МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
Факультет кібербезпеки, комп'ютерної та програмної інженерії
Кафедра комп'ютерних інформаційних технологій

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри
_____Аліна САВЧЕНКО
«___» _____2021 р.

ДИПЛОМНА РОБОТА

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ

"МАГІСТРА"

ЗА ОСВІТНЬО-ПРОФЕСІЙНОЮ ПРОГРАМОЮ "ІНФОРМАЦІЙНІ УПРАВЛЯЮЧІ СИСТЕМИ ТА ТЕХНОЛОГІЇ"

Тема: **"Метод навчання програмуванню із застосуванням web-технологій"**

Виконавець: Кравченко Артем Олегович

Керівник: к.т.н., Колісник Олена Василівна

Нормоконтролер: _____ Ігор РАЙЧЕВ

Київ 2021

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп’ютерної та програмної інженерії

Кафедра Комп’ютерних інформаційних технологій

Галузь знань, спеціальність, освітньо-професійна програма: 12 “Інформаційні технології”, 122 “Комп’ютерні науки”, “Інформаційні управляючі системи та технології”

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Аліна САВЧЕНКО

«____»_____2021р.

ЗАВДАННЯ

на виконання дипломної роботи студента

Кравченка Артема Олеговича

(прізвище, ім'я, по батькові)

1. Тема роботи: «Метод навчання програмуванню із застосуванням web-технологій» затверджена наказом ректора від 12.10.2021 за № 2228/ст.

2. Термін виконання роботи: з 12.10.2021 по 31.12.2021.

3. Вихідні дані до роботи: теоретичні відомості та основи проектування інформаційних систем, множина відомих архітектур програмних компонентів і додатків та програмних систем (ПС), множина відомих патернів.

4. Зміст пояснювальної записки: вступ, аналіз існуючих рішень в предметній області, огляд основних методологічних підходів в навчанні та аналіз інструментів і технологій для реалізації web-додатку, опис та технологія розроблення методу за допомогою веб-технологій.

5. Перелік обов'язкового ілюстративного матеріалу: слайди, презентація.

6. Календарний план-графік

№ п/п	Завдання	Термін виконання завдання	Підпис керівника
1.	Отримання завдання на дипломну роботу та побудова плану графіка виконання робіт.	12.10.2021 – 15.10.2021	
2.	Пошук ресурсів з навчання програмуванню.	16.10.2021 – 19.10.2021	
3.	Вибір найбільш успішних проектних рішень для їх подальшого розгляду.	20.10.2021 – 24.10.2021	
4.	Опис вибраних ресурсів з навчання програмуванню.	25.10.2021 – 31.10.2021	
5.	Написання Розділу 1 дипломної роботи.	01.11.2021 – 07.11.2021	
6.	Розробка та реалізація програмного комплексу управління репозитарієм патернів. Написання Розділу 2 дипломної роботи.	08.11.2021 – 17.11.2021	
7.	Написання Розділу 3 дипломної роботи. Завершення створення пояснювальної записки дипломної роботи.	18.11.2021 – 01.12.2021	
8.	Оформлення та друк пояснювальної записки.	02.12.2021 – 11.12.2021	
9.	Створення презентації, доповіді та підготовка до захисту дипломної роботи.	12.12.2021 – 20.12.2021	

7. Дата видачі завдання: 12.10.2021р.

Керівник дипломної роботи _____ Олена КОЛІСНИК

(підпис керівника)

Завдання прийняв до виконання _____ Артем КРАВЧЕНКО

(підпис випускника)

РЕФЕРАТ

Пояснювальна записка до дипломної роботи «Метод навчання програмуванню із застосуванням web-технологій» складається із вступу, трьох розділів, загальних висновків, списку бібліографічних посилань і містить 92 сторінку, 38 рисунків та 1 таблиці. Список бібліографічних посилань містить 18 найменувань.

Мета роботи: розробка методу для навчання програмуванню із застосовунням web-технологій.

Методи дослідження: об’єктно орієнтоване моделювання, розробка фронт-енд частини, розробка бек-енд частини, розробка та підключення бази даних.

Об'єкт дослідження: методи навчання прграмуванню, web-технології для реалізації методу.

Предмет дослідження: метод навчання програмуванню.

Метою магістерської роботи є дослідження та розробка методу навчання програмуванню із використанням веб-технологій. Актуальність дослідження обумовлена необхідністю вирішення проблем підвищення якості освіти шляхом вдосконалення підходу до навчання студентів програмуванню з використанням веб-технологій. Результатом виконання магістерської роботи є розроблений веб-додаток, який реалізує, власне сам метод навчання програмуванню. Функціональність додатку передбачає можливість навчання як з мобільніх пристроїв так і з інших. Реалізована функціональність компіляції та запуску консольних додатків (режим Пісочниця), додавання коментарів під кожним блоком конентку для кращої взаємодії користувачів між собою та авторами навчальних ресурсів.

Ключові слова: ІНФОРМАЦІЙНА СИСТЕМА, МЕТОД НАВЧАННЯ, ВЕБ-ДОДАТОК, ANGULAR, NESTJS, КОМПІЛЯТОР КОДУ, MONGODB, VS CODE.

ЗМІСТ

ВСТУП 8
РОЗДІЛ 1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ В ПРЕДМЕТНІЙ ОБЛАСТІ..... 10
1.1. Codeacademy – інтерактивна онлайн-платформа з навчання 7 мовам програмування. 10
1.2. Tynker – навчальна платформа з програмування спрямована на навчання дітей. 11
1.3. Udacity – web-сервіс з масових відкритих онлайн курсів з інформаційних технологій 12
1.4. Ресурс LRN з вивчення мов програмування і мови розмітки гіпертексту. 14
1.5. Swift Playgrounds – освітній інструмент та середовище розробки мови програмування Swift для платформи IOS 15
1.6. Courséra – веб-ресурс для навчання інформаційним технологіям онлайн. 16
1.7. W3Schools – безкоштовний навчальний веб-сайт для навчання кодуванню в Інтернеті. 17
1.8. EdX – інтернет платформа масових відкритих інтерактивних курсів онлайн 18
1.9. Codewars – web-ресурс з безліччю завдань по програмуванню і їх вирішинню 20
1.10. CodeCombat – HTML5 рольова гра, для навчання базовим концептам програмування. 21
1.11. Аналіз порівняння функціоналу систем з вивчення мов програмування. 22
1.12. Висновок за розділом 1. 24
РОЗДІЛ 2. МЕТОДОЛОГІЧНІ ПІДХОДИ В НАВЧАННІ ТА АНАЛІЗ ІНСТРУМЕНТІВ І ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ ВЕБ-ДОДАТКУ 25
2.1. Основні методологічні підходи до навчання. 25
2.1.1. Особистісний підхід до навчання. 25
2.1.2. Діяльнісний підхід до навчання. 26
2.1.3. Системний підхід до навчання. 27
2.1.4. Індивідуальний підхід до навчання. 28
2.2. Аналіз інструментів і технологій для реалізації web-сервісу. 30

2.2.2. Sass – скриптова метамова, яка інтерпретується в каскадні таблиці стилів. 32
2.3.3. JavaScript – динамічна, об'єктно-орієнтована прототипна мова програмування. 34
2.3.4. Angular – JavaScript фреймворк для реалізації клієнтської частини. 39
2.3.5. MongoDB – документо-орієнтована система керування базами даних. 41
2.3.6. Node.js – платформа для виконання мережевих застосунків, написаних мовою JavaScript. 43
2.3.7. Express.js – Node.js фреймворк для реалізації серверної частини. 47
2.3.8. Visual Studio Code — середовище для створення, редагування та завантаження програм. 48
2.3.9. Капмілятор коду в режимі реального часу Sandbox. 49
2.3.10. Технологія Drag-and-drop. 51
2.4. Висновок за розділом 2. 52
РОЗДІЛ 3. РЕАЛІЗАЦІЯ ПРОЕКТНИХ РІШЕНЬ. 54
3.1. Загальний опис розробленого продукту. 54
3.2. Реалізація клієнтської частини веб-додатку. 70
3.3. Реалізація серверної частини веб-додатку 77
3.4. Висновок за розділом 3. 88
ВИСНОВКИ 90
СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ 91

ВСТУП

На 2021 рік існує багато різних методів навчанню програмуванню починаючи з простих та безкоштовних і закінчуючи повноцінними які є платними, або частково безкоштовними. Такі методи використовують різні підходи та інструменти.

