МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХЕРСОНСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
Факультет комп'ютерних наук, фізики та математики
Кафедра інформатики, програмної інженерії та економічної кібернетики

Створення програмної системи підтримки проведення лабораторних робіт з програмування в умовах дистанційного навчання

Кваліфікаційна робота (проєкт)
на здобуття ступеня вищої освіти "магістр"

Виконав:здобувач 2 курсу, 261М групи
Спеціальності 121 Інженерія програмного забезпечення
Освітньо-професійної програми
«Інженерія програмного забезпечення»
другого (магістрського) рівня освіти
Болокан Владислав Анатолійович
Керівники: доктор фізико-математичних наук, професор
Львов Михайло Сергійович
кандидат фізико-математичних наук,
доцент Єрмолаєв Вадим Анатолійович
Рецензент: професор
Кузьменков С.Г.

Херсон – 2021

ЗМІСТ

ВСТУП

Актуальність дослідження. Розвиток глобальної комп’ютерної мережі Інтернет відкрив нові перспективи еволюційного вдосконалення світової системи освіти. Сьогодні традиційні методи освіти доповнюються новими методами навчання на основі, електронних комп’ютерних мереж та телекомунікацій. Дистанційна освіта та навчання телебаченню, засновані на Інтернет-технологіях, виконують багато нових функцій, що передбачають реалізацію певних принципів, серед яких принципи розподіленої співпраці, інтеграції та приєднання до глобальної мережевої спільноти є дуже важливими. За сучасних умов необхідно сформувати гнучку розподілену систему навчання протягом усього життя. За допомогою цієї системи людина може виконувати лабораторні роботи.

Об’єкт дослідження: Дистанційне навчання з програмування.

Предмет дослідження: Технології створення дистанційної системи.

Мета дослідження: Полягає у визначені способів та засобів реалізації програмної системи для підтримки проведення лабораторних робіт з програмування в умовах дистанційного навчання, та створенні самої системи.

У зв'язку з поставленою метою були визначені **завдання**:

1. Проаналізувати існуючі системи та зробити висновки над їх актуальністю
2. Розглянути детально готові рішення
3. Знайти найбільш ефективні інструменти для реалізації продукту
4. Розглянути бібліотеки для розробки
5. Спроєктувати серверну та клієнтську частини сайту.

Структура роботи: Робота складається зі вступу, трьох розділів, висновків, списку використаних джерел.

РОЗДІЛ 1
ПОНЯТТЯ ДИСТАНЦІЙНОГО НАВЧАННЯ ТА ОФОРМЛЕННЯ ЙОГО ІНТЕРФЕЙСУ

1.1. Аналіз систем дистанційного навчання

Дистанційне навчання це – сукупність технологій, що забезпечують зв'язок з викладачем та студентом використовуючи ІКТ(інформаційно-комунікаційні технології)

LMS (Система управління навчанням) - це програмне забезпечення, яке дозволяє створювати онлайн -курси, керувати курсами та навчати людей, дозволяючи користувачам отримувати доступ до матеріалів, тестів тощо. Подібно до соціальної мережі, яка дозволяє керувати своїм профілем, публікувати фотографії та спілкуватися, LMS дозволяє керувати освітою. Ви можете відкрити доступ до навчального контенту в потрібному порядку, перевірити та відстежити знання учнів і навіть створити власні критерії для результатів оцінювання. Якщо ви ніколи не використовували систему управління навчанням, саме час вивчити цей інструмент. Багато програмних рішень LMS впроваджуються у найвідоміших та найуспішніших компаніях світу. В рамках корпоративного навчання. Справа в тому, що ці системи є не лише новою тенденцією, але й інструментом, який може збільшити прибуток, заощадити час та полегшити життя керівництва та співробітників.

Якщо переглянути платформу LMS то існує два основних типи таких систем: серверна та хмарна. Давайте зрозуміємо різницю між ними. Сервер LMS встановлюється на сервері власника і працює з користувачем через браузер. Це перший тип LMS, який з’явився в університетах і все ще потрібен. Це також вибір власників великих компаній, які цінують повний контроль над системною та інформаційною безпекою. Недоліком системи

віддаленого навчання є вартість придбання серверів, програмного забезпечення та управління. Хмарні рішення - це найпоширеніший тип платформи дистанційного навчання. Через їх простоту та простоту використання власники приватних онлайн - шкіл та керівники компаній обирають їх для покращення корпоративного навчання. За допомогою цього методу немає необхідності купувати сервер та керувати ним, оскільки весь вміст зберігається у «хмарі», і вам потрібно лише сплатити оренду.

1.2. LMS та СДН

LMS і СДН поняття хоч і схожі, але невеликі відмінності є. Давайте розберемося докладніше.

LMS або Система управління навчанням - додаток для адміністрування навчальних курсів в рамках взаємодії між викладачами та учнями між собою на відстані. Це програмне рішення необхідно для планування і управління всіма навчальними заходами в організації.

Мета - управління процесом навчання, відстеження прогресу вивчення матеріалів і розвитку учнів за всіма типами заходів.

Частим помилкою є те, що LMS має на увазі тільки дистанційне навчання. Це не так, вона допомагає організувати всі форми навчання: очну, дистанційну, індивідуальну і групову. Це можна здійснити завдяки тому, що за допомогою LMS можна організовувати різні форми комунікації. Наприклад: онлайн-трансляції, форуми, чати і розсилки.

СДН або Система дистанційного навчання - сукупність організаційних, телекомунікаційних та педагогічних ресурсів, що забезпечують залучення в створення і практичне здійснення освітніх програм з використанням технології дистанційного навчання (ДН).

ДН - процес взаємодії викладача та учнів між собою на відстані, що містить всі властиві навчальному процесу компоненти:

- мета
- зміст
- методи
- засоби навчання

Мета СДН - систематизація та організація взаємодії викладачів і учнів за допомогою інтернет-технологій.

Важливим фактом є те, що дистанційне навчання виключає очні формати, звідси і нюанс в значеннях. Тому що LMS дозволяє проводити курси не тільки віддалено, але і офлайн.

1.3. Огляд сучасних LMS-систем

Тепер, коли ми розумієте, що таке система управління навчанням, ми хочемо познайомити вас з найбільш популярними LMS системами і розповісти про їх основні переваги та недоліки.

1. eTutorium LMS

Хмарний сервіс для організації дистанційного навчання з конструктором тестів і опитувань, вбудованої вебінарной платформою і інструментами мотивації. eTutorium відмінно підходить для проведення вебінарів, нарад, тренінгів та інших форм навчання.

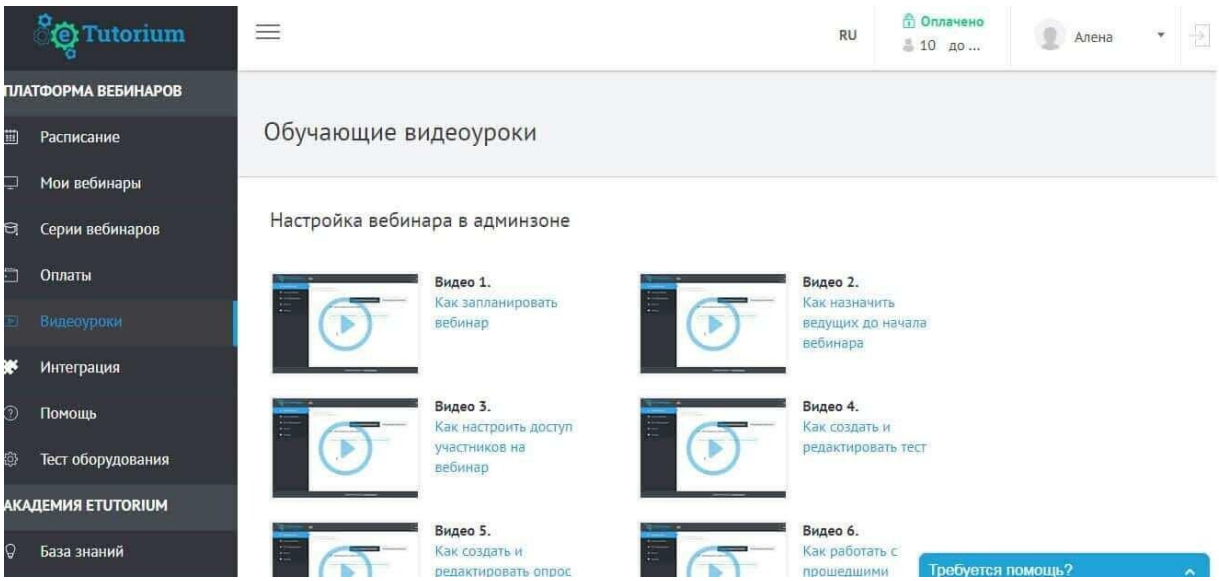

Рис. 1.1 eTutorium система

Плюси: необмежений простір для зберігання, сертифікати для друку.
Мінуси: немає можливості самостійної кастомізації, обмежена кількість типів користувачів, відсутність версії коробочки

2. iSpring Learn

Це хмарна LMS з простим і зрозумілим інтерфейсом, яка дозволяє швидко запустити дистанційний курс і тестування учнів. Підтримує всі види навчальних матеріалів, вебінари і статистику.[1] Є редактор курсів і зручний додаток, яке дозволяє вчитися прямо зі свого смартфона.

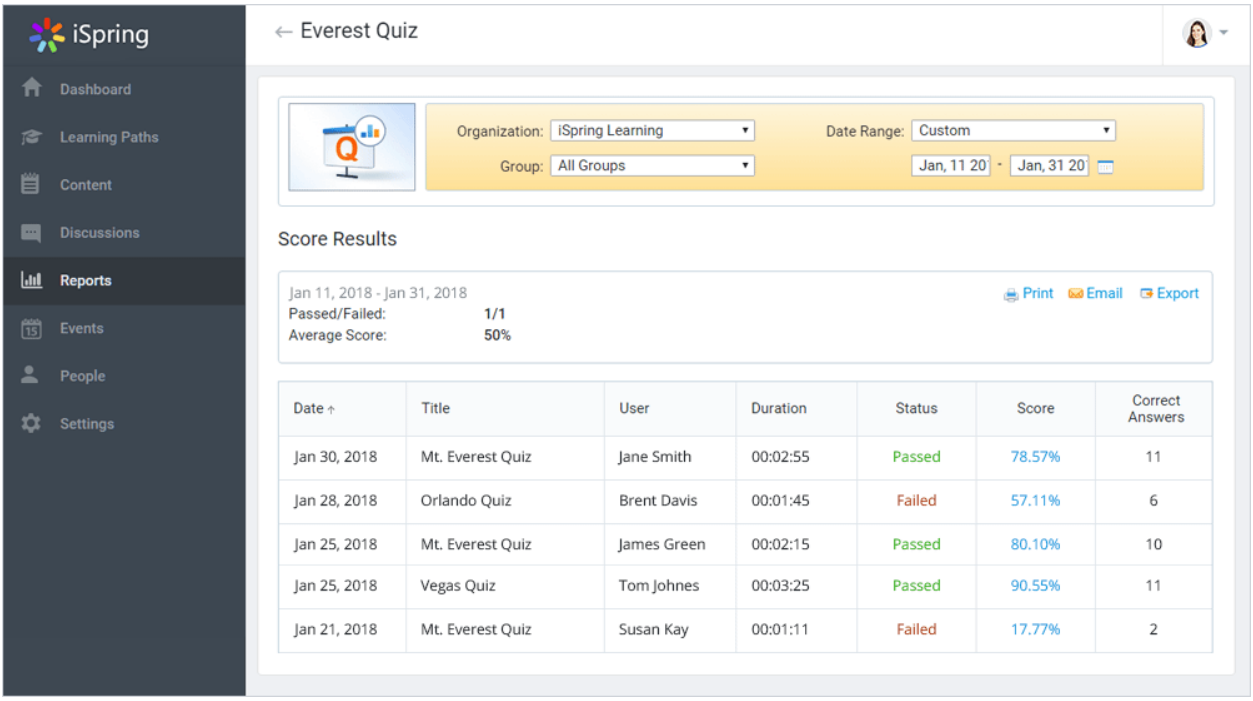

Рис. 1.2 iSpring Learn система

Плюси: необмежений простір для зберігання, сертифікати для друку. Мінуси: немає можливості самостійної кастомізації, обмежена кількість типів користувачів, відсутність версії коробочки.

