ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра інженерії програмного забезпечення

Пояснювальна записка

до бакалаврської роботи
на ступінь вищої освіти бакалавр

на тему: **«РОЗРОБКА ДОДАТКУ ДЛЯ НАВЧАННЯ ДИСЦИПЛІНИ "АЛГОРИТМИ ТА СТРУКТУРИ ДАНИХ" МОВОЮ JAVASCRIPT»**

Виконав: студент 5 курсу, групи ППЗ-51
спеціальності
121 Інженерія програмного забезпечення
(шифр і назва спеціальності)
Комаров О.В.
(прізвище та ініціали)
Керівник Негоденко О.В.
(прізвище та ініціали)
Рецензент
(прізвище та ініціали)

Київ – 2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти – «Бакалавр»

Спеціальність підготовки – 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ

Завідувач кафедри

Інженерії програмного забезпечення

_____ Негоденко О.В.

“_ ”_____ 2023 року

З А В Д А Н Н Я

НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТА

КОМАРОВА ОЛЕКСАНДРА ВАЛЕРІЙОВИЧА

(прізвище, ім’я, по батькові)

1. Тема роботи: «Розробка додатку для навчання дисципліни "Алгоритми та структури даних" мовою JavaScript»

Керівник роботи: Негоденко О.В., к.т.н., доцент

(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом вищого навчального закладу від «24» лютого 2023 року № 26.

2. Строк подання студентом роботи «1» червня 2023 року

3. Вхідні дані до роботи:

3.1 Науково-технічна література з питань, пов’язаних з програмним забезпеченнямщодо розробки веб-додатків.

3.2 Офіційна документація мови JavaScript.

4. Офіційна документація Mongo DB.
5. Зміст розрахунково-пояснювальної записки(перелік питань, які потрібно розробити).

5.1 Аналіз програмного забезпечення для навчання.

5.2 Інструментальні програмні засоби для розробки додатку для навчання дисципліни «Алгоритми та структури даних».

5.3 Інструментальні засоби роботи з базою даних.

5.4 Тестування розробленого програмного забезпечення.

5.5 Висновки.

6. Перелік демонстраційного матеріалу (назва основних слайдів)

6.1 Титульний слайд.

6.2 Мета, об’єкт та предмет дослідження.

6.3 Задачі дипломної роботи.

6.4 Аналіз аналогів.

6.5 Вимоги до додатку.

6.6 Програмні засоби реалізації.

6.7 Діаграма варіантів використання.

6.8 Діаграма класів.

6.9 Схема бази даних.

6.10 Екранні форми.

6.11 Апробація результатів дослідження.

6.12 Висновки.

6.13 Кінцевий слайд.

7. Дата видачі завдання «25» лютого 2023р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури	25.02.23 – 03.03.23	Виконано
2	Дослідження аналогів та актуальності додатку	04.03.23 – 18.03.23	Виконано
3	Аналіз та вибір інструментів для розробки додатку	19.03.23 – 25.03.23	Виконано
4	Проєктування та реалізація	26.03.23 – 30.04.23	Виконано
5	Вступ, висновки, реферат	1.05.23 – 10.05.23	Виконано
6	Розробка обов‘язкових демонстраційних матеріалів	11.05.23 – 18.05.23	Виконано
7	Попередній захист роботи	31.05.2023	Виконано
8	Здача роботи	1.06.23	Виконано

Студент_____ Комаров О.В.

(підпис) (прізвище та ініціали)

Керівник роботи_____ Негоденко О.В.

(підпис) (прізвище та ініціали)

РЕФЕРАТ

Текстова частина бакалаврської роботи 73 с., 63 рис., 2 табл., 21 джерело.

РОЗРОБКА ДОДАТКУ ДЛЯ НАВЧАННЯ ДИСЦИПЛІНИ "АЛГОРИТМИ ТА СТРУКТУРИ ДАНИХ" МОВОЮ JAVASCRIPT, ВЕБ-ДОДАТОК.

Об'єктом дослідження – процес навчання з дисципліни "Алгоритми та структури даних".

Предмет дослідження – додаток для навчання з дисципліни "Алгоритми та структури даних".

Метою роботи – спрощення процесу навчання з дисципліни "Алгоритми та структури даних" засобами додатку, створеного мовою JavaScript.

Методи дослідження – методи проєктування та розробки програмного забезпечення, методи опрацювання та аналізу отриманих результатів, методи тестування програмного забезпечення.

У роботі проведено аналіз існуючих програмних засобів, таких як VisuAlgo, Algorithm Visualizer, LeetCode, Exercism. Визначено переваги та можливості сучасних програмних засобів для навчання. Вибрано інструментальні програмні засоби для розробки додатку для навчання. Вибрано інструментальні засоби роботи з базою даних. Розроблено додаток для підтримки навчального процесу дисципліни «Алгоритми та структури даних». Протестовано розроблене програмне забезпечення для навчання дисципліни.

Практична значущість результатів: Даний продукт може бути використаний для навчання курсу "Алгоритми та Структури даних".

Галузь використання — навчальний процес.

ЗМІСТ

ВСТУП

Знання алгоритмів та структур даних є необхідною складовою професійної компетентності програміста. Для розробки ефективного та швидкого програмного забезпечення вміння обирати доцільний алгоритм дій та використовувати відповідні структури даних є незамінними.

Структури даних і алгоритми допомагають розуміти, як працюють системи, додатки та програми, та як можна покращити їх ефективність.

Крім того, знання алгоритмів і структур даних дозволяє підвищити ефективність роботи програміста, розвинути аналітичні здібності та алгоритмічне мислення. Знання загальних принципів алгоритмів і структур даних допомагає програмісту ухвалювати правильні рішення та обирати для розв'язання завдань найбільш оптимальні інструменти.

Об’єкт – процес навчання з дисципліни "Алгоритми та структури даних".

Предмет дослідження – додаток для навчання з дисципліни "Алгоритми та структури даних".

Мета роботи – спрощення процесу навчання з дисципліни "Алгоритми та структури даних" засобами додатку, створеного мовою JavaScript.

Для досягнення поставленої мети слід виконати наступні завдання:

1. Провести аналіз програмного забезпечення для навчання.
2. Визначити переваги та можливості сучасних програмних засобів для навчання.
3. Вибрати інструментальні програмні засоби, що мають забезпечити розробку додатку для навчання дисципліни «Алгоритми та структури даних».
4. Вибрати інструментальні засоби роботи з базою даних.
5. Протестувати розроблене програмне забезпечення.

РОЗДІЛ 1

АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Переваги вивчення дисципліни "Алгоритми та Структури даних" за допомогою програмного забезпечення

Використання програмного забезпечення для вивчення будь-якої дисципліни має багато переваг. Воно дозволяє зробити процес вивчення цієї дисципліни простішим, цікавішим. А також покращує організацію дистанційного та індивідуального навчання. Та допомагає організувати наочний контроль за проходженням навчання для студентів та викладачів.

Крім того, це покращує рівень інформаційної компетентності викладачів та студентів, та підвищує ефективність їх взаємодії.

Програмний продукт створюється на базі курсу "Алгоритми та Структури даних" Державного університету телекомунікацій.

Програма має модульну структуру, так що можна додавати навчальні модулі.

Кожна із розглянутих тем складається з таких розділів:

- Теорія
- Приклади
- Завдання
- Аналіз результатів

Перевагою даного продукту являється можливість використовувати його різними мовами.

1.2 Аналіз аналогів

Розглянемо програми, у яких є можливість вивчати Алгоритми та

Структури даних. Слід зазначити, що деякі з них є комерційними – потребують плату за використання.

1.2.1 VisuAlgo

Це веб-інструмент вивчення та візуалізації алгоритмів

Рисунок 1.1 – Види візуалізованих структур даних у VisuAlgo

VisuAlgo має два основні компоненти: 24 сторінки візуалізації та пов’язаний з ними компонент онлайн-вікторини.

Кожна сторінка візуалізації має «режим електронної лекції», який доступний у верхньому правому кутку.

Наприклад, для зв’язаного списку (Linked List)

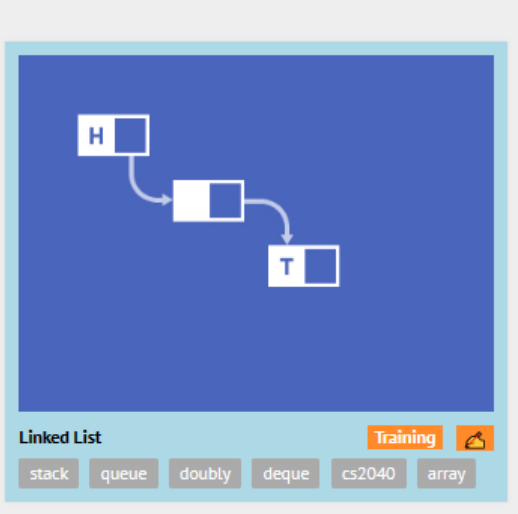


Рисунок 1.2 – Візуалізація зв'язаного списку у VisuAlgo

Можна розглянути теорію:

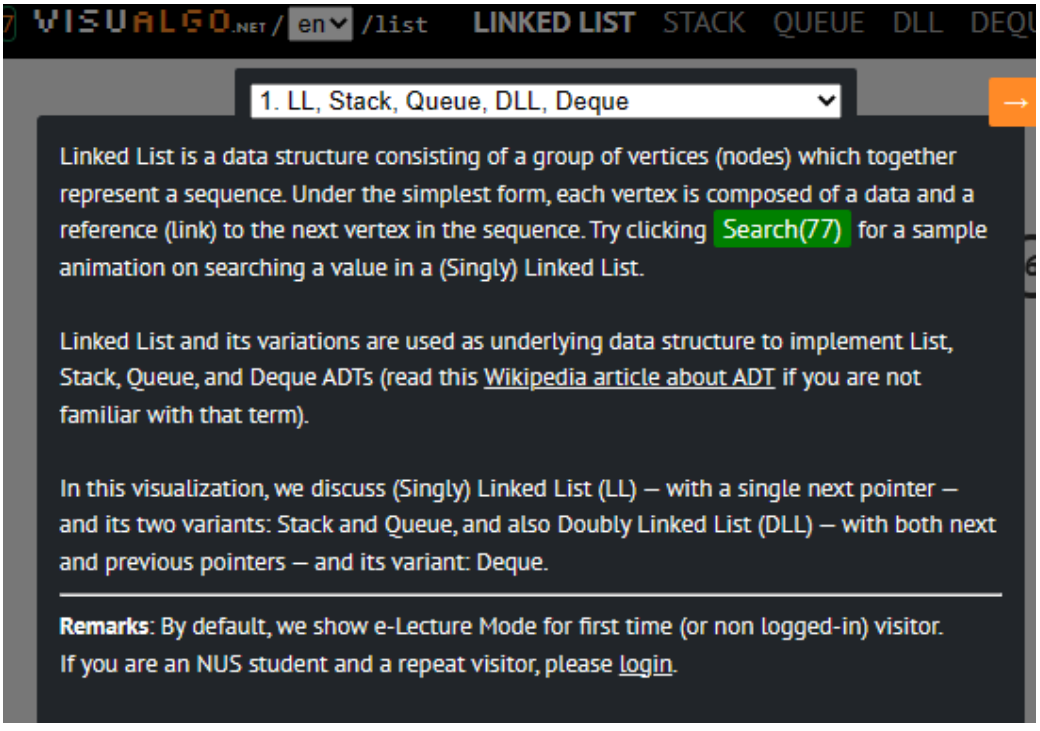


Рисунок 1.3 – Представлення теоретичного матеріалу у VisuAlgo

Розв'язати задачі за цією темою:

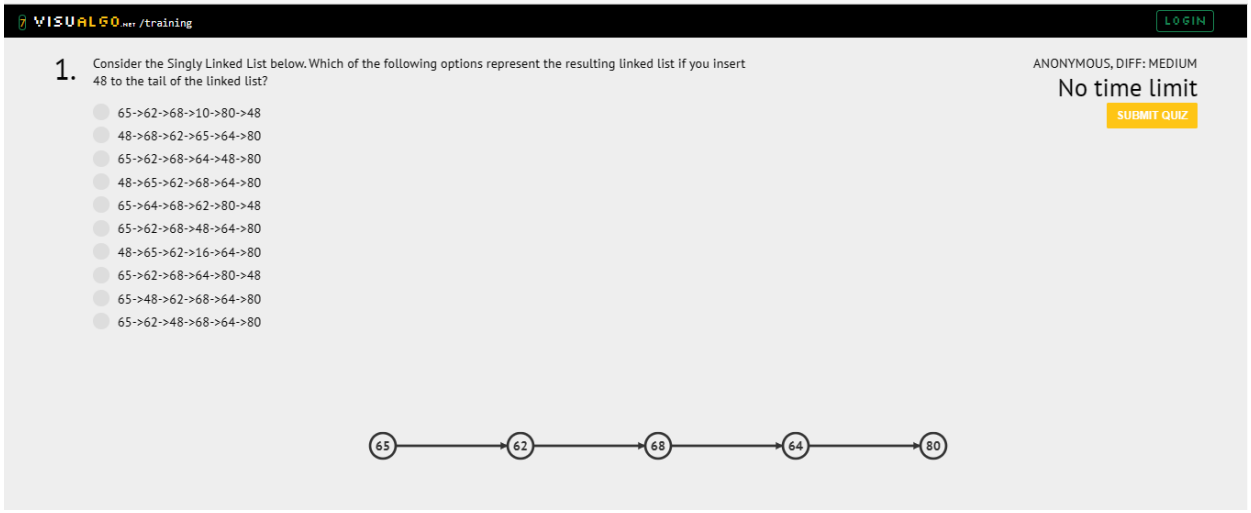
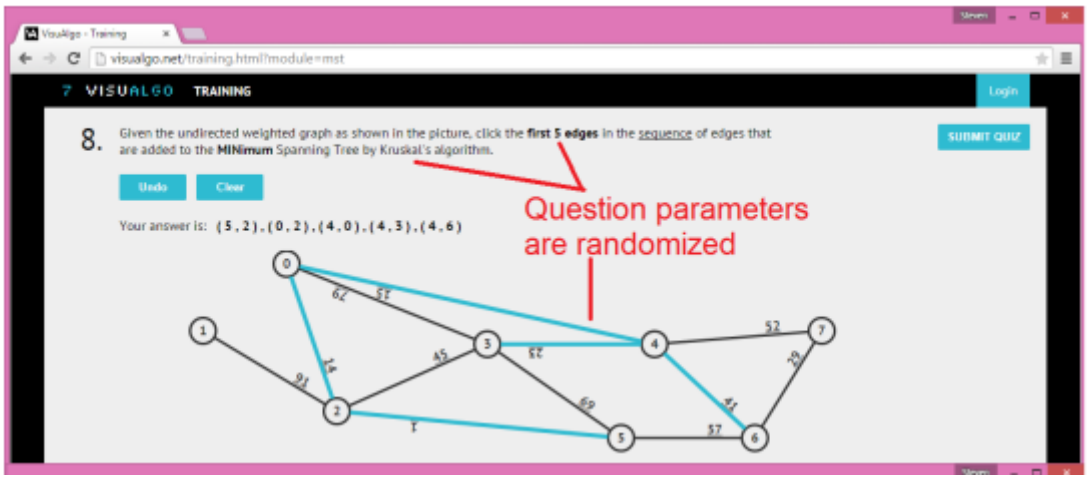


Рисунок 1.4 – Тест за темою у VisuAlgo

VisuAlgo має такі сильні сторони:

- Це веб-інструмент, тому не потребує інсталювати додаткове програмне забезпечення.
- Він використовує нові веб-технології: HTML5, CSS3, JavaScript.
- Це набір візуалізацій алгоритмів з єдиним інтерфейсом.
- Він дозволяє користувачам створювати власні вхідні алгоритми, і візуалізація працюватиме з цими входами.

Ось наприклад:



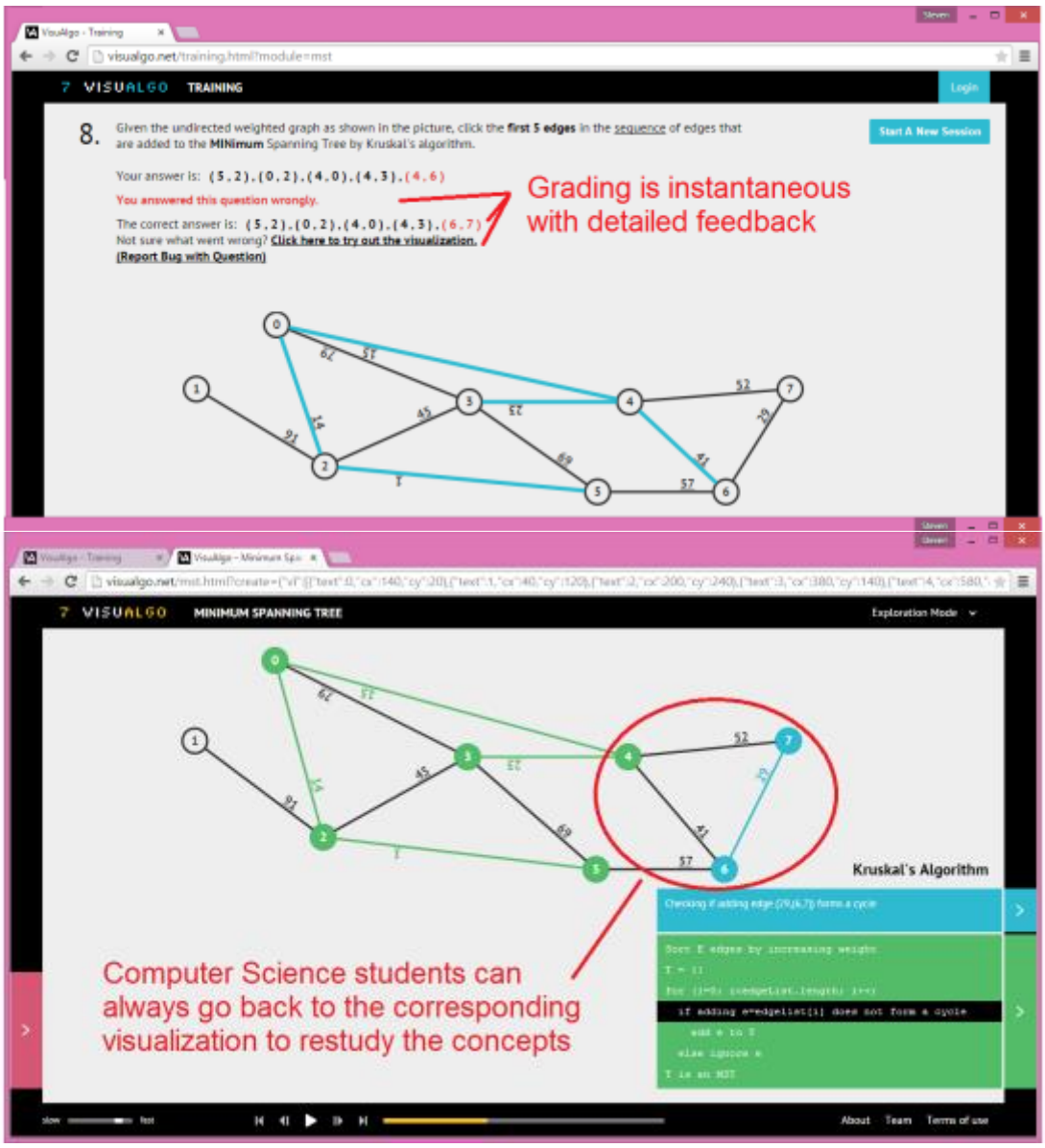


Рисунок 1.5 – Можливість задавати вхідні дані алгоритму у VisuAlgo

1.2.2 Algorithm Visualizer

Algorithm Visualizer – це інтерактивна онлайн-платформа, яка візуалізує алгоритми з коду.