В наші дні учню важко впоратися з колосальним потоком інформації, який щодня обрушується на нього. Навіть хороша пам'ять не завжди в змозі зберегти гігантський масив інформації. Саме тому з'являються нові завдання в навчанні, пов'язані з чітким відбором навчального матеріалу, структуруванням курів програмування, складання удосконалених методик вивчення, що дозволяють за малу кількість часу отримувати максимум інформації.

Мета магістерської роботи – дослідження та розробка методу навчання програмуванню з використанням web-технологій. В роботі передбачається розробка веб-сервісу для вивчення мов програмування.

Актуальність роботи зумовлена тим, що в наш час, різко зростає попит на навчальні методи та ресурси з програмування, а сфера інформаційних технологій показує найбільшу динаміку зростання і попит на спеціалістів в цій сфері постійно зростає.

Грамотна реалізація подібного методу надає можливість всім охочим ознайомитись з основами програмування та практично закріплювати теоретичний матеріал. Цей метод також орієнтується на те щоб студент зміг після проходження всіх необхідних матеріалів претендувати на складання валідного резюме та подальшого працевлаштування. Реалізація зручного інтерфейсу зробить процес навчання більш комфортним та зрозумілим для людей що тільки починають знайомитись з сферою інформаційних технологій.

Для досягнення поставленої мети в роботі необхідно вирішити наступні завдання:

− провести аналіз подібних існуючих методів навчання програмування;

− обґрунтувати вибір підходів, що будуть використовуватися для вирішення проблеми ефективного вивчання програмування;

− надати методологічні основи вирішення поставленої проблеми;

− виконати проектування web-сервісу із зазначеними технологіями;

− розробити алгоритми та програмні реалізації web-сервісу для вивчення програмування.

РОЗДІЛ 1

АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ В ПРЕДМЕТНІЙ ОБЛАСТІ

1.1. Codeacademy – інтерактивна онлайн-платформа з навчання 7 мовам програмування

Codecademy – онлайн-платформа для навчання 12 мовам програмування (рис.1.1): Ruby, Python, Java, PHP, JavaScript та іншим, а також робота з JavaScript бібліотекою jQuery та мовою гіпертекстової розмітки HTML і стильового оформлення веб-сторінок CSS.

Як заявляють адміністратори цієї онлайн-платформи, за станом на січень 2014 року, 24 мільйони зареєстрованих користувачів виконали більше ніж 100 мільйонів вправ. Онлайн-платформа Codecademy отримала безліч позитивних відгуків в багатьох спільнотах мережі інтернет, а також таких виданнях як New York Times і TechCrunch.

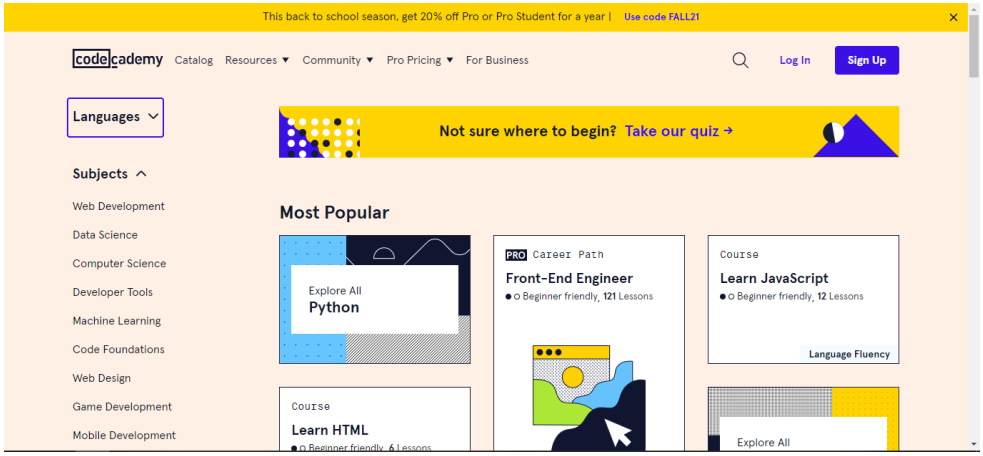

Рис. 1.1. Головна сторінка онлайн-платформи Codecademy

Для того щоб мати повноцінний доступ до функціоналу користувачі повинні мати свій власний профіль, що створюється за допомогою декількох простих кроків.

В онлайн-платформі існує система досягень за виконання завдання, а також індикатор прогресу проходження курсу навчання, який можна побачити всім охочим користувачам. Також на платформі доступні словники CSS та HTML в межах одного курсу. На сайті існує функціонал, який дозволяє користувачам створювати і публікувати свої курси, використовуючи фуркціонал «Творця курсу».Також на даному ресурсі існує форму, на якому починаюці та вже з досвідом розробники мають можливість обімінюватись думками та різними ресурсами для допомоги один одному. Для деяких навачальних курсів передбачений функціонал «пісочниці», в якому користувачі мають змогу тестувати свой програмне рішення в виді коду. Прийнявши участь в Тижні Освітньої Інформатики напочатку зими 2013 року, Codecademy створила свій перший iOS-додаток «Година Кодексу». В цілому цей додаток орієнтується на людей, які хочуть вивчати програмування в ігровому вигляді.

1.2. Tynker – навчальна платформа з програмування спрямована на навчання дітей

Навчальна платформа Tynker – це одне з найпопулярніших рішень в сегменті навчання дітей веб-технологіям (рис.1.2). Платформа використовується більше більш ніж у 8000 шкіл і сприяє понад 6 млн дітям почати знайомство та виввчення з інформаційними технологіями. В рамках платформи також розроблений iOS-додаток для користуваців Apple. Компанія Tynker заснована в 2012 році в США в місті Маунтін-В'ю, що в штаті Каліфорнія. З фондами, залученими від окремих фінансових агентів та інституційних інвесторів, Tynker for Schools вийшов в світ в квітні 2013 року, а "Tynker for Home" одним роком пізніше.

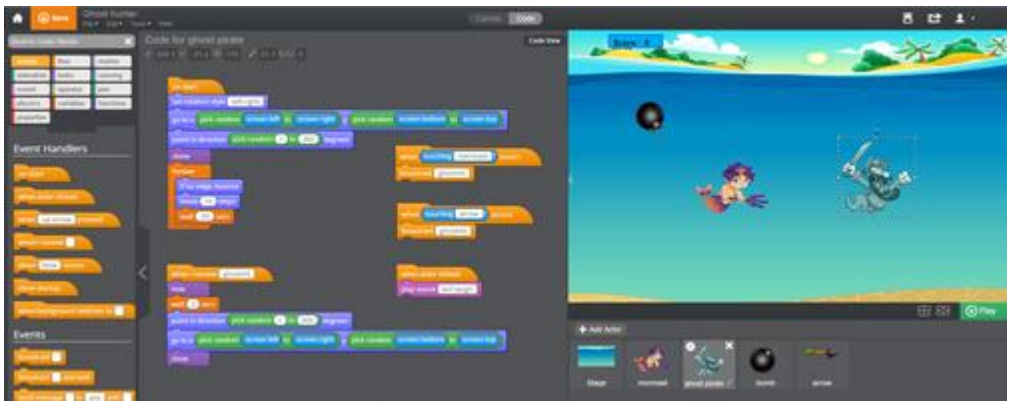

Рис. 1.2. Навчальна платформа Tynker

Tynker – платформа що спрямована на навчання програмуванню з орієнтацією на дітей, а саме, як створювати ігри та програми. Замість того, щоб писати вхідний код, можна візуально перетягувати блоки коду для більшої зручності дітям. Інтрфейс користувача та принципи користивування засновані на безкоштовній платформі Scratch, Hopscotch та Snap. На відміну від Scratch, навчальна платформа Tynker не заснована на власних технологіях Adobe Flash, але використовує HTML5 і JavaScript, і може працювати в браузері без плагінів, а також на планшетах і смартфонах. Ще одна відмінність полягає в тому, що Scratch – це безкоштовний продукт із відкритим кодом, а Tynker – комерційний проект, призначений для продажу навчальних курсів. У середині літа 2014 року Tynker випущений для iPad та Android. Проекти можна скачати з мережі і вони будуть працювати на будь-якій платформі.

1.3. Udacity – web-сервіс з масових відкритих онлайн курсів з інформаційних технологій

Web-сервіс Udacity – ресурс для навчання інформаційним технологіям, що створив безкоштовний додаток, в якому користувач має моєливість істотно підвищити рівень своїх знань, проходячи безкоштовні курси від провідних експертів IT-галузі (рис.1.3). Додаток доступний для iOS та Android.