3. GuruCan

Платформа створена з метою створення і продажу онлайн-курсів. Доступна на двох мовах - англійській і російській. [2] Ставши користувачем системи, ви можете проводити вебінари, автоматизувати маркетингові кампанії, залучати учнів за допомогою гейміфікації, перевіряти домашні завдання і багато іншого.

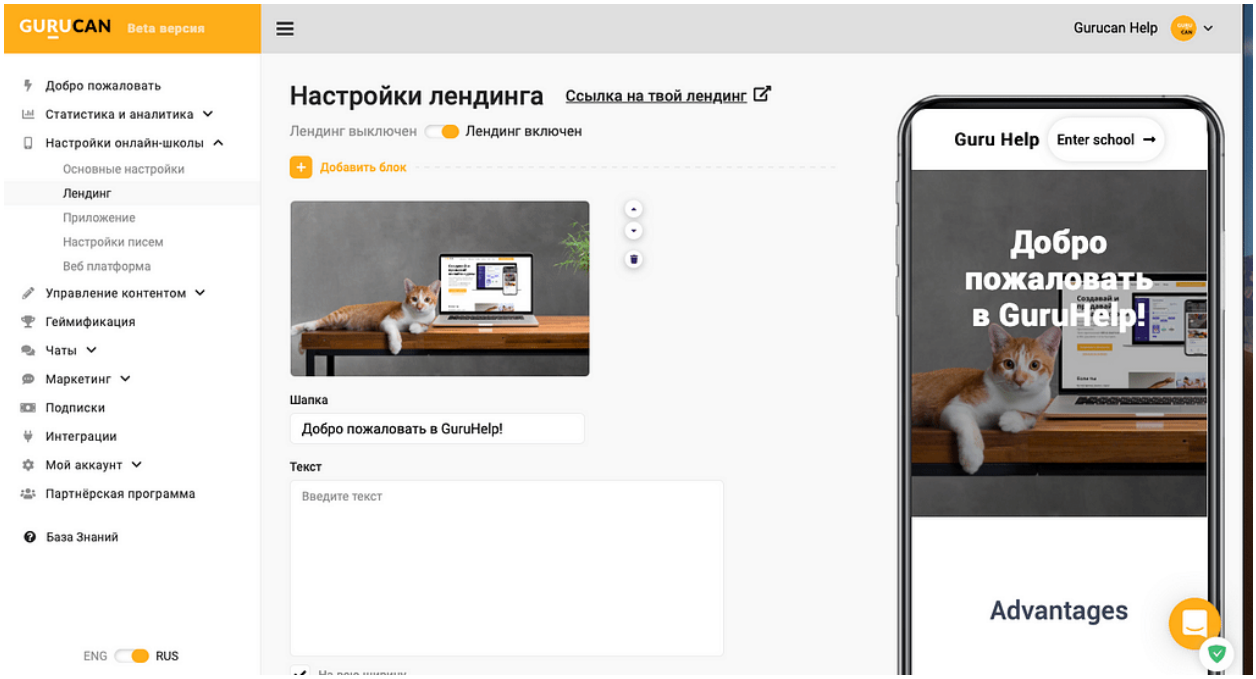

Рис. 1.3 GuruCan система

Плюси: є безкоштовний тестовий період, підійде для бізнесу в сфері фітнесу і ЗСЖ, тому що має спеціалізовані модулі для тренувань, марафонів, планів харчування і рецептів.

Мінуси: не підходить для корпоративного навчання, відсутня реєстрація учнів через Google і соцмережі. Немає можливості створити сертифікат про проходження курсу.

4. GetCourse

Платформа для проведення семінарів, тренінгів, курсів, очних і онлайн-занять. [3] Всі модулі в GetCourse пов'язані між собою, тому немає необхідності імпортувати дані в інші сервіси і платити за кожен з них окремо.

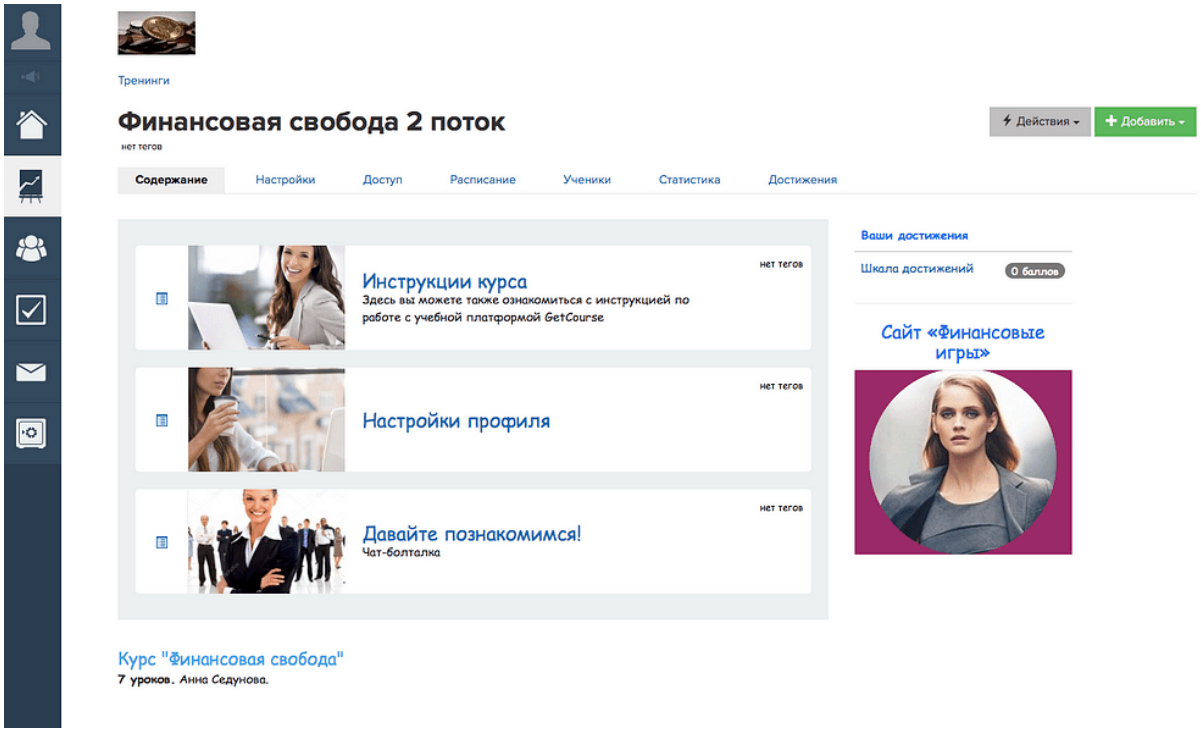

Рис. 1.4

Плюси: безкоштовний тестовий доступ, вбудований функціонал для продажів.

Мінуси: високі тарифи, складний інтерфейс

1.4. Вибір способу програмування(React)

React.js був розроблений Джорданом Уолком, програмістом Facebook, який намагався вирішити проблеми, з якими ця соціальна мережа зіткнулася під час розміщення оголошень. Хоча робота Уолка розпочалася у 2010 році, лише у травні 2013 року компанія Марка Цукерберга запустила React як рішення з відкритим кодом. [4] З тих пір було багато ентузіастів цієї бібліотеки, які продовжують сприяти її вдосконаленню.

Завдяки роботі Walke, Facebook та всіх програмістів, які внесли внесок у її оптимізацію, тепер у нас є інструмент, який дозволяє нам розробляти веб-

додатки, в яких перегляди інтерфейсу безпосередньо пов'язані з даними сервера, які вони отримують . Інші варіанти маніпулювання елементами DOM, такі як JQuery або навіть чистий Javascript, призводять до заплутаності та складності в обслуговуванні коду. React уникає цих проблем, пропонуючи архітектуру на основі компонентів, які є фрагментами коду, які використовують HTML, CSS та Javascript, щоб вони містили логіку та презентацію. [5]

Таким чином, ці компоненти можна відтворювати та використовувати у різних частинах програми та у різних інтерфейсах, створюючи більш впорядкований, зрозумілий та добре організований контроль потоку. Ось чому React є однією з найкращих альтернатив, коли йдеться про створення "додатків для однієї сторінки" або SPA, які, як і з інтерфейсом Facebook, завантажують різні візуалізації в одному інтерфейсі, без необхідності перезавантажувати сторінку. [6]
React - найпотужніша бібліотека інтерфейсу для розробки частини View з MVC. Нижче наведено кілька причин:

Крива малого навчання: Усі веб -розробники вже знають JavaScript. навчитись досить легко. Тим, хто не має досвіду веб -розробки або не знайомий з веб -розробкою, потрібно спочатку вивчити JavaScript. особливо останній javascript ES6.

Архітектура на основі компонентів:

Архітектура на основі компонентів дає вам можливість проєктувати складні програми та розбивати їх на невеликі фрагменти, що містять невеликі частини, і ці невеликі фрагменти значно спрощують створення, обслуговування, тестування та налагодження програми.

Він швидко відтворює та повторно відтворює програми та оновлює останні зміни даних.

Використовуючи алгоритм рендеринга, React робить його дійсно дуже швидким і швидким. Крім того, реагуйте за допомогою Virtual Dom (VDOM), представлення в реальній пам’яті Real Dom.

Масштабування React:

Додаток на основі React працює швидше, що забезпечує чудовий досвід користувача до кінця дня, коли продуктивність вашого додатка-це питання, яке робить вашого користувача щасливим.

Просто подивіться Facebook, він дуже масштабований, Facebook виробляє близько 5000 компонентів, але він працює без проблем з продуктивністю, підхід на основі компонентів, використання JSX, використання віртуального DOM, велика доступність сторонніх компонентів, і все це робить React масштабованим.