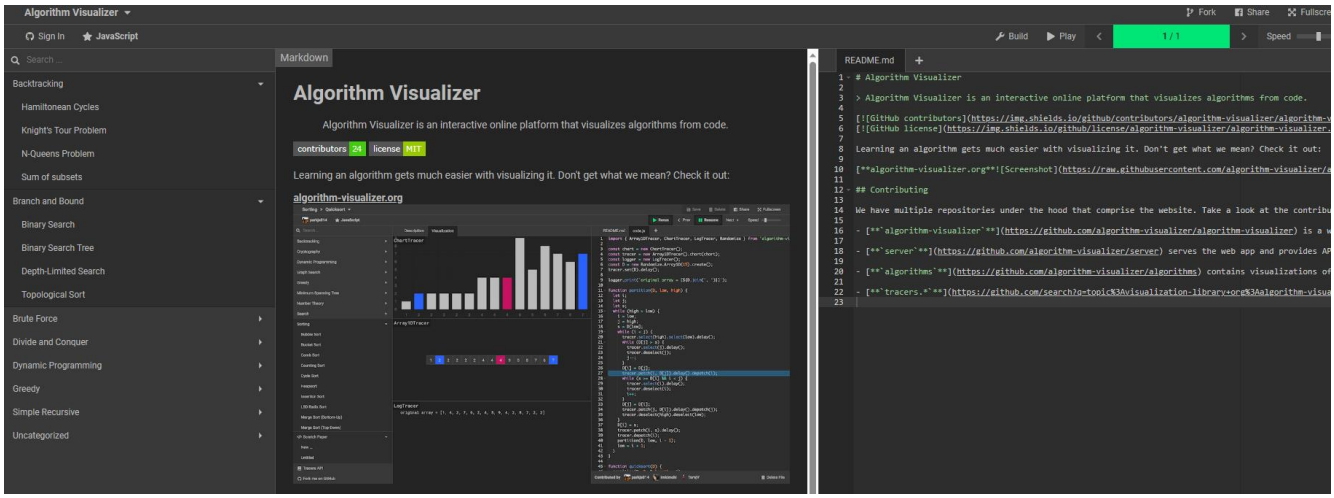


Рисунок 1.6 – Вікно довідки Algorithm Visualizer

Вивчити алгоритм стає набагато простіше, візуалізувавши його.

Тут є можливість працювати з кількома репозиторіями

Це веб-програма, написана на React. Він містить компоненти інтерфейсу користувача та інтерпретує команди для візуалізації.

Сервер обслуговує веб-програму та надає API, які їй потрібні на льоту. (наприклад, вхід на GitHub, компіляція/запуск коду тощо)

Програма містить візуалізацію алгоритмів, показаних у бічному меню веб-сайту.

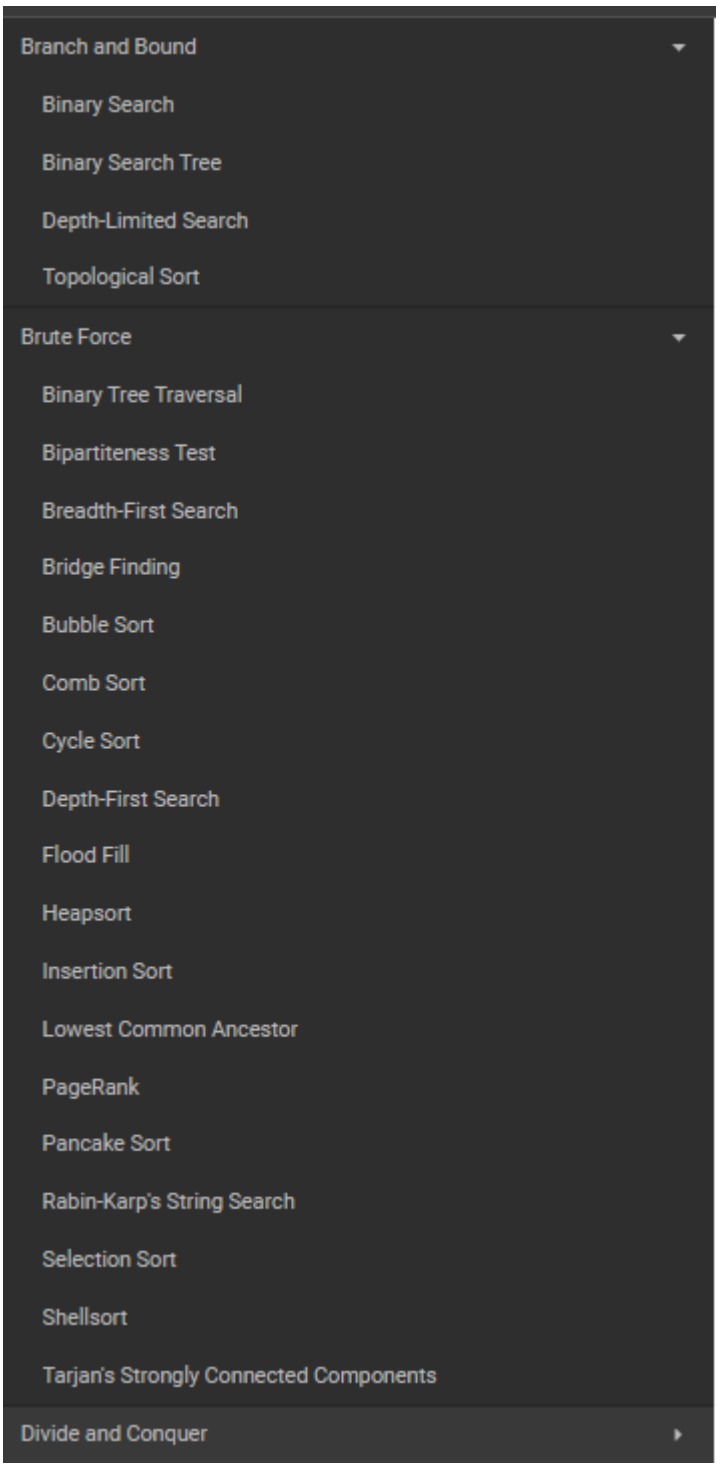
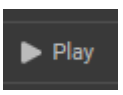


Рисунок 1.7 – Перелік алгоритмів, візуалізованих в Algorithm Visualizer

Код на JavaScript, що відображається в правій частині екрану, можна змінювати і побачити, до чого призведуть ці зміни, натиснувши кнопку 

В нижній частині виводиться лог виконуваного коду.

Ось як це виглядає, наприклад для алгоритму обходу бінарного дерева (Binary Tree Travel)

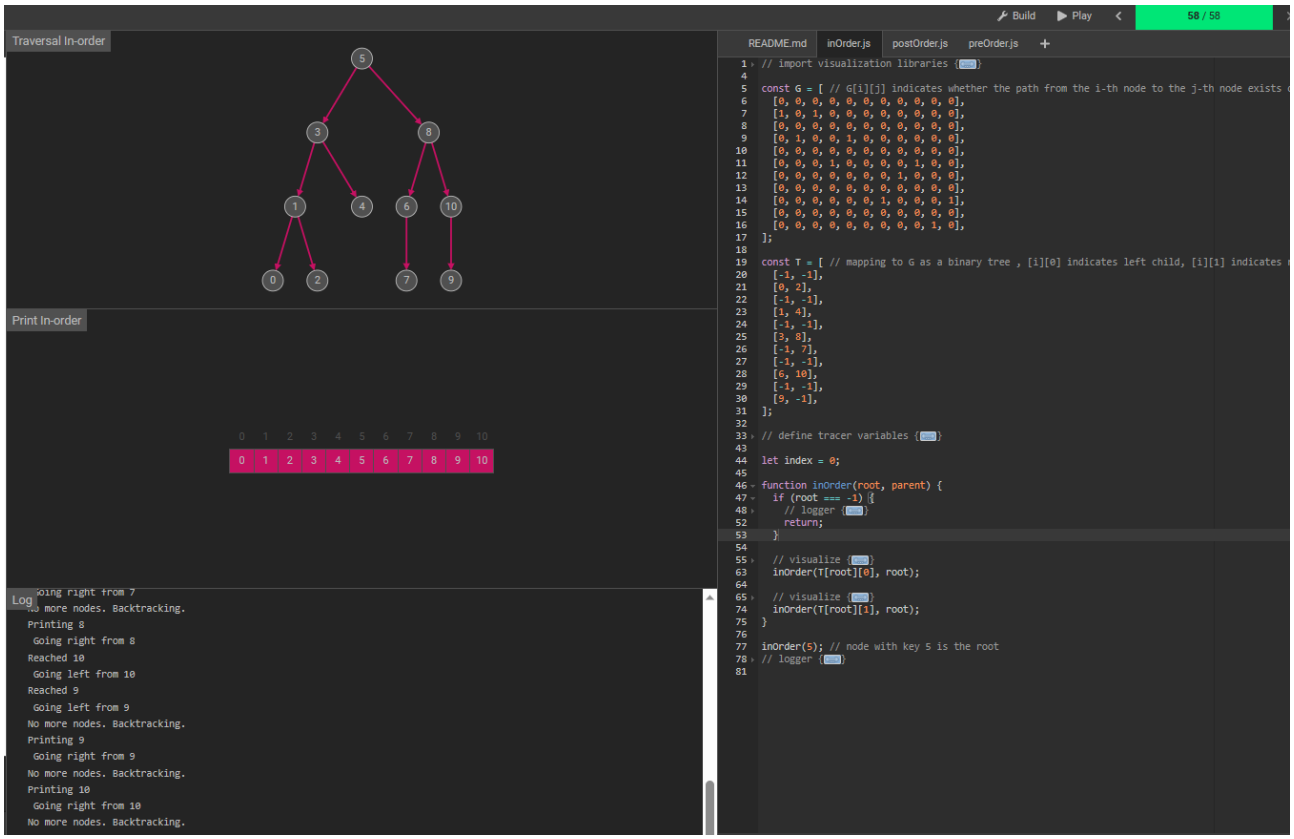


Рисунок 1.8 – Приклад візуалізації алгоритму обходу бінарного дерева в Algorithm Visualizer

1.2.3 LeetCode

Це онлайн-платформа з алгоритмічними завданнями по програмуванню, що може використовуватись як тренажер для підготовки до технічних співбесід.

Вважається, що якщо регулярно з ним займатися, то через 6–12 місяців шанси влаштуватися в будь-якому серйозну компанію помітно виростуть.

При пошуках роботи розробника програмного забезпечення обов’язковим кроком буде співбесіда з кодування.

LeetCode містить 93+ покрокових інструкцій прикладів задач. Ці задачі використовуються для демонстрації загальних шаблонів, способів їх розпізнавання та реалізації рішень.

67+ підібраних практичних задач.

120+ наочних посібників. Вони варіюються від анімації до повних відео, які проходять через алгоритм.

12 вікторин. Кожна структура даних або алгоритм має вікторину в кінці розділу, щоб перевірити знання

137+ бонусних задач.

Платформа дає доступ до сервера чату, де можна зустрітися з іншими слухачами курсу та автором курсу. Там можна формувати навчальні групи, організовувати пробні інтерв'ю та задавати запитання про курс, структури даних і алгоритми чи інтерв'ю загалом.

Інтегроване середовище кодування LeetCode, яке підтримує 19 різних мов.

LeetCode має такі можливості:

Explore – це добре організований інструмент, який допоможе отримати максимальну віддачу від LeetCode, пройти навчання, розвя'зати багато задач по програмуванню, та набути практичного досвіду.

Questions, Community & Contests (Запитання, спілкування та конкурси)

Більше 2750 запитань, щоб потренуватися. Можна приєднатися до однієї з найбільших технічних спільнот із сотнями тисяч активних користувачів і брати участь у конкурсах, щоб випробувати себе та отримати нагороди.

Компанії та кандидати

LeetCode не тільки готує кандидатів до технічних співбесід, але також допомагає компаніям визначити найкращі технічні таланти.

Developer (Розробник)

Зараз підтримує 14 популярних мов програмування. Інструменти розробки, такі як Playground, допомагають тестувати, налагоджувати та навіть писати власні Проєкти онлайн.

Leetcode Explore дозволяє покращити свої знання та навики з багатьох тем.

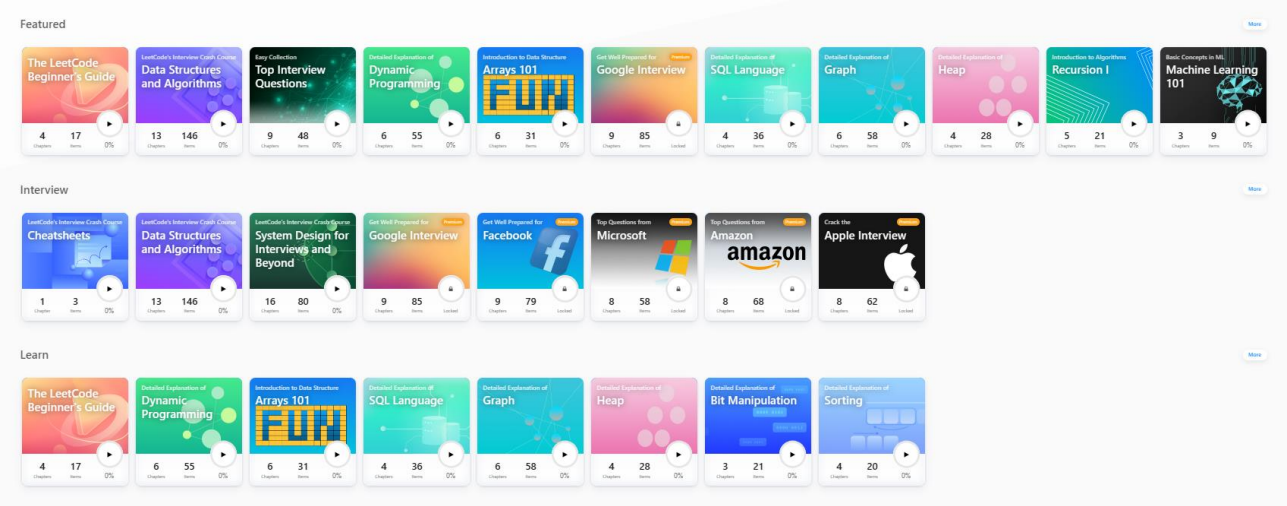


Рисунок 1.9 – Теми в Leetcode

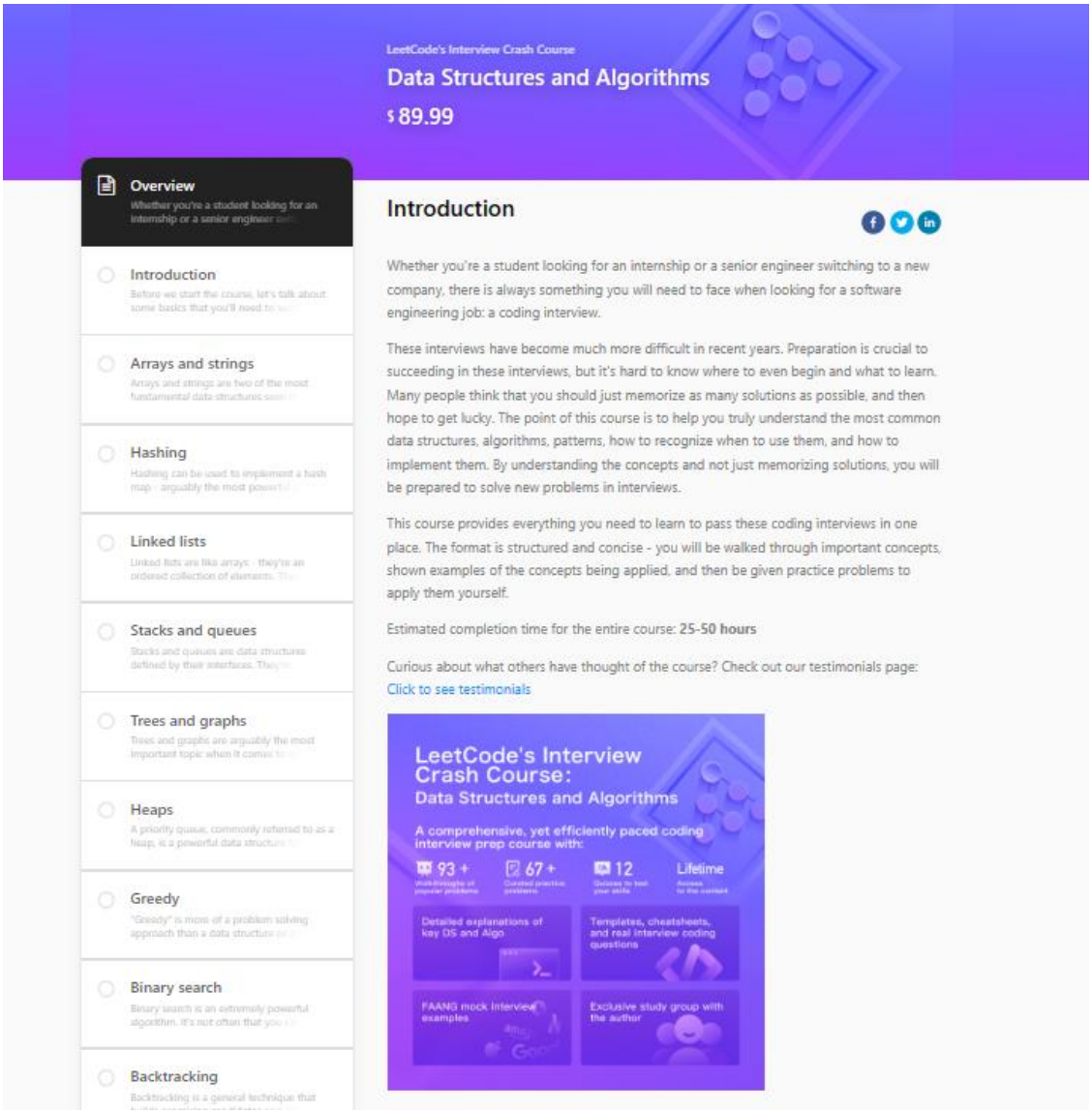


Рисунок 1.10 – Вікно огляду розділу Структури даних та Алгоритми в Leetcode

Видно, що обсяг тем достатній для досить глибокого вивчення цього розділу

- Arrays and strings (Масиви та рядки)
- Hashmaps and sets (Хеш-карти та набори)
- Linked lists (Зв'язані списки)
- Stacks and queues (Стеки та черги)
- Trees and graphs (Дерева та графи)
- Heaps (Купи)
- Greedy algorithms (Жадібні алгоритми)
- Binary search (Бінарний пошук)
- Backtracking (Відстеження_
- Dynamic programming (Динамічне програмування)

Суть цього курсу полягає в тому, щоб допомогти зрозуміти найпоширеніші структури даних, алгоритми, шаблони, як розпізнати, коли їх використовувати, і як їх реалізувати.

Розуміючи поняття, а не просто запам’ятовуючи рішення, можна буде підготуватись вирішувати нові проблеми під час співбесід.

Цей курс містить усе, що вам потрібно, щоб навчитися проходити співбесіди з кодування в одному місці. Формат є структурованим і лаконічним – вас ознайомлять із важливими поняттями, покажуть приклади застосовуваних понять, а потім отримають практичні завдання, щоб застосувати їх самостійно.

Курс розділений на розділи. Перший розділ – це короткий вступ, у якому розповідається про фундаментальні навички, необхідні для успішного вивчення цього курсу, а також опис курсу.

У кожному з цих розділів можна знайти таку інформацію:

Пояснення структури даних/алгоритму, для чого він корисний, як його можна використовувати для вирішення проблем, а також деталі реалізації та складність часу/простору.

Загальні шаблони та прийоми, пов’язані зі структурою даних або алгоритмом.

Кілька прикладів покрокових інструкцій відповідних проблем, які допоможуть проілюструвати концепції, доповнені наочними посібниками, стислими поясненнями та аналізом складності часу/простору.

Точно підібрані практичні завдання, які допоможуть вам розвинути пам’ять завдяки тому, що ви навчилися.

У кінці кожного розділу також надається список бонусних завдань. Ці завдання є необов’язковими і не вважаються частиною курсу, але їх можна використовувати для закріплення ідей, викладених у кожному розділі.

Тут також міститься кілька корисних інструментів, які можна використовувати в майбутньому. Існують шаблони коду для всіх загальних шаблонів, шпаргалки щодо часових і просторових складнощів, а також блок-схема, яку можна використовувати як загальне керівництво, намагаючись з’ясувати, яку структуру даних або алгоритм слід використовувати.

