Тема:

«Онлайн платформа дистанційного навчання, з модулем рейтингу успішності студента, виконання практичних робіт та системою тестування і перевірки знань»

ЗМІСТ

РОЗДІЛ 1. АНАЛІЗ АНАЛОГІВ ТА ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ ПЛАТФОРМИ ДИСТАНЦІЙНОГО НАВЧАННЯ З ДОДАТКОВИМИ МОДУЛЯМИ ПЕРЕВІРКИ ЗНАНЬ

1.1.Постановка задачі

Платформи та сервіси дистанційного навчання представляють собою віртуальну освітню систему, яка надає викладачам та студентам розширені можливості використання сучасних технологій для навчання на відстані. Вони забезпечують організацію взаємодії між викладачами та студентами в інтерактивному форматі, а також спрощують керування процесом дистанційного навчання. Це середовище та інструмент для передачі знань [1].

Основною метою створення платформи дистанційного навчання є спрощення процесу обміну навчальними матеріалами, керування ними та розповсюдження за допомогою інтернету. На ній викладачі можуть завантажувати матеріали курсів, створювати тести та оцінювати роботи студентів. Студенти можуть переглядати матеріали курсів й власну успішність, виконувати практичні завдання, проходити тести та отримувати зворотній зв’язок.

Назва платформи, що розробляється – Elers.

Передбачається, що цільовою аудиторією платформи дистанційного навчання будуть студенти 17-30 років та працівники закладів вищої освіти (ЗВО) 25-60 років.

Для того, щоб успішно завершити створення даної платформи потрібно виконати наступні завдання:

- проаналізувати аналоги програмних продуктів для дистанційного навчання;
- визначити функціональні та нефункціональні вимоги до платформи;
- обрати оптимальну архітектуру для функціонування системи;
- обрати мови програмування та фреймворки для розробки;
- визначити акторів та варіанти використання системи;
- спроектувати базу даних, яку буде легко масштабувати;

- розробити програмний комплекс з наступними функціональними можливостями:
  - створення та редагування курсів, практичних робіт та тестів;
  - завантаження та перегляд навчальних матеріалів;
  - виставлення успішності студентів та відвідування занять;
  - завантаження та оцінювання практичних робіт;
  - підтримка різних типів запитань у тестах.
- створити адаптивний, зрозумілий та простий дизайн;
- провести тестування роботи додатку на відповідність поставленим вимогам.

Важливим кроком є аналіз аналогів, тобто додатків, що мають подібний функціонал. На цьому етапі потрібно визначити переваги та недоліки інших систем, які варто врахувати при розробці власної, а також знайти корисні та додаткові можливості. Виконання даного кроку дозволить уникнути помилок, які наявні в інших системах й отримати якісний та унікальний продукт.

Наступним кроком потрібно визначити функціональні та нефункціональні вимоги до системи електронного навчання, зокрема можливості перегляду та завантаження навчальних матеріалів, проведення тестування, оцінювання студентів тощо.

Розробку програмного коду цієї платформи можна розділити на три ключові етапи: проектування бази даних, створення користувацького інтерфейсу та розробка RESTful сервера. Останнє забезпечує можливість взаємодії з функціями додатку через визначені URL-адреси та HTTP-методи, такі як GET, POST, PUT, PATCH, DELETE та інші [8].

Результатом успішної реалізації поставленого завдання є онлайн платформа дистанційного навчання, яка забезпечує доступ до широкого спектру навчальних матеріалів, тестів, практичних завдань та успішності студентів.

1.2. Дослідження аналогів систем дистанційного навчання

Для проведення дослідження аналогів систем дистанційного навчання було використано метод порівняльного аналізу. За допомогою цього методу можна визначити як спільні, так і відмінні характеристики різних платформ, шляхом порівняння їхніх однотипних властивостей, таких як функціональність, дизайн інтерфейсу, зручність використання та інші. При цьому, системи, які порівнюються, повинні бути схожими за своїми цілями, цільовою аудиторією та функціональними можливостями. Застосування порівняльного аналізу дозволяє виявити переваги та недоліки кожної системи, а також знайти нові ідеї та рішення для покращення функціональності та дизайну власного проєкту [15].

Після детального аналізу і перегляду подібних систем стало зрозуміло, що світовий ринок має багато схожих продуктів. Існує безліч сервісів, які пропонують різні функціональні можливості, які іноді є подібними, а іноді представляють унікальні рішення.

Отже, розглянемо найбільш популярні платформи для дистанційного навчання з метою виокремлення ключових особливостей та визначення аспектів, які можна впровадити у власний продукт.

1.2.1. Moodle

Moodle (Modular Object-Oriented Dynamic Learning Environment, вимовляється «Мудл») – це модульне об’єктно-орієнтоване динамічне навчальне середовище, яке називають також системою управління навчанням, курсами, віртуальним навчальним середовищем або просто платформою для навчання, яка надає викладачам та студентам великий набір інструментів для дистанційного навчання. Цю платформу можна використовувати для навчання здобувачів освіти, при підвищенні кваліфікації, бізнес-навчанні, як в комп’ютерних аудиторіях, так і для самостійної роботи вдома [2].

Moodle надає багато різного функціоналу для створення онлайн-курсів та взаємодії зі студентами. Використовуючи дану платформу, викладачі мають змогу створювати власні дистанційні курси та публікувати навчальні матеріали (тексти лекцій, практичні завдання, самостійні роботи, тести для студентів

тощо) та контролювати прогрес учнів. Вона широко використовується багатьма закладами вищої та середньої освіти, адже є відмінним помічником в організації дистанційного навчання. Ця система може бути використана не тільки для організації традиційних дистанційних курсів, але й для підтримки очного навчання [3].