Завдяки наявності чудових сторонніх компонентів це економить багато часу та витрат на розробку.

У React чудова спільнота: з сильною спільнотою також з’являються великі ресурси, доступні тисячі відеороликів, навчальних посібників, публікацій у блогах, які допоможуть вам вивчити та розробити будь -який тип програми за допомогою React. [7]

Основні особливості ReactJs

1. React використовує Virtual DOM замість Real DOM.

Використання Virtual DOM підвищує продуктивність програми. Він працює як одностороннє зв'язування даних.

Ось робота Virtual DOM.

- Поки відбуваються будь-які зміни даних, весь інтерфейс користувача відтворюється у віртуальному DOM

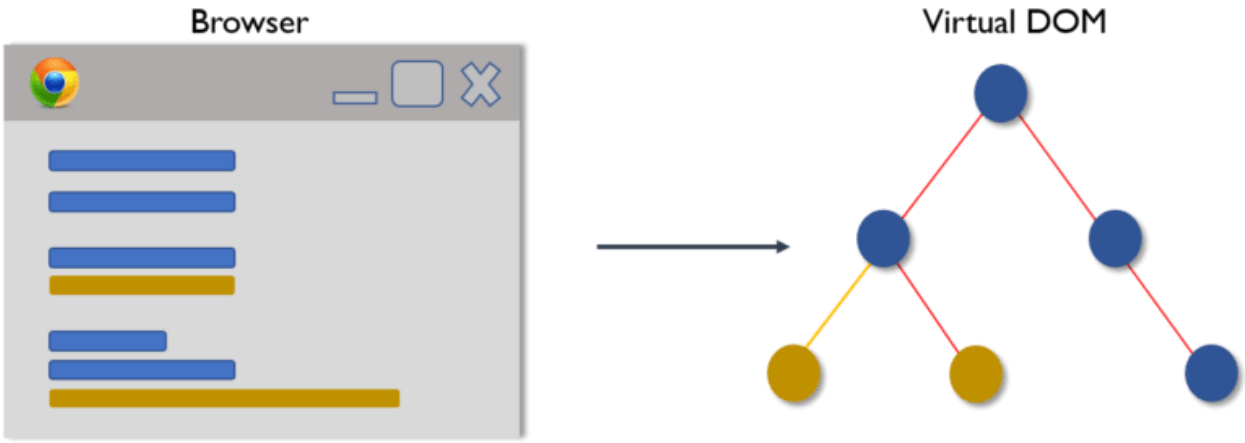

Рис. 1.5

- Потім обчисліть різницю між попереднім представленням DOM і новим.

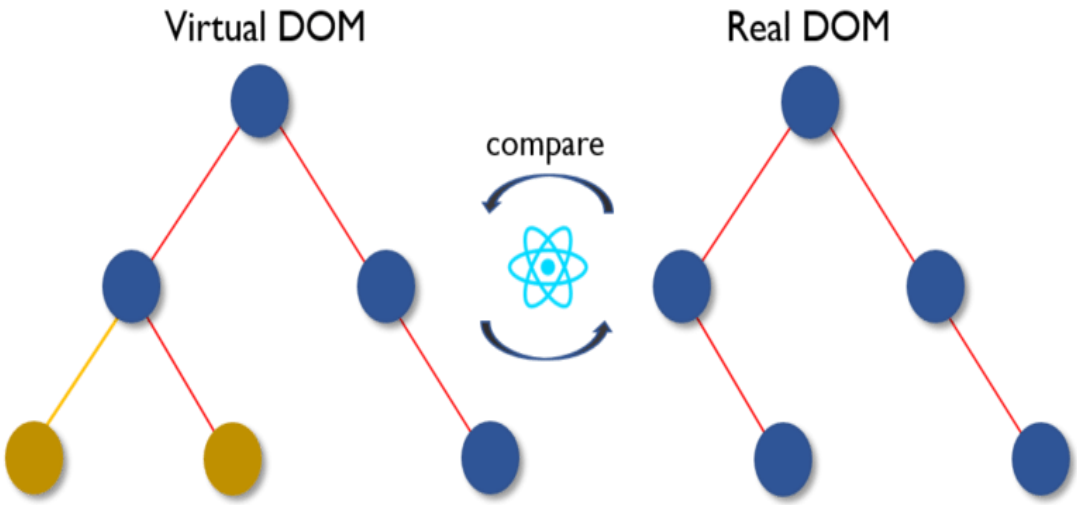

Рис. 1.6

Після завершення розрахунку Real DOM оновлюється лише зі змінами.

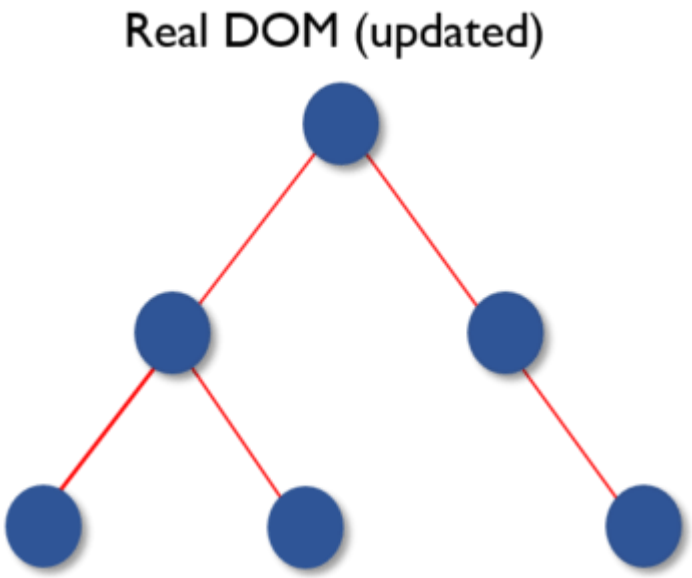

Рис. 1.7

1.5. Архітектура Flux

React відповідає за V або Перегляд у MVC. Flux, шаблон проє

ктування, відповідає M в MVC.

Це архітектура, відповідальна за створення даних у JavaScript-додатках та розробку серверних сторін у веб-додатках. Flux доповнює складові компоненти видань Перегляд у React, за допомогою одно направлених потік даних. [8]

Також можна сказати, що Flux більше чім шаблон, більше чім фреймворк і має 4 головні компоненти (більш детально вони будуть розглядати привітання):

- Диспетчер (Dispatcher)
- Хранилище (Stores)
- Представлення (Views) (React компонент)
- Дія (Action)

Це не схоже на привичний MVS, який ми привикли бачити в інших фреймворках. Дійсно, там є Контролер, але в більшій частині це контролер, відповідаючи за Перегляди. Перегляди знаходяться у верху ієрархії, і вони передають функціонал та дані елементам-потомкам.

Flux відповідає концепціям одно направленого потоку даних, що робить його простим для пошуку помилок. Данні проходять через прямий поток ваших додатків. React і Flux - на даний момент два найпопулярніших фреймворки, які використовують принцип одно направленого потоку даних.

В цей час, як React використовує віртуальний DOM для відображення змін, Flux робить це трохи інакше. Взаємодія з користувальницьким інтерфейсом вибирає ряд дій, які можуть змінювати дані додатків. [9]
Він має відкритий код, і це скоріше шаблон проєктування, а не фреймворк, тому його можна використовувати відразу. Те, що відрізняє його від інших, то відрізняє його і від шаблону проєктування MVC.

Допомагає зробити код більш передбачуваним, в порівнянні з MVC фреймворками. Розробники можуть створювати додатки, не турбуючись про складні взаємодії між джерелами даних.

Вигідно відрізняється більш організованим потоком даних - односпрямованим. Те, що він односпрямований, головна особливість. Ці дії поширюються на нову систему щодо взаємодії з користувачем.

Також ви можете почати використовувати Flux, не використовуючи при цьому весь новий код, на відміну від React.
Компоненти архітектури Flux взаємодіють між собою скоріше як EventBus і менш, ніж MVC.

Як згадувалося раніше, це не справжня бібліотека або фреймворк, це новий вид архітектури, яку Facebook використовує для роботи з React

зсередини. Отже, основна функція Flux буде доповнити React і реалізувати односпрямований потік даних. [10]

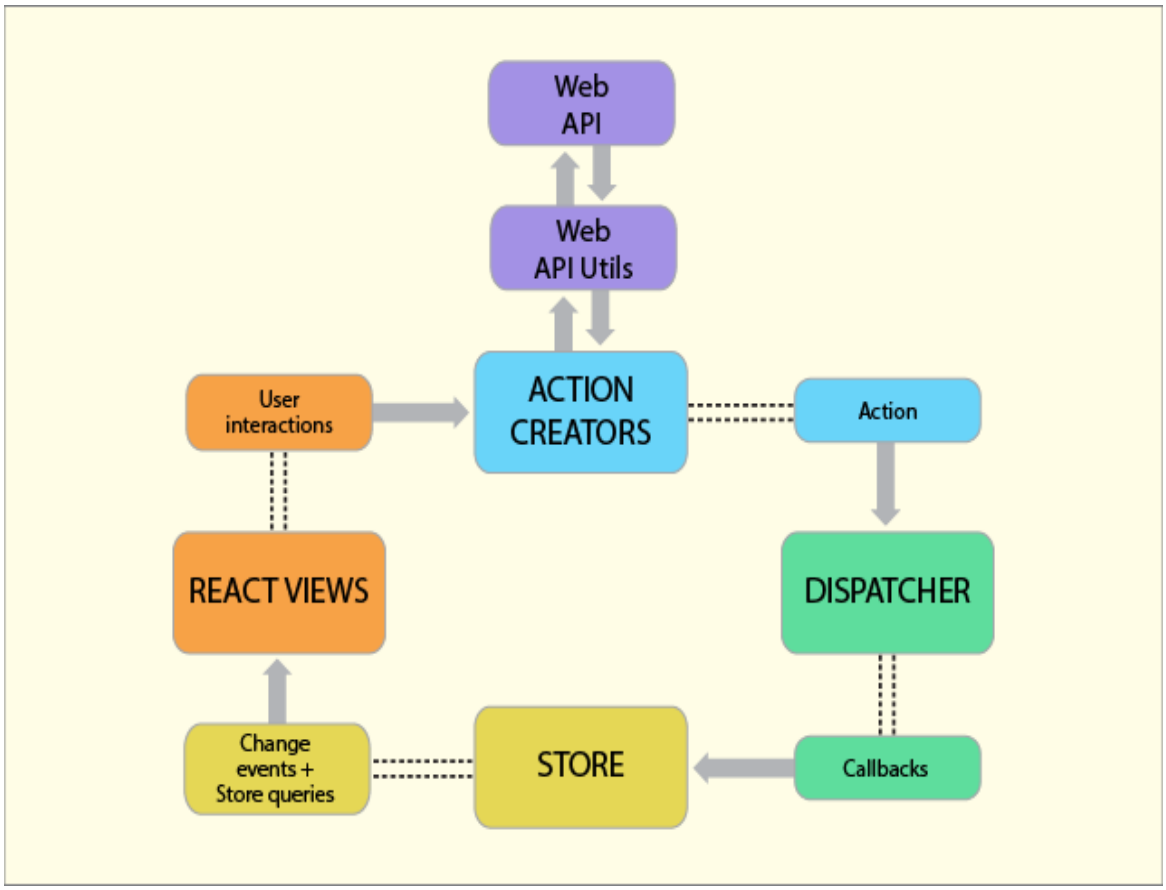

Рис. 1.7

У стандартній архітектурі наступні компоненти:

- Actions - помічники, які передають дані в Dispatcher
- Dispatcher - отримує ці дії і передає корисне навантаження зареєстрованим callback-му.
- Stores - діють як контейнери для стану програми та логіки. Реальна робота додатка відбувається в Stores. Stores, зареєстровані для прослуховування дій Dispatcher, будуть відповідно і оновлювати View.
- Controller Views - React компоненти захоплюють стан з Stores, а потім передають дочірнім компонентів.