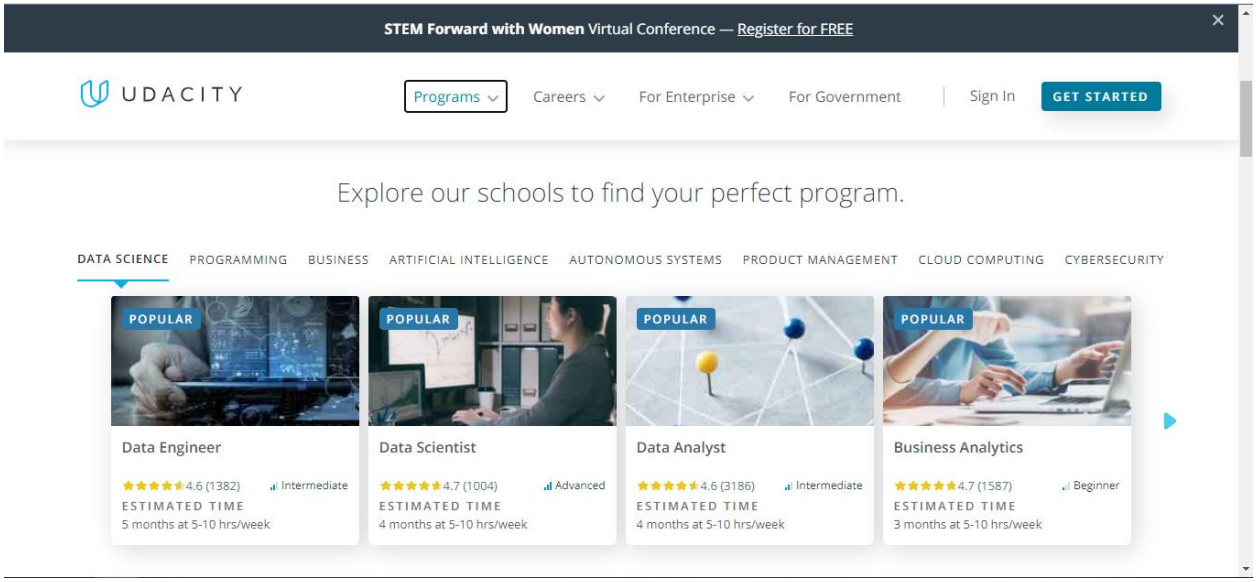

Рис. 1.3. Головна сторінка ресурсу Udacity

Udacity – приватна освітня організація, заснована Майклом Сокольський Себастьяном Труном і Девідом Ставенсом з метою збільшення достопності освіти. Компанія створена в результаті розширення програми з інформатики Стенфордського університету. Курси що є на ресурсі - безкоштовні і доступні через мережу Інтернет, прослухати їх можуть всі бажаючі. На початку існування ресурсу пропонувалося всього шість курсів. Станом на 1 жовтня 2012 року сервіс Udacity пропонує вже 14 курсів. Тоді число студентів вже становило десятки тисяч. Про заснування Udacity було оголошено на конференції Digital Life Design в 2012 році. Того ж року Себастьян Трун відзначений газетою The Guardian як людина, що вніс значний внесок в розвиток відкритого Інтернету. Лекції англійською мовою з субтитрами в поєднанні з функціоналом вбудованих тестів і домашніми роботами, засновані на моделі «вчитися на практиці». Лекції мають функціонал вбудованих тестів, щоб дати студентам можливість краще зрозуміти концепції та ідеї які пропонують лекції. До фізики, робототехніки та штучного інтелекту додалися курси з створення стартапів і ведення блогів, але, на жаль не всі курси були таким ж успішними. Багато ідей виявились провальними, наприклад, курс з дискретної математики – рівень, якого дещо низьким порівняно з іншими курсами інших ресурсів. Також до неоднозначних показників відноситься те

що, на форумах багато незадоволених користувачі: курси, на їх погляд, перебувають за межею можливостей і, часто, штучно ускладнені. Команда сервісу приймає це до відома – простіше, навряд чи буде, але методи викладання матеріалу і тести перманентно переглядаються, щоб якомога більше студентів мали можливість освоїти предмет, незважаючи на складність навчального метеріалу.

1.4. Ресурс LRN з вивчення мов програмування і мови розмітки гіпертексту

Lrn – це веб-сервіс, який допоможе вам навчитися впевнено кодувати однією з популярних мов програмування: HTML, CSS, Python, Ruby, Javascript і багато інших (рис. 1.4). Доступний як додаток для iOS (версія Android в розробці).

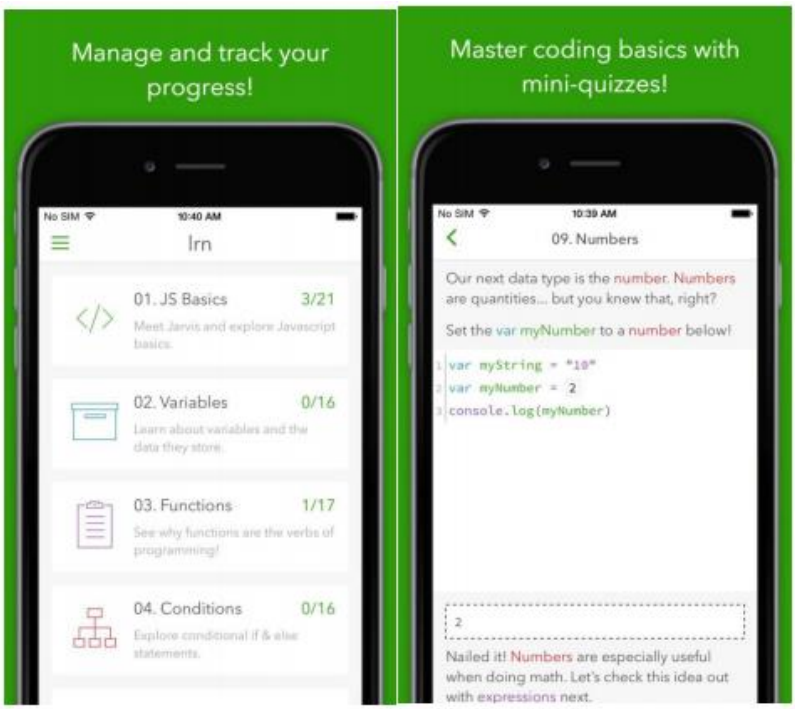

Рис. 1.4. Додаток Lrn

Цей додаток допоможе вивчити код у HTML, CSS, Ruby, Python та Javascript за допомогою функціоналу інтерактивних міні-тестів. Користувачі зрозуміють, як читати, писати та говорити мовою програмування. Попередній досвід, а також інтернет чи клавіатура не обов’язкові в наявності. Більше 400

безкоштовних міні-головоломок та понад 200 інших міні -вікторини можна розблокувати в додатку. Курс включає наступні розділи: Курс HTML: Введення в HTML, макети, форми, таблиці, списки; Курс CSS: властивості, селектори, інтервали, позиціонери; Курс Javascript: основи JS, змінні, цикли, оператори умови, функції, масиви, багатомірні масиви, об'єкти, замикання; Курс Python: основи PY, змінні, функції, терміни, списки, словники, цикли; Курс Ruby: Введення в Ruby, змінні, методи, умови, цикли, набори, класи.

1.5. Swift Playgrounds – освітній інструмент та середовище розробки мови програмування Swift для платформи IOS

Swift Playgrounds – це додаток для iPad для вивчення кодування методом експерементування з кодом (рис. 1.6.). Користувачі вирішують інтерактивні вікторини на заняттях "Управління кодом", щоб вивчити основи кодування. Також є додаткові завдання, які дозволяють вивчати код і створювати програми, які є складнішими.