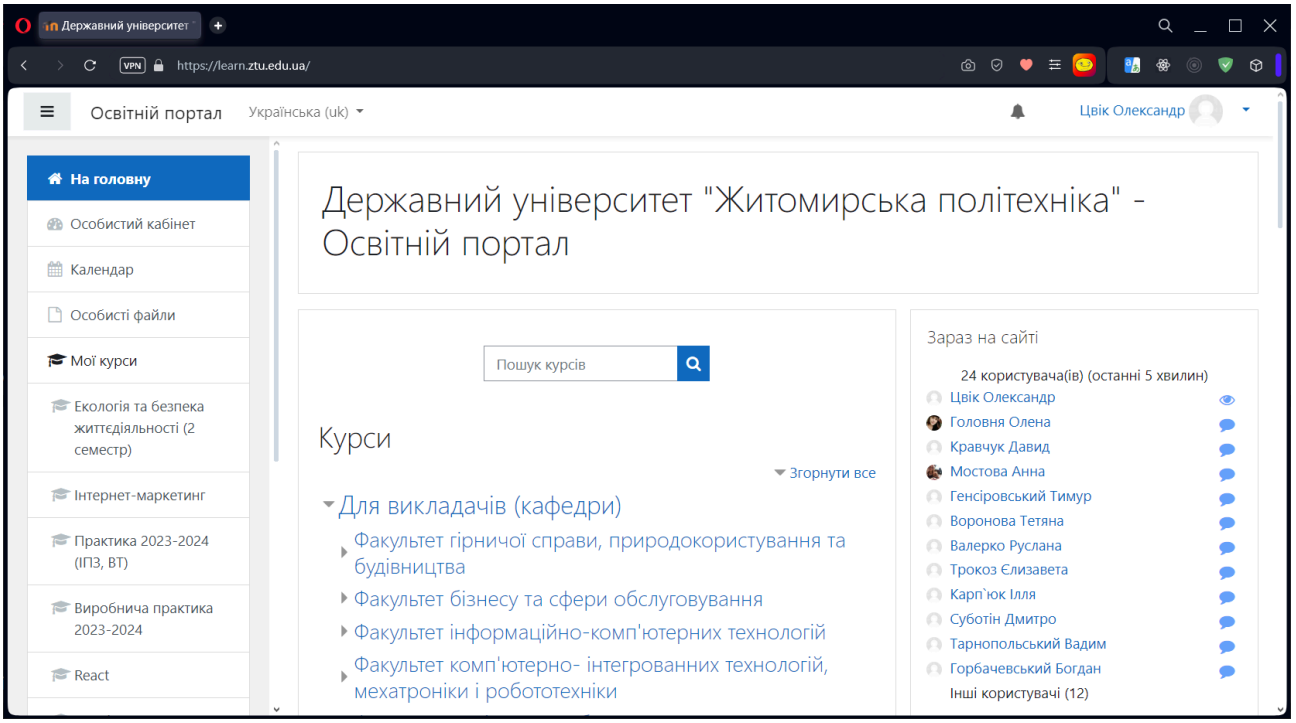

Рисунок 1.1 – Інтерфейс Moodle налаштований для Житомирської політехніки

Однак, на відміну від Google Classroom, ця платформа вимагає більш серйозного підходу і більш глибокого вивчення інструментів роботи.

Moodle повністю безкоштовна платформа, яку можна вільно завантажувати, встановлювати та змінювати. Вона відноситься до Open Source систем, тобто системам з відкритим вихідним кодом, що дозволяє багатьом програмістам створювати додаткові, дуже корисні розширення або модулі.

Основні функціональні можливості Moodle:

– створення курсів та наповнення різноманітними матеріалами;

– можливість публікації навчального контенту різного формату – аудіо, відео, текст, різні формати файлів, каталоги тощо;

– велика кількість інструментів для управління курсами;

– містить багато функціоналу для створення різноманітних тестів;

- містить налаштування варіантів керування доступом користувачів до курсу та розміщених матеріалів;
- відстеження прогресу студентів;
- наявність особистого кабінету, який містить всі дані про користувача;
- користувачі мають можливість обмінюватися повідомленнями;
- перегляд активних користувачів на поточній сторінці.

Оскільки Moodle програма з відкритим вихідним кодом, платформа має велику кількість плагінів та доповнень до системи. Такі доповнення як правило безкоштовні, їх можна просто завантажити і встановити для своєї системи.

Прикладами таких плагінів є:

- модулі відеоконференції;
- аудіо та відео чати;
- масова розсилка повідомлень;
- засоби проектної роботи;
- налаштування зовнішнього вигляду курсів;
- різні інтерактивні елементи гейміфікації та ігри (кросворд, вікторина, мільйонер, судоку та інші);
- формування електронного портфоліо.

1.2.2. Google Classroom

Google Classroom є популярною платформою для електронного навчання, розробленою компанією Google. Цей інструмент значно спрощує процес організації та створення навчальних завдань для вчителів та студентів, забезпечуючи ефективний зворотній зв’язок.

Цей продукт інтегрується з іншими Google-сервісами, такими як Google Диском, Документами, Календарем, Формами та Gmail. У сервісі викладач може коментувати та оцінювати отримані роботи, відсилати їх студенту на доопрацювання та повторно оцінювати після внесення правок. Крім того, викладач має можливість публікувати оголошення, які потім можуть коментувати учасники курсу [3].

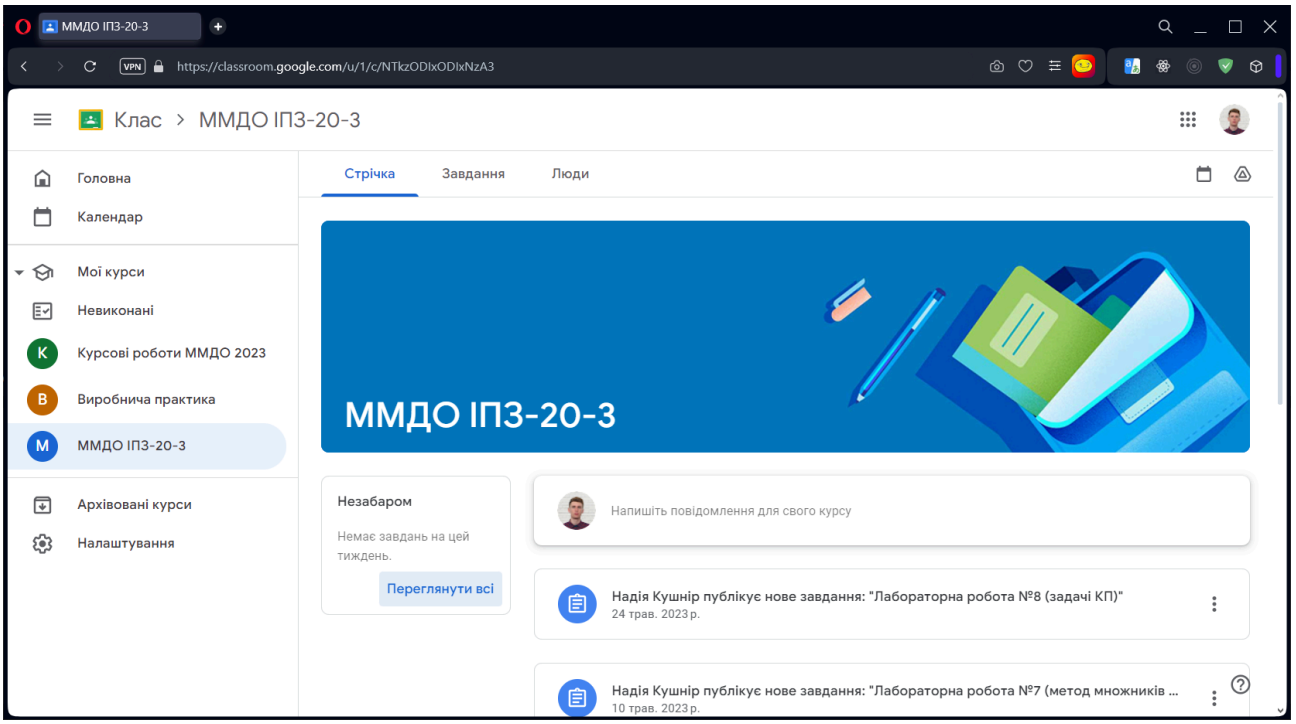

Рисунок 1.2 – Курс в Google Classroom

Google Classroom навряд чи можна назвати класичною системою дистанційного навчання, це скоріше середовище для спільної роботи з можливістю обмінюватися файлами, де Google об’єднали всі необхідні інструменти для ефективної освіти в одному місці. Але дана система відмінно підійде для знайомства з онлайн-навчанням. Для того щоб скористатися функціями даного сервісу, достатньо лише створити обліковий запис в Google. Інтерфейс платформи дуже зручний і простий, тому складнощів у знайомстві з функціями не виникне.