Контролери в MVC і Flux розрізняються. Тут контролери є Controller-View і знаходяться на самій вершині ієрархії. View - це React компоненти. [11]

Весь функціонал, як правило, знаходиться в Store. В Store виконується вся робота і повідомляється Dispatcher, які події / дії він прослуховує. [12]

Коли відбувається подія, Dispatcher відправляє "корисне навантаження" в Store, який зареєстрований для прослуховування конкретно цієї події. Тепер в Store необхідно оновити View, що в свою чергу викликає дію. Дествие, точно також як і ім'я, тип події та багато іншого відомі заздалегідь.

View поширює Action через центральний Dispatcher, і це буде відправлено в різні Stores. Ці Stores будуть відображати бізнес-логіку програми та інші дані. Store оновлює все View.

Найбільш добре це працює спільно зі стилем програмування React, тоді Store відправляє поновлення без необхідності детально описувати, як змінювати уявлення між станами.

Це доводить, що шаблон проєктування Flux слід за односпрямованим потоком даних. Action, Dispatcher, Store і View - незалежні вузли з конкретними вхідними та вихідними даними. Дані проходять через Dispatcher, центральний хаб, який в свою чергу управляє всіма даними.

Dispatcher діє як реєстр із зареєстрованими callback-ами, на які відповідають Store. Store будуть давати зміни, які обрані Controller-Views.

Це відбувається, коли View поширюється в системі:

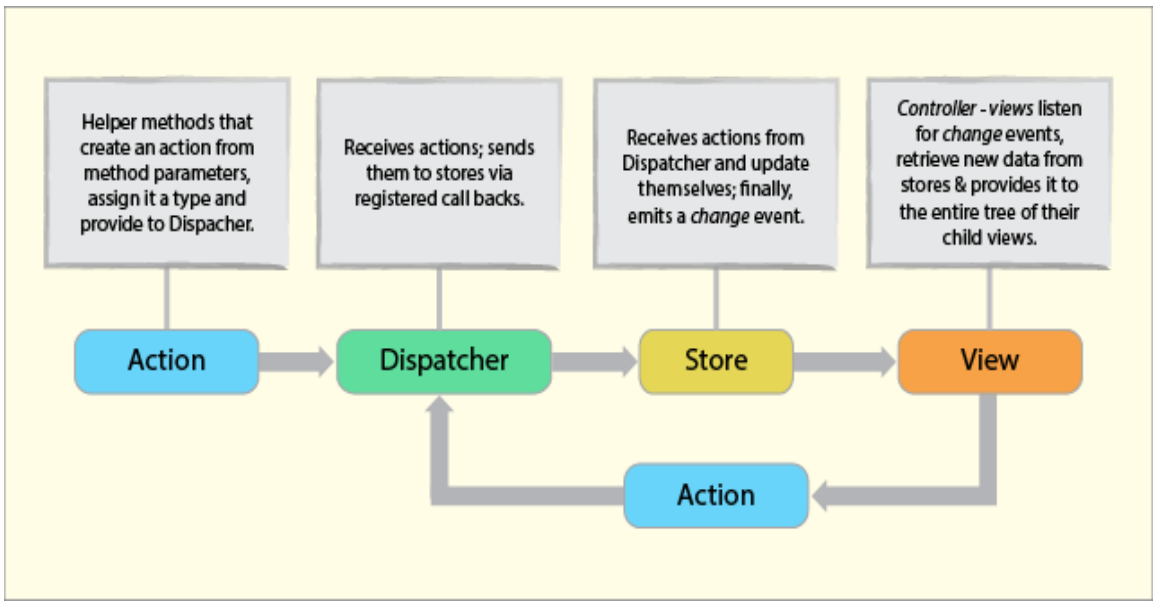

Рис. 1.8

Це доводить, що в Flux немає двосторонніх прив'язок (bind), що структура на кшталт функціональному відносного програмування, і ще щось на зразок потокового програмування.

Залежності, які відбуваються в Store, зберігаються в суворій ієрархії, тоді як Dispatcher обробляє оновлення. [13] Ця структура також вирішує проблеми, які природним чином виникають при двосторонньої прив'язці.

РОЗДІЛ 2
РЕВЮ СУЧАСНИХ ТЕХНОЛОГІЙ ДЛЯ СТВОРЕННЯ ДОДАТКУ

2.1. Що таке Node.js

Node.js - є платформою розробки з відкритим кодом для виконання коду JavaScript на стороні сервера. Node корисний для розробки програм, які вимагають постійного з'єднання браузера з сервером, і часто використовується для таких програм у режимі реального часу, як чат, стрічки новин та веб-сповіщення. [14]

Node.js призначений для роботи на виділеному сервері HTTP і для використання одного потоку з одним процесом одночасно. Додатки Node.js базуються на подіях і працюють асинхронно. Код, побудований на платформі Node, не відповідає традиційній моделі прийому, обробки, надсилання, очікування, отримання. Натомість Node обробляє вхідні запити у постійному стеку подій і надсилає невеликі запити один за одним, не чекаючи відповідей.

Це відхід від основних моделей, які запускають більші, більш складні процеси та запускають декілька потоків одночасно, причому кожен потік чекає відповідної відповіді, перш ніж рухатися далі.

Однією з головних переваг Node.js є те, що він не блокує введення/виведення (введення -виведення). Деякі розробники вкрай критично ставляться до Node.js і відзначають, що якщо для одного процесу потрібна значна кількість циклів процесора, програма блокуватиметься і блокування може призвести до збою програми. Прихильники моделі Node.js стверджують, що час обробки процесора не викликає занепокоєння через велику кількість невеликих процесів, на яких базується код Node.

2.2. Дослідження бібліотек

Популярність JavaScript різко зросла через Node.JS так як він представляє ідею парадигми "JavaScript всюди", яка об'єднує всю розробку веб-додатків навколо однієї мови програмування, а не має іншу мову для серверних та клієнтських сценаріїв. Також має в собі низку бібліотек для програмування. [15]

Дуже поширеними та популярним бібліотеками являються Angular, Vue, React

1. React.

React робить безболісний створення призначених для користувача інтерфейсів. Створюйте прості уявлення для кожного стану вашого застосування, і React буде ефективно оновлювати і відображувати тільки потрібні компоненти при зміні ваших даних.

Основні особливості React:

- Простота - Зрозуміти ReactJS просто простіше відразу. Підхід на основі компонентів, чітко визначений життєвий цикл та використання простого JavaScript роблять React дуже простим у вивченні, створюють професійну мережу (та мобільні додатки) та підтримують її. React використовує спеціальний синтаксис під назвою JSX, який дозволяє змішувати HTML з JavaScript. Це не є вимогою; Розробник все ще може писати простим JavaScript, але використання JSX набагато простіше. [16]
- Легко вчитися - Кожен, хто має базові попередні знання в галузі програмування, може легко зрозуміти React, тоді як Angular та Ember називають «мовою, специфічною для домену», що означає, що їх важко вивчити. Щоб реагувати, вам потрібні лише базові знання CSS та HTML
- Нативність - React можна використовувати для створення мобільних додатків (React Native). A React - завзятий шанувальник багаторазового використання, тобто підтримується широке повторне

використання коду. Тож одночасно ми можемо створювати IOS, Android та веб -додатки ;

- Прив’язка даних - у React використовується одностороннє зв'язування даних, а архітектура програми під назвою Flux контролює потік даних до компонентів через одну контрольну точку-диспетчер. Простіше налагоджувати автономні компоненти великих програм ReactJS; [17]
- Продуктивність - React не пропонує жодної концепції вбудованого контейнера для залежностей. Ви можете використовувати модулі Browserify, Require JS, EcmaScript 6, які ми можемо використовувати через Babel, ReactJS для автоматичного введення залежностей;
- Тестування - Додатки ReactJS дуже легко перевірити. React можна розглядати як функції стану, тому ми можемо маніпулювати станом, який ми передаємо до представлення ReactJS, і подивитися на вихідні дані та викликані дії, події, функції тощо. [18]

2. Vue.

Фреймворк покликаний полегшити життя і позбавити вас від написання однотипного коду. Творцем Vue.js є Evan You, колишній співробітник Google і Meteor Dev Group. Почав він розробляти фреймворк в 2013-му, а в лютому 2014 го відбувся перший публічний реліз. Vue широко використовується серед китайських компаній, наприклад: Alibaba, Baidu, Xiaomi, Sina Weibo і ін. Він входить в ядро Laravel і PageKit. Нещодавно вільна система управління репозиторіями GitLab теж перейшла на Vue.js. Але в міру того, як кодова база деяких фреймворків сильно зростає, вони починають вносити свою частку в проект. [23] Через це при плануванні розвитку ми повинні враховувати два фактори:

- складність застосування;
- складність фреймворка, який ми використовуємо;

Основними концепціями Vue є;

- конструктор
- компоненти
- директиви
- переходи

3. Angular.

Angular-це платформа та фреймворк для створення односторінкових клієнтських програм за допомогою HTML та TypeScript. Він реалізує основні та додаткові функції як набір бібліотек TypeScript, які ви імпортуєте у свої програми. Архітектура програми Angular спирається на певні фундаментальні концепції. [20]

Загалом фреймворки підвищують ефективність та продуктивність веб -розробки, забезпечуючи послідовну структуру, щоб розробникам не доводилося відновлювати код з нуля. Фреймворки - це економія часу, яка пропонує розробникам безліч додаткових функцій, які можна додати до програмного забезпечення без особливих зусиль.