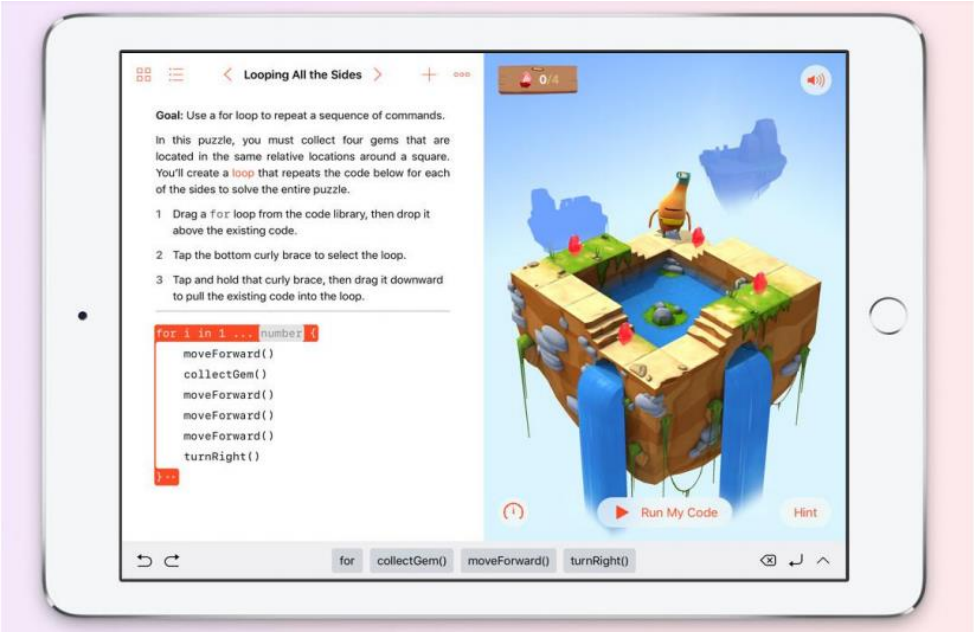

Рис. 1.5. Додаток Swift Playgrounds

Додаток Swift Playgrounds не вимагає знань кодування, тому підходить для початківців. Користувачі можуть постійно вивчати Swift – мову програмування, створену Apple, якою користуються розробрики на платформі

iOS для створення багатьох популярних сьогодні програм. А оскільки він створений для того, щоб отримати максимальну віддачу від iPad та справжнього iOS SDK, Swift Playgrounds - це перший у своєму роді навчальний досвід. Вбудовані навчальні заняття Apple допоможуть вам: розібрати основні концепції програмування за допомогою коду для вирішення головоломок; Переглядати свій код у гарному та інтерактивному 3D - вигляді, який можна прокрутити пальцем для збільшення; анімації застосовують кожну нову концепцію кодування високого рівня, перш ніж зануритися в таємниці; вибирати одного з трьох анімованих символів, для виконання кодових дій; скористатись словником, який містить повний перелік поширених термінів, для цього потрібно просто торкнітися слова в інструкціях, для отримання додаткової інформації.

1.6. Courséra – веб-ресурс для навчання інформаційним технологіям онлайн

Онлайн-платформа Coursera заснована професорами Стенфордського університету та Дафною Коллер, Ендрю Нгом, яка пропонує курси, спеціалізації та ступені (рис. 1.6). Coursera співпрацює з університетами та іншими установами, щоб пропонувати онлайн -курси, спеціалізації та ступені в різних предметних областях, таких як гуманітарні науки, інженерія, біологія, медицина та суспільствознавство, бізнес, інформатика, математика, цифровий маркетинг, комп’ютерні науки багато інших. Станом на червень 2018 року в Courséra зареєстровано понад 33 мільйони користувачів та понад 2400 курсів з багатьох дисциплін.

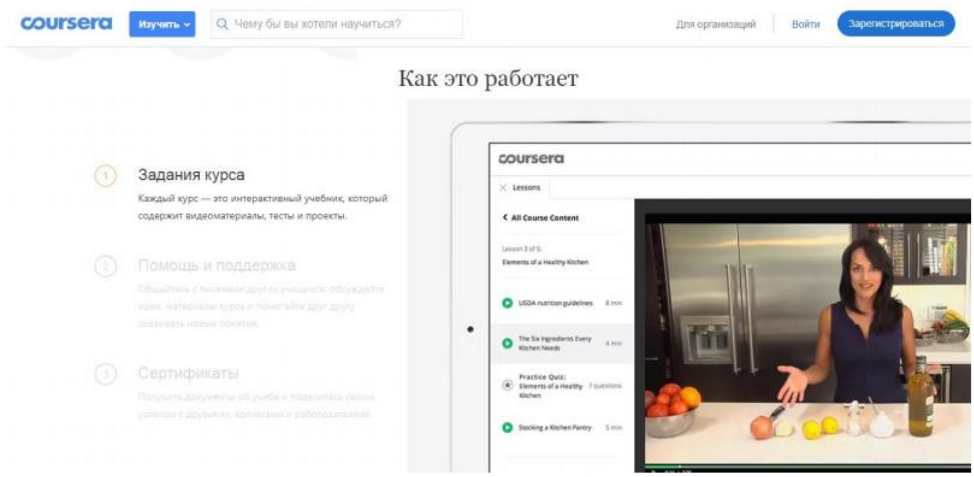

Рис. 1.6. Головна сторінка ресурсу Coursera

Coursera заснована в 2012 році професорами комп'ютерних наук Стенфордського університету Дафною Коллер і Ендрю Нгом, Стенфордський, Мічиганський, Прінстонський та Пенсільванський університети є одними з перших університетів, які пропонують контент на платформі. Пропозиції з початку розширення ресурсу включають спеціалізації - набори курсів, які формують навички з певної теми, а також кваліфікації та продукти розвитку праці для організацій, бізнесу та уряду. Початкове фінансування проекту склало 16 мільйонів доларів за підтримки Byers & Kleiner Perkins Caufield та New Enterprise Associates. У 2015 році EDB Investments очолив Серію С у галузі венчурного фінансування на загальну суму понад 60 мільйонів доларів. На 2019 рік компанія отримала 146,1 млн доларів фінансування.

1.7. W3Schools – безкоштовний навчальний веб-сайт для навчання кодуванню в Інтернеті

W3Schools – це веб-ресурс для веб-розробників, який пропонує підручники та посилання на такі мови та технології веб-розробки, як HTML, CSS, JavaScript, jQuery, Bootstrap, SQL, Python, PHP, C#, AngularJS та Java що охоплюють більшість аспектів веб-розробки.

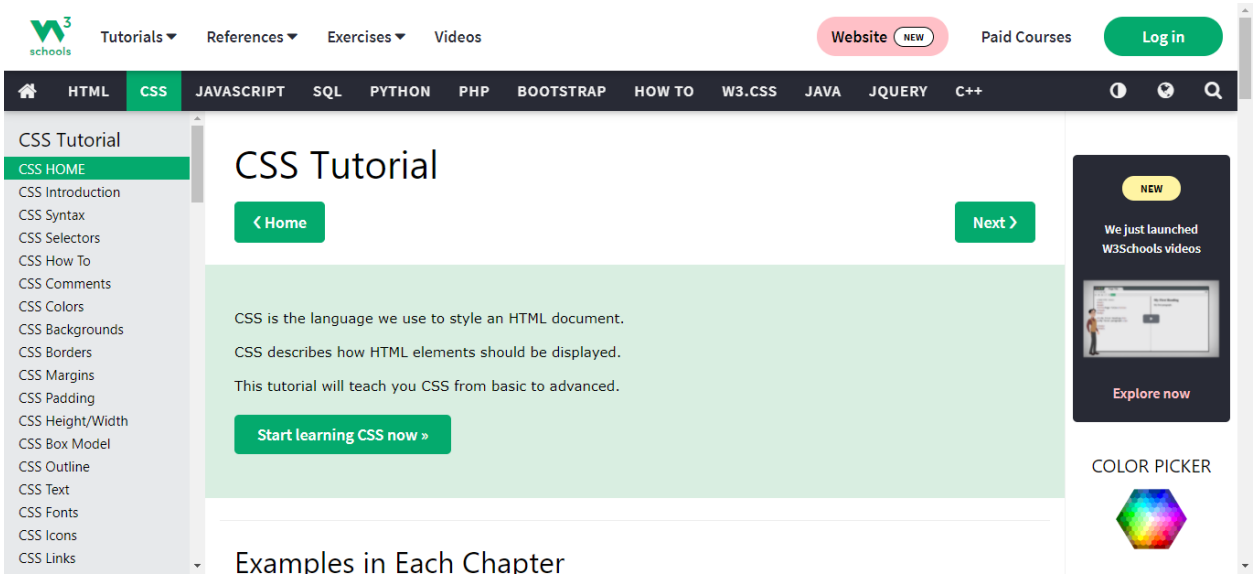

Рис. 1.7. Головна сторінка ресурсу W3Schools

Ресурс отримав свою назву від World Wide Web, але на пряму не пов'язаний з цим поняттям.

W3Schools – в 1998 році норвезькою компанією, яка спеціалізувалася на розробці програмного забезпечення та консалтингу деяких компаній.

W3Schools – це школа для веб-розробників, що охоплює всі аспекти веб-розробки, та надає можливості навчання як для початківців, так і для більш досвідчених розробників. Ресурс часто використовується розробниками різних рівней кваліфікації в якості документації по написанню певних технологічних моментів в реалізації програмного функціоналу. Найбільш популярними документаціями на цьому ресурсі є HTML та CSS.