Основні функціональні можливості Google Classroom:

– викладачі можуть створювати курси, додавати студентів і ділитися різними навчальними матеріалами;

– над курсом можуть працювати декілька викладачів;

– можливість створення завдань та тестів;

– оцінювання виконаних студентами завдань і перегляд їх прогресу;

– викладач може залишати коментарі стосовно виконаного завдання;

– легка інтеграція з іншими інструментами Google для зручного обміну документами та іншими матеріалами.

1.2.3. ClassDojo

ClassDojo – це сервіс, який допомагає забезпечити комунікацію між вчителями, учнями та їхніми батьками. Дана платформа дає змогу максимально відтворити шкільне середовище вдома й намагається зацікавити дітей шкільного віку за допомогою гейміфікації та заохочувальних бейджів. Кожен учень на платформі має свою власну анімовану аватарку, яка «радіє», коли отримано похвалу від учителя і «сумує», коли ставлять негативну оцінку.

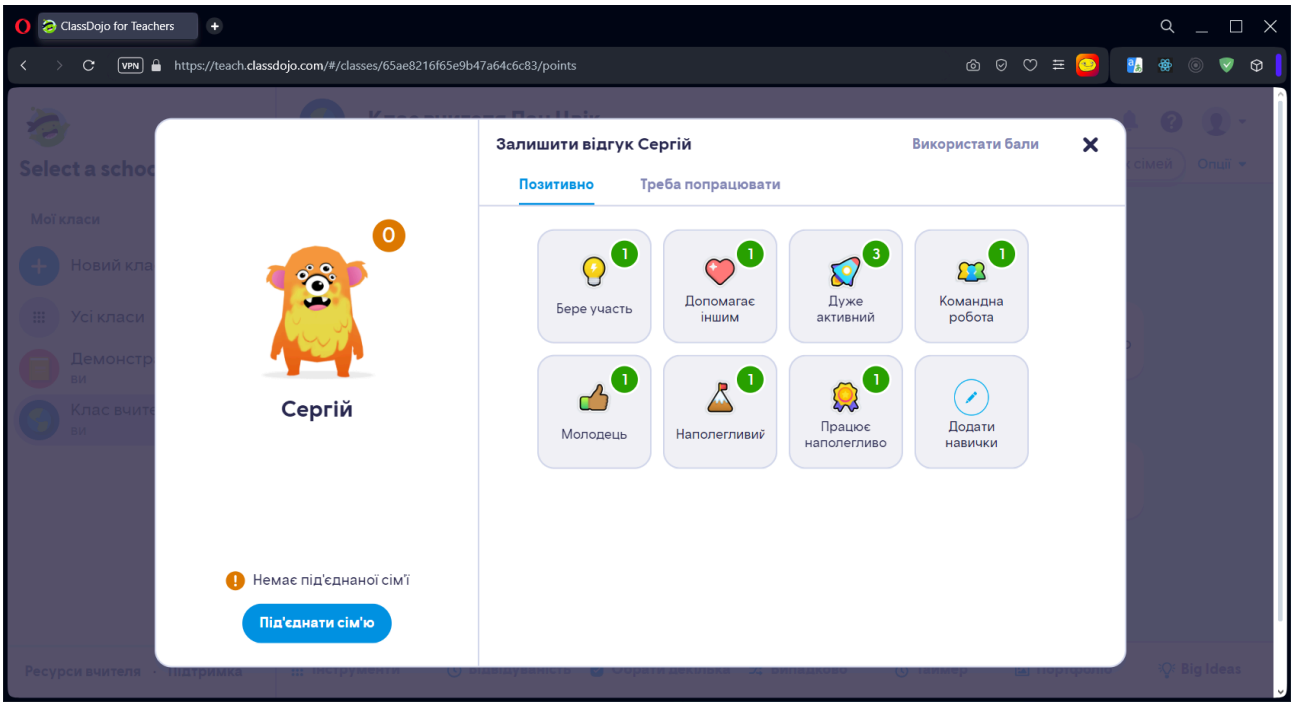

Рисунок 1.3 – Можливість залишити відгук учню в ClassDojo

Заохочувальні бейджи можуть давати як позитивні оцінки, так і негативні з закликом до праці (див. рис. 1.3). Також вчителі можуть створювати нові бейджи та визначати скільки балів вони будуть додавати чи віднімати учню.

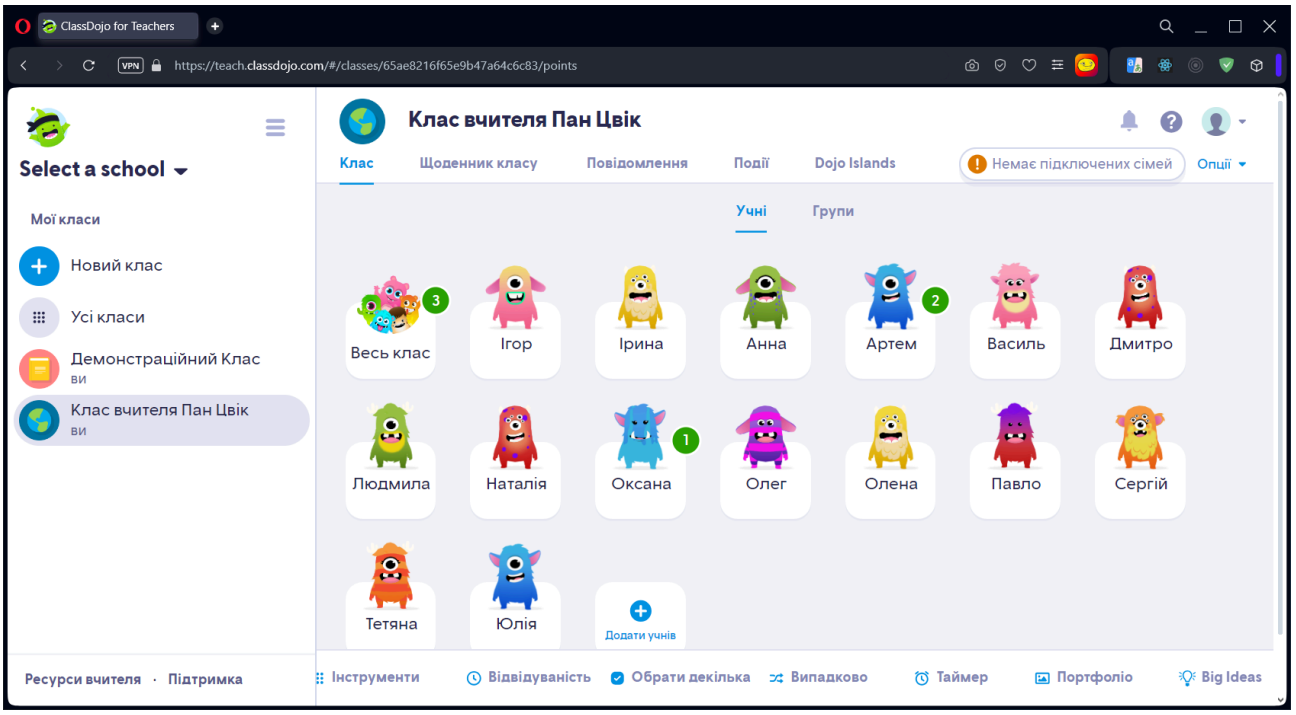

Рисунок 1.4 – Інтерфейс класу в ClassDojo

Основні функціональні можливості ClassDojo:

– легко запрошувати учнів і їх батьків, за допомогою згенерованого QR-коду, готового посилання для входу в акаунт або запрошень на електронну пошту;

– яскрава анімація аватарів, яка привертає увагу (див. рис. 1.4);

– стрічка подій для інформування класу;

– створення завдань з різними типами відповідей (текст, відео, фото, малювання, робочий лист);

– створення індивідуальних завдань для кожного учня окремо або спільне завдання для всього класу;

– бейджи з різними зображеннями для позитивних та негативних оцінок;

– генерація статистики прогресу кожного учня і всього класу для обраного періоду;

– батьки можуть переглядати успішність їхньої дитини.