Порівняння бібліотеки

	Angular	**Vue**	**React**
Творець	Google	Evan You	Facebook
Розмір	237.8 KB	51.32 KB	87.4 KB
Рік	2010	2014	2013
Рейтинг на Github	173 тис.	61 тис.	151 тис.
Час скачування	1.65	1.12	1.21
бистрота с ДОМ	1.41	1.35	1.27

Фінальна збірка	12.2.8 September 30, 2021	3.0 "One Piece" Dec 13, 2019	17.0.2 March 22, 2021
Ужиткували	Microsoft Office Deutsche Bank Santander	Upwork Wizzair Adobe Portfolio	Airbnb Dropbox Codecademy

Таблиця 2.1 Порівняння

2.3. Огляд Redux

Redux - бібліотека для JavaScript з відкритим кодом, призначена для управління станом додаток. Час всього використовується у парі з React або Angular для розробки клієнтської частини. Забезпечує ряд інструментів, що дозволяють значно спрощувати передачу даних зберігання через контекст. Допомагає писати програми, які поводяться послідовно, працюють у різних середовищах (клієнтських, серверних та рідних) та їх легко перевірити. Крім того, він забезпечує чудовий досвід розробників, наприклад редагування коду в режимі реального часу в поєднанні з налагоджувачем(debugger). [21]

Ви можете використовувати Redux разом з React або з будь -якою іншою бібліотекою представлень. Він крихітний (2 кБ, включаючи залежності), але має велику доступну екосистему доповнень.

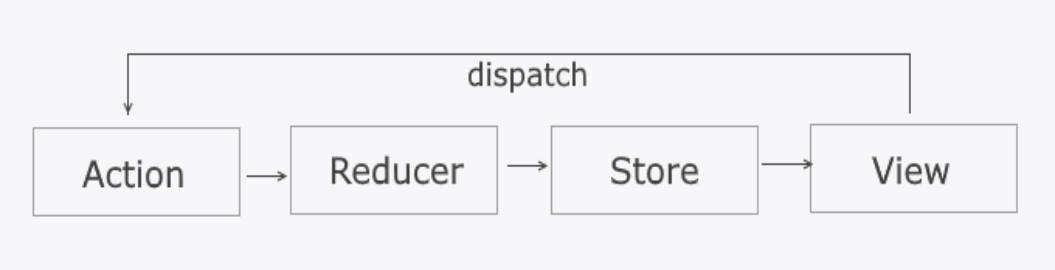

Рис. 2.1 Redux архітектура

Store – містить усе дерево стану вашої програми. Єдиний спосіб змінити стан всередині нього - надіслати дію над ним. [19]

Action - як і у випадку зі станом, дії, що піддаються серіалізації, включають кілька визначальних функцій, запису та відтворення дії.

Reduce - функція, яка отримує попередній стан і Action і повертає новий стан. завдяки цьому вдається налаштувати гнучку взаємодію клієнту та сервера і організувати зрозумілий потік даних в веб-додатку.

Особливості:

- Більше вільного місця
- Має дуже велику базу відкритої інформації для навчання, і ознайомлення з функціоналом
- Часті оновлення які виконує розробник, та підтримка спільноти
- Підтримує більшість бібліотек для розробки

Принципи Redux:

1. Єдине джерело істини.

Це полегшує створення універсальних програм, оскільки стан із вашого сервера можна серіалізувати та перетворити на клієнта без зайвих зусиль щодо кодування. Одне дерево станів також полегшує налагодження або перевірку програми; це також дозволяє зберігати стан вашого додатка в процесі розробки для прискорення циклу розробки. Деякі функціональні можливості, які традиційно було важко реалізувати - скасувати/повторити, наприклад, - можуть раптом стати тривіальними для реалізації, якщо всі ваші стани зберігаються в одному дереві. Стан доступний лише для зчитування.

2. Статус доступний лише для читання. Єдиний спосіб змінити стан, це викликати дію, об’єкт який описує те що сталося. Це гарантує що зміни стану будуть чітко контролюватися і не викличуть додаткових помилок. Тому що всі зміни централізовані і відбуваються чітко. Дії - це

прості об’єкти, які можна серіалізувати, зберегти та скопіювати пізніше для налагодження.

3. Зміни вносяться за допомогою чистих функцій.

Як змінюється дерево станів за допомогою чистого редуктора (Reducer). Це чиста функція, яка приймає попередній стан і дії як параметри і повертає новий стан зі змінами.

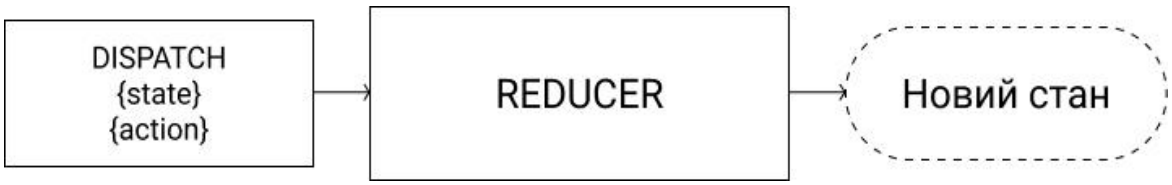

Рис. 2.2. Демонстрація роботи Reducer

Використання з React.

Механізм локального сховища компонента, який поставляється разом з базовою бібліотекою (React) незручний тим, що таке сховище ізольовано. Наприклад, якщо ви хочете, щоб різні незалежні компоненти реагували на будь-яку подію, вам доведеться або передавати локальне стан у вигляді пропсів дочірнім компонентів, або піднімати його вгору до найближчого батьківського компонента. В обох випадках робити це не зручно. Код стає брудніше, складним для розуміння, а компоненти залежними від їх вкладеності. Redux знімає цю проблему так як всі стани доступні усім компонентом без особливих труднощів.

Redux є універсальним засобом розробки і може бути використаний в зв'язці з різними бібліотеками і фреймворками.

РОЗДІЛ 3
ПЛАНУВАННЯ ТА ОПРАЦЮВАННЯ ДОДАТКУ

3.1. Постановка задач

1. Проєктування серверної та клієнтської бази
2. Розробка back-end
3. Розробка front-end
4. Поєднання компонентів
5. Налагодження

Завданням поставленим перед кваліфікаційною роботою являється розробка веб-додатка для контролю виконання лабораторних робіт з програмування.

На прикладі сервісів Moodle, ISpring, Blackboard веб додаток буде мати одне направлення, та дозволить розгорнути свій сервіс для перевірки виконання лабораторних робіт. В даному веб-додатку буде використовуватися сучасні технології.[24]

Загальна структура веб-додатка.

Веб-додаток буде виконаний з трьох модулів, представлених на рис. 3.1.

Клієнт-сервер - позначає зв'язок між програмами, що співпрацюють у додатку, що складається з клієнтів, які ініціюють запити на послуги та серверів, що надають цю функцію або послугу. В даному веб-додатку це буде бібліотека React

Серверна частина - це процес на сервері для обробки запитів користувача та взаємодії з даними. В проекті кваліфікаційної роботи було використано бібліотеку Express для програмування серверної частини.

База даних - упорядкований набір структурованої інформації або даних, які зазвичай зберігаються в електронному вигляді в комп'ютерній

системі. База даних зазвичай управляється системою управління базами даних (СКБД). Дані разом з СУБД, а також додатки, які з ними пов'язані, називаються системою баз даних, або, для стислості, просто базою даних. В даному веб-додатку це буде бібліотека MongoDB

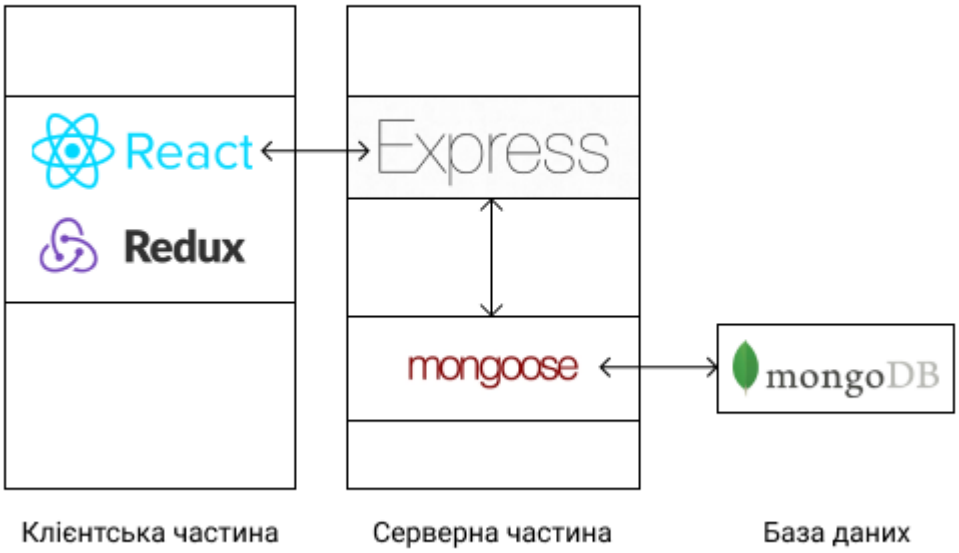

Рис. 3.1 Модулі

3.2. Проєктування додатку

Діаграмма була створена з допомогою мови UML. Для того щоб описати вимоги до додатку.

Таблиця відображає зв'язок даних і відносин.