1.8. EdX – інтернет платформа масових відкритих інтерактивних курсів онлайн

EdX – це освітня платформа, яка пропонує безкоштовні онлайн-курси найкращих університетів світу (рис. 1.8.). Також доступний як додаток для iOS та Android. EdX – постачальник багатьох відкритих онлайн-курсів (MOOC). Він пропонує студентам у всьому світі онлайн-курси вищих навчальних

закладів з широкого спектра дисциплін, включаючи безкоштовні курси. EdX також проводить дослідження у сфері навчання на основі аналізу начання користувачів платформи, та її відгуків і коментарів. EdX-це некомерційна організація, яка працює на безкоштовній платформі OpenX з відкритим кодом OpenX.

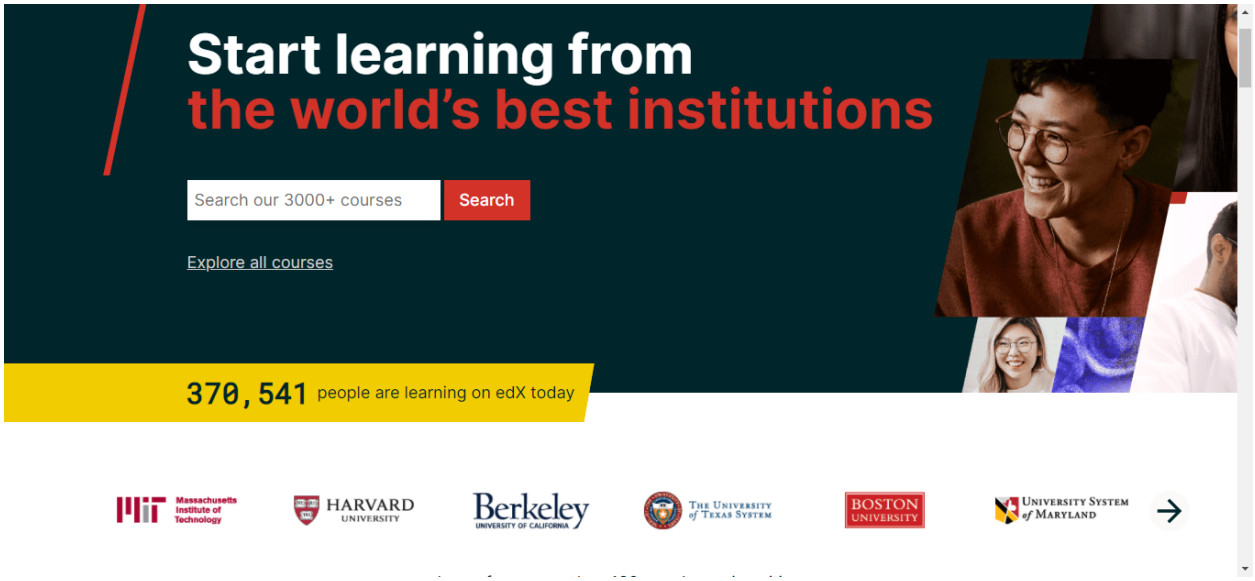

Рис. 1.8. Інтернет платформа EdX

У травні 2012 року в Гарвардському університеті та Массачусетському технологічному інституті створено EDX. Більше 70 шкіл, корпорацій та некомерційних організацій пропонують або планують пропонувати курси на веб-сайті edX. Станом на 29 грудня 2017 року в EdX навчається приблизно 14 мільйонів студентів, які проходять понад 1800 курсів. Курси EdX включають щотижневі послідовності навчання. Кожен навчальний цикл містить короткі відеоролики, включені до інтерактивного уроку, де учні можуть одразу попрактикуватися у концепціях що викладені в відео. Курси часто включають в себе навчальні відеоролики, подібні до дискусійних груп для малих шкіл, онлайн-підручники та онлайн-дискусійні форуми, де учні можуть розміщувати та переглядати запитання та коментарі один до одного, а також асистентів. При необхідності до курсу включаються онлайн-лабораторії. Наприклад, у перших MOOC EdX – курсах схем та електроніки – студенти створювали віртуальні схеми в онлайн -лабораторії. EdX пропонує

сертифікати про успішне закінчення та деякі курси, які підлягають кредитуванню. Рішення про те, чи пропонує коледж чи університет кредит для онлайн-курсу, вирішує школа. EdX пропонує різноманітні способи проходження курсів, включаючи перевірені курси, де студенти мають можливість ознайомитись з курсом (безкоштовно) або працювати на сертифікаті edX (вартість формується від конкретного курсу). Для курсів, опублікованих до 7 грудня 2015 р., Можна отримати курси з відзнакою поверх сертифіката з відзнакою (безкоштовно). EdX також пропонує сертифікат XSeries для проходження набору з двох-семи перевірених курсів з предмета (вартість залежить від курсу).

1.9. Codewars – web-ресурс з безліччю завдань по програмуванню і їх вирішинню

CodeWars – це сервіс, де ви можете конкурувати в програмуванні з іншими учасниками (Рис. 1.9). Користувачі мають доступ до різних мов програмування та завдань, які поділяються на рівні складності.

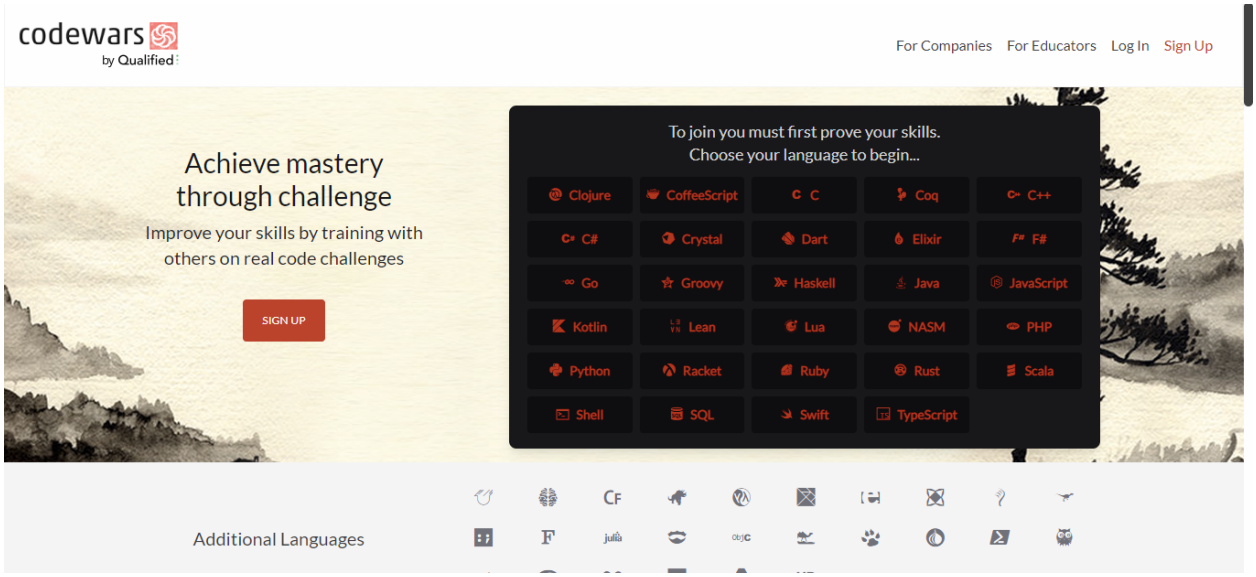

Рис. 1.9. Головна сторінка ресурсу Codewars

На цій освітній платформі розробники програмного забезпечення готуються до програмних завдань, відомих як ката. Ці завдання з

програмування допомогають розвивати логічне мислення в імперативній парадигмі мислення. Всі завдання можна виконувати в рамках сервісного онлайн IDE. Також, якщо завдання занадто складне, то користувач може подивитися рішення до цього завдання в інших користувачів, які розміщенні нижче завдання, але за це дещо знімається рейтинг користувача, який формується на основі проходження завдань.

Компанія Codewars заснована Натаном Доктором та Джейком Хоффнером в листопаді 2012 року, проект спочатку розпочався на конкурсі Startup Weekend того року, де він створений в якості прототипу. Він отримав перше місце в цьому конкурсі, привернувши увагу спеціалістів та зацікавившись фінансуванням від двох суддів конкрусу Пейджа Крейга (інвестор) та Брайана Лі (підприємець).

Після створення першої версії платформи вона запущена до спільноти Hacker News, отримавши значну увагу завдяки її формату та залучивши близько 10000 нових користувачів протягом Startup Weekend.