Також розбираються етапи співбесіди з програмування та те, що слід робити на кожному етапі. Щоб продемонструвати цю пораду, є приклади відео інтерв’ю FAANG.

Цей курс розроблений, щоб допомогти кожному підвищити свої навички та впевненість у кодуванні під час співбесід.

1.2.4 Exercism

Ціль цього інструменту – допомогти людям покращити свої навики програмування на різних мов.

На сайті організовано навчання різним мовам програмування. Найкраще всього сайт підійде для того, хто вже знає одну мову і хоче вивчити нову. Для навчання доступно 67 мов, найбільше завдань на C#, Python і JavaScript.

У нього є код сайту, код API і код командного рядка. У нього є інструменти та сервіси, метадані та вправи.

У Exercism є 40+ автономних репозиторіїв на 40+ різних мовах.

Ось приклад мов, що можна вивчати:

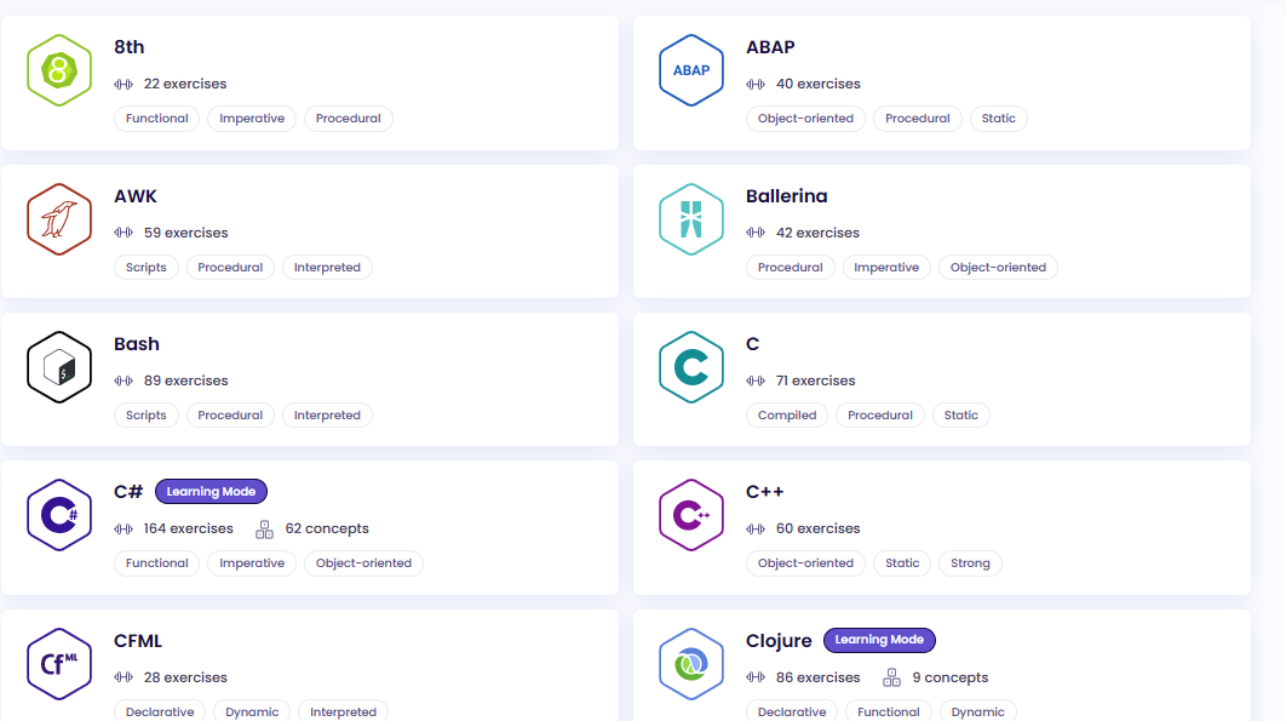


Рисунок 1.10 – Вікно з переліком мов програмування, які можна вивчати на платформі Exercism

Сайт Exercism безкоштовний, працює завдяки спільноті. Користувачі діляться на учнів та наставників. Учні виконують завдання, а викладачі проводять перевірку коду і дають поради. Коментарі від наставників корисні та продумані

Виконання завдань відбувається таким чином:

Після вибору розділу треба встановити консольне додаток. Треба запустити команду, зазначену в описі завдання на сайті, і додаток формує каталог, файл із шаблоном для виконання завдання та файл із тестами.

Виконуємо завдання, перевіряємо, що рішення проходить тести. Виконати завдання і провести тести можна в будь-якому IDE

Виконуємо ще одну консольну команду і завдання відправляється на перевірку.

Після виправлення зауважень від наставника, рішення знову відправляється на перевірку. І так поки наставник не схвалить рішення. Потім відкривається наступна задача.

Такий принцип застосовується до завдань із основного трека по мові. Крім них є ще додаткові завдання. Огляд коду від наставника в них не обов'язковий, тому можна переглянути рішення інших учасників, а також поспілкуватися та обговорити варіанти рішень.

РОЗДІЛ 2

АНАЛІЗ ІНСТРУМЕНТАЛЬНИХ ЗАСОБІВ РЕАЛІЗАЦІЇ

2.1 Огляд використаних технологій програмування

2.1.1 HTML

HTML (HyperText Markup Language – мова розмітки гіпертексту) – це код, який використовується для структурування веб-сторінки та її вмісту. Наприклад, вміст може бути структурований у вигляді набору абзаців, списку маркованих пунктів або за допомогою зображень і таблиць даних. Як випливає з назви, ця стаття дасть вам базове розуміння HTML та її функцій.

HTML – це мова розмітки, яка визначає структуру вашого контенту. HTML складається з низки елементів, які ви використовуєте, щоб вкласти або обернути різні частини контенту, щоб він виглядав певним чином або діяв певним чином. За допомогою тегів-обгорток можна зробити слово або зображення гіперпосиланням на інше місце, виділити слова курсивом, зробити шрифт більшим або меншим і так далі.

Структура елемента HTML

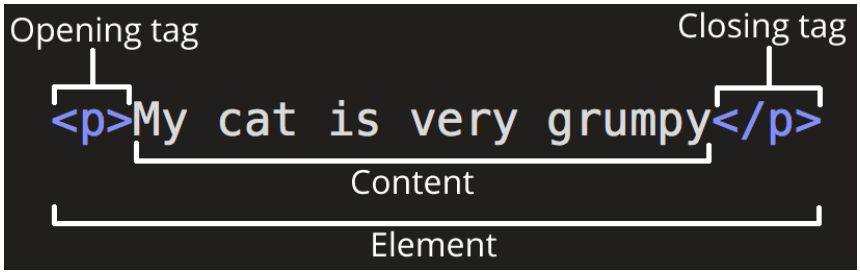


Рисунок 2.1 – Складові частини тегу

Основні частини елемента наступні:

1. Відкриваючий тег: .
2. Закриваючий тег: .
3. Вміст (Content): Текст між тегами.
4. Елемент (Element): Все разом.

Елементи також можуть мати атрибути, які виглядають наступним чином:

Рисунок 2.2 – Атрибути тегу

Атрибути містять додаткову інформацію про елемент. Тут class – це назва атрибута, а editor-note – значення атрибута. Атрибут class дозволяє маркувати елементи однаковим чином.

Вкладення елементів

Ви також можете поміщати елементи всередину інших елементів, але при цьому необхідно слідкувати за порядком відкриття та закриття тегів.

Елементи повинні відкриватися і закриватися правильно, щоб вони були чітко всередині або зовні один одного.

Структура HTML-документа

```
<!DOCTYPE html>
<html lang="en-US">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width" />
    <title>My test page</title>
  </head>
  <body>
    <img src="images/firefox-icon.png" alt="My test image" />
  </body>
</html>
```

Рисунок 2.3 – Структура HTML-документа

Тут бачимо такі елементи:

- <!DOCTYPE html> – тип документа.
- <html></html> – елемент <html>. Цей елемент обгортає весь вміст на всій сторінці і іноді його називають кореневим елементом. Він також включає

атрибут lang, який задає основну мову документа.

- <head></head> – елемент <head>. Цей елемент містить технічну інформацію про HTML-сторінку (ключові слова та опис сторінки, які ви хочете відобразити в результатах пошуку, CSS для стилізації вмісту, оголошення наборів символів тощо).
- <meta charset="utf-8"> – Цей елемент встановлює кодування символів, яку повинен використовувати ваш документ, на UTF-8, яка включає більшість символів з переважної більшості письмових мов.
- <meta name="viewport" content="width=device-width"> – Цей елемент забезпечує відображення сторінки по ширині вікна перегляду, не дозволяючи мобільним браузерам відображати сторінки ширше за вікно перегляду, а потім зменшувати їх.
- <title></title> – елемент <title>. Він задає заголовок вашої сторінки, який з'являється у вкладці браузера, в якій завантажується сторінка. Він також використовується для опису сторінки, коли ви додаєте її до закладок/вибраного.
- <body></body> – елемент <body>. Він містить видимий контент сторінки.

2.1.2 CSS

CSS (Cascading Style Sheets) – це код, який стилізує веб-сторінку.

Як і HTML, CSS не є мовою програмування. Це також не мова розмітки. CSS – це мова таблиць стилів. CSS – це те, що ви використовуєте для вибіркового оформлення елементів HTML. Наприклад, цей CSS виділяє текст абзацу, встановлюючи червоний колір:

```
p {
  color: red;
}
```

Рисунок 2.4 – Приклад правила CSS

Щоб код працював, потрібно застосувати цей CSS (вище) до HTML-документа. Для цього треба вставити наступний рядок у заголовок (між тегами <head> і </head>):

```
<link href="styles/style.css" rel="stylesheet" />
```

Рисунок 2.5 – Підключення CSS до сторінки

Структура набору правил CSS

Давайте розберемо код CSS для тексту червоного абзацу, щоб зрозуміти, як він працює:

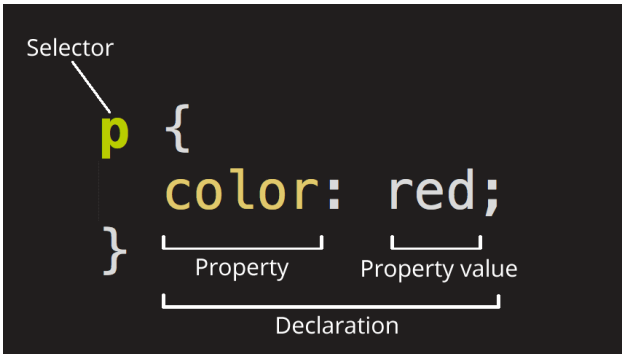


Рисунок 2.6 – Структура набору правил CSS

Вся структура називається набором правил. (Термін "набір правил" часто називають просто "правило".)

Селектор

Це ім'я HTML-елемента на початку набору правил. Вона визначає елемент(и), який(і) буде(уть) стилізовано (у цьому прикладі – елементи <p>).

Декларація

Це єдине правило, як color: red;. Воно визначає, яку з властивостей елемента ви хочете стилізувати.

Параметри

Це способи, за допомогою яких ви можете змінити стиль елемента HTML. (У цьому прикладі колір є властивістю елементів <p>.) У CSS ви вибираєте, на які властивості ви хочете вплинути у правилі.

Значення параметра

Праворуч від параметра – після двокрапки – знаходиться його значення. Вибирається одне з багатьох можливих значень для даного параметру.

Ось інші частини наборів правил:

- Окрім селектора, кожен набір правил має бути загорнутий у фігурні дужки. ({})
- Усередині кожного визначення у наборі правил ви повинні використовувати двокрапку (:), щоб відокремити властивість від її значення або значень.
- Усередині кожного набору правил ви повинні використовувати крапку з комою (;), щоб відокремити кожну декларацію від наступної.

Різні типи селекторів

Таблиця 2.1 – Типи селекторів

Назва селектора	**Що він обирає**	**Приклад**
Селектор елементів (іноді його називають тегом або селектором типу)	Всі HTML-елементи вказаного типу.	p вибирає <p>
Селектор ідентифікатора	Елемент на сторінці з вказаним ідентифікатором. На даній HTML-сторінці кожне значення ідентифікатора має бути унікальним.	#my-id вибирає <p id="my-id"> або <a id="my-id">

Продовження таблиці 2.1 – Типи селекторів

Назва селектора	**Що він обирає**	**Приклад**
Вибір класу	Елемент(и) на сторінці з вказаним класом. На сторінці може бути декілька екземплярів одного класу.	.my-class вибирає <p class="my-class"> та <a class="my-class">
Селектор атрибутів	Елемент(и) на сторінці з вказаним атрибутом.	img[src] вибирає <img src="myimage.png">, але не <img>
Псевдо-селектор класів	Вказаний(і) елемент(и), але тільки коли він перебуває у вказаному стані. (Наприклад, коли курсор наведено на посилання).	a:hover вибирає <a>, але тільки коли вказівник миші наведено на посилання.

CSS: все про коробки

Дещо, що ви можете помітити при написанні CSS: багато чого в ньому стосується блоків. Сюди входить встановлення розміру, кольору та положення. Більшість HTML-елементів на вашій сторінці можна уявити собі як блоки, що стоять на інших блоках.

CSS-верстка здебільшого базується на моделі блоків. Кожен блок, що займає місце на вашій сторінці, має такі властивості, як

- padding, простір навколо вмісту. У наведеному нижче прикладі це простір навколо тексту абзацу.
- border, суцільна лінія, яка знаходиться безпосередньо за межами відступів.
- margin, простір ззовні від межі.

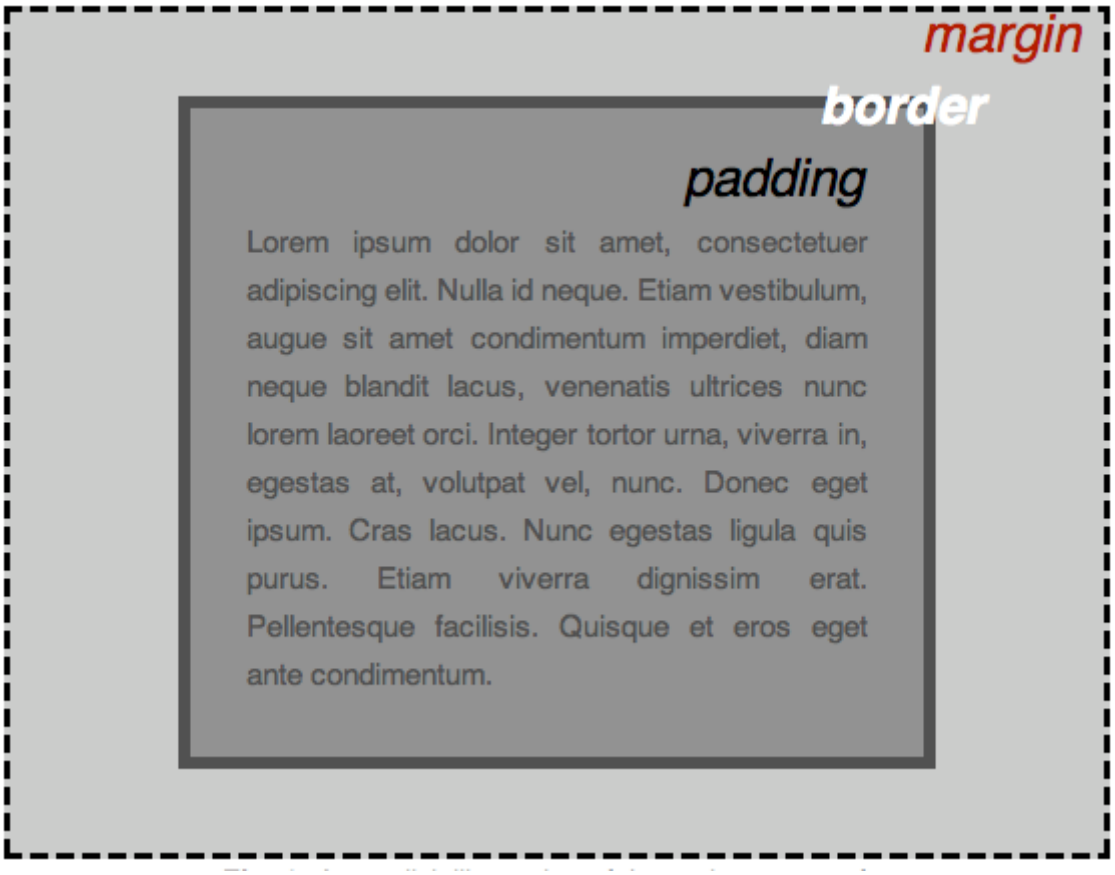


Рисунок 2.6 – Властивості блоку

У цьому розділі ми також використовуємо:

- width (ширина елемента).
- background-color, колір за вмістом елемента та відступами.
- color, колір вмісту елемента (зазвичай тексту).
- text-shadow задає тінь для тексту всередині елемента.
- display задає режим відображення елемента.

2.1.3 JavaScript

JavaScript – це мова програмування, яка дозволяє додавати на веб-сторінку взаємодію з користувачем. Наприклад, анімацію або програвання відео, перегляд інтерактивних мап тощо. Вихідний код сторінки містить посилання на файли JavaScript або окремі функції, що додає сторінці функціоналу програми: вона має змінні, пам’ять та інші особливості.

В основному, веб-сторінка складається з трьох компонентів: HTML, CSS та JavaScript.

JavaScript – це мова, яка дає змогу вам застосовувати складні речі на web-сторінці – щоразу, коли на web-сторінці відбувається щось більше, ніж просто її статичне відображення, – відображення періодично оновлюваного контенту, чи інтерактивних карт, чи анімація 2D/3D-графіки, чи прокручування відео в програвачі, і т.д. – можете бути впевнені, що швидше за все, не обійшлося без JavaScript. Це третій шар листкового пирога стандартних web-технологій, два з яких (HTML і CSS) ми детально розкрили в інших частинах навчального посібника.

Ядро мови JavaScript складається з деякої кількості звичайних можливостей, які дають змогу робити таке:

- Зберігати дані всередині змінних. У прикладі вище, ми, наприклад, запитували введення нового імені, яке потрібно було ввести, потім зберігали ім'я у змінній name.
- Операції над фрагментами текстів (відомими в програмуванні як "рядки"). У прикладі вище ми брали рядок "Player 1: " і приєднали його до значення змінної name для отримання повного тексту, наприклад: "Player 1: Chris".
- Запускати код відповідно до певних подій, що відбуваються на web-сторінці. У нашому прикладі вище, ми використовували click (en-US) подія, для визначення моменту, коли кнопка була клацнута, відповідно до цього запускався код, який оновлював текст.
- І багато іншого!

Ще більш захопливою є функціональність, створена поверх основної мови JavaScript. Так звані інтерфейси прикладного програмування (API) надають вам додаткові надздібності для використання у вашому коді JavaScript.

API – це готові набори блоків коду, які дають змогу розробнику реалізовувати програми, які в іншому разі було б важко або неможливо реалізувати. Вони роблять те саме для програмування, що готові комплекти

меблів роблять для домашнього будівництва – набагато простіше брати готові панелі та скручувати їх разом, щоб зробити книжкову полицю, ніж самому розробляти дизайн, ходити в пошуках правильної деревини, вирізати всі панелі необхідного розміру та форми, знайти відповідні гвинти, а потім зібрати їх разом, щоб зробити книжкову полицю.

API-інтерфейси браузера вбудовані у ваш веб-браузер і можуть відображати дані з навколишнього комп'ютерного оточення або робити корисні складні речі. Наприклад:

- API-інтерфейс DOM (Document Object Model) дає змогу вам маніпулювати HTML і CSS, створювати, видаляти і змінювати HTML, динамічно застосовувати нові стилі до вашої сторінки тощо. Кожного разу, коли ви бачите спливаюче вікно на сторінці або якийсь новий вміст, як ми бачили вище в нашому простому демо), наприклад, це DOM у дії.
- API геолокації витягує географічну інформацію. Так Google Maps може знайти ваше місце розташування і нанести його на карту.
- API Canvas і WebGL дають змогу створювати анімовані 2D і 3D-графіки. Люди роблять деякі дивовижні речі, використовуючи ці веб-технології – див. Chrome Experiments і webglsamples.
- Аудіо та відео API, як-от HTMLMediaElement і WebRTC, дають змогу робити справді цікаві речі з мультимедіа, як-от відтворення аудіо та відео просто на вебсторінці, або захоплення відео з веб-камери та відображення його на чужому комп'ютері (спробуйте наш простий демонстраційний знімок, щоб зрозуміти ідею).