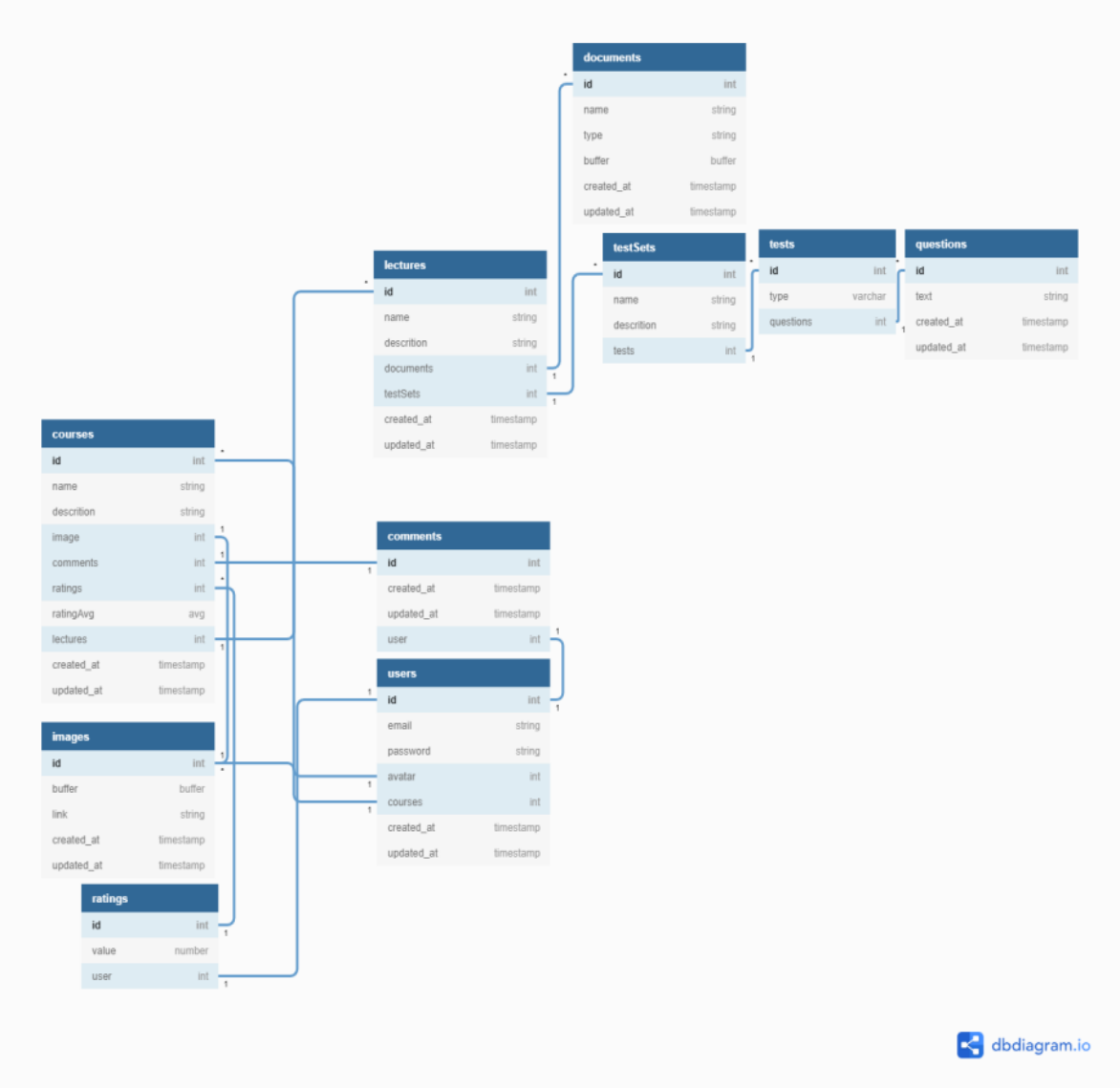

Рис 3.2 Діаграма бази даних

У системі є 2 ролі: Вчитель та студент. Було досліджено предметну область, і на цій основі був побудований графік попиту з використанням варіантів.

Об’єкт	**Документація**
Вчитель	Відповідає за розміщення лабораторних. Редагування лекцій, додавання робіт.
Студент	Взаємодіє з створеними

	роботами.
Автентифікація	Перевірка даних для входу в обліковий запис
Створення / редагування лабораторних Створення / редагування файлів	Розміщення записів виконується через адмін панель

Таблиця 3.1. Документування об'єктів

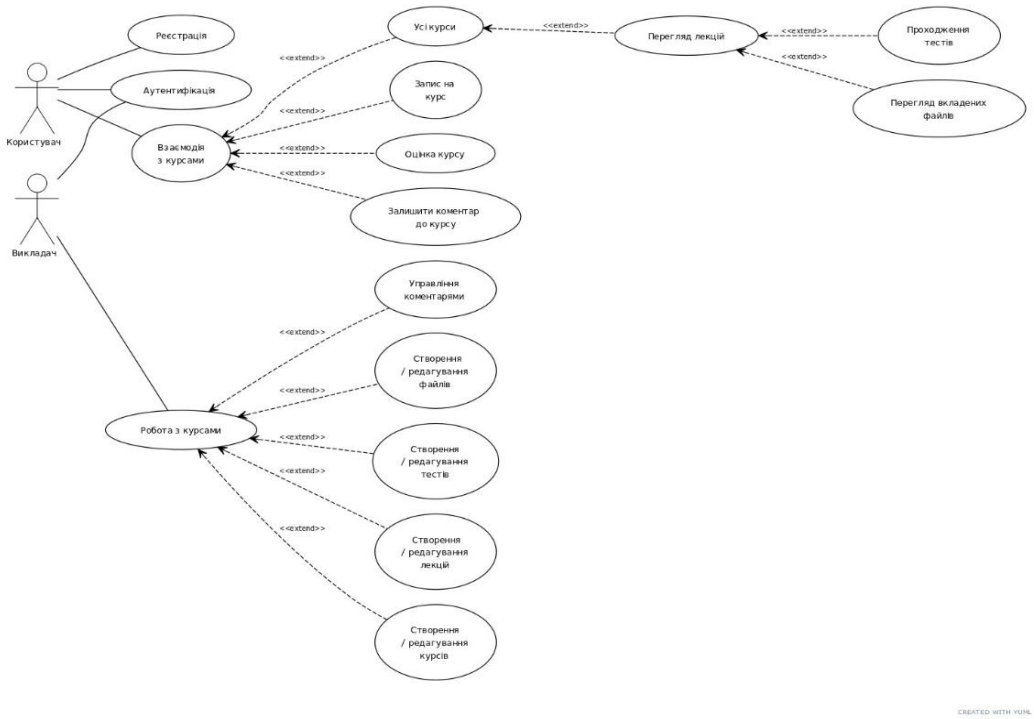

Рис 3.3 Діаграма варіантів використання

3.3. Розробка серверної частини

MongoDB реалізує новий підхід до побудови баз даних, де немає таблиць, схем, запитів SQL, зовнішніх ключів і багатьох інших речей, які притаманні об'єктно-реляційних баз даних.

З часів динозаврів було звичайною справою зберігати всі дані в реляційних базах даних (MS SQL, MySQL, Oracle, PostgresSQL). При цьому було не так важливо, а чи підходять реляційні бази даних для зберігання даного типу даних чи ні. [25]

На відміну від реляційних баз даних MongoDB пропонує документо-орієнтовану модель даних, завдяки чому MongoDB працює швидше, має кращу масштабованість, її легше використовувати.

Але, навіть враховуючи всі недоліки традиційних баз даних і гідності MongoDB, важливо розуміти, що завдання бувають різні і методи їх вирішення бувають різні. В якійсь ситуації MongoDB дійсно поліпшить продуктивність вашої програми, наприклад, якщо треба зберігати складні за структурою дані. В іншій же ситуації краще буде використовувати традиційні реляційні бази даних. Крім того, можна використовувати змішані підхід: зберігати один тип даних в MongoDB, а інший тип даних - в традиційних БД.

Вся система MongoDB може представляти не тільки одну базу даних, що знаходиться на одному фізичному сервері. Функціональність MongoDB дозволяє розташувати кілька баз даних на декількох фізичних серверах, і ці бази даних зможуть легко обмінюватися даними і зберігати цілісність.

Формат даних в MongoDB

Одним з популярних стандартів обміну даними та їх зберігання є JSON (JavaScript Object Notation). JSON ефективно описує складні за структурою дані. Спосіб зберігання даних в MongoDB в цьому плані схожий на JSON, хоча формально JSON не використовується. Для зберігання в MongoDB застосовується формат, який називається BSON (Бісон) або скорочення від binary JSON.[38]

BSON дозволяє працювати з даними швидше: швидше виконується пошук і обробка. Хоча треба зазначити, що BSON на відміну від зберігання даних в форматі JSON має невеликий недолік: в цілому дані в JSON-форматі

займають менше місця, ніж в форматі BSON, з іншого боку, даний недолік з лишком окупається швидкістю.

MongoDB написана на C, тому її легко перенести на найрізноманітніші платформи. MongoDB може бути розгорнута на платформах Windows, Linux, MacOS, Solaris. Можна також завантажити вихідний код і самому скомпілювати MongoDB, але рекомендується використовувати бібліотеки. [31]

База даних Oracle дозволяє швидко і безпечно зберігати та отримувати дані. Ось переваги інтеграції бази даних Oracle:

- Ось переваги інтеграції бази даних Oracle:
- База даних Oracle є кроссплатформенною.
- Він може працювати на різному обладнанні в різних операційних системах, включаючи Windows Server, Unix та різні дистрибутиви GNU/Linux.
- База даних Oracle має свій мережевий стек, що дозволяє додаткам з іншої платформи безперебійно спілкуватися з базою даних Oracle. Наприклад, програми, що працюють у Windows, можуть підключатися до бази даних Oracle, що працює під управлінням Unix.
- ACID сумісний Oracle ACID-сумісна база даних, яка допомагає підтримувати цілісність та надійність даних.
- Прихильність відкритим технологіям Oracle одна з перших баз даних, яка підтримувала GNU/Linux наприкінці 1990 -х років до того, як GNU/Linux стала комерційним продуктом.
- З тих пір вона підтримує цю відкриту платформу.

MySQL - одна з найбільш впізнаваних технологій у сучасній екосистемі великих даних. Часто називається найпопулярнішою базою даних і в даний час користується широким і ефективним використанням незалежно від галузі, зрозуміло, що будь -хто, хто займається корпоративними даними

або загальними IT, повинен принаймні прагнути до базового знання MySQL.

З MySQL навіть ті, хто новачок у реляційних системах, можуть негайно створити швидкі, потужні та безпечні системи зберігання даних. Програмний синтаксис та інтерфейси MySQL також є ідеальними воротами у широкий світ інших популярних мов запитів та структурованих сховищ даних.

Тому можна зробити висновок що база даних Mongodb найкраще підходить для вирішення поставлених задач.[32]

Документи замість рядків приклад схеми документа:

```
const {Schema, model, Types} = require('mongoose')

const schema = new Schema({
    name: {type: String, required: true, unique: true},
    description: {type: String, required: true},
    image: [{type: Types.ObjectId, ref: ‘Image}],
    comments: [{type: Types.ObjectId, ref: 'Comment'}],
    ratings: [{type: Types.ObjectId, ref: 'Rating'}],
    ratingAvg: {type: Number},
    lectures: [{type: Types.ObjectId, ref: 'Lecture'}]
},{ timestamps: true })

module.exports = model('Course', schema)
```

Рис 3.4 Приклад опису схеми документу MongoDB

На рисунку наведеному вище можна побачити як створюється властивості для подальшого використання з допомогою Express.js для обробки запитів

Це безкоштовна платформа з відкритим вихідним кодом для Node.js. Він використовується для швидкого та легкого проєктування та створення веб-додатків. Оскільки Express.js вимагає лише javascript, програмістам та

розробникам стає простіше створювати веб-програми та API без будь-яких зусиль.[27]