Після проведеного аналізу аналогів систем дистанційного навчання, було виконано порівняння їх основних функціональних можливостей з платформою Elers (див. табл. 1.1).

Таблиця 1.1

Порівняння аналогів систем дистанційного навчання

Характеристика	Moodle	Google Classroom	ClassDojo	Elers
Створення курсів та наповнення матеріалами	+	+	-	+
Створення тестів	+	+	-	+
Створення та оцінювання завдань	+	+	+	+
Перегляд успішності студентів	+	+	+	+
Чат	+	-	+	-
Налаштування доступу до курсу	+	-	-	+
Керувати курсом можуть декілька викладачів	+	+	+	+
Календар подій	+	+	-	-
Батьки мають змогу переглядати успішність своїх дітей	+	+	+	-
Можливість групувати матеріали курсу та налаштовувати відображення кожної окремої вкладки	-	-	-	+
Наявність відзнак за навчання	+	-	+	+
Можливість зміни рамки навколо зображення облікового запису	-	-	-	+
Можливість зміни фону профіля	-	-	-	+

Платформа Moodle має широкий набір інструментів для управління курсами. Цей набір включає можливість публікації різних типів матеріалів курсу, створення тестів та практичних завдань, перегляду успішності студентів на курсі, налаштування доступу до курсу та формування календаря подій. Крім того, за допомогою додаткових налаштувань та плагінів, можна підключити

модуль чату для обміну повідомленнями між учасниками платформи, додати отримання відзнак за навчання, а також створити акаунти батьків, які матимуть доступ до перегляду успішності своїх дітей.

Сервіс Google Classroom має спрощений функціонал для управління курсами. Завдяки інтеграції з Google Формами, викладачі можуть легко створювати тести, а з Google Документами – в реальному часі спостерігати за роботою студентів, переглядати документи, над якими ті працюють, а також виправляти помилки. У сервісі викладачі можуть коментувати виконані завдання студентів та виставляти оцінки, а також відправляти роботу студентів на доопрацювання. Після внесення правок студентом, викладач може повторно оцінити роботу. Щоб додати батьків до курсу, викладачеві необхідно мати обліковий запис Google Workspace for Education, оскільки звичайні акаунти не надають такої можливості.

ClassDojo не пропонує функцію створення курсів та їх наповнення матеріалами. Натомість, він дозволяє користувачам створювати класи, публікувати нові пости та планувати події. Цей сервіс також не підтримує створення тестів, а доступ до класів обмежений лише учасниками, які можуть переглядати опубліковані матеріали. Календар подій також відсутній. Даний сервіс пропонує окремі акаунти для батьків, де вони можуть переглядати успішність своїх дітей та їх виконані роботи. Однією з переваг цього сервісу є гейміфікація, яка реалізована за допомогою мотиваційних бейджів.

За результатами порівняння аналогів проведемо формування функціональних та нефункціональних вимог.

Функціональні вимоги:

- відсутність реєстрації, лише адміністратори можуть реєструвати нових користувачів з різними ролями;
- вхід для студентів і викладачів з різними рівнями доступу;
- створення курсів та можливість додавати різні матеріали (файли, посилання, текстове наповнення);

− можливість завантаження практичних завдань студентами та їх оцінювання викладачем;

− можливість групування матеріалів курсу викладачами за тематичними вкладками;

− створення тестів з різними типами питань (вибір однієї чи декількох правильних відповідей із запропонованих варіантів, відкрита відповідь, встановлення відповідності) та автоматизована перевірка результатів;

− відображення прогресу студентів у навчанні, їхні оцінки та досягнення;

− візуалізація успішності студентів у вигляді графіків та діаграм;

− можливість змінити рамку навколо зображення облікового запису та фон профілю.

Нефункціональні вимоги:

− платформа повинна бути сумісною з різними браузерами та пристроями користувачів;

− мінімалістичний та легкий у використанні інтерфейс;

− інтерфейс платформи повинен мати українську та англійську локалізацію;

− можливість вибрати світлу чи темну тему сайту;

− масштабованість для збільшення кількості користувачів та обсягу навчального матеріалу.

Впровадження елементів гейміфікації позитивно впливає на всіх учасників навчального процесу, адже стимулює їх бути більш активними. З розглянутих аналогів ClassDojo містить мотиваційні бейджі, які отримують учні за виконання або невиконання завдань та за їх поведінку на заняттях. У Moodle існує можливість налаштувати нагородження студентів відзнаками за успіхи у навчанні або, за допомогою плагіну «Level up!», відстежувати рівень проходження курсу студентом. Отже, важливо додати елементи гейміфікації до власної розробки [9].

На платформі Elers буде реалізовано можливість зміни рамки навколо профільного зображення та фону профілю за накопичені рейтингові бали, які було отримано під час навчання. У системі будуть зберігатися кілька готових варіантів дизайну, які можна буде придбати та застосувати до свого облікового запису. Для покупки візерунків достатньо буде набрати певну кількість балів та обміняти їх на вибраний візерунок. Візерунки будуть доступні для всіх користувачів, незалежно від їхнього рейтингу.

Підсумовуючи результати дослідження було виділено основні переваги та недоліки розглянутих систем (див. табл. 1.2).

Таблиця 1.2

Переваги та недоліки систем дистанційного навчання

Вебплатформа	Переваги	Недоліки
Moodle	– створення курсів, завдань та тестів; – наявні чати; – налаштування дозволів для користувацьких ролей; – календар подій; – наявність відзнак за навчання; – відкритий вихідний код та велика кількість додаткових плагінів.	– інтерфейс не є інтуїтивно зрозумілим; – вивчення системи займає багато часу.
Google Classroom	– створення курсів, завдань та тестів; – календар подій; – легка інтеграція з іншими сервісами Google.	– відсутні чати; – відсутні відзнаки за навчання; – відсутність налаштування дозволів для користувацьких ролей.
ClassDojo	– створення класів та завдань; – публікація постів та подій; – наявні чати; – анімовані аватари; – мотиваційні бейджи; – швидка реєстрація для учнів та батьків.	– відсутність тестів; – відсутність налаштування дозволів для користувацьких ролей; – відсутній календар подій; – відсутня можливість групувати пости за тематикою.
Elers	– створення курсів, завдань та тестів; – налаштування дозволів для користувацьких ролей; – групування матеріалів курсу по вкладках;	– відсутні чати; – відсутній календар подій; – відсутні акаунти батьків.

	– наявність відзнак за навчання; – зміна рамки навколо зображення облікового запису та фон профілю.	