Код JavaScript виконується JavaScript-движком браузера, після того як код HTML і CSS був оброблений і сформований у веб-сторінку. Це гарантує, що структура і стиль сторінки вже сформовані до моменту запуску JavaScript.

JavaScript є інтерпретованою мовою – код запускається зверху вниз і результат запуску негайно повертається. Вам не потрібно перетворювати код в іншу форму, перед запуском у браузері.

Клієнтський код – це код, який запускається на комп'ютері користувача. Під

час перегляду веб-сторінки, клієнтський код завантажується, а потім запускається і відображається браузером.

З іншого боку, серверний код запускається на сервері, потім його результати завантажуються і відображаються в браузері. Приклади популярних серверних веб-мов включають PHP, Python, Ruby і ASP.NET. І JavaScript! JavaScript так само може використовуватися, як серверна мова, наприклад у популярному середовищі Node.js.

Так само, як і в HTML і CSS, можливо писати коментарі у вашому JavaScript-коді, що буде проігноровано браузером, та існує тільки для того, щоб давати підказки вашим друзям-розробникам про те, як працює код (і особисто вам, якщо ви повернетеся до коду через 6 місяців і не зможете згадати, що ви робили).

2.1.2 React

React – це декларативна, ефективна та гнучка JavaScript-бібліотека для створення користувацьких інтерфейсів. Вона дає змогу вам збирати складний UI з маленьких ізольованих шматочків коду, які називаються "компонентами".

React має кілька різних видів компонентів, але ми почнемо з підкласів `React.Component`:

```
class ShoppingList extends React.Component {
  render() {
    return (
      <div className="shopping-list">
        <h1>Список покупок для {this.props.name}</h1>
        <ul>
          <li>Instagram</li>
          <li>WhatsApp</li>
          <li>Oculus</li>
        </ul>
      </div>
    );
  }
}

// Пример использования: <ShoppingList name="Марк" />
```

Рисунок 2.2 – Приклад класу, написаного з використанням бібілотеки React

Ми використовуємо компоненти, щоб повідомити React, що ми хочемо побачити на екрані. Щоразу, коли наші дані змінюються, React ефективно оновлює і повторно рендерить наші компоненти.

ShoppingList є прикладом класового компонента React. Компонент приймає параметри, які називаються пропсами (props, скорочення від properties – властивості), і повертає з методу render() ієрархію подань для відображення.

Метод render повертає опис того, що ви хочете побачити на екрані. React бере цей опис і відображає результат. Якщо точніше, render повертає React-елемент, який є легковажним описом того, що потрібно відрендерити. Більшість React-розробників використовують спеціальний синтаксис під назвою "JSX" для спрощення опису структури. Під час компіляції синтаксис перетворюється на React.createElement('div'). Приклад вище рівнозначний ось цьому:

```
return React.createElement('div', {className: 'shopping-list'},
  React.createElement('h1', /* ... h1 children ... */),
  React.createElement('ul', /* ... ul children ... */)
);
```

Рисунок 2.3 – Приклад альтернативного написання класу з використанням бібілотеки React

JSX має всі можливості JavaScript: У JSX ви можете використовувати будь-який вираз JavaScript у круглих дужках; всі елементи React – це об'єкти JavaScript, які можна зберігати у змінній або використовувати в додатку Елементи React – це об'єкти JavaScript, їх можна зберігати у змінних або використовувати в додатку.

Компонент ShoppingList вище рендерить тільки вбудовані DOM-компоненти, такі як і
. Однак ви також можете створювати і відображати власні компоненти. Наприклад, ви можете викликати весь компонент ShoppingList, набравши<ShoppingList />. Кожен React-компонент є інкапсульованим і може використовуватися незалежно. Таким чином, складні користувацькі інтерфейси можуть бути побудовані з простих компонентів.

