічне обґрунтування

- Оцінка ринку
- Аналіз конкуренції
- Маркетингова стратегія
- Виробничий план
- Фінансовий план
- Оцінка ризиків

# Висновки

- Розробка веб-додатку insync є важливим внеском до поліпшення освітнього процесу в Україні. Проект вирішує проблеми обміну навчальними матеріалами, організації індивідуальних завдань та підвищення ефективності навчання.
- У першому розділі класифіковано об’єкт дослідження та обґрунтовано актуальність. Проведено аналіз проблем, конкурентів, побудовано дерево цілей, визначено вимоги.
- У другому розділі досліджено функціональні та інтерфейсні особливості, розроблено базу даних і діаграми.
- У третьому розділі описано структуру веб-додатку, створено схему, проведено тестування.
- Четвертий розділ визначає вимоги до програмного забезпечення та інтерфейсу, здійснено техніко-економічне обґрунтування.