1.10. CodeCombat – HTML5 гра, для навчання базовим концептам програмування

CodeCombat – одна з найпопулярніших навчальних ігор з програмування (рис. 1.10.). Додаток часто використовуються в навчальних закладах для знайомства з розрозбкою програмного забезпечення. Включає в себе курси для всіх вікових категорій. CodeCombat – це навчальна ігрова компанія, заснована в Сан -Франциско, Каліфорнія. Компанія створила та підтримує браузерну відеогру, яка навчає гравців навичкам програмування.

CodeCombat заснована у лютому 2013 року Ніком Зімом, Скоттом Еріксоном та Джорджем Сейнсом, які раніше працювали над розробкою програмного забезпечення для вивчення мов Skritter.

У 2014 році компанія отримала підтримку на етапі раннього венчурного капіталу Y Combinator. За інформацією CodeCombat, їхня продукція безкоштовна, і компанія планує заробляти прибуток, за рахунок реклами та

пожертв від користувачів. Також CodeCombat запровадив щомісячну платну підписку, яка надає доступ до додаткового ігрового контенту.

Рис. 1.10. Головна сторінка ресурсу CodeCombat

CodeCombat – це рольова гра на основі браузера, яка навчає мовам програмування Javascript та Python, а також основамкомп’ютерних наук. Щоб підняти рівень в грі, гравець повинен перевірити свої знання, написавши код. Гра включає однокомпонентні та багатокористувацькі компоненти і розрахований на аудиторію середніх шкіл. Гра позитивно оцінена журналом PC Magazine, хоча рецензенти відзначили, що вміст гри нагадує "казуальну гру-браузер" і "переважно не запам'ятовується". У січні 2014 року CodeCombat створив власне програмне забезпечення з відкритим кодом та реалізував редактор ігрових рівнів, щоб користувачі могли створювати власний ігровий вміст.

1.11. Аналіз порівняння функціоналу систем з вивчення мов програмування

У рамках проведення досліджень різних систем та їх методів навчання, складемо таблицю з найважливішим функціоналом для кожної інформаційної системи з вивчення мов програмування. Результати порівняльного аналізу

функціональних можливостей досліджуваних навчальних систем наведені в таблиці. 1.1.

Таблиця 1.1

Порівняльний аналіз функціональних можливостей навчальних систем, що досліджуються

Система	Khan	Tynker	Lrn	Udacity	Coursera	W3schools	CodeCombat	Codewars	Swift Playgrounds	EdX
Теорія	-	-	-	+	+	+	-	-	+	+
Сертифікування	+	+	+	+	-	+	-	-	+	-
Дизайн	-	+	+	+	+	+	+	+	+	+
Тестування	-	+	+	+	-	+	-	+	+	-
Можливість програмування в реальному часі	+	+	+	+	+	+	+	+	+	+
Офлайн работа	-	+	-	-	+	-	-	-	-	+
БД	+	-	+	+	+	+	+	+	+	+
Пісочниця	+	+	+	-	-	+	+	+	+	-

Порівняльний аналіз показав, що більшість існуючих систем здійснюють вивчення та закріплення матеріалу за допомогою веб додатків, які інколи мають мобільні версії, але як правило веб-додатки розробляються таким чином щоб можна було користуватися ними не тільки з персонального комп’ютеру, але й з мобільних пристроїв без скачування додаткового додатуку з мобільних маркетів. Зазвичай в цих системах є можливість проглядати теоретичний матеріал та проходити тестування. Всі створені роботи зберігаються в базах данних тих чи інших навчальних систем. Є можливість їх як зберігати так і видаляти, але також існують і мінуси, наприклад: немає можливості іпортировати і експортувати данні та продукти при проходженні навчання; немає структуровані програми для більш ефективного вивчання матеріалу; відсутність функціоналу так званої пісочниці для програмування в режимі оффлайн. На сьогоднішній день всі програмні рішення, які схожі на запланований мною навчальний ресурс більш схожі на ігри. Найбільш вдалим

продуктом, в якомі є все що дійсно потрібно для легкого розуміння та вивчення необхідної інформації є веб-сервіс W3chools.

1.12. Висновок за розділом 1

Проаналізувавши безліч навчальних веб-сервісів, можна зробити висновок, що кожний з них використовує як свої власні, унікальні методи навчання так і ті, які вже показали свою ефективність та стали популярними в навчальних системах веб-сервісів. Найбільш цікавими на мою думку мені здалися два сервіси, в саме : w3schools та codewars. Але інші проаналізовані ресурси також мають свої цікаві особливості, які заслуговують на увагу при проектуванні власної навчальної системи.

Скомбінувавши навчальні концепції проаналізованих веб-ресурсів можна створити сервіс, що буде навчальним посібником та одночасно конкурентним середовищем в опануванні тієї чи іншої мови програмування представленої на цьому ресурсі. Це можна досягти з’єднавши навчальну частину з практичною, в якій буде не просто можливість закріплення матерів курсу за допомогою тестувань по конкретній темі, але й проходженням додаткових завдань різного рівня складності з можливістю опублікування рішення завдання, для його коментування, та оцінювання іншими користувачами своїми голосами. Рішення за найбільшою кількістю голосів будуть в окремій вкладці розміщенні впорядку зменшення голосів. Таким чином студент зможе не тільки вирішувати сам завдання, але й дивитися й коментувати рішення інших, що збільшить якість опанованих практичних навичок.

РОЗДІЛ 2

МЕТОДОЛОГІЧНІ ПІДХОДИ В НАВЧАННІ ТА АНАЛІЗ ІНСТРУМЕНТІВ І ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ ВЕБ-ДОДАТКУ.

2.1. Основні методологічні підходи до навчання

Термін «підхід» у дидактиці визначається як сукупність принципів, що визначають навчальну стратегію, при цьому кожен принцип регулює вирішення певних протиріч, що виникають в процесі навчання. Таким чином, змістом поняття «підхід» є певна ідея, сукупність принципів, концепція, що зумовлюють організацію того чи іншого процесу, явища, наприклад, процесу навчання програмуванню.

Існує ряд зарубіжних досліджень, присвячених розробці методичних підходів до навчання програмуванню, серед яких можна виділити: системний підхід (І. Одинцов і ін.), Діяльнісний підхід (Е.А. Ракітіна і ін.), Когнітивний підхід (J. Reinfelds і ін.), семіотичний підхід (P. Andersen, К. І. Баумана, Н.І. Рижова і ін.), проблемний підхід (Є. Касьянова, К.Ю. Поляков та ін.).

2.1.1. Підхід до навчання

Цей підхід визнає особистість як продукт суспільно-історичного розвитку і носія культури, і не допускає зведення особистості до натури індивіда. Особистість – мета, суб'єкт, результат і головний критерій ефективності педагогічного процесу при такому підході. Унікальність особистості – її морально-вольові якості. Завдання вихователя: створення умов для саморозвитку задатків і творчого потенціалу індивіда.

Особистісний підхід вимагає прийняття унікальності особистості, її інтелектуальної та моральної свободи, права на повагу та самореалізації.

Він передбачає опору в навчанні на природний процес саморозвитку задатків і творчого потенціалу особистості, створення для цього відповідних умов.

2.1.2. Діяльнісний підхід до навчання

Діяльнісний підхід в навчанні ґрунтується на "принциповій позиції, що людська психіка нерозривно пов'язана з її діяльністю та обумовлена нею". Категорія діяльності є базисом цього підходу. Водночас під діяльністю розуміють цілеспрямовану діяльність людини, виражену в процесі її взаємодії із зовнішнім світом, ця взаємодія включає вирішення життєво важливих завдань, що визначають існування та розвиток людини.

Діяльність – засіб та умова розвитку особистості, вона підходить для трансформації шаблону сприйняття навколишньої дійсності. Завдання вихователя – підібрати та організувати діяльність учнів з точки зору суб’єкта розпізнавання діяльності та спілкування (самої діяльності). Це включає: усвідомлення, постановку цілей, планування дій, організацію, оцінку результатів та самоаналіз (рефлексія). А.Н. Леонтьєв: "Для того, щоб опанувати досягнення людської культури, кожне нове покоління має здійснювати діяльність, подібну (хоча і не ідентичну) досягненням, що стоять за цими досягненнями", – написав він. Ось чому, щоб підготувати учнів до самостійного вирішення задач, необхідно максимально залучити їх до цієї діяльності, тобто організувати повноцінну в моральному і соціальному відношенні життєдіяльність. Діяльнісний підхід вимагає переведення учня на предмет пізнання, праці та комунікації. Важливим при цьому є те, що активність особистості та її потреби в самовдосконаленні повинні розглядатися як базис.