2.1.3 NodeJS

Node.js – асинхронне середовище виконання JavaScript, кероване подіями, призначене для створення масштабованих мережевих додатків. У прикладі "hello world", наведеному нижче, багато з'єднань можуть оброблятися одночасно. На кожне з'єднання робиться зворотній виклик, але якщо немає чого обробляти, Node.js переходить у режим сну.

```
const http = require('http');

const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

Рисунок 2.4 – Приклад коду для створення додатку на NodeJS

Це контрастує з більш поширеною сьогодні моделлю розпаралелювання, яка використовує потоки операційної системи. Мережі на основі потоків відносно неефективні і дуже складні у використанні. Крім того, оскільки немає блокування, користувачам Node.js не потрібно турбуватися про те, що їхні процеси будуть заблоковані: У Node.js дуже мало функцій, які виконують ввід/вивід безпосередньо, тому, якщо вони не виконують ввід/вивід за допомогою методів синхронізації в стандартній бібліотеці Node.js, Node.js не буде цього робити. Дуже розумно розробляти масштабовані системи на Node.js, оскільки вона ніколи не блокується.

Node.js схожий за дизайном на Ruby's Event Machine та Python's Twisted, але Node.js просуває модель подій трохи далі. Вона представляє цикл подій як конструкцію часу виконання, а не як бібліотеку. В інших системах для запуску циклу завжди є виклик блоку. Зазвичай, поведінка визначається за допомогою

зворотного виклику на початку коду і запускається в кінці сервера за допомогою блочного виклику, такого як EventMachine::run(). У Node.js немає такого виклику для ініціалізації циклу обробки подій. Node.js є дуже простою системою, тому що коли виконується вхідний скрипт, він просто входить в цикл обробки подій. Коли більше немає викликів для виконання, Node.js виходить з циклу. Така поведінка схожа на JavaScript у браузері, де цикл обробки подій прихований від користувача.

HTTP – це першокласний громадянин Node.js, розроблений з урахуванням потокової передачі даних і низької затримки. Це робить Node.js ідеальним для створення веб-бібліотек та фреймворків.

Те, що Node.js не має потоків, не означає, що ви не можете скористатися перевагами багатоядерності у вашому середовищі. Дочірні процеси можуть бути створені за допомогою API child_process.fork() і призначені для легкої взаємодії. Кластерні модулі створюються з тим самим інтерфейсом, який розподіляє сокети між процесами і дозволяє балансувати навантаження між ядрами.

2.1.4 Mongo DB

Mongo DB – документно-орієнтована БД.

Запис у MongoDB – це документ, який є структурою даних, що складається з пар полів і значень. Документи MongoDB схожі на об'єкти JSON. Значеннями полів можуть бути інші документи, масиви та масиви документів.

Рисунок 2.5 – Приклад документа для бази даних Mongo DB

Переваги використання документів полягають у наступному:

- Документи відповідають нативним типам даних у багатьох мовах програмування.
- Вбудовані документи та масиви зменшують потребу в дорогих з'єднаннях.
- Динамічна схема підтримує вільний поліморфізм.

Collections/Views/On-Demand Materialized Views

MongoDB зберігає документи в колекціях. Колекції є аналогом таблиць у реляційних базах даних.

На додаток до колекцій, MongoDB підтримує колекції:

- Read-only Views (Starting in MongoDB 3.4)
- On-Demand Materialized Views (Starting in MongoDB 4.2).

Ключові особливості

Висока продуктивність

MongoDB забезпечує високу продуктивність збереження даних. Зокрема,

- Підтримка вбудованих моделей даних зменшує активність вводу/виводу в системі баз даних.
- Індекси підтримують швидші запити і можуть включати ключі з вбудованих документів та масивів.

API запитів

API запитів до MongoDB підтримує операції читання та запису (CRUD), а також:

- Агрегація даних
- Текстовий пошук та геопросторові запити.

Висока доступність

MongoDB надає можливість реплікації, яка називається набором реплік:

- автоматичне переключення на інший ресурс
- надлишковість даних.

Набір реплік – це група серверів MongoDB, які підтримують один і той

самий набір даних, забезпечуючи надмірність і підвищуючи доступність даних.

Горизонтальна масштабованість

MongoDB забезпечує горизонтальне масштабування як частину своєї основної функціональності:

- Шардинг розподіляє дані між кластером машин.
- Починаючи з версії 3.4, MongoDB підтримує створення зон даних на основі ключа шарда. У збалансованому кластері MongoDB спрямовує читання і запис, що покриваються зоною, тільки на ті шарди, що знаходяться всередині зони. Докладнішу інформацію можна знайти на сторінці посібника Зони.

Підтримка декількох механізмів зберігання даних

MongoDB підтримує кілька механізмів зберігання даних:

- WiredTiger Storage Engine (включаючи підтримку Encryption at Rest)
- In-Memory Storage Engine.

Крім того, MongoDB надає API, що підключається, який дозволяє третім особам розробляти механізми зберігання даних для MongoDB.

2.1.5 Mongoose

Mongoose – це ODM (Object Document Manager), який подібно до реляційних ORM (Object Relationship Managers) відображає дані в нашій базі даних в об'єкти для більш звичних шаблонів програмування. Mongoose – це ODM MongoDB для додатків на Javascript, і ця стаття має на меті стати ресурсом для роботи з MongoDB в JS за допомогою Mongoose.

URI Mongoose

URI розшифровується як універсальний ідентифікатор ресурсу – рядок, який дозволяє програмам надсилати повідомлення іншим програмам. Ми постійно використовуємо HTTP URI у вигляді URL-адрес наших улюблених веб-сайтів.

Шаблон виглядає наступним чином:

protocol://username:password@host:port/resource?query=string&key=value

- protocol:// тип повідомлення, що надсилається, наприклад, https://, mongodb://, postgresql://
- username:password@ ім'я користувача та пароль, якщо вони потрібні для цілі, зазвичай потрібні для баз даних, оскільки більшість веб-сторінок відкриті для публічного доступу, а веб-сайти зазвичай не використовуються.
- host – це домен і піддомени, які можуть бути пов'язані з ним, що є псевдонімом для IP-адреси сервера, на якому розміщено цільову програму (наприклад, localhost є псевдонімом для 127.0.0.1, тобто машини, яку ви зараз використовуєте).
- port Кожен сервер може приймати повідомлення на різних портах, пронумерованих до 65535 (найбільше 16-бітне ціле число без знаку). Зазвичай ви не вказуєте порт для URL-адрес, оскільки браузери знають, що http-трафік надходить на порт 80, а https-трафік – на порт 443. Більшість баз даних мають порт за замовчуванням, який вони запускають на mongodb -> 27017 | postgresql -> 5432 | mysql -> 3306
- /resource вказує програмі-одержувачу, до яких ресурсів отримати доступ у місці призначення. Для веб-додатків це зазвичай певна веб-сторінка, файл або JSON-дані. Для додатків баз даних це, як правило, посилання на конкретну базу даних, до якої здійснюється доступ.
- ?query=string&key=value це рядок запиту, який можна використовувати для передачі додаткової інформації, наприклад, даних з форми або конфігурацій бази даних.

РОЗДІЛ 3

ПРОЄКТУВАННЯ І РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

3.1 Аналіз варіантів використання

Діаграма варіантів використання – це графічне представлення взаємодії між користувачем і системою, що показує взаємозв'язок між користувачем і різними варіантами використання, в яких він бере участь. Діаграми варіантів використання можуть описувати різні типи користувачів і різні варіанти використання системи і часто використовуються в поєднанні з іншими типами діаграм. Варіанти використання представлені колами або овалами.

Мета діаграми використання – проілюструвати динамічні аспекти системи. Додаткові діаграми та документація можуть бути використані для надання повного функціонального та технічного уявлення про систему. Вони надають спрощене схематичне зображення того, що насправді має робити система.

Ці діаграми складаються з наступних елементів

- Межа системи – прямокутник з назвою у верхній частині та еліпсом (прецедентом) всередині,
- Актор – стилізований людський персонаж, що представляє роль користувача (в загальному випадку людини, зовнішньої сутності, класу або іншої системи), який взаємодіє з сутністю (системою, підсистемою або класом). Актори не можуть бути пов'язані один з одним (за винятком відносин обробки/дослідження),