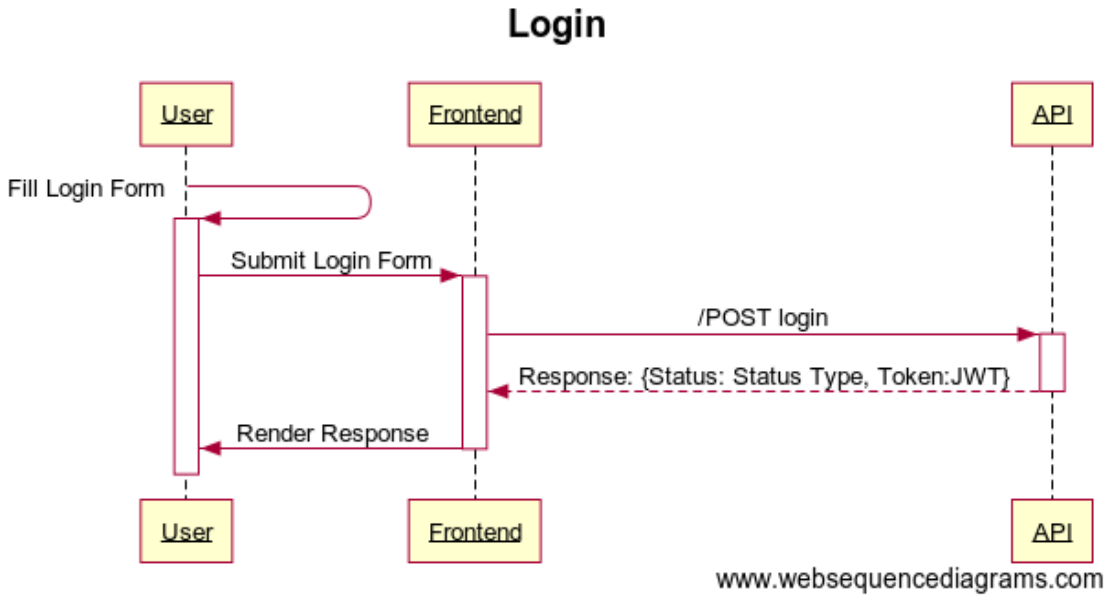

Рис 3.5 Схема авторизації

```
app.use('/api/courses', require('./routes/course.routes'))

// GET /api/courses/
router.get('/', async (req, res) => {
    try {
        let { limit } = req.body
        const courses = await Course.find().limit(limit[0], limit[1])
        res.status(200).json({"courses": courses})
    } catch (e) {
        console.log(e)
        res.status(500).json({'message': "Error!"})
    }
})
```

Рис 3.6 Get запит для виведення списку

Деякі параметри обробника запиту:

- Router (об'єкт маршрутизатора) - Метод route є похідним від одного з методів HTTP і приєднується до примірника класу express. req - об'єкт запиту.

- res - відправки бажаної відповіді HTTP.
- res.status(...),json(...) - це вбудована функція посереднього програмного забезпечення в Express. Він аналізує вхідні запити з корисними навантаженнями JSON і базується на аналізі тіла.
- Course – Схема бази даних

3.4. Розробка інтерфейсу користувача

Для програмування було використано бібліотеку React. Для керуванням серидовища Redux.[36]

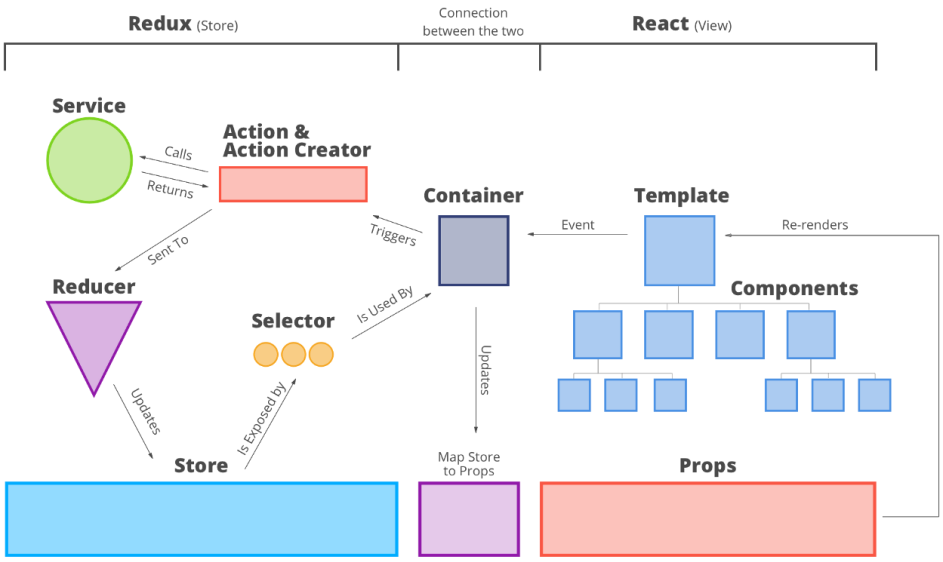

Рис 3.7 Загальна архітектура для React + Redux

Веб-компоненти React - забезпечують надійну інкапсуляцію для повторного використовуваних компонентів, в той час, як React надає декларативну бібліотеку для синхронізації даних с DOM.

Container - виконує дві функції, представляючи інтерфейс користувача та обробляючи логіку компонента.

Service - сервіс, спрямовує API запити

Selector - можуть обчислювати похідні дані, дозволяючи Redux зберігати мінімально можливий стан

Action - це звичайний об’єкт JavaScript, який повинен мати атрибут type для позначення типу виконуваної дії. Зробивши висновки ми створили таблицю структури додатку рис. 3.3.

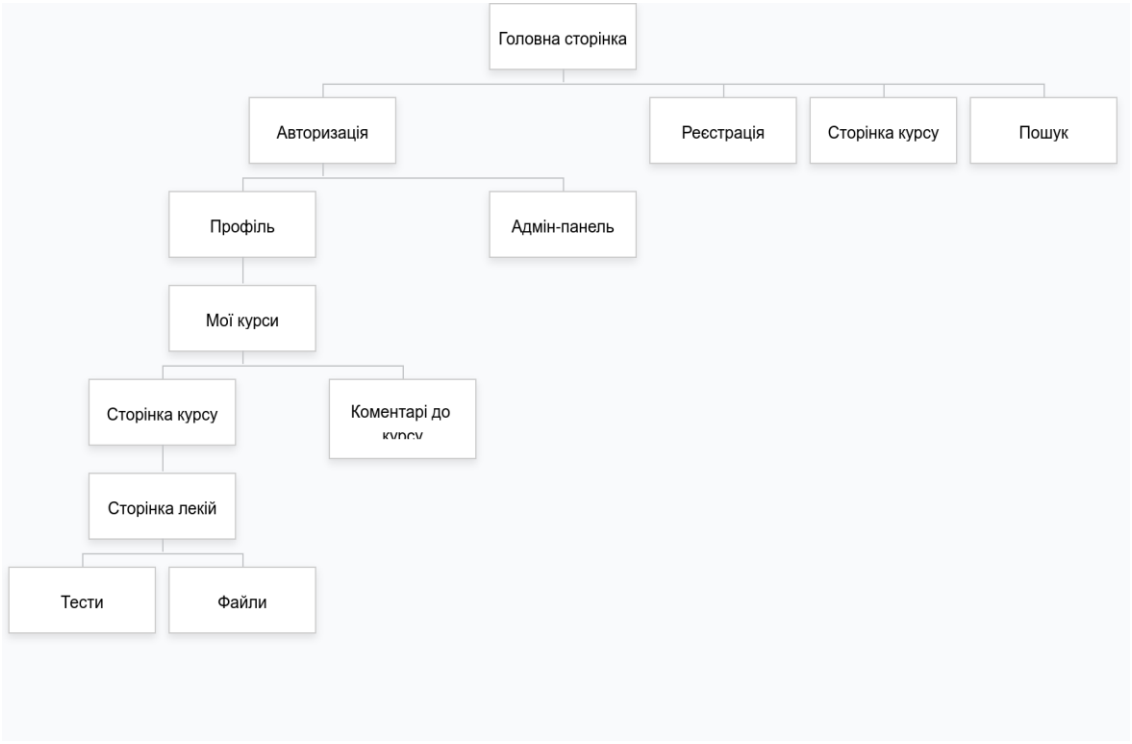

Рис. 3.8 Структура

Розробка візуальної частини

Під час цієї частини було виконана розробка макету для швидкого розуміння і виконання поставленої роботи. Тому для цього було вибрано React розробку так як він підтримує безперервний процес поліпшення. Компоненти, які були створені під час роботи над тим чи іншим проектом, не мають додаткових залежностей. Таким чином, ніщо не заважає використовувати їх знову і знову в проектах різного типу. Весь попередній досвід може бути з легкістю застосований при роботі над новим сайтом або навіть при створенні мобільного додатка. Використовуючи передові можливості, такі як Virtual DOM або ізоморфний JavaScript, React розробники можуть з високою швидкістю створювати високопродуктивні додатки, незважаючи на рівень їх складності.

```
import CourseList from '../components/courses/CourseList'
import Search from '../components/Search'
export class Main extends Component {
    render() {
        return (
            <div>
                <Search/>
                <h2>Нові курси</h2>
                <CourseList courses={this.props.courses} />
                <Footer/>
            </div>
        )
    }
}
```

Рис. 3.9 Приклад сторінки додатку

3.5. Клієнт-сервер

Виконавши підготовку і налагодження візуальної частини було виконано, тепер буде виконуватися налаштування сервера.

Для цього і було обрано бібліотеку Redux, яку ми розглянули у другому розділі.

Асинхронні запити для подальшого імпортування.

```
export const FETCH_COURSES_REQUEST = 'FETCH_COURSES_REQUEST'
export const FETCH_COURSES_SUCCESS = 'FETCH_COURSES_SUCCESS'
export const FETCH_COURSES_FAILURE = 'FETCH_COURSES_FAILURE'
```

Рис. 3.10

```
const fetchCoursesRequest = courses => {
    return {
        type: FETCH_COURSES_REQUEST
    }
}
```

Рис. 3.11 зразок Actions

Actions – це звичайні JavaScript-об'єкти. Екшени повинні мати поле type, яке вказує на тип виконуваного екшену. Типи повинні бути, як правило, задані, як рядкові константи. Після того, як ваш додаток розростеться, ви можете захотіти перемістити їх в окремий модуль.

```
export const fetchCourses = () => {
    return (dispatch) => {
        dispatch(fetchCoursesRequest())
        axios.get('/courses').then((response) => {
            const courses = response.data
            dispatch(fetchCoursesSuccess(courses))
        })
        .catch((error) => {
            const { message } = error
            dispatch(fetchCoursesFailure(message))
        })
    }
}
```

Рис. 3.12 запит Action Creator

Action-и приймають тип і опціонально корисне навантаження (type і payload).

Більшість змін в додатку, що використовує Redux, починаються з події, яка прямо або побічно запускається користувачем. Події - це, наприклад, натискання кнопки, вибір елемента у спадному меню, наведення курсору на певний елемент або запит AJAX, який тільки що повернув якісь дані. Навіть початкова завантаження сторінки може бути

приводом для відправки action-а. Дії часто відправляються за допомогою action creator-а.

```
const reducer = (state = initialState, action) => {
    switch (action.type) {
        case FETCH_COURSES_SUCCESS: {
            return {
                ...state,
                courses: action.payload,
                loading: false
            }
        }
        default: {
            return state
        }
    }
}
```

Рис. 3.13 функція Reducer

Redux надає функцію під назвою combReducers, яка виконує два завдання: генерує функцію, яка викликає наші reducer-и з фрагментом стану, обраним відповідно до їх ключем, а потім знову об'єднує результати в один об'єкт.

ВИСНОВКИ

Метою кваліфікаційної роботи є розробка створення програмної системи підтримки проведення лабораторних робіт з програмування в умовах дистанційного навчання.