2.1.3. Системний підхід до навчання

Системний підхід базується на розгляді об’єктів як систем. Він орієнтує дослідження у напрямку дослідження цілісності об’єкта, виявлення різних типів зв’язків у ньому та об’єднання їх в єдину теоретичну картину. Аналіз наукової літератури показує, що основними категоріями системного підходу є система, структура та середовище.

Система – це сукупність елементів, які мають зв’язки між собою, утворюючи певну єдність та цілісність.

Крім того, дослідники системного підходу (В. С. Леднєв, В. Г. Афанасьєв та ін.) наголошують, що система - це сукупність об’єктів, взаємодія яких викликає появу нових інтеграційних якостей не характерних для окремо взятих компонентів які складають систему.

Структура – це стійкий набір зв’язків між елементами об’єкта, що забезпечує його цілісність та однорідність, тобто зберігаючи основні властивості при багатьох зовнішніх та внутрішніх змінах.

Структура є головною ознакою дійсності – «все влаштовано певним чином: усі об’єкти дійсності складаються з чогось, мають свої складові частини, і в той же час, якими б складними вони не були, вони самі є частиною чогось більш складного» І так далі, як у збільшенні обсягу та масштабу, так і в їх зменшенні. Більше того, все рухається – все тече, все змінюється».

Тому кожен об’єкт реальності вимагає їх перевірки двома способами: статичним та динамічним.

Середовище – група всіх об’єктів/сутностей, відсутні в системі, зміни їх властивостей або поведінки, що впливають на досліджувану систему, а також об’єкти чи сутності, властивості або поведінка яких змінюється залежно від поведінки системи.

Слід зазначити, що системний підхід до розпізнавання та перетворення будь-якого об’єкта є основним загальнонауковим підходом. Використання цього підходу у навчанні програмуванню дає змогу визначити компонент як

систему програмування навчання з усіма її характеристиками: цілісність, зв’язок, структуру та організацію, системні рівні та їх ієрархію, управління, самоорганізацію системи, її функції та розвитку.

Ідеї та принципи застосування системного підходу до викладання інформатики та програмування присвячені працям Н.В. Макарова, І.О. Одинцова та інші.

Основна ідея системного підходу в контексті викладання програмування полягає у розгляді навчальних завдань, тісно пов’язаних із засобами, методами програмування та взагалі з технологічним процесом. Дослідження кожної з цих галузей не слід проводити окремо, а так щоб між ними відбувався тісний зв'язок і взаємозалежність.

Враховуючи міжпредметні зв’язки програмування з іншими предметами та областями наукових знань, можна стверджувати, що системний підхід є фундаментальним для його викладання, а інші методологічні підходи, які можна застосувати для навчання програмуванню, мають доповнювати та поширювати його основні ідеї та принципи і не викликати протиріч по відношенню до цього підходу.

2.1.4. Індивідуальний підхід до навчання

Індивідуальний підхід передбачає, що в центрі навчання знаходиться сам учень – його мотиви, цілі, унікальний психологічний склад, тобто учень як особистість. Виходячи з інтересів учнів, рівня знань та вмінь, вчитель (той хто навчає) визначають освітню мету навчання та формують, спрямовують та коригують весь навчальний процес з метою розвитку особистості учня. Індивідуальний підхід передбачає, що в процесі навчання кожного предмета в повній мірі враховуються національні особливості, стать та вік, індивідуальні психологічні аспекти та статус студента.

Цей підхід здійснюється через зміст і форму самого навчального завдання, через характер спілкування зі студентами. Питання, коментарі,

завдання вирішуються для учнів в індивідуальному підході, що стимулює їх індивідуальну, інтелектуальну діяльність. Слід також зазначити, що в індивідуальній складовій цей метод можна пов'язати зі студентсько-орієнтованим підходом, сформованим на основі гуманістичної психології А. Маслоу, К. Роджерса-методом гуманістичної психології. У сучасних побутових творах (І. С. Якиманська, Є. В. Бондаревська, В. В. Серіков) представлена цілісна концепція індивідуально-орієнтованого навчання.

Індивідуальна модель передбачає таку організацію навчання яка буде, зосереджуватись переважно на особистості учня, оригінальності, самобутності, унікальності, суб’єктивності учня.

У педагогічній роботі вихователь (учитель) організовує свою діяльність з урахуванням особливостей кожного вихованця так як всі учні різні, а значить і якістні характеристики в опануванні дисципліни у них теж різні.

Методологічні підходи педагогіки як галузі гуманітарного знання дозволяють:

1) визначити її дійсні проблеми і способи їх вирішення;

2) проаналізувати всю суму освітніх проблем і встановити їх порядок значущості (ієрархію);

3) реалізувати гуманістичну парадигму освіти. Сприйняття як безпосереднє відображення світу класифікується за різними підставами.

Як правило виділяють п'ять видів сприйняття інформації (по модальності сприйняття): візуальне, аудиальное, дотикове (тактильну), смакове, нюхове. Розрізняють також види сприйняття в залежності від об'єкта сприйняття, наприклад, сприйняття часу, простору, швидкості, руху; музики, творів живопису; основних явищ соціального життя особи, подій суспільного життя, тощо.

2.2. Аналіз інструментів і технологій для реалізації web-сервісу

2.3.1. HTML – мова розмітки вебсторінок для мережі інтернет

HTML (англ. HyperText Markup Language) – це мова тегів, за допомогою яких здійснюється формування структури вебсторінок в мережі Інтернет. Браузери отримують документи в форматі HTML з вебсервера або з локальної пам'яті й завантяжують документи в мультимедійні вебсторінки. HTML описує структуру вебсторінки семантично, а також підказки в яких системах кодування потрібно відобразити документ.

Елементи в HTML являють собою будівельні блоки сторінок HTML. За допомогою конструкцій HTML, мультемидійні дані та інші об'єкти, наприклад інтерактивні форми, можуть бути вбудовані у сторінку. HTML надає засоби для створення структурованих документів, визначаючи структурну семантику тексту, наприклад списки, абзаци, цитати, посилання, заголовки та інші наявні елементи. Елементи HTML визначаються тегами, написаними з використанням кутових дужок та слешом. Теги на зразок <img /> або <input /> безпосередньо виводять контент на сторінку. Інші теги, такі як , оточують контент і надають інформацію для його опису, а також можуть включати інші, вкладені теги як піделементи, які вважаються наслідниками і по замовчуванні наслідують правила описів контенту. Браузери не відображують теги HTML на сторінці, але використовують їх для інтерпретації контенту сторінки.

В HTML є можливість вбудовувати програми, які написані на скриптових мовах. Наприклад JavaScript, що впливає на поведінку та вміст вебсторінки. Включення в структуру стилів CSS визначає зовнішній вигляд і компонування контенту на сторінці. World Wide Web Consortium (W3C), який визначає стандарти HTML та CSS.

HTML надає засоби для:

- створення структурованих документів шляхом визначення структурного складу тексту: списки, абзаци, заголовки, цитати, таблиці та інші;
- отримання інформації зі мережі Інтернет за допомогою гіперпосилань;
- створення інтерактивних форм;
- включення звуку, зображень, відео, та інших об'єктів.

2.2.2. Sass – скриптова метамова, яка інтерпретується в каскадні таблиці стилів

Рис. 2.1. Лого технології Sass

Sass (англ. Syntactically Awesome Stylesheets) – скриптова метамова, яка інтерпретується в каскадні таблиці стилів (CSS). Sass використовується для спрощення файлів CSS та підвищення рівня абстракції коду.

Метамова Sass має два синтаксиси:

- sass (оригінальний) – відрізняється тим, що не має фігурних дужок, в ньому елементи, які вкладені, реалізовані за допомогою відступів, а правила відокремлюються одне від одного переведенням на новий рядок;
- scss (новий) – використовує фігурні дужки подібно як в CSS.

Файли що мають sass-синтаксис використовують розширення .sass, а scss-синтаксис – .scss.

Sass розширює каскадні стилі, за допомогою надавання деяких механізмів, доступних в традиційних об'єктно-орієнтованих мовах програмування, але недоступних для CSS. Sass-інтерпретатор кампілює SassScript у блоки правил CSS. По своїй суті, Sass – це синтаксична надбудова для каскадних стилів.

Sass дозволяє визначати та ініціалізувати змінні. Змінні в Sass починаються зі знака долара $. Ініціалізація значень змінних здійснюється за допомогою двокрапки. SassScript підтримує чотири типи даних:

- числовий
- булевий (логічний) тип
- рядок (з лапками чи без)
- колір (назва)

Змінна може бути в якості аргументу чи результу однієї чи декількох функцій. Під час інтерпретації значення змінних вставляються у результуючий документ CSS.