- Прецедент – позначений еліпс, що показує систематичну дію (яка також може включати можливі альтернативи), що призводить до результату, який спостерігається актором. Прецедент може бути назвою або (з точки зору актора) описом того, "що" робить система (на відміну від того, "як" вона це робить). Ім'я прецеденту може бути пов'язане з послідовним (атомарним) сценарієм, тобто певною послідовністю дій, що описує повідомлення. Під час сценаріїв актори обмінюються системними повідомленнями. Сценарії можуть

бути представлені на діаграмі прецедентів UML-опису відео; з одним прецедентом може бути пов'язано кілька різних сценаріїв.

На рисунку 3.1 зображено діаграму варіантів використання, яка описує можливі дії Користувача та Адміністратора в системі.

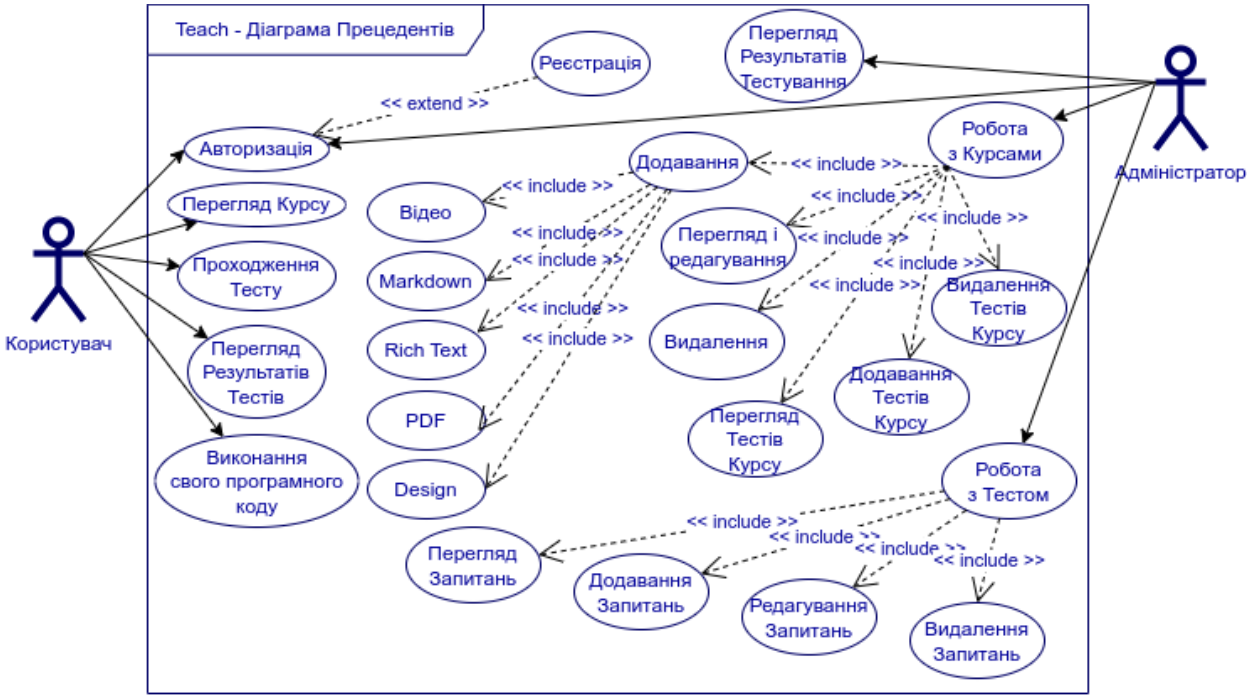


Рис. 3.1 – Діаграма варіантів використання

3.2 Проєктування структури додатку

Діаграма класів є одним з основних інструментів в області моделювання об'єктно-орієнтованих систем. Вона використовується для візуалізації класів, їх взаємозв'язків і структури системи.

Діаграма класів надає уявлення про класи, об'єкти, атрибути і методи, що входять до системи або програми. Вона дозволяє моделювати структуру класів і їх взаємозв'язки, що допомагає розуміти, як об'єкти взаємодіють між собою.

На діаграмі класів класи зображаються у вигляді прямокутників, де вказуються назва класу, його атрибути (змінні) і методи (операції або функції), що його характеризують. Взаємозв'язки між класами показуються зв'язками, які можуть вказувати агрегацію, композицію, спадкування, залежності та інші типи взаємодій.

Діаграми класів використовуються для Проєктування архітектури програмних систем, аналізу вимог, документування коду, комунікації між членами команди розробників і для вивчення взаємодії між класами в об'єктно-орієнтованому програмуванні. Вони дозволяють зрозуміти структуру системи і полегшують розробку, тестування і супровід програмного забезпечення.

З огляду на те, що розроблявся веб-додаток, Діаграма класів на рисунку представлена Діаграма структури додатку у вигляді умовної карти сайту.

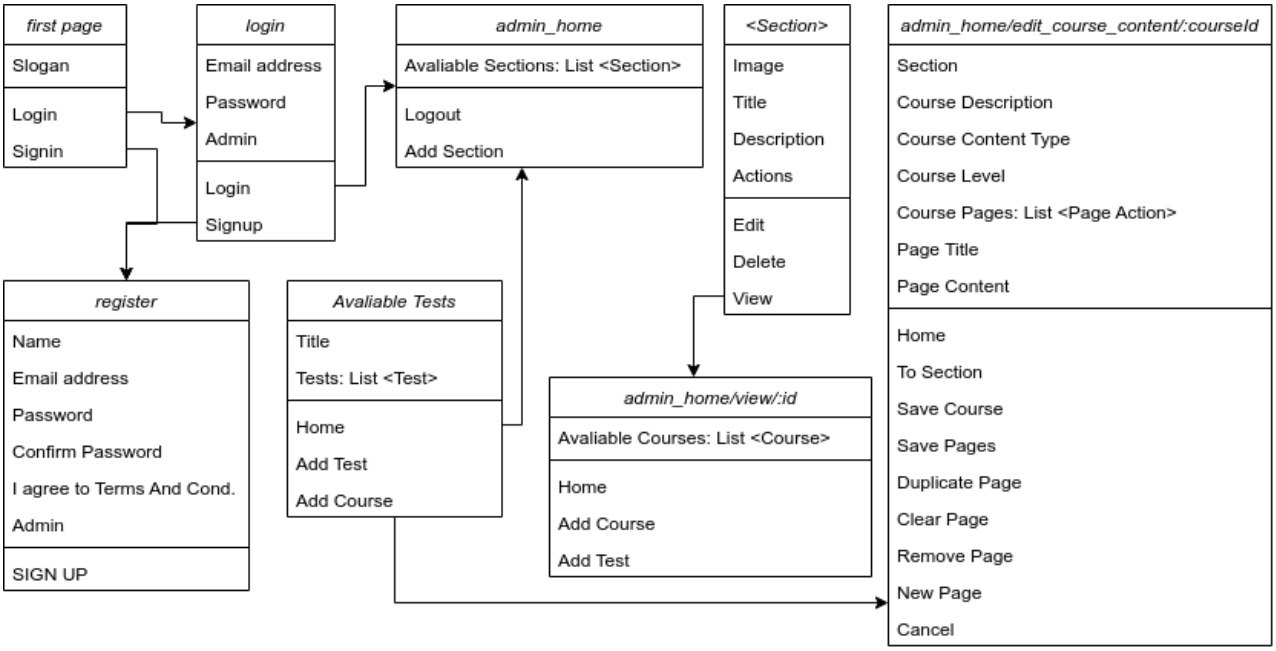


Рис. 3.2 – Діаграма класів

3.3 Проєктування бази даних додатку

Діаграма структури бази даних використовується для візуалізації структури бази даних, що включає таблиці, відношення між ними, атрибути та зв'язки між таблицями. Вона дозволяє описати логічну або фізичну структуру бази даних і представити її у зручній формі.

Основні цілі використання діаграм структури бази даних:

- Візуалізація структури: Діаграма допомагає розібратися в структурі бази даних, її складових елементах та залежностях між ними. Вона надає зрозумілу візуальну модель бази даних, що полегшує сприйняття та аналіз.

- Аналіз і Проєктування: Діаграми структури баз даних використовуються при аналізі та Проєктуванні бази даних. Вони дозволяють ідентифікувати сутності, атрибути та зв'язки, що впливають на логічну модель даних. За допомогою діаграм можна виявити потреби у нових таблицях, зв'язках або атрибутах.
- Комунікація: Діаграми структури баз даних є ефективним інструментом комунікації між розробниками, аналітиками та стейкхолдерами. Вони допомагають узгодити сприйняття структури бази даних і забезпечити зрозуміле спілкування між різними учасниками Проєкту.
- Документація: Діаграми структури баз даних служать як документація для баз даних. Вони фіксують структуру бази даних на папері або електронному носії, що полегшує зрозуміння та збереження інформації про базу даних для майбутніх Проєктів або технічної підтримки.

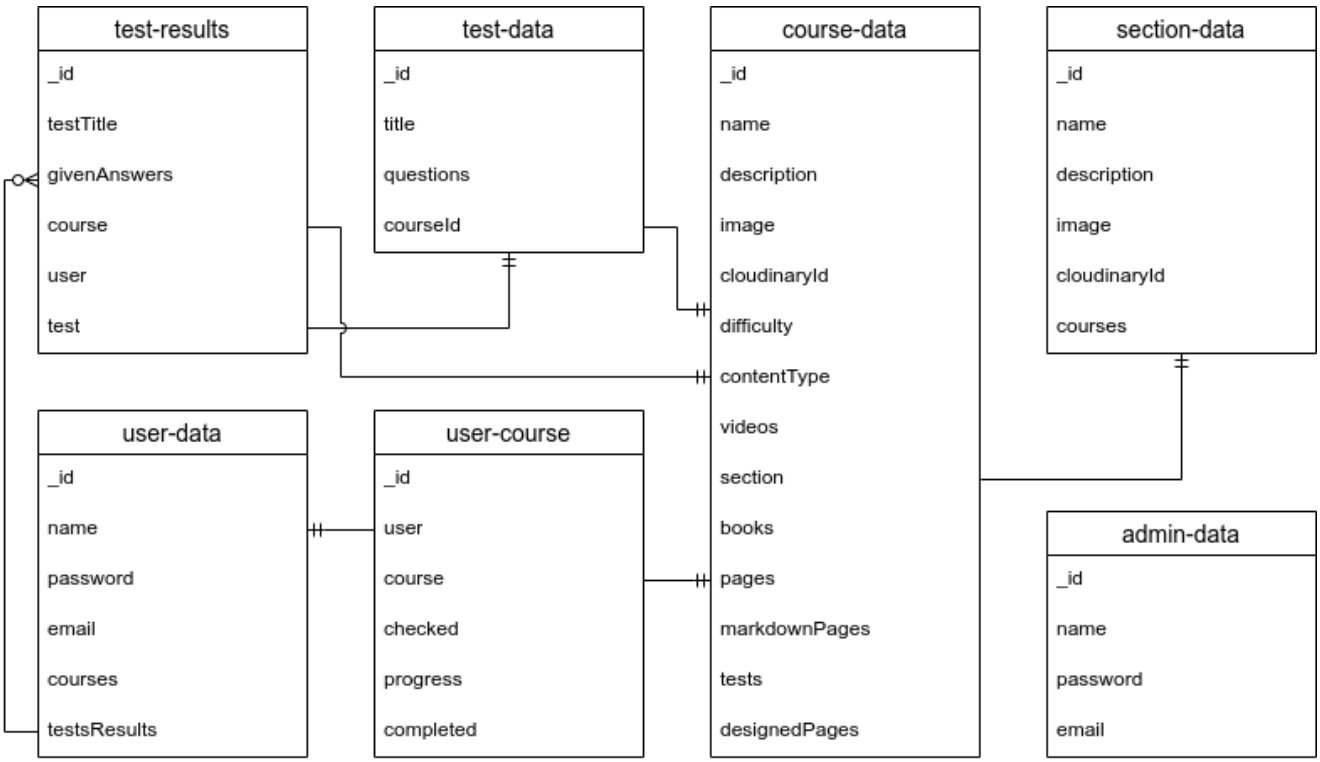


Рис. 3.3 – Структура бази даних системи

3.4 Розробка графічного інтерфейсу системи

Графічний інтерфейс системи складається з таких основних сторінок:

- Сторінка реєстрації (рис. 3.3);
- Сторінка авторизації (рис. 3.4);
- Головна сторінка Адміністратора (рис. 3.5);
- Сторінка курсів Адміністратора (рис. 3.6);
- Сторінка тестів Адміністратора (рис. 3.7);
- Сторінка редагування курсу Адміністратором (рис. 3.8);
- Сторінка редагування тесту Адміністратором (рис. 3.9);
- Сторінка редагування питань тесту Адміністратором (рис. 3.10);
- Сторінка редагування варіанту відповіді Адміністратором (рис. 3.11);
- Головна сторінка Користувача (рис. 3.12).
- Сторінка користувача з доступними розділами (рис. 3.13).
- Сторінка користувача з розпочатими ним курсами (рис. 3.14).
- Сторінка перегляду курсу Користувачем (рис. 3.15).
- Сторінка проходження тесту Користувачем (рис. 3.16).
- Сторінка результатів проходження тестів Користувачем (рис. 3.17).
- Сторінка практики програмування свого коду Користувачем (рис. 3.18а).
- Сторінка практики програмування свого коду Користувачем (рис. 3.18б).

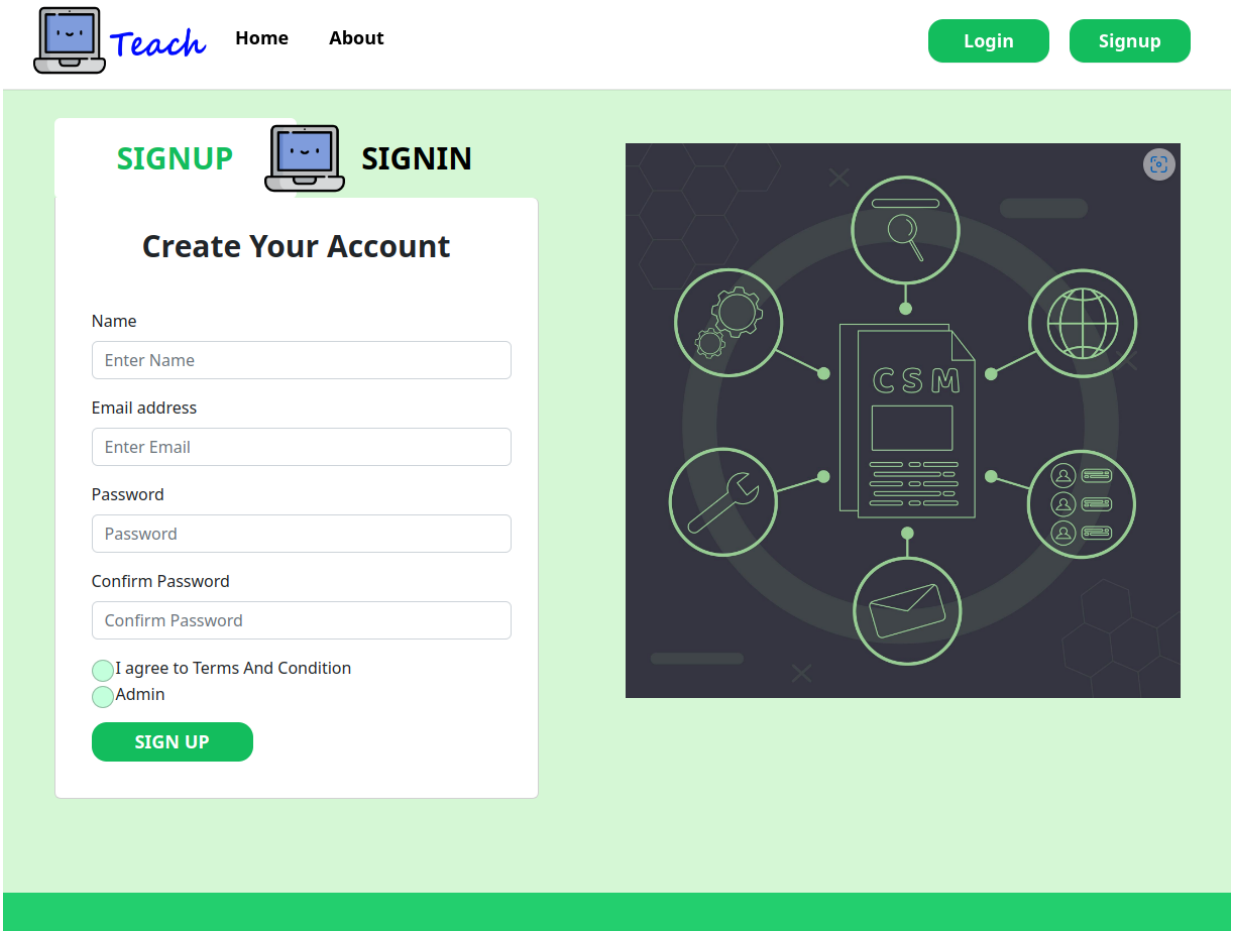


Рисунок 3.3 – Сторінка реєстрації

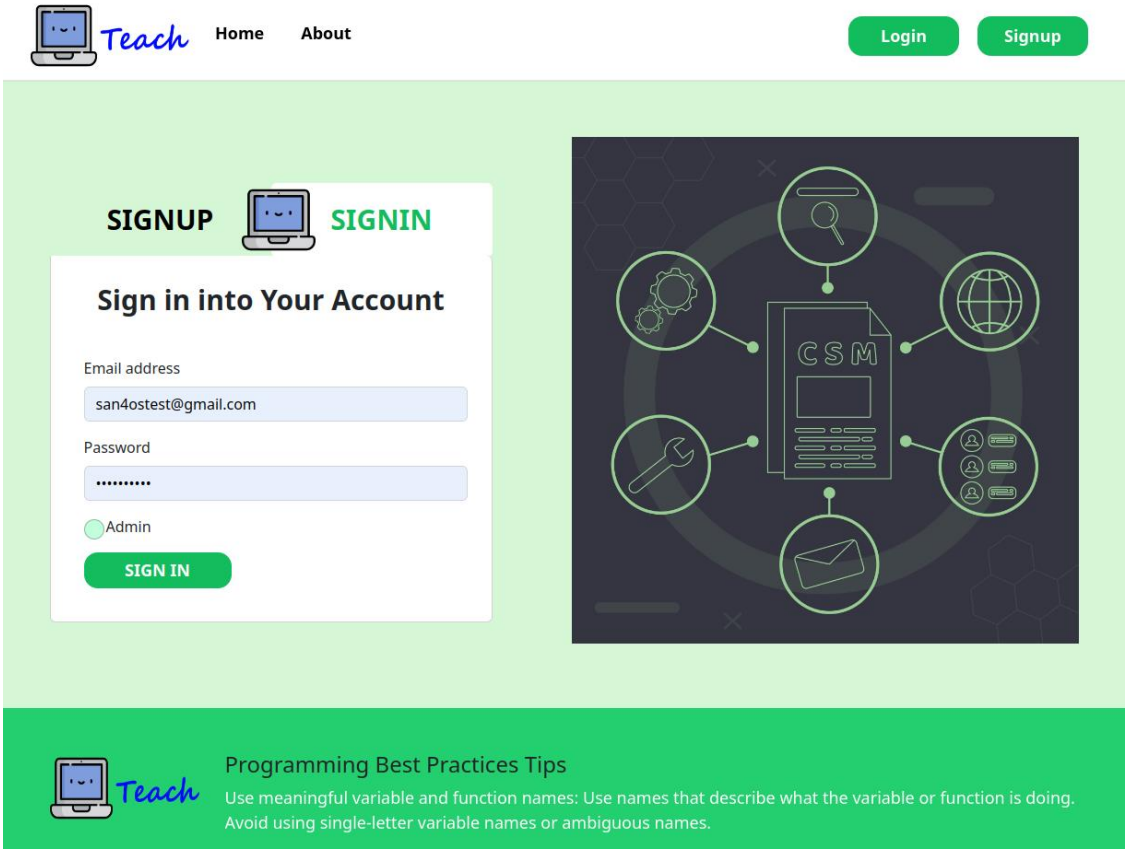


Рисунок 3.4 – Сторінка авторизації

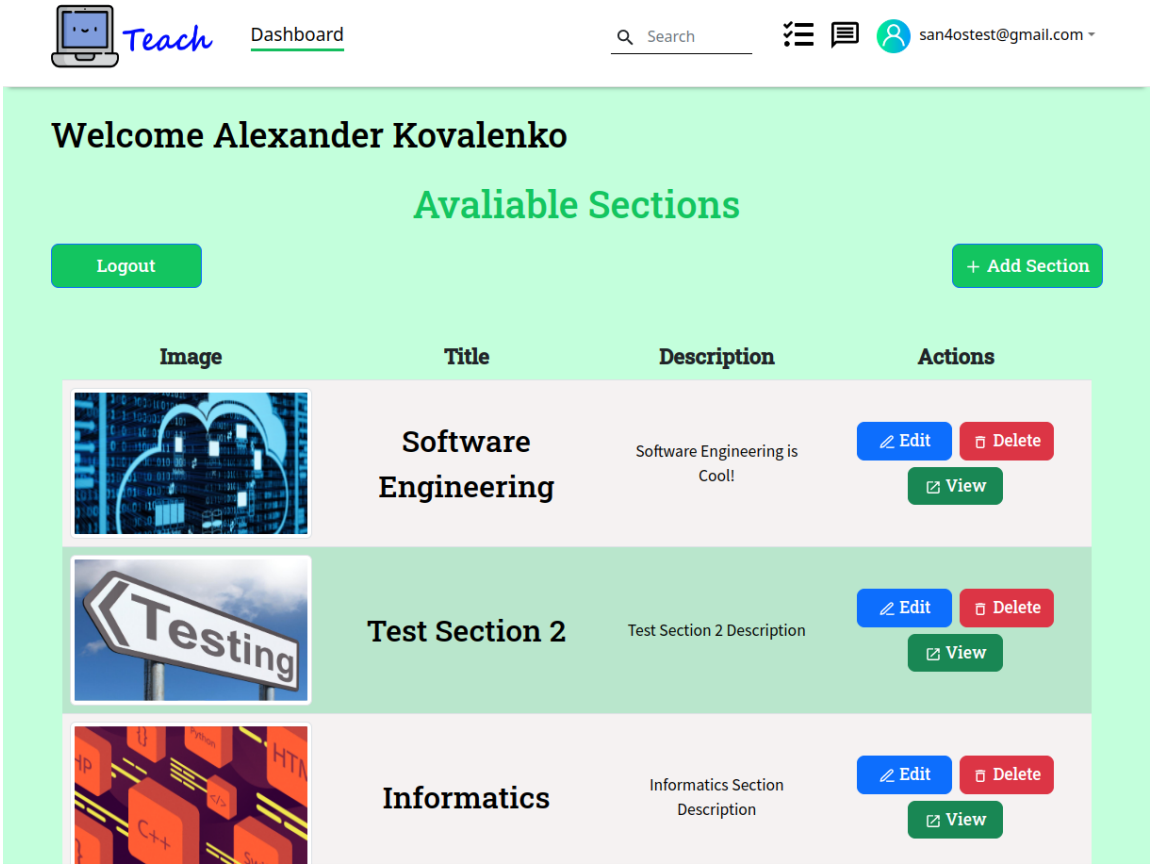


Рисунок 3.5 – Головна сторінка Адміністратора

Рисунок 3.6 – Сторінка курсів Адміністратора

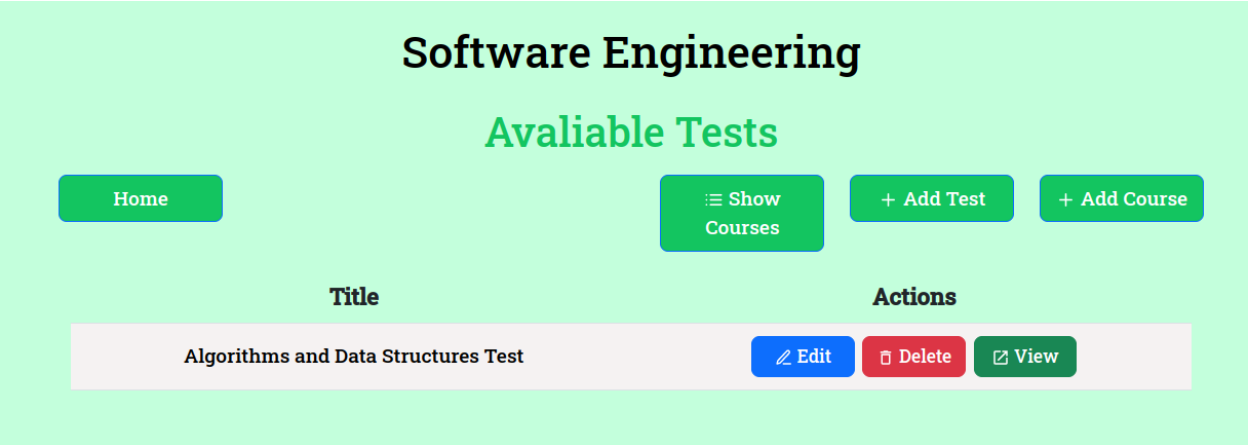


Рисунок 3.7 – Сторінка тестів Адміністратора

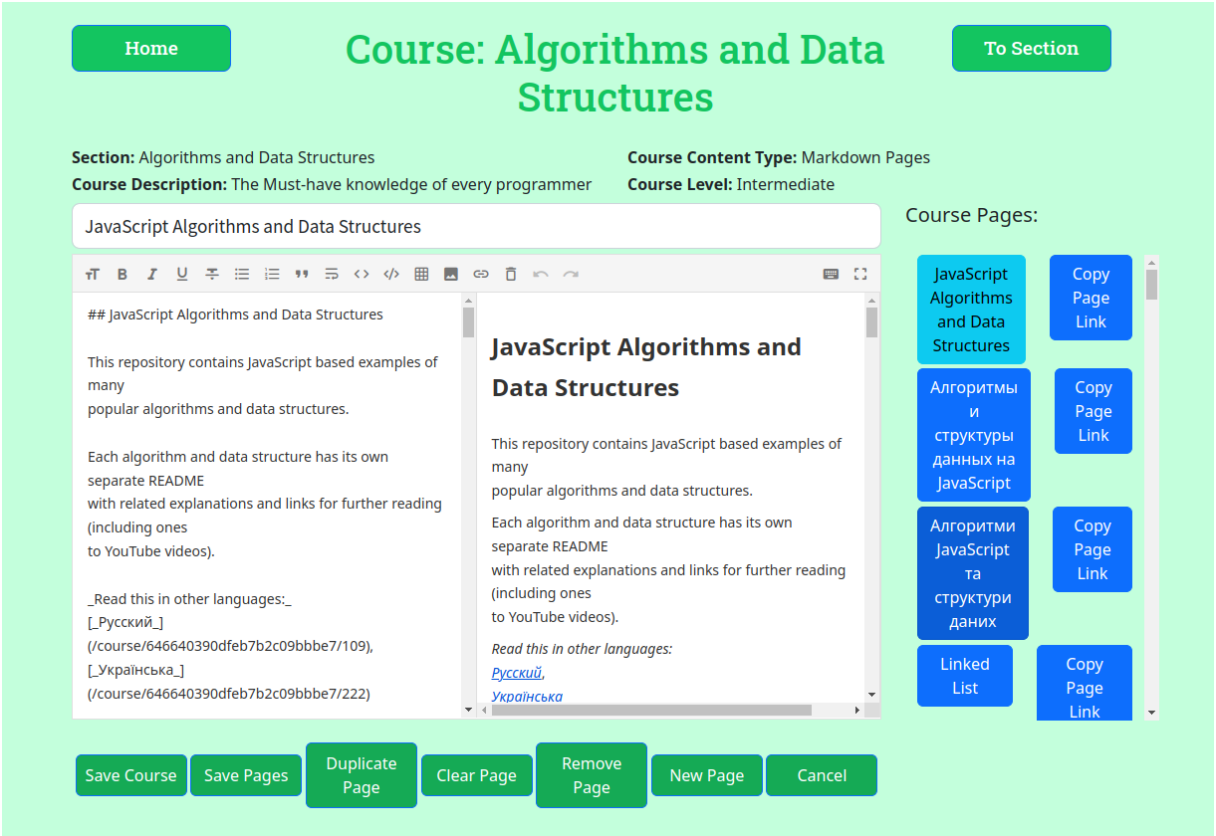


Рисунок 3.8 – Сторінка редагування курсу Адміністратором

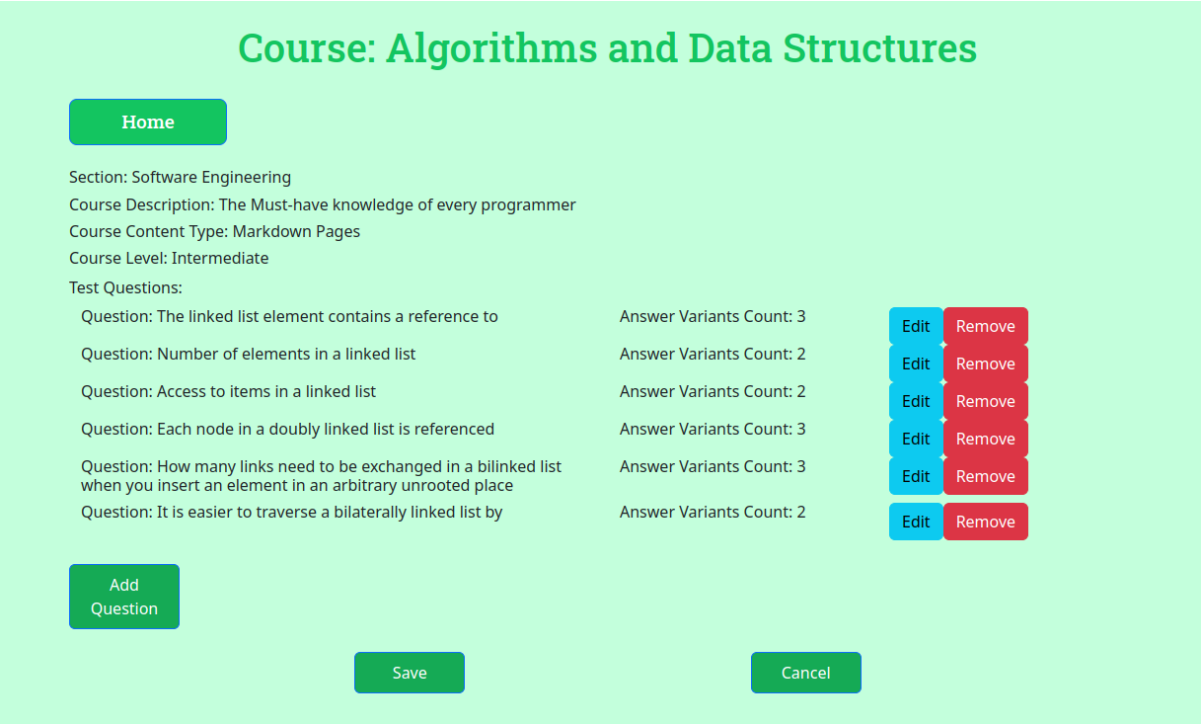


Рисунок 3.9 – Сторінка редагування тесту Адміністратором

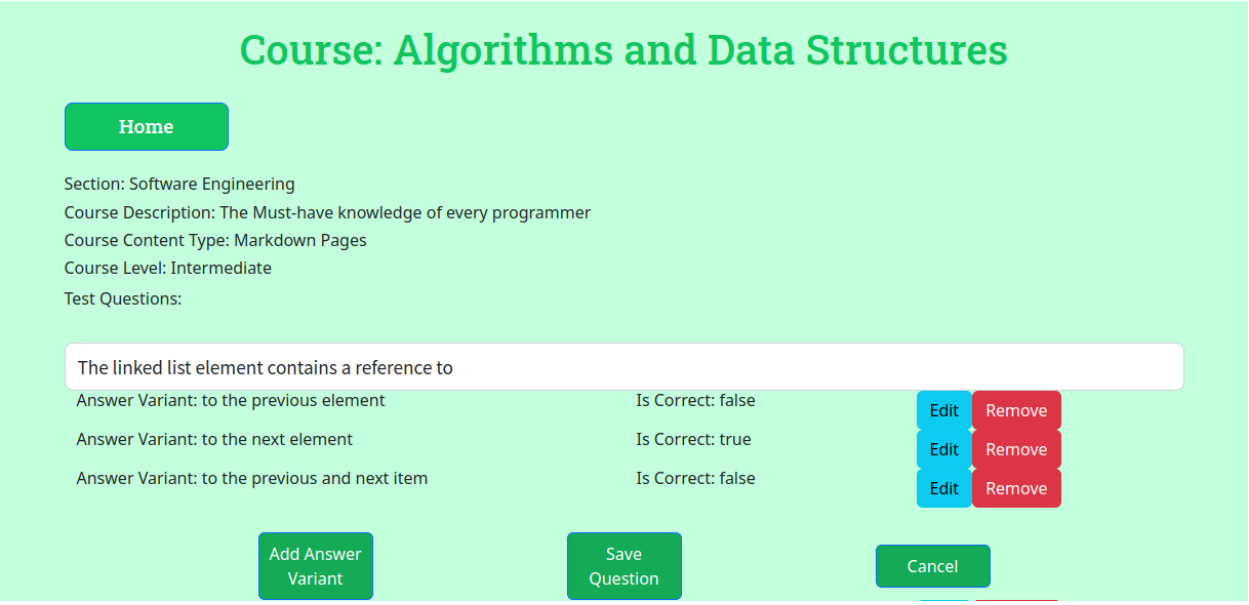


Рисунок 3.10 – Сторінка редагування питань тесту Адміністратором

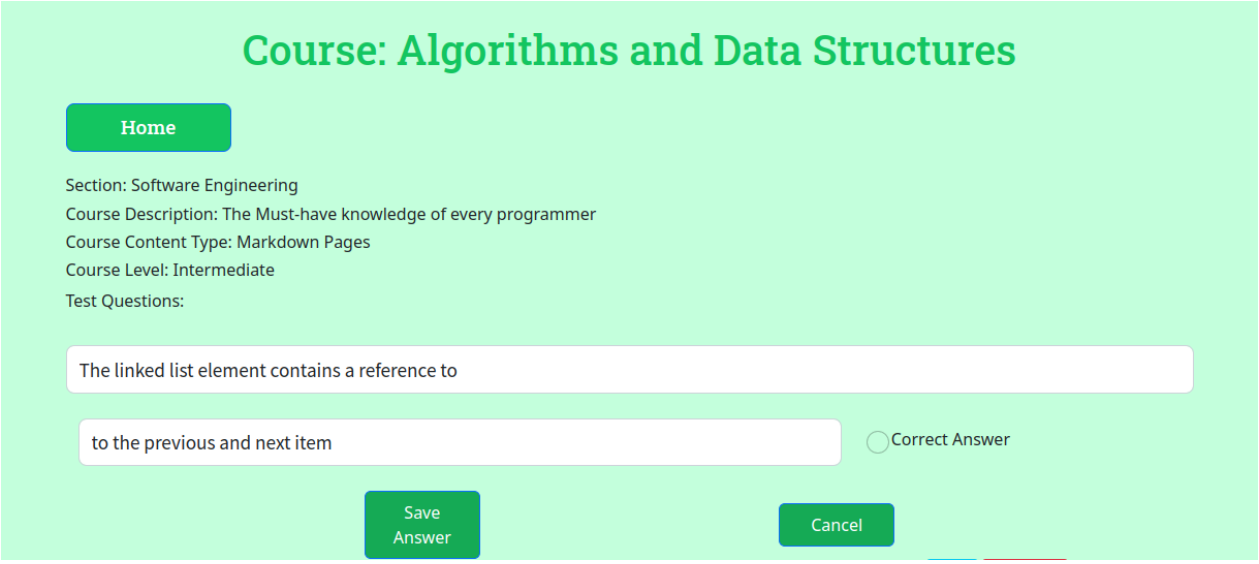


Рисунок 3.11 – Сторінка редагування варіанту відповіді Адміністратором

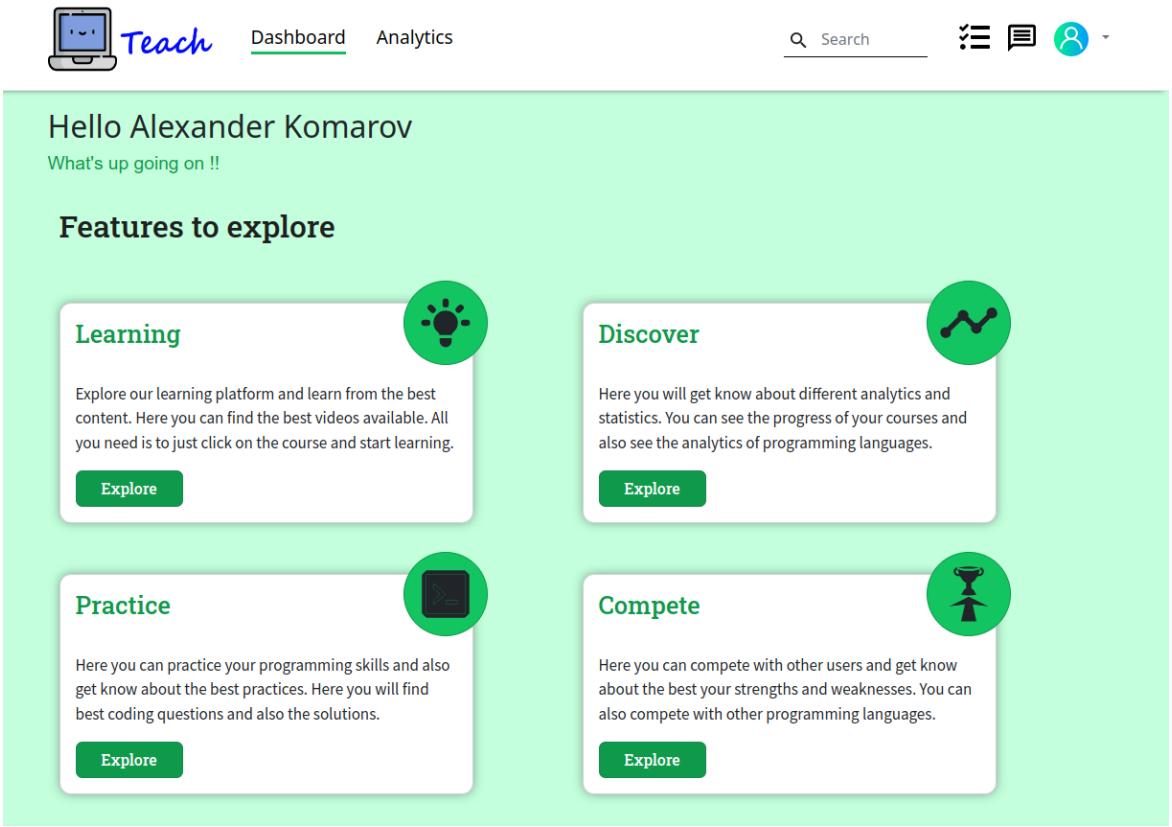


Рисунок 3.12 – Головна сторінка Користувача

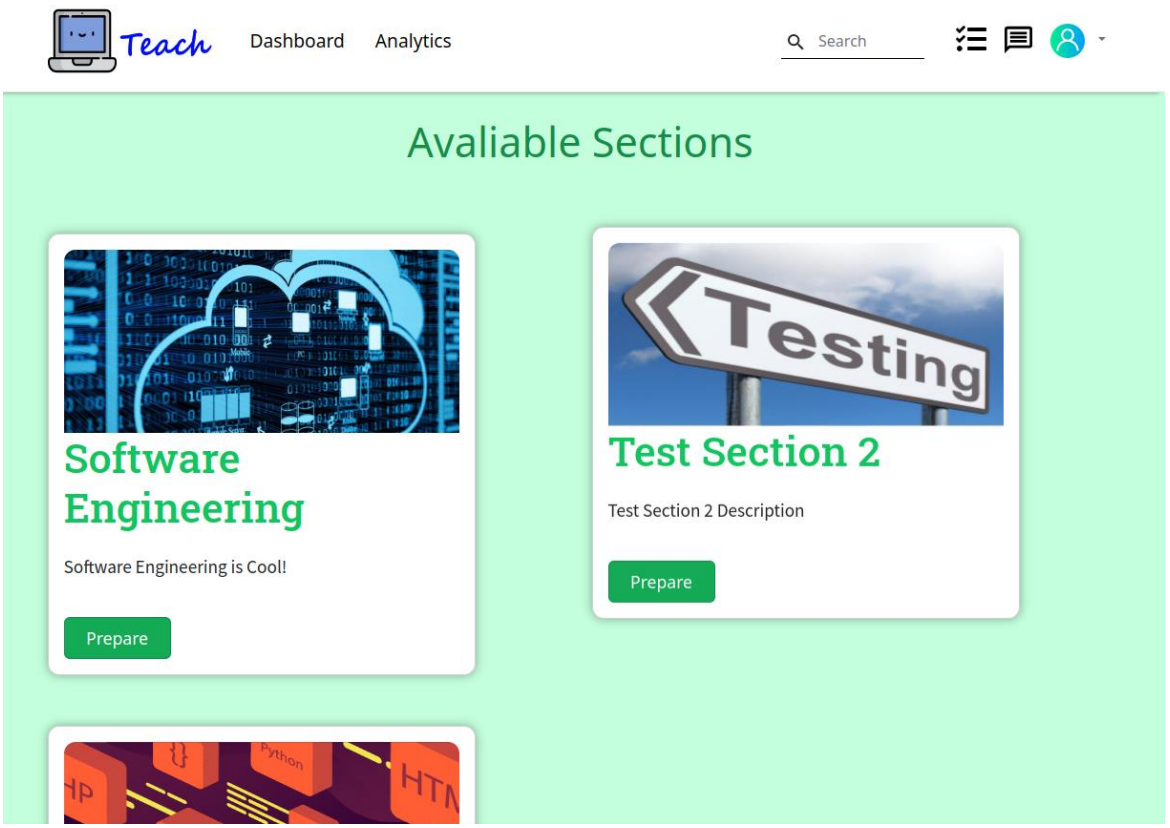


Рисунок 3.13 – Сторінка Користувача з доступними розділами

Рисунок 3.14 – Сторінка Користувача з розпочатими ним курсами

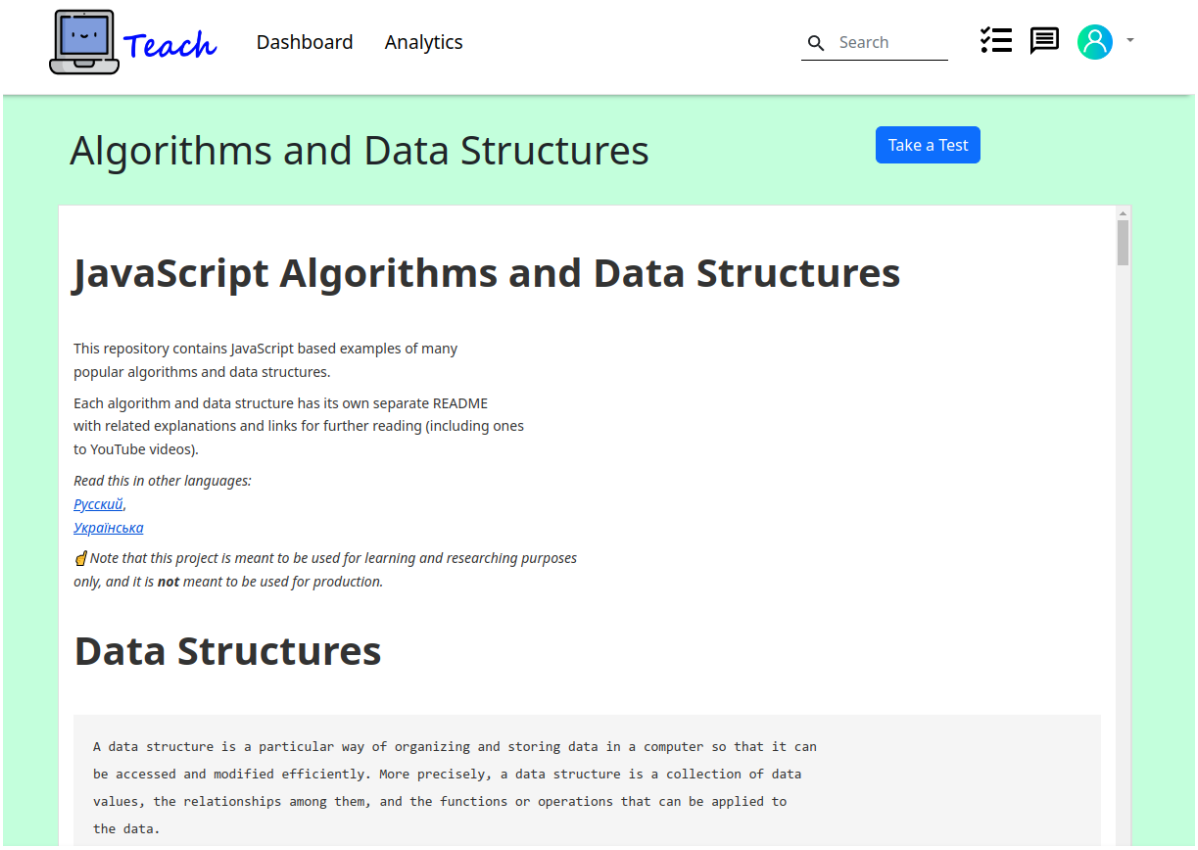


Рисунок 3.15 – Сторінка перегляду курсу Користувачем

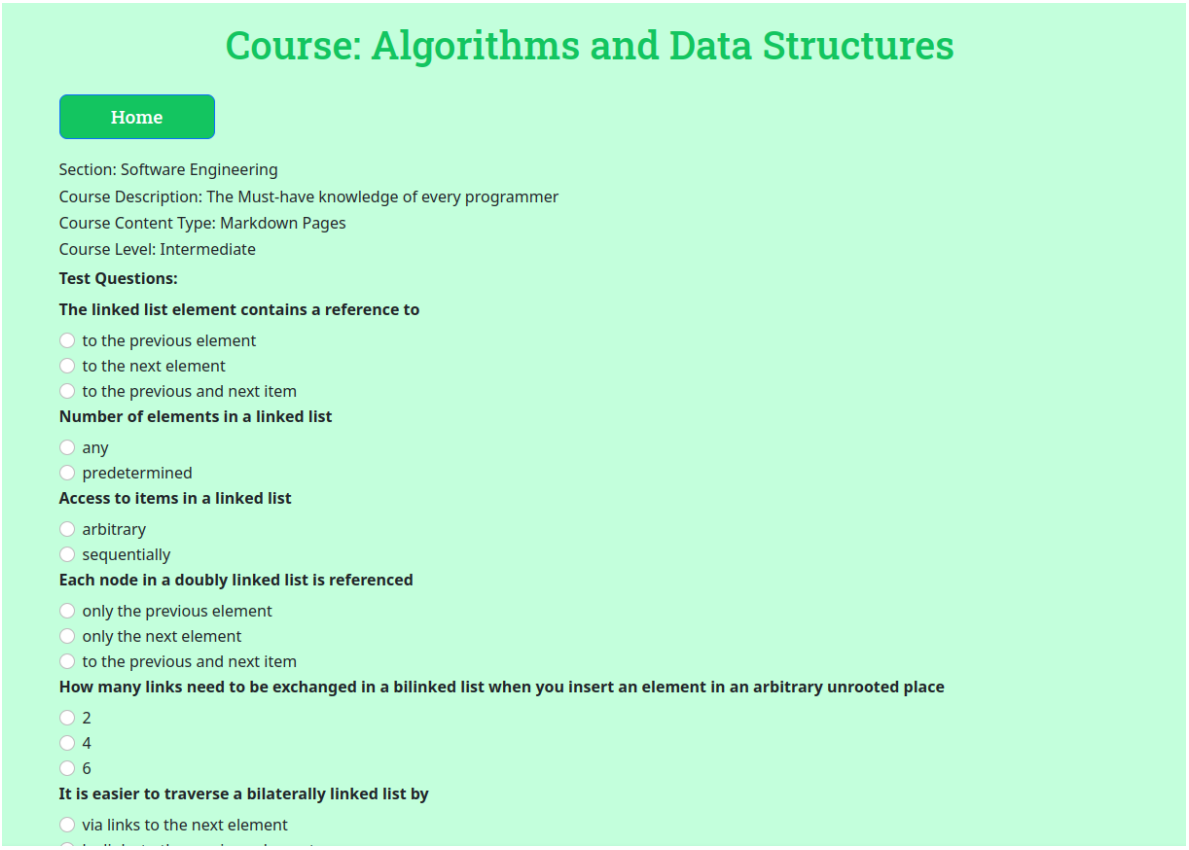


Рисунок 3.16 – Сторінка проходження тесту Користувачем

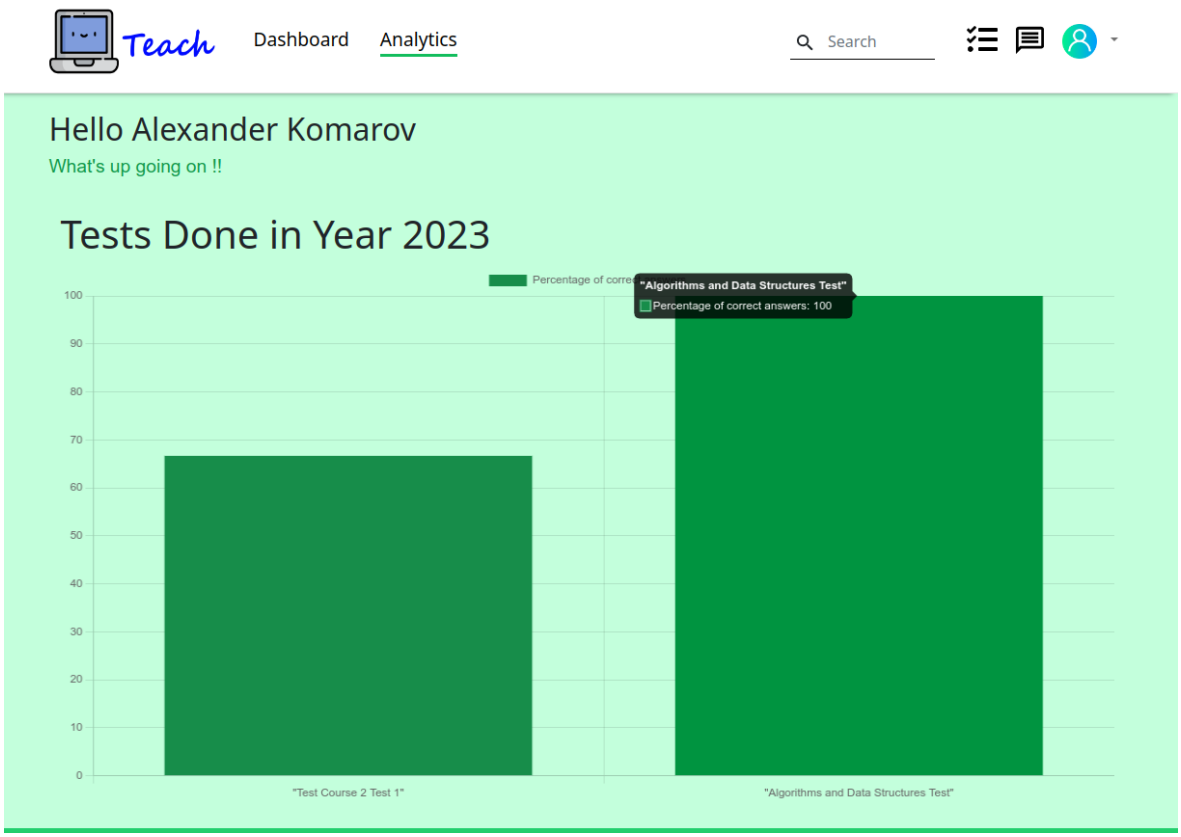


Рисунок 3.17 – Сторінка результатів проходження тестів Користувачем

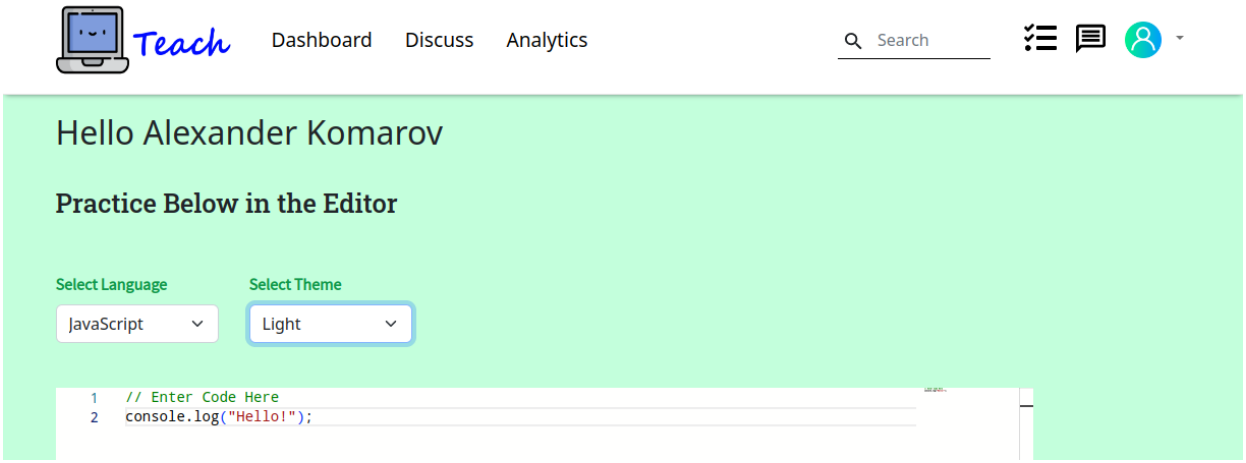


Рисунок 3.18а – Сторінка практики програмування свого коду Користувачем

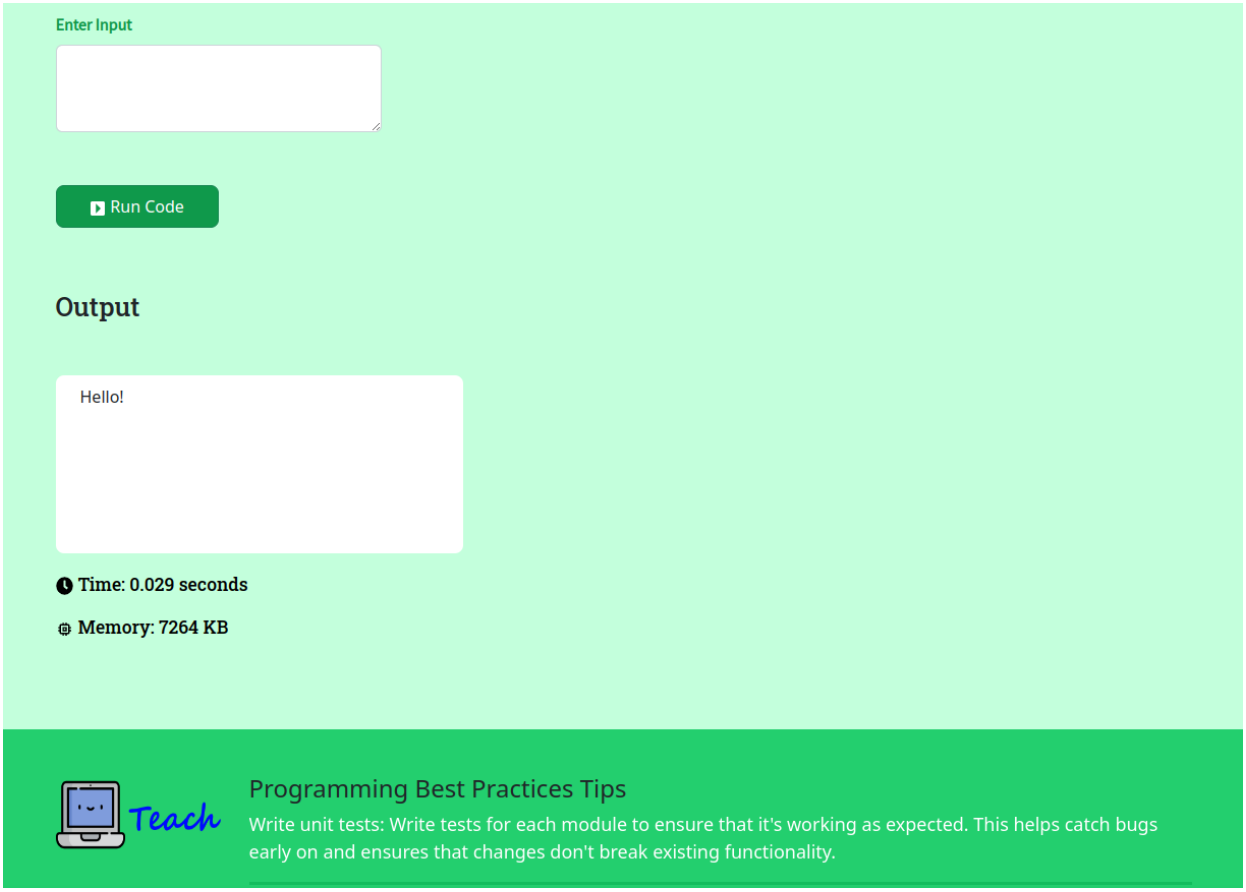


Рисунок 3.18б – Сторінка практики програмування свого коду Користувачем

3.5 Тестування

Тестування програмного забезпечення є важливим процесом у розробці програмного забезпечення, який включає в себе оцінку функціональності, якості та продуктивності програмної системи. Існують різні методи тестування, що застосовуються на різних етапах життєвого циклу розробки програмного забезпечення, включаючи тестування "чорного ящика", тестування "білого ящика", тестування "сірого ящика", модульне тестування, інтеграційне тестування, системне тестування, приймально-здавальне тестування та регресійне тестування.

Види тестування програмного забезпечення

- Тестування чорної скриньки:

Цей метод зосереджений на тестуванні зовнішньої поведінки програмної системи без урахування її внутрішньої структури або деталей реалізації.

Тестувальники вводять певні вхідні дані і вивчають вихідні, щоб визначити, чи поводиться система так, як очікувалося.

Він базується на вимогах і специфікаціях, гарантуючи, що програмне забезпечення відповідає бажаній функціональності та очікуванням користувачів.

- Тестування білої скриньки:

Тестування "білої скриньки", також відоме як структурне тестування або тестування "скляної скриньки", досліджує внутрішню структуру та реалізацію програмної системи.

Тестувальники знають внутрішній код і використовують такі методи, як покриття коду, тестування гілок і тестування шляхів, щоб переконатися, що всі шляхи коду протестовані.

Воно спрямоване на виявлення проблем, пов'язаних зі структурою, логікою та реалізацією програмного забезпечення.

- Тестування сірої скриньки:

Тестування сірої скриньки поєднує в собі елементи як чорної, так і білої скриньки.

Тестувальники мають обмежені знання про внутрішню структуру і використовують їх для розробки більш ефективних тестових кейсів.