1.3. Планування та проектування архітектури вебплатформи дистанційного навчання

Для реалізації вебзастосунків використовується клієнт-серверна архітектура. Модель такої системи полягає в тому, що клієнт, використовуючи браузер, відправляє запит на віддалений сервер. На стороні сервера отриманий запит обробляється, формується відповідь і далі готовий результат відправляється клієнтові [4].

Перевагами такого архітектурного стилю є [4]:

- простота в обслуговуванні, якщо один сервер виходить з ладу через ремонт, оновлення або переміщення, це не впливає на роботу клієнта, за умови, що інші, резервні сервери будуть налаштовані та доступні для продовження роботи;
- забезпечує безпеку даних, адже вся інформація зберігається на віддаленому сервері, який має кращий захист, ніж клієнтські пристрої;
- централізований доступ до даних, оскільки дані зберігаються тільки на сервері.

Основний недолік даної архітектури є те, що несправність сервера зробить систему недоступною для користувачів, якщо відсутні резервні сервери. Також, якщо будь-який один модуль матиме більше навантаження ніж інші, то при недостатній потужності сервера з’являються сильні затримки в роботі всієї системи [4].

Даний архітектурний підхід розділяє функціональні обов’язки між клієнтською та серверною частинами.

Функції, які реалізуються на сервері:

- зберігання, доступ, захист і резервне копіювання даних;
- обробка клієнтського запиту;

− відправлення результату (відповіді) клієнту у вигляді вебсторінки (html) або іншого файлу (json, зображення тощо).

Функції, які реалізуються на стороні клієнта:

− надання користувальницького інтерфейсу;

− формулювання запиту до сервера і його відправка;

− отримання та відображення результатів запиту.

Для розробки платформи дистанційного навчання буде використано шаблон проектування Модель-Представлення-Контролер (англ. Model-View-Controller, MVC). Цей шаблон є ідеальним вибором для створення вебдодатків, оскільки дозволяє відокремити інтерфейс користувача, тобто HTML-розмітку, від бізнес-логіки системи [11].

Основна концепція шаблону MVC полягає в розподілі відповідальності, де кожна частина архітектури MVC є чітко визначеною та автономною. Цей шаблон розділяє систему на три взаємопов’язані складові (див. рис. 1.5): модель даних, представлення даних та контролер, який керує обміном даних між ними. Кожна складова виконує лише свої визначені функції [11]:

− Модель відповідає за роботу з даними. Вона може бути простою моделлю представлення, яка лише передає дані між представленням і контролером, або моделлю предметної області, яка містить бізнес-дані, а також операції та правила для їх обробки.

− Представлення відповідає за візуалізацію даних, зазвичай частини моделі, у формі користувацького інтерфейсу.

− Контролер обробляє запити користувачів, виконує операції з моделлю і вибирає відповідне представлення для оновлення інтерфейсу користувача.

Кожен з цих компонентів має чітко визначені обов’язки та виконує лише певні функції, що забезпечує легкість у супроводі та гнучкість розробки.

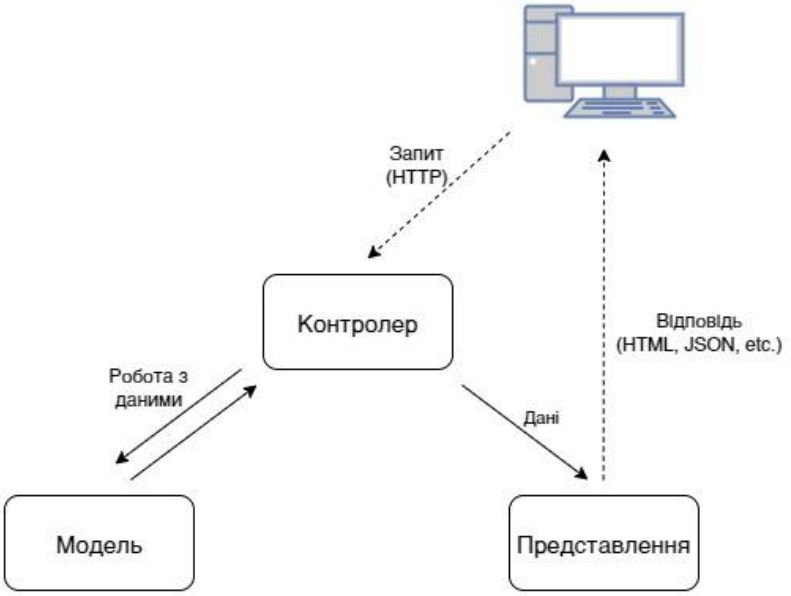

Рисунок 1.5 – Схема MVC шаблону

Щоб прискорити та спростити розробку системи буде використано монолітну архітектуру. Цей підхід зазвичай використовується у невеликих командах та проектах з невеликим навантаженням. Монолітна архітектура має простий процес деплою на сервер, оскільки весь код розглядається як єдина програма. Однак, це є й головним недоліком, оскільки він впливає на можливість масштабування та може призвести до єдиної точки відмови, що може вивести всю програму з ладу [12].

Для проектування серверної частини доцільно використати основні принципи чистої архітектури Роберта Мартіна для розділення програмного забезпечення на різні рівні. Головним правилом, що приводить цю архітектуру в дію, є правило залежностей (англ. Dependency Rule): залежності у вихідному коді мають бути спрямовані лише всередину (див. рис. 1.6) [10, с. 215].

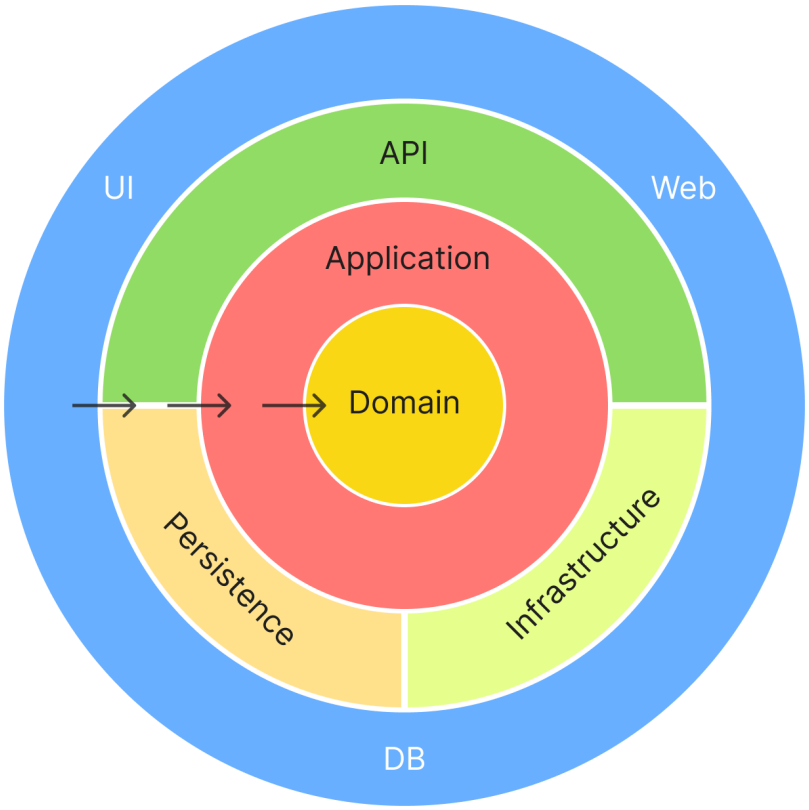

Рисунок 1.6 – Схема слоїв чистої архітектури

Усе, що перебуває у внутрішньому колі, взагалі не може нічого знати про зовнішні кола. Наприклад, імена, оголошені в зовнішніх колах, не повинні згадуватись в коді, що міститься у внутрішніх колах. Сюди належать функції, класи, змінні та будь-які інші іменовані елементи програми [10, с. 215-216].