У ході кваліфікаційної роботи було виконано:

- Проаналізовано системи LMS та СДН
- Виконаний пошук оптимальних способів для реалізації додатку
- Була проаналізована архітектура
- Ревю сучасних технологій для створення додатку
- Розробка серверної частини
- Розробка інтерфейсу користувача
- Клієнт – сервер
- Сучасність використаних технологій

В ході виконання роботи було розглянуто LMS та СНД системи такі як eTutorium LMS, iSpring Learn, GuruCan. GetCourse. LMS це то платформа для електронного навчання. Ключові принципи її роботи криються в самій абревіатурі. Learning - навчання. Для створення цієї системи було прийнято рішення використати сучасний підхід. Базою даних було вибрано як MongoDB переваги цієї системи зберігання використовуваних в даний момент даних використовується внутрішня пам'ять, що дозволяє отримувати більш швидкий доступ. Дані зберігаються у вигляді JSON документів. MongoDB підтримує динамічні запити документів (document-based query) Відсутність складних JOIN запитів. Немає необхідності маппінга об'єктів додатки в об'єкти БД. Бібліотекою для інтерфейсу React – її переваги це компонентний підхід, компактність, гнучкість і розвинена екосистема, зворотня сумістність.

Розроблений веб-додаток задовольняє всі вимоги, сформовані на етапі постановки завдання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. React [Електронний ресурс] – Режим доступу до ресурсу: https://ru.reactjs.org/
2. Ispring [Електронний ресурс] – Режим доступу до ресурсу: https://www.ispring.com/ua/elearning-insights/chto-takoe-lms
3. Что такое LMS и как с ее помощью управлять обучением [Електронний ресурс] – Режим доступу до ресурсу: https://timeweb.com/ua/community/articles/lms-sistema-chto-eto-takoe
4. Webtutor – это система управления обучением, разработанная для самых требовательных. [Електронний ресурс] – Режим доступу до ресурсу: https://coloris.com.ua/webtutor-lms/?utm_source=google&utm_medium=cpc&utm_campaign=websoft-hcm-12-07-2021&utm_term
5. Системы Дистанционного Обучения [Електронний ресурс] – Режим доступу до ресурсу: https://lmslist.ru/sdo/
6. The Best Websites Examples Built With React JS [Електронний ресурс] – Режим доступу до ресурсу: https://procoders.tech/blog/popular-react-js-websites-examples/
7. React Native [Електронний ресурс] – Режим доступу до ресурсу: https://reactnative.dev/
8. Flutter for React Native developer [Електронний ресурс] – Режим доступу до ресурсу: https://flutter.dev/docs/get-started/flutter-for/react-native-devs
9. Что такое Angular. Первый проект [Електронний ресурс] – Режим доступу до ресурсу: https://metanit.com/web/angular2/1.1.php
10. Angular [Електронний ресурс] – Режим доступу до ресурсу: https://www.jetbrains.com/help/webstorm/angular.html#create_new_angular_app

11. Using Angular in Visual Studio Code [Електронний ресурс] – Режим доступу до ресурсу: https://code.visualstudio.com/docs/nodejs/angular-tutorial
12. A Predictable State Container for JS Apps [Електронний ресурс] – Режим доступу до ресурсу: https://redux.js.org/
13. Redux is a predictable state container for JavaScript apps. [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/reduxjs/redux
14. Where software teams [Електронний ресурс] – Режим доступу до ресурсу: https://rajdee.gitbooks.io/redux-in-russian/content/docs/introduction/
15. JS: Redux (React) [Електронний ресурс] – Режим доступу до ресурсу: https://ru.hexlet.io/courses/js-redux
16. Get started developing for Android using React Native [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/windows/dev-environment/javascript/react-native-for-android
17. Get started build a desktop app with React Native for Windows [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/windows/dev-environment/javascript/react-native-for-windows
18. Vue.js для сомневающихся. Все, что нужно знать [Електронний ресурс] – Режим доступу до ресурсу: https://home.pearsonvue.com/
19. The Flux Architecture [Електронний ресурс] – Режим доступу до ресурсу: https://www.coursera.org/lecture/front-end-react/the-flux-architecture-IprLd
20. Flux In ReactJS [Електронний ресурс] – Режим доступу до ресурсу: https://www.w3spoint.com/flux-reactjs
21. A Simple React-Flux Example Using ES6 [Електронний ресурс] – Режим доступу до ресурсу: https://www.codeproject.com/Articles/1179395/A-Simple-React-Flux-Example-Using-ES

22. VueJS Tutorial [Електронний ресурс] – Режим доступу до ресурсу: https://www.tutorialspoint.com/vuejs/index.htm
23. VueJS - Components [Електронний ресурс] – Режим доступу до ресурсу: https://www.tutorialspoint.com/vuejs/vuejs_components.htm
24. The Unified Modeling Language [Електронний ресурс] – Режим доступу до ресурсу: https://www.uml-diagrams.org/
25. Use Case Diagram [Електронний ресурс] – Режим доступу до ресурсу: https://www.smartdraw.com/use-case-diagram/
26. Use Case Diagram Tutorial (Guide with Examples) [Електронний ресурс] – Режим доступу до ресурсу: https://creately.com/blog/diagrams/use-case-diagram-tutorial/
27. UML Use Case Diagram Tutorial [Електронний ресурс] – Режим доступу до ресурсу: https://www.lucidchart.com/pages/uml-use-case-diagram
28. MongoDB Tutorial [Електронний ресурс] – Режим доступу до ресурсу: https://www.tutorialspoint.com/mongodb/index.htm
29. Redux Fundamentals, Part 3: State, Actions, and Reducers [Електронний ресурс] – Режим доступу до ресурсу: https://redux.js.org/tutorials/fundamentals/part-3-state-actions-reducers
30. Redux Toolkit TypeScript Quick Start [Електронний ресурс] – Режим доступу до ресурсу: https://redux.js.org/tutorials/typescript-quick-start.
31. Steve F. The LMS Guidebook: Learning Management Systems Demystified / Foreman Steve., 2017. – 274 с.
32. Robin W. The Road to React: Your journey to master React.js in JavaScript / Wieruch Robin., 2017. – 340 с.
33. Eve P. Learning React: Modern Patterns for Developing React Apps 2nd Edition / P. Eve, B. Alex., 2020. – 312 с.
34. Pedro T. Professional Node.js: Building Javascript Based Scalable Software / Teixeira Pedro., 2012. – 495 с.
35. Basarat S. Beginning Node.js / Syed Basarat., 2014. – 326 с.

36. Greg L. Beginning Node.js, Express & MongoDB Development / Lim Greg., 2019. – 155 с.
37. Greg L. Beginning MERN Stack: Build and Deploy a Full Stack MongoDB, Express, React, Node.js App / Lim Greg., 2021. – 159 с.
38. MongoDB Fundamentals: A hands-on guide to using MongoDB and Atlas in the real world / P.Amit, A. Juned, H. Michael, N. Liviu., 2020. – 748 с.
39. David F. JavaScript: The Definitive Guide: Master the World's Most-Used Programming Language / Flanagan David., 2020. – 706 с. – (O'Reilly Media).
40. Jonathan W. Get Programming with Node.js / Wexler Jonathan., 2019. – 480 с. – (Manning). – (1st edition).

Додаток 1

КОДЕКС АКАДЕМІЧНОЇ ДОБРОЧЕСНОСТІ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ ХЕРСОНСЬКОГО ДЕРЖАВНОГО УНІВЕРСИТЕТУ

Я, _Болокан Владислав Анатолійович_, учасник(ця) освітнього процесу Херсонського державного університету, **УСВІДОМЛЮЮ**, що академічна доброчесність – це фундаментальна етична цінність усієї академічної спільноти світу.

ЗАЯВЛЯЮ, що у своїй освітній і науковій діяльності **ЗОБОВ'ЯЗУЮСЯ**:

– дотримуватися:

- вимог законодавства України та внутрішніх нормативних документів університету, зокрема Статуту Університету;
- принципів та правил академічної доброчесності;
- нульової толерантності до академічного плагіату;
- моральних норм та правил етичної поведінки;
- толерантного ставлення до інших;
- дотримуватися високого рівня культури спілкування;

– надавати згоду на:

- безпосередню перевірку курсових, кваліфікаційних робіт тощо на ознаки наявності академічного плагіату за допомогою спеціалізованих програмних продуктів;
- оброблення, збереження й розміщення кваліфікаційних робіт у відкритому доступі в інституційному репозитарії;
- використання робіт для перевірки на ознаки наявності академічного плагіату в інших роботах виключно з метою виявлення можливих ознак академічного плагіату;

– самостійно виконувати навчальні завдання, завдання поточного й підсумкового контролю результатів навчання;

– надавати достовірну інформацію щодо результатів власної навчальної (наукової, творчої) діяльності, використаних методик досліджень та джерел інформації;

– не використовувати результати досліджень інших авторів без використання покликань на їхню роботу;

– своєю діяльністю сприяти збереженню та примноженню традицій університету, формуванню його позитивного іміджу;

– не чинити правопорушень і не сприяти їхньому скоєнню іншими особами;

– підтримувати атмосферу довіри, взаємної відповідальності та співпраці в освітньому середовищі;

– поважати честь, гідність та особисту недоторканність особи, незважаючи на її стать, вік, матеріальний стан, соціальне становище, расову належність, релігійні й політичні переконання;

– не дискримінувати людей на підставі академічного статусу, а також за національною, расовою, статевою чи іншою належністю;

– відповідально ставитися до своїх обов'язків, вчасно та сумлінно виконувати необхідні навчальні та науково-дослідницькі завдання;

– запобігати виникненню у своїй діяльності конфлікту інтересів, зокрема не використовувати службових і родинних зв'язків з метою отримання нечесної переваги в навчальній, науковій і трудовій діяльності;

– не брати участі в будь-якій діяльності, пов'язаній із обманом, нечесністю, списуванням, фабрикацією;

– не підроблювати документи;

– не поширювати неправдиву та компрометуючу інформацію про інших здобувачів вищої освіти, викладачів і співробітників;

– не отримувати і не пропонувати винагород за несправедливе отримання будь-яких переваг або здійснення впливу на зміну отриманої академічної оцінки;

– не залякувати й не проявляти агресії та насильства проти інших, сексуальні домагання;

– не завдавати шкоди матеріальним цінностям, матеріально-технічній базі університету та особистій власності інших студентів та/або працівників;

– не використовувати без дозволу ректорату (деканату) символіки університету в заходах, не пов'язаних з діяльністю університету;

– не здійснювати і не заохочувати будь-яких спроб, спрямованих на те, щоб за допомогою нечесних і негідних методів досягати власних корисних цілей;

– не завдавати загрози власному здоров'ю або безпеці іншим студентам та/або працівникам.

УСВІДОМЛЮЮ, що відповідно до чинного законодавства у разі недотримання Кодексу академічної доброчесності буду нести академічну та/або інші види відповідальності й до мене можуть бути застосовані заходи дисциплінарного характеру за порушення принципів академічної доброчесності.

20.10.2021		Болокан В.А
(дата)	(підпис)	(ім'я, прізвище)