Це дозволяє глибше зрозуміти систему і може виявити приховані проблеми, які не можуть бути виявлені лише за допомогою тестування "чорного ящика".

- Юніт-тестування:

Модульне тестування передбачає тестування окремих компонентів або блоків коду, щоб переконатися, що вони функціонують правильно в ізоляції.

Воно зосереджене на перевірці найменших тестованих частин програмного забезпечення, таких як функції або методи.

Розробники часто виконують модульне тестування під час процесу розробки, щоб на ранніх стадіях виявляти помилки і забезпечувати якість коду.

- Інтеграційне тестування:

Інтеграційне тестування перевіряє взаємодію та співпрацю між різними компонентами або модулями програмної системи.

Воно гарантує, що окремі модулі працюють разом, як очікується, коли вони об'єднані.

На цьому етапі тестування тестувальники перевіряють інтерфейси, потоки даних і зв'язок між компонентами.

- Системне тестування:

Системне тестування оцінює всю програмну систему в цілому і перевіряє, чи відповідає вона заданим вимогам.

Тестується поведінка, продуктивність, безпека та інші нефункціональні аспекти системи.

Тестувальники проводять системне тестування в середовищі, яке максимально нагадує виробниче.

- Приймальне тестування:

Приймальне тестування перевіряє, чи відповідає програмна система вимогам замовника і чи готова вона до розгортання.

Воно передбачає, що кінцеві користувачі або зацікавлені сторони виконують тести, щоб визначити, чи відповідає програмне забезпечення їхнім

очікуванням.

Основна увага приділяється тому, щоб переконатися, що програмне забезпечення відповідає своєму призначенню і задовольняє потреби користувача.

- Регресійне тестування:

Регресійне тестування гарантує, що зміни або оновлення програмного забезпечення не призведуть до появи нових дефектів або порушення існуючої функціональності.

Воно повторно тестує раніше працюючі функції, щоб переконатися, що вони все ще функціонують так, як очікувалося.

Регресійне тестування зазвичай виконується після внесення змін або оновлень до програмного забезпечення.

Переваги тестування програмного забезпечення

Тестування програмного забезпечення надає кілька переваг користувачам програмних продуктів, серед яких:

- Покращена якість: Завдяки ретельному тестуванню виявляються та усуваються дефекти та помилки програмного забезпечення, що призводить до підвищення якості продукту. Тестування допомагає гарантувати, що програмне забезпечення працює за призначенням, відповідає вимогам користувача і працює надійно.
- Розширена функціональність: Тестування перевіряє, чи всі функції та можливості програмного забезпечення працюють правильно. Це допомагає виявити будь-які проблеми або невідповідності в поведінці системи, гарантуючи, що користувачі можуть ефективно використовувати передбачені можливості програмного забезпечення.
- Підвищення надійності: Тестування має на меті зробити програмне забезпечення більш надійним шляхом виявлення та виправлення дефектів. Проводячи різні типи тестування, такі як системне тестування та регресійне тестування, можна мінімізувати потенційні проблеми та ризики, що призведе до створення більш стабільного та надійного програмного продукту.
- Краща продуктивність: Тестування продуктивності оцінює, як

програмне забезпечення працює за різних умов, таких як високе навантаження або одночасна робота користувачів. Виявляючи вузькі місця в роботі, тестування програмного забезпечення допомагає оптимізувати продуктивність системи, гарантуючи, що вона зможе ефективно справлятися з очікуваними робочими навантаженнями.

- Підвищена безпека: Тестування безпеки є важливим аспектом тестування програмного забезпечення. Воно допомагає виявити вразливості та слабкі місця в програмному забезпеченні, які можуть бути використані зловмисниками. Вирішуючи проблеми безпеки за допомогою тестування, користувачі можуть бути впевнені, що їхні дані та інформація захищені.
- Покращення користувацького досвіду: Тестування відіграє важливу роль у забезпеченні позитивного користувацького досвіду. Перевіряючи зручність використання, функціональність і продуктивність програмного забезпечення, тестування допомагає гарантувати, що користувачі можуть легко взаємодіяти з програмним забезпеченням, ефективно виконувати завдання і мати задовільний досвід роботи.
- Скорочення часу простою та витрат: Виявляючи та вирішуючи проблеми на етапі тестування, можна мінімізувати потенційні збої або простої системи. Це призводить до підвищення доступності програмного забезпечення та зниження витрат, пов'язаних з обслуговуванням, підтримкою та доопрацюванням.
- Відповідність вимогам і правилам: Залежно від галузі або предметної області, певні програмні продукти повинні відповідати певним нормам і стандартам відповідності. Тестування допомагає гарантувати, що програмне забезпечення відповідає цим вимогам, дозволяючи користувачам працювати в межах правових і регуляторних норм.

Таблиця 3.1 – Тестування додатку

№	**Тест-кейс**	**Очікуваний результат**	**Отриманий результат**
1	Невірні данні при авторизації	При введенні невірних даних система повідомляє користувача про те, що такого користувача не існує, або дані введені невірно	При введенні невірних даних система повідомляє користувача про те, що такого користувача не існує, або дані введені невірно
2	Порожні поля при авторизації	При спробі авторизації з порожнімиполями система повідомляє користувача про те, що ці поля необхідно заповнити.	При спробі авторизації з порожнімиполями система повідомляє користувача про те, що ці поля необхідно заповнити.
3	Неспівпадаючі паролі при реєстрації	При введенні неспівпадаючих паролів система повідомляє користувача про те, що паролі не співпадають.	При введенні неспівпадаючих паролів система повідомляє користувача про те, що паролі не співпадають.

Продовження таблиці 3.1 – Тестування додатку

№	Тест-кейс	Очікуваний результат	Отриманий результат
4	Порожні поля при реєстрації	При спробі реєстрації з порожніми полями система повідомляє користувача про те, що ці поля необхідно заповнити.	При спробі реєстрації з порожніми полями система повідомляє користувача про те, що ці поля необхідно заповнити.
5	Створення курсу з порожніми полями	При спробі створення курсу з порожніми полями система повідомляє користувача про те, що ці поля необхідно заповнити.	При спробі створення курсу з порожніми полями система повідомляє користувача про те, що ці поля необхідно заповнити.
6	Створення тесту з порожніми полями	При спробі створення тесту з порожніми полями система повідомляє користувача про те, що ці поля необхідно заповнити.	При спробі створення тесту з порожніми полями система повідомляє користувача про те, що ці поля необхідно заповнити.

Продовження таблиці 3.1 – Тестування додатку

№	Тест-кейс	Очікуваний результат	Отриманий результат
7	Спроба перейти в панель адміністратора без наявності прав адміністратора	При спробі користувача без прав адміністратора перейти в панель адміністрування система переводить користувача на форму авторизації.	При спробі користувача без прав адміністратора перейти в панель адміністрування система переводить користувача на форму авторизації.

Результати тестування показують, що програма функціональна і готова до використання.

ВИСНОВКИ

Мета роботи полягала в підтримці процесів навчання з дисципліни "Алгоритми та структури даних" засобами додатку, створеного мовою JavaScript.

Для досягнення поставленої мети було виконано наступні завдання:

1. Проведено аналіз потреб навчального процесу дисципліні "Алгоритми та Структури даних". Виявлено спосіб спрощення процесу навчання. Виявлено та зафіксовано ключові вимоги до розроблюваного програмного продукту.
2. Розглянуто та проаналізовано наявні програмні засоби, що дозволяють вивчати Алгоритми та Структури даних. Проведено порівняння їх функціональних можливостей. Виявлено переваги та недоліки.
3. Розроблено структуру бази даних створюваного додатку.
4. Розроблено програмне забезпечення для навчання курсу «Алгоритми та Структури даних» з урахуванням поставлених вимог до додатку.
5. Проведено тестування користувацького інтерфейсу додатку та функціоналу запису створюваного контенту в базу даних, а також відображення існуючого контенту на відповідних сторінках.