Усі головні компоненти системи чітко розділені та інкапсульовані, утворюючи кілька шарів з односторонніми залежностями:

1. Domain. Компонент Domain – це фундамент системи, навколо нього будується весь проект. Цей шар немає жодних залежностей від зовнішніх і містить лише сутності, перерахування, константні дані і, можливо, деякі спеціальні винятки на рівні Domain.

2. Application. Разом із доменним рівнем (Domain) прикладний рівень (Application) утворює ядро рішення, яке повинно мати можливість працювати та забезпечувати бізнес-логіку незалежно від зовнішніх рівнів і залежить виключно від Domain. Він створює об’єкти сутностей, виконує бізнес-правила та повертає результати для API.

3. Infrastructure. Це рівень, у якому має бути реалізована вся логіка зв'язку із зовнішніми системами, як-от надсилання електронних листів, зв'язок із стороннім API тощо. Він залежить лише від Application та реалізує його інтерфейси.

4. Persistence. Порівняно з Infrastructure, рівень Persistence також містить логіку для зв'язку із зовнішніми системами, але його конкретна мета – зв'язуватися з базами даних. Цей рівень зазвичай залежить від Application та Domain.

5. API. Це рівень взаємодії із зовнішнім світом, який дозволяє клієнтам отримувати видимі результати після запиту даних. Цей рівень може бути у будь-якій формі, наприклад: веб-API, консольна програма, графічний інтерфейс тощо. Він також залежить лише від Application, але також може використовувати елементи з Persistence. Наприклад, при першому запуску програми потрібно завантажити початкові дані (англ. seed data) в БД, які зберігаються в Persistence. Він не містить бізнес-логіки, його єдине завдання – перетворити запити користувача на зрозумілу для Application форму та повернути отримані результати у зручному форматі для компонента Web.

6. Web. Шар Web на самому верхньому рівні та відповідає за реалізацію клієнтської частини, яка може включати в себе HTML, CSS, JavaScript та будь-які інші фронтенд технології для створення вебінтерфейсу. Користувачі можуть використовувати даний компонент для взаємодії з системою за допомогою HTTP запитів.

Застосування підходу чистої архітектури з її чітко визначеними компонентами у монолітному проєкті може спростити його майбутнє перенесення на мікросервісну архітектуру. Мікросервісна архітектура забезпечує більш зручну роботу для великої команди розробників, легше масштабується та дозволяє використовувати різні мови програмування для кожного окремого сервісу. Головними її недоліками є складність проектування та деплою на сервер, що вимагає більше часу та ресурсів порівняно з монолітною архітектурою [12].

Клієнтську частину, яка відповідає за складний інтерфейс з багатьма інтерактивними елементами, такими як таблиця успішності або інструменти створення та проходження тестів, варто реалізувати у вигляді односторінкового додатку (англ. Single Page Application, SPA). Завдяки цьому підходу, користувач отримує швидке та плавне відображення змін інтерфейсу на його дії. Крім того, технології для розробки SPA мають більше можливостей ніж звичайний JavaScript, HTML та CSS.

SPA – це вебдодаток, який завантажує сторінку лише один раз та динамічно змінює її вміст без необхідності перезавантаження всієї сторінки. SPA використовує підхід AJAX (англ. Asynchronous JavaScript and XML) для обміну даними з сервером та оновлення лише необхідних частин сторінки. SPA забезпечує такі переваги, як [13]:

1. Постійна взаємодія з користувачем, за допомогою динамічної зміни контенту й без завантаження нової сторінки з сервера. Це дає змогу зменшити навантаження на сервер, адже серверу достатньо відправити дані у JSON форматі, а не генерувати всю HTML сторінку.
2. При завантаженні нових модулів (сторінок) контент на них оновлюється лише частково, тому що немає необхідності повторно завантажувати елементи, які не було змінено.
3. Розробникам доступні різні фреймворки та бібліотеки, які спрощують створення архітектури проєкту та дають чимало готових елементів для роботи.

Односторінкові додатки мають і деякі недоліки, які можна виправити шляхом використання сторонніх інструментів та проведення додаткової оптимізації [13]:

1. Проблеми з індексацією динамічного контенту, тобто складнощі з налаштуванням SEO.
2. Тривале завантаження під час першого відвідування, через велику кількість скриптів, які генерують контент на сторінці.
3. Важливо, щоб у користувачів було увімкнено підтримку JavaScript у браузері.

Після того, як було обрано архітектуру платформи, необхідно розробити її діаграму компонентів. Розуміння точної поведінки кожної частини програмного забезпечення є важливим для будь-якого програміста. Діаграми компонентів можуть допомогти іншим розробникам зрозуміти структуру конкретної системи. Крім того, ці діаграми можуть описувати програмні системи, реалізовані будь-якою мовою чи стилем програмування.

Головна мета цих діаграм – показати взаємозв’язок між різними компонентами в системі. Тут «компоненти» можуть означати модулі класів, які представляють незалежні системи або підсистеми з можливістю взаємодії з рештою системи, також у ролі компонента можуть виступати файли, бібліотеки, модулі, виконувані файли, пакети тощо [14].