Розроблену систему можна використовувати для покращення навчального процесу. Можна додавати нові курси та матеріали.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. What are the benefits of using educational software in schools? [Електронний ресурс]: [Веб-сайт]. – електронні дані. – Режим доступу: https://www.researchgate.net/post/What-are-the-benefits-of-using-educational-software-in-schools
2. Sampling Strategies for Educational Research [Електронний ресурс]: [Веб-сайт]. – електронні дані. – Режим доступу: http://www.vkmaheshwari.com/WP/?p=236
3. Використання сучасних освітніх інструментів для підвищення рівня цифрової компетентності педагога НУШ [Електронний ресурс]: [Веб-сайт]. – електронні дані. – Режим доступу: https://medialiteracy.org.ua/vykorystannya-suchasnyh-osvitnih-instrumentiv-dlya-pidvyshhennya-rivnya-tsyfrovoyi-kompetentnosti-pedagoga-nush/
4. Head First HTML and CSS by Elisabeth Robson and Eric Freeman Copyright © 2012. Printed in Canada. Published by O'Reilly Media, Inc., 1005 Gravenste in Highway North, Sebastopol, CA 95472.
5. The Essential Guide to HTML5: Using Games to Learn HTML5 and JavaScript © 2018. Library of Congress. Control Number: 2018962546 Copyright © 2018 by Jeanine Meyer.
6. HTML5 and CSS3, Seventh Edition: Visual QuickStart Guide, Elizabeth Castro and Bruce Hyslop Peachpit Press 1249, Eighth StreetBerkeley, CA 94710.
7. Web Development with Node and Express by Ethan Brown Copyright © 2014 Ethan Brown. All rights reserved. Printed in the United States of America. Published by O'Reilly Media, Inc., 1005 Gravenste in Highway North, Sebastopol, CA 95472.
8. Learning Web Application Development by Semmy Purewal Copyright © 2014 Semmy Purewal. All rights reserved. Printed in the United States of America. Published by O'Reilly Media, Inc., 1005 Gravenste in Highway North, Sebastopol, CA 95472.

9. Supercharged JavaScript Graphics by Raffaele Cecco Printed in the United States of America. Published by O'Reilly Media, Inc., 1005 Gravenste in Highway North, Sebastopol, CA 95472.
10. Building Node Applications with MongoDB and Backbone by Mike Wilson Copyright © 2013 Mike Wilson. All rights reserved. Printed in the United States of America. Published by O'Reilly Media, Inc., 1005 Gravenste in Highway North, Sebastopol, CA 95472.
11. MongoDB: The Definitive Guide, Second Edition by Kristina Chodorow Copyright © 2013 Kristina Chodorow. All rights reserved. Printed in the United States of America. Published by O'Reilly Media, Inc., 1005 Gravenste in Highway North, Sebastopol, CA 95472.
12. Pro MERN Stack Vasan Subramanian Bangalore, Karnataka, India Library of Congress Control Number: 2017933833 Copyright © 2017 by Vasan Subramanian
13. HTML & CSS Design and buiLd Websites, Published by John Wiley & Sons, Inc. 10475 Crosspoint Boulevard Indianapolis, © 2011 by John Wiley & Sons, Inc., Indianapolis, Indiana Manufactured in the United States of America Published simultaneously in Canada.
14. Beginning Web Programming with HTML, XHTML, and CSS Copyright © 2004 by Wiley Publishing, Inc. All rights reserved. Published simultaneously in Canada.
15. HTML basics [Електронний ресурс]: [Веб-сайт]. – електронні дані. – Режим доступу: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics
16. CSS basics [Електронний ресурс]: [Веб-сайт]. – електронні дані. – Режим доступу: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/CSS_basics
17. JavaScript basics [Електронний ресурс]: [Веб-сайт]. – електронні дані. – Режим доступу: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/JavaScript_basics

18. Що таке React? [Електронний ресурс]: [Веб-сайт]. – електронні дані. – Режим доступу: https://uk.legacy.reactjs.org/tutorial/tutorial.html#what-is-react
19. About Node.js® [Електронний ресурс]: [Веб-сайт]. – електронні дані. – Режим доступу: https://nodejs.org/en/about
20. Introduction to MongoDB [Електронний ресурс]: [Веб-сайт]. – електронні дані. – Режим доступу: https://www.mongodb.com/docs/manual/introduction/
21. What is Mongoose? [Електронний ресурс]: [Веб-сайт]. – електронні дані. – Режим доступу: https://dev.to/alexmercedcoder/2022-mongoosejs-cheatsheet-9b8

Додаток А

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка додатку для навчання дисципліни "Алгоритми та структури даних" мовою JavaScript

Виконав студент 5 курсу
групи ППЗ-51
Комаров Олександр Валерійович
Керівник роботи
К.т.н, доцент ІПЗ Негоденко Олена Василівна

Київ – 2023

МЕТА, ОБ’ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Мета роботи –** підтримка процесів навчання з дисципліни "Алгоритми та структури даних" засобами додатку, створеного мовою JavaScript.
- **Об’єкт дослідження –** процес навчання з дисципліни "Алгоритми та структури даних".
- **Предмет дослідження –** додаток для навчання з дисципліни "Алгоритми та структури даних".

2

ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

1. Провести аналіз програмного забезпечення для навчання.
2. Визначити переваги та можливості сучасних програмних засобів для навчання.
3. Вибрати інструментальні програмні засоби для розробки додатку для навчання.
4. Вибрати інструментальні засоби роботи з базою даних.
5. Розробити додаток для підтримки навчального процесу дисципліни «Алгоритми та структури даних».
6. Протестувати розроблене програмне забезпечення.

3

АНАЛІЗ АНАЛОГІВ

Функціонал	VisuAlgo	Algorithm Visualizer	LeetCode	Exercism	Teach Розроблено
Відображення інтерфейсу англійською та українською мовами	Тільки англійською	Тільки англійською	Тільки англійською	Тільки англійською	Англійською та українською
Можливість створення сторінок курсу у форматі Markdown	-	+	-	+	+
Можливість завантаження та перегляду курсу у форматі PDF	-	-	-	-	+
Можливість створення сторінок курсу у форматі HTML у Rich Text	-	-	-	-	+
Можливість створення дизайну сторінок курсу	-	+	-	+	+
Відображення статистики результатів тестувань	-	-	+	+	+
Можливість створення тестів для курсу	-	-	-	+	+

4

ВИМОГИ ДО ДОДАТКУ

Функціональні вимоги

1. Можливість підтримки бази даних тестових завдань.
2. Аутентифікація користувача.
3. Наявність ролей Адміністратора та Користувача.
4. Ведення бази даних Користувачів та Адміністраторів.
5. Запам'ятовування результатів тестувань в базі даних.
6. Виведення результатів тестування.

Нефункціональні вимоги

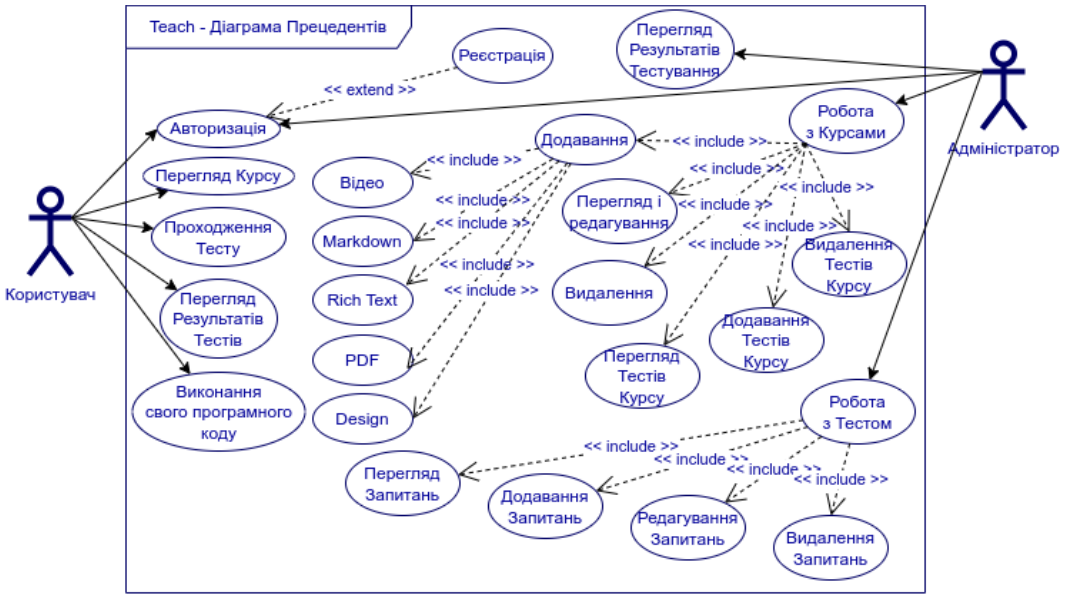
1. Можливість розширення: додавання інших курсів та матеріалів.
2. Додавання матеріалів курсу в різних форматах.
3. Відображення інтерфейсу додатку та матеріалів курсів різними мовами.

5

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

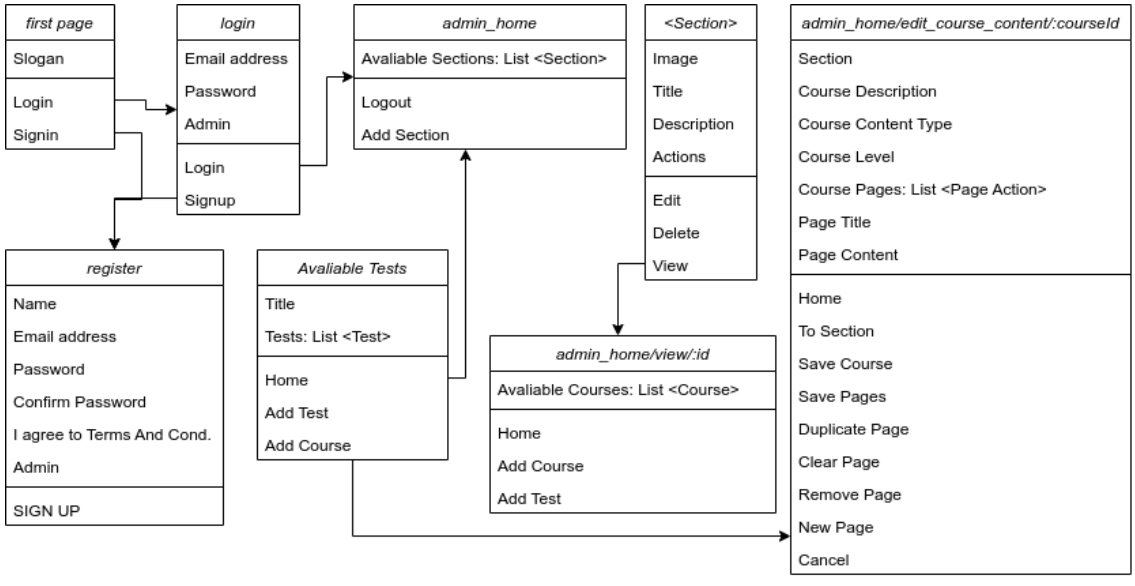
6

ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ



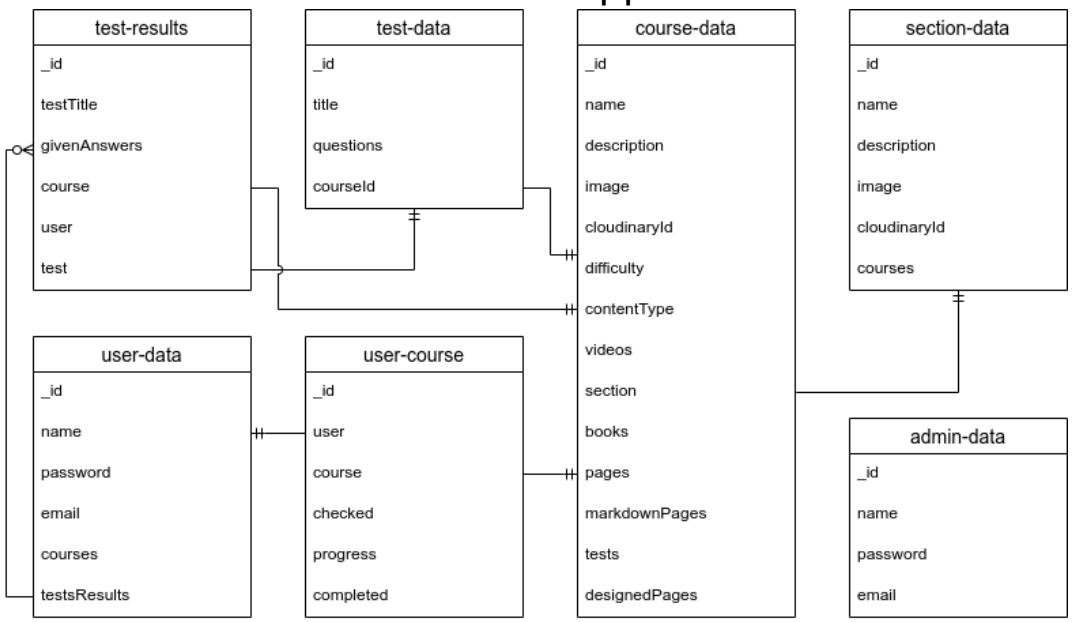
7

ДІАГРАМА КЛАСІВ



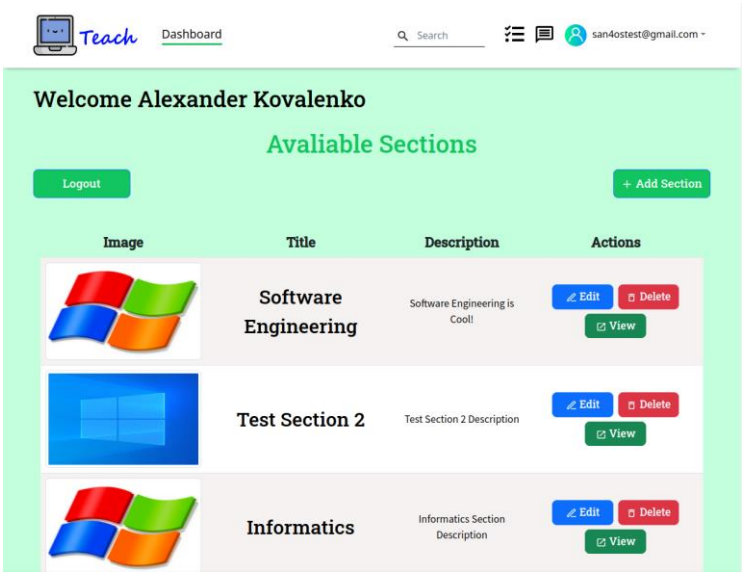
8

СХЕМА БАЗИ ДАНИХ



9

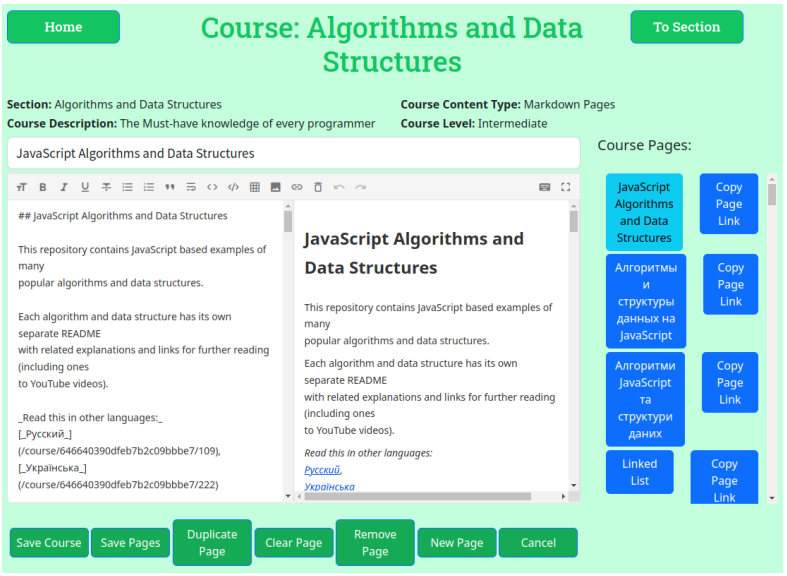
ЕКРАННІ ФОРМИ



Домашня сторінка Адміністратора

10

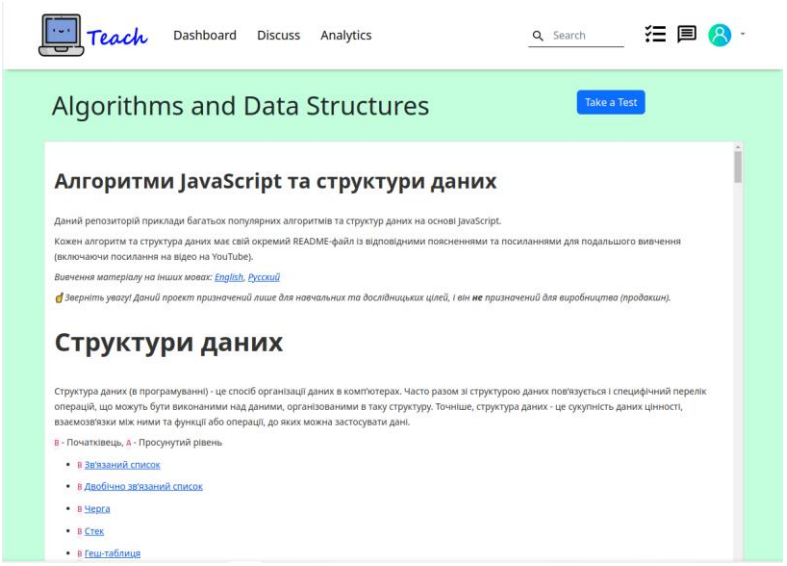
ЕКРАННІ ФОРМИ



Сторінка редагування курсу в форматі Markdown

11

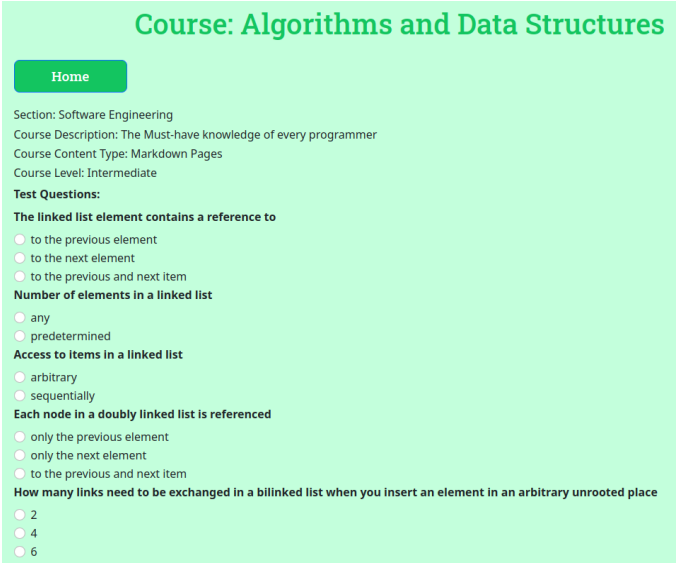
ЕКРАННІ ФОРМИ



Сторінка перегляду курсу Користувачем

12

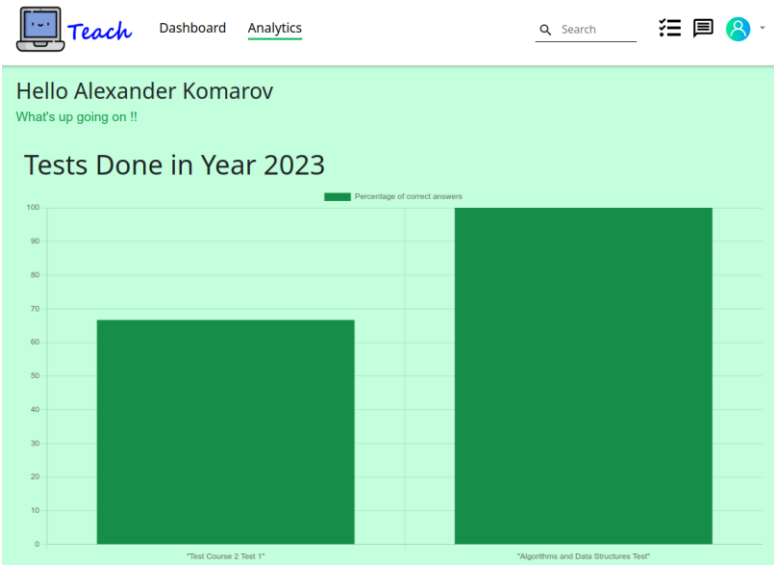
ЕКРАННІ ФОРМИ



Сторінка складання тесту курсу Користувачем

13

ЕКРАННІ ФОРМИ



Сторінка з виводом результатів тестування

14

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Негоденко О.В. Переваги вивчення дисципліни "Алгоритми та структури даних" за допомогою програмного забезпечення / О.В. Комаров, О.В. Негоденко // Застосування програмного забезпечення в інфокомунікаційних технологіях: Матеріали всеукраїнської науково-технічної конференції. Збірник тез. 20.04.2023, ДУТ, м. Київ — К.: ДУТ, 2023. — С. 92.

15

ВИСНОВКИ

1. Проведено аналіз особливостей розробки та практичної реалізації програмного забезпечення для навчання.
2. Розглянуто програмне забезпечення, яке може бути використано для навчання дисципліні «Алгоритми та Структури даних».
3. Проведено аналіз засобів розробки програмного забезпечення. обрано мову програмування JavaScript.
4. Визначено об’єкт, предмет та мету бакалаврської роботи.
5. Розроблено програмне забезпечення для навчання курсу «Алгоритми та Структури даних».

16

ДЯКУЮ ЗА УВАГУ!