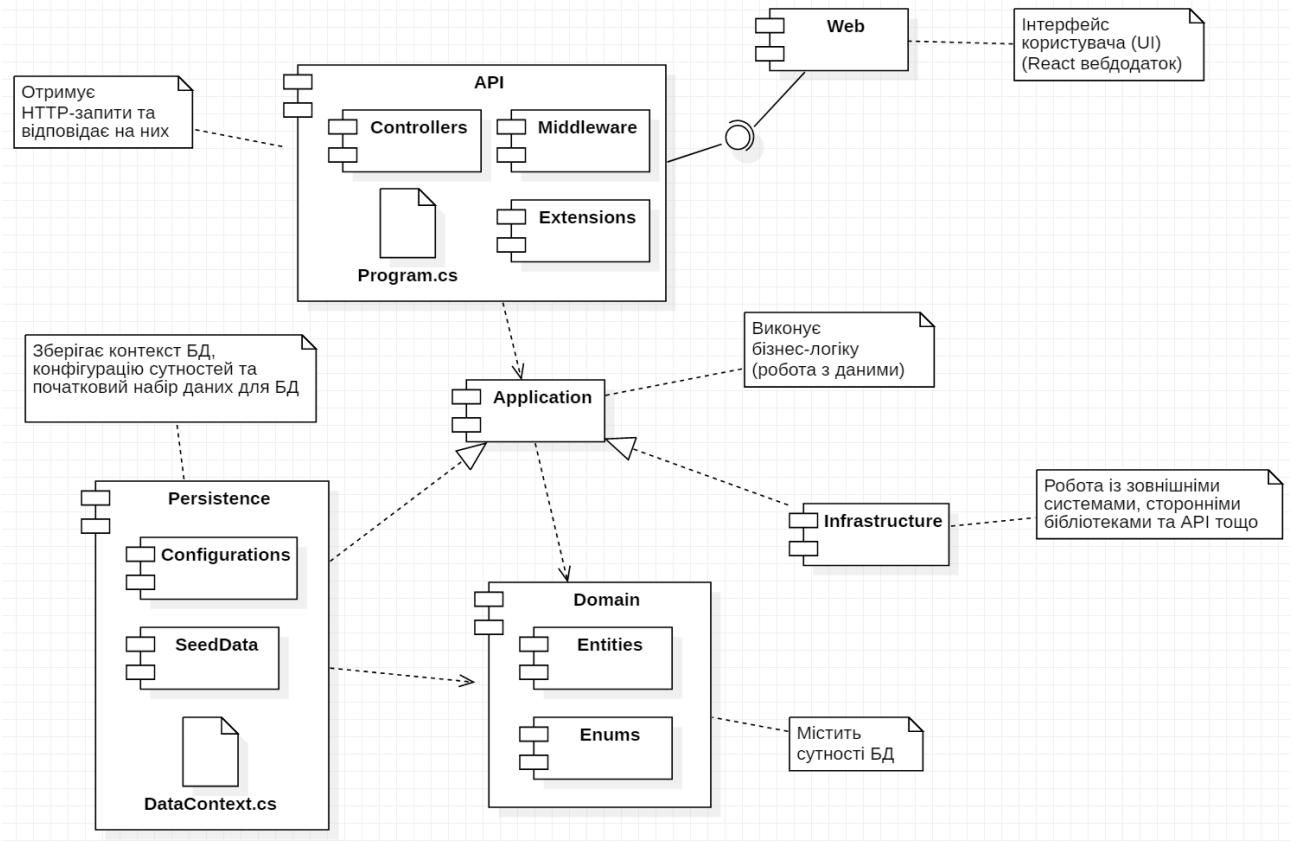

Рисунок 1.6 – Діаграма компонентів

Діаграма компонентів зображена на рис. 1.6 показує загальну структуру платформи дистанційного навчання. Для зручності сприйняття діаграми компонентів, до кожного компонента було додано відповідні нотатки.

Компонент «Web» зв’язаний з компонентом «API» через інтерфейс тому, що взаємодія між ними відбувається через певний набір методів HTTP запитів. Ці методи визначають спосіб передачі даних та обмін інформацією між компонентами, що дозволяє взаємодіяти без прив’язки до конкретних реалізацій методів.

Компонент «API» використовує компонент «Application» для доступу до даних та основної функціональності системи. В свою чергу «Application» використовує «Domain», де зберігаються класи сутностей, що відповідають таблицям в базі даних. Компоненти «Persistence» та «Infrastructure» реалізують різні інтерфейси компонента «Application» для роботи з контекстом БД та зовнішніми системами відповідно. Також «Persistence» використовує сутності з «Domain» для того, щоб провести конфігурацію БД.

Компонент «Web» відповідає за представлення (View) з шаблону MVC. У компоненті «API» зберігаються контролери (Controller), а компоненти «Application», «Domain», «Persistence» та «Infrastructure» містять правила для маніпулювання даними (Model).

1.4. Обґрунтування використання технологій та засобів

Для серверної частини обрано **ASP.NET Core** – це кросплатформний фреймворк для створення вебзастосунків на платформі .NET з відкритим вихідним кодом, де в якості мови програмування можна використовувати C#. Має всі необхідні інструменти для роботи сучасного вебдодатку: маршрутизація, конфігурація, логування, можливість роботи з різними системами баз даних, налаштування авторизації, інтеграція зі сторонніми API та інші. Він пропонує ефективний підхід до розробки програми, спеціально орієнтований на роботу в стилі REST (англ. Representation State Transfer, передача стану уявлення).

Проектування бази даних буде виконано за допомогою ORM (англ. object-relational mapping, реляційне відображення об’єктів), тобто відображення даних на реальних об’єктах, **Entity Framework Core** (EF Core) та підходу Code

First. Тобто спочатку потрібно написати код на мові програмування C#, а потім з цього коду буде автоматично згенеровано таблиці для бази даних. Для цього підходу дуже важливо визначити класи сутностей, що будуть відповідати таблицям, які будуть зберігатися в базі даних.

Конфігурацію таблиць зручно налаштовувати за допомогою **Fluent API**, адже він надає більший діапазон параметрів конфігурації, ніж атрибути анотації даних. Конфігурація Fluent API також сприяє чистому коду, оскільки конфігурацію можна зберігати в окремих файлах.

EF Core підвищує ефективність та швидкість розробки, оскільки тепер програмісти мають справу не з таблицями та полями, а з класами та об’єктам, що дозволяє не писати складних SQL-запитів. Даний фреймворк підтримує міграції бази даних, що дозволяє легко вносити зміни в схему бази даних та оновлювати дані в базі даних, не втрачаючи існуючої інформації. Це особливо корисно при розробці проекту, коли схема бази даних може змінюватися. EF Core є одним з популярних ORM-фреймворків для .NET розробників, адже підтримує різні СУБД, включаючи PostgreSQL, MySQL, SQLite, Oracle, MongoDB та інші.

Бібліотека **MediatR** – спрощує реалізацію патернів CQRS та Mediator. Вона дозволяє розділити відповідальність між контролерами та обробниками команд і запитів, що робить код більш структурованим та гнучким.

FluentValidation – це бібліотека .NET для створення типізованих правил валідації даних. Допомагає уникнути помилок та виконання некоректних операцій з даними.

Обрано **PostgreSQL** як об’єктно-реляційну систему керування базами даних, яка має високу продуктивність та підтримує широкий спектр функцій, включаючи зберігання процедур, тригери, індекси, віртуальні таблиці (view) та налаштування обмежень (constraints). Гарантує цілісність даних завдяки ACID-транзакціям, що робить її безпечним вибором для зберігання важливої інформації про користувачів та курси.

MongoDB обрано як NoSQL базу даних, яка забезпечує високу швидкість та продуктивність при роботі з великими обсягами даних. Надає горизонтальне масштабування, що дозволяє практично необмежено збільшувати обсяги зберігання, що чудово підходить для створення платформи дистанційного навчання, адже майже щодня буде створюватися велика кількість нових записів про відвідуваність та успішність студентів, проходження тестів та завантаження виконаних практичних робіт.

Для реалізації клієнтського SPA додатку обрано бібліотеку **React**, яка дозволяє повторно використовувати раніше написані компоненти та спрощує процес створення динамічних вебсторінок. Наприклад, таблиці з успішністю студентів та інструменти створення та проходження тестів повинні мати багато динамічних елементів для зручного відображення та оновлення даних. Крім того, React краще використовувати разом з мовою програмування **TypeScript**, адже строга типізація призводить до написання більш надійного коду та прискорює розробку.

Взято **Redux Toolkit** для керування станом додатку й **RTK Query** для відправки запитів до сервера. Бібліотеку компонентів **Ant Design** для стилізації компонентів, а також додатково бібліотеку **date-fns** – для роботи з датами та часом, й відображення їх на сторінці у різних форматах.

Redux є популярною бібліотекою управління станом для React додатків. Його використання дозволяє легко управляти станом додатка, адже Redux зберігає стан додатку в одному об’єкті, який можна легко оновлювати за допомогою виклику функції dispatch. Особливо його зручно використовувати, коли однаковим та актуальним станом потрібно користуватися у різних компонентах або коли є необхідність у взаємодії між кількома компонентами.

Наприклад, у стані додатка можна зберігати інформацію про автентифікованого користувача, для того щоб відкривати лише ті частини сайту до яких користувач має доступ. При першому завантаженні сторінки відбувається запит до сервера для отримання даних про поточного користувача,

поки користувач чекає можна показати індикатор завантаження, компонент з бібліотеки Ant Design.

Для навігації та маршрутизації в React додатках використовується бібліотека **React Router** (react-router-dom). Вона дозволяє легко визначати шляхи, обробляти параметри URL і ефективно керувати навігацією між сторінками. Має можливість гнучкого налаштування перенаправлення користувачів на інші сторінки, якщо вони намагаються перейти до неіснуючої сторінки або сторінки, яка вимагає авторизації.

Щоб покращити якість написання коду варто використати **Prettier** та **ESLint** – це інструменти, які допомагають дотримуватися загальних стандартів у процесі розробки програмного забезпечення. Prettier використовується для форматування коду, працює за допомогою правил, які можна налаштувати відповідно до потреб проєкту. ESLint допомагає виявити помилки у коді, підтримує широкий спектр правил, включаючи правила стилю, продуктивності та інші.

У якості середовища розробки обрано **Visual Studio Code** (VS Code). Цей редактор швидко аналізує структуру коду, показує підказки під час написання коду, підтримує автодоповнення і рефакторинг, що сприяє уникненню помилок та прискорює процес розробки. Має великий список доступних розширень, які дозволяють зручно працювати з великим переліком технологій, таких як C#, JavaScript, TypeScript, HTML, CSS тощо. Крім того, за допомогою VS Code можна легко та зручно відлагоджувати власний код.

Висновки до першого розділу

Проаналізовано вимоги до програмного продукту та предметної області. Визначено завдання, які необхідно вирішити для успішної реалізації проєкту.

Після дослідження аналогів Moodle, Google Classroom та ClassDojo, було визначено їхні ключові функціональні можливості та створено порівняльну таблицю у якій перераховано переваги та недоліки. На основі цього аналізу

було сформовано функціональні та нефункціональні вимоги до майбутньої системи.

Для реалізації системи обрано монолітну архітектуру, в якій клієнтську частину буде реалізовано у вигляді SPA додатку, а сервер буде використовувати чисту архітектуру Роберта Мартіна.

Використання SPA дозволить забезпечити користувачам зручний, динамічний та швидкий інтерфейс, зменшуючи навантаження на сервер та забезпечуючи більш ефективну обробку запитів. Застосування чистої архітектури на сервері дозволяє розділити компоненти системи на незалежні шари, що полегшить тестування, розширення та підтримку коду.

Було обрано основні інструменти для розробки серверної частини, такі як ASP.NET Core, Entity Framework Core, MediatR та FluentValidation. У якості СКБД обрано PostgreSQL та MongoDB, а VS Code як середовище розробки. Головні технології для реалізації клієнтського додатку: React, TypeScript, Redux Toolkit, RTK Query, React Router, Ant Design, date-fns, Prettier та ESLint.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1.Платформи та сервіси дистанційного навчання [Електронний ресурс] – Режим доступу до ресурсу: https://www.krok.edu.ua/ua/pro-krok/pidrozdili/navchalni/tsentr-distantsijnogo-navchannya/platformi-ta-servisi-distantsijnogo-navchannya.

2.Що таке Moodle [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: https://moodle.org/mod/page/view.php?id=8174.

3.Ткаченко Л. В. ОСОБЛИВОСТІ ВПРОВАДЖЕННЯ ДИСТАНЦІЙНОГО НАВЧАННЯ В ОСВІТНІЙ ПРОЦЕС ЗАКЛАДУ ВИЩОЇ ОСВІТИ [Електронний ресурс] / Л. В. Ткаченко, О. С. Хмельницька. – 2021. – Режим доступу до ресурсу: http://pedagogy-journal.kpu.zp.ua/archive/2021/75/part_3/20.pdf.

4.КОВТУН Л. О. ВИБІР АРХІТЕКТУРИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ІНФОРМАЦІЙНОЇ НАВЧАЛЬНОЇ СИСТЕМИ [Електронний ресурс] / Л. О. КОВТУН, Р. ФРАНЧУК, В. М. ТКАЧУК // Вісник Хмельницького національного університету. – 2019. – Режим доступу до ресурсу: http://journals.khnu.km.ua/vestnik/wp-content/uploads/2021/01/42-6.pdf.

5.Classdojo [Електронний ресурс] – Режим доступу до ресурсу: https://www.classdojo.com/uk-ua/.

6.Google Classroom [Електронний ресурс] – Режим доступу до ресурсу: https://classroom.google.com/.

7.Moodle [Електронний ресурс] – Режим доступу до ресурсу: https://moodle.org/.

8.Жаврук Н. В. ВИКОРИСТАННЯ ТЕХНОЛОГІЇ REST ДЛЯ ПОБУДОВИ ВЕБСЕРВІСІВ [Електронний ресурс] / Н. В. Жаврук – Режим доступу до ресурсу: http://eprints.zu.edu.ua/21125/1/ZhavrukAPSI2016.pdf.

9.Гришуніна М. В. ГЕЙМІФІКАЦІЯ ДИСТАНЦІЙНОГО КУРСУ ЗАСОБАМИ MOODLE [Електронний ресурс] / М. В. Гришуніна – Режим доступу до ресурсу: https://2020.moodlemoot.in.ua/course/view.php?id=14.

10. Мартін Р. Чиста архітектура: Мистецтво створення програмного забезпечення / Роберт Мартін. – Харків: Фабула, 2019. – 368 с. – (Видання друге).

11. Глабець І. Використання архітектурного шаблону MVC при проектуванні програмного забезпечення [Електронний ресурс] / І. Глабець. – 2013. – Режим доступу до ресурсу: https://api.core.ac.uk/oai/oai:elartu.tntu.edu.ua:123456789/9470.

12. Боков О. Г. Аналіз архітектури сучасних ВЕБ-додатків [Електронний ресурс] / О. Г. Боков – Режим доступу до ресурсу: https://openarchive.nure.ua/handle/document/21107.

13. Плоха О. Б. ОСНОВНІ ПЕРЕВАГИ ТА НЕДОЛІКИ ОДНОСТОРІНКОВИХ ТА БАГАТОСТОРІНКОВИХ ВЕБ-ДОДАТКІВ. Тези доповідей, 8. [Електронний ресурс] / О. Б. Плоха, А. Ю. Верещака. – 2020. – Режим доступу до ресурсу: https://it.hneu.edu.ua/wp-content/uploads/2021/10/tezy-dopovidej-mizhnarodnoyi-naukovo-praktychnoyi-konferencziyi-informaczijni-tehnologiyi-ta-systemy-2020.pdf#page=8.

14. Єфремов М. Ф. ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ З ВИКОРИСТАННЯМ UML [Електронний ресурс] / М. Ф. Єфремов, Ю. М. Єфремов, В. М. Єфремов. – 2016. – Режим доступу до ресурсу: http://eztuir.ztu.edu.ua/123456789/3296.

15. Савельєв Ю. Б. Аналіз порівняльний [Електронний ресурс] / Ю. Б. Савельєв, В. В. Чепак // Велика українська енциклопедія. – 2019. – Режим доступу до ресурсу: https://vue.gov.ua/Аналіз_порівняльний.

16.