Синтаксис для SCSS:

```scss
$blue: #3bbfce;
$margin: 16px;

.content-navigation {
  border-color: $blue;
  color:
    darken($blue, 20%);
}

.border {
  padding: $margin / 2;
  margin: $margin / 2;
  border-color: $blue;
}
```

Інтерпретується в:

```scss
.content-navigation {
  border-color: #3bbfce;
  color: #2b9eab;
}

.border {
  padding: 8px;
  margin: 8px;
  border-color: #3bbfce;
}
```

2.3.3. JavaScript – динамічна, об'єктно-орієнтована прототипна мова програмування.

Рис. 2.2. Лого мови програмування JavaScript

JavaScript (JS) – прототипна, динамічна, об'єктно-орієнтована мова програмування. Являється реалізацією стандарту ECMAScript. Як правило використовується в створенні вебсторінок, що надає можливість на боці пристроя кінцевого користувача взаємодіяти з користувачем, здійснювати керування браузером, асинхронно надсилати та приймати дані з серверу, динамічно змінювати структуру та зовнішній вигляд вебсторінки без перевантаження.

JavaScript класифікують як прототипну скриптову мову програмування з динамічною типізацією. Окрім прототипної, JavaScript також частково або повністю підтримує інші парадигми програмування (імперативну та частково функціональну) і деякі відповідні архітектурні властивості, зокрема: динамічна та слабка типізація яку можна віднести як до плюсів так і до мінусів, автоматичне керування пам'яттю та збір сміття, , функції як об'єкти першого класу, концепція прототипного наслідування.

Мова JavaScript застосовується для створення:

- системи поводеження вебсторінок для надання їм інтерактивності та динаміки;
- односторінкових (SPA – single page application) та прогресивних вебзастосунків (PVA – progressive web application)
- серверної частини застосунку (Node.js)
- стаціонарних застосунків (Electron)
- мобільних застосунків (Cordova, React Native)

Не двивлячись на те, що Java та JavaScript мають схожі назви, вони є двома різними мовами, що мають різну семантику, хоча й мають загальні риси в стандартних бібліотеках та правилах іменування сутностей.

JavaScript, на 2020 рік є однією з найпопулярніших мов програмування в мережі інтернет. На початкових роках існування цієї мови, більшість професіоналів в програмуванні скептично ставилися до JavaScript, цільова аудиторія якої складалася в більшості з програмістів-аматорів. Поява технології AJAX суттєво змінила ситуацію та звернула увагу професійної спільноти до цієї мови, а її подальші модифікації за стандартами ES6+ імплементували багато корисних можливостей, яких небуло достатньо для ефективного вирішення тієї чи іншої проблеми при створенні програмних рішень. В результаті цього, були розроблені та модернізовані багато практик використання JavaScript (в тому числі, налагодження та тестування), були також створені бібліотеки та фреймворки цієї мови, значно поширилося використання JavaScript поза браузером.

Мова JavaScript має ряд властивостей об'єктно-орієнтованих мов, але завдяки концепції прототипого наслідування підтримка об'єктів в ній відрізняється від традиційних мов ООП. Окрім того, JavaScript має кілька властивостей, які притаманні функціональним мовам – анонімні функції, замикання (closures), об'єкти як списки, каррінг, функції як об'єкти першого класу – що додає мові додаткову гнучкість в програмуванні.

JavaScript має синтаксис подібній до мови C, але в порівнянні з мовою C має такі принципові відмінності:

- автоматичне приведення типів;
- анонімні та стрілочні функції;
- автоматичне збирання сміття;
- об'єкти, з можливістю інтроспекції і динамічної зміни типу через механізм прототипів;
- обробка винятків;
- функції як об'єкти першого класу;

JavaScript містить декілька десятків вбудованих об'єктів, які поділяються на групи: фундаментальні (Object, Function, Boolean, Symbol), помилки (група об'єктів Error), текстові (String, RegExp), числа та дати (Number, BigInt, Math Date), індексовані (група об'єктів Array), ключові (Map, Set, WeakMap, WeakSet), абстрактні (Promise, Generator), для роботи з структурованими даними (ArrayBuffer, Atomics, DataView, JSON), рефлекційні (Reflect, Proxy), групи Intl та WebAssembly. Крім того, мова JavaScript містить набір вбудованих операцій, що здійснюють керування логікою виконання програм. Синтаксис JavaScript в основному схожий з мовою Java (в результаті успадкування від C), але, дещо, спрощений у порівнянні з ним, щоб зробити мову сценаріїв легкою для освоєння починаючим користувачам. Наприклад, оголошення змінної не буде містити її типу, властивості об’єктів також не мають типів, а оголошення функції може знаходитися в тілі програми після її виклику.

Приклад оголошення і використання класу в JavaScript:

```
class MyClass {
  constructor() {
    this.myValue1 = 1;
    this.myValue2 = 2;
  }
}

const mc = new MyClass();
mc.myValue1 = mc.myValue2 * 2;
```

Технологія, яка стала популярною, та що значно додала сторінкам динаміки і забезпечила нові можливості (асихнронні запити до серверу) — це динамічне завантаження і вставка даних в документ, що має назву AJAX.

При використанні в рамках технології DHTML JavaScript код імпортується в HTML-код сторінки і виконується вбудованим браузерним інтерпретатором. Код JavaScript вставляється в теги <script></script> , хоча в більшості сучасних браузерів мова сценаріїв за умовчанням налаштована як JavaScript. Скрипт, що виводить модальне вікно в браузері з написом «Hello, World!»:

```
<script>
  alert("Hello, World!");
</script>
```

Мова JavaScript є інтерпретованою мовою програмування, з слабкою типізацією, і може виконуватися в різних середовищах, які можуть бути зі своїми власними особливостями сумісності, програміст повинен бути пильним, і має переконуватись, що його код виконується як очікується в широкому переліку можливих конфігураційних налаштувань. Типізація вважається одною з базових проблем JavaScript, тому восени 2012 року, компанія Microsoft презентувала міжнародній спільноті мову програмування під назвою TypeScript, що в кінцевому випадку компілюється в JavaScript та містить декілька важливих для користувачів JavaScript доповнень, що полегшують роботу на етапі розробки.

При розробці великих і нетипових вебзастосунків з використанням мови JavaScript, важливим є доступ до інструментів відлагодження коду. Так як браузери, від різних виробників, в деяких моментах відрізняються у поведінці JavaScript і реалізації об'єктної моделі документа, то необхідно мати відлагоджувач для кожного браузера окремо, якщо вебзастосунок орієнтовано на нього.

На 2020 рік Google Chrome, Firefox, Edge, Opera та Safari мають свої власні відлагоджувачі.

Також в наявності є такі корисні інструменти, як:

- Babel – компілятор JavaScript-коду для старіших версій стандарту ECMAScript, який допомагає розробникам застосовувати найновіші можливості мови для оточення, яке не встигло реалізувати останній стандарт.
- ESLint – перевіряє якість коду, скануючи JavaScript-програму, шукаючи розбіжності з правилами написання і підсвічуючи їх для розробника;
- Prettier – автоматичне форматування коду у коректний стандарт вигляду, для кращої читанності коду розробниками;

Кожен блок коду інтерпретатор розбирає окремо. На вебсторінках, коли треба комбінувати блоки JavaScript та HTML, синтаксичні помилки знайти простіше, якщо зберігати функції в окремому блоці коду, або, ще краще використовувати багато невеликих, пов'язаних між собою файлів з розширенням .js. Таким чином, синтаксична помилка не буде причиною «падіння» всієї вебсторінки та дозволить оповістити користувача про проблему яка виникла в системі.

2.3.4. Angular – JavaScript фреймворк для реалізації клієнтської частини

Рис. 2.3. Лого фреймворка Angular

Angular (як правило так іменують фреймворк Angular 2-гої версіх та вище) – написаний на мові TypeScript являє собою front-end фреймворк з відкритим кодом, який підтримується під керівництвом Angular Team у компанії Google, а також корпораціями та спільнотами приватних розробників. Angular – це AngularJS, який перероблений тією ж командою розробників за для покращення якісних показників в багатьох аспектах.

На початку переписаний AngularJS отримав назву Angular 2 від команди розробників, яка над ним працювала, але це викликало плутанину серед розробників, які користувалися цим інструментом. Аби показати різницю між ними та наголосити, що це окремі проекти, команда вирішила для фреймворків версій 1.X використати назву AngularJS, а для версій, починаючи з 2.0, – Angular без постфіксу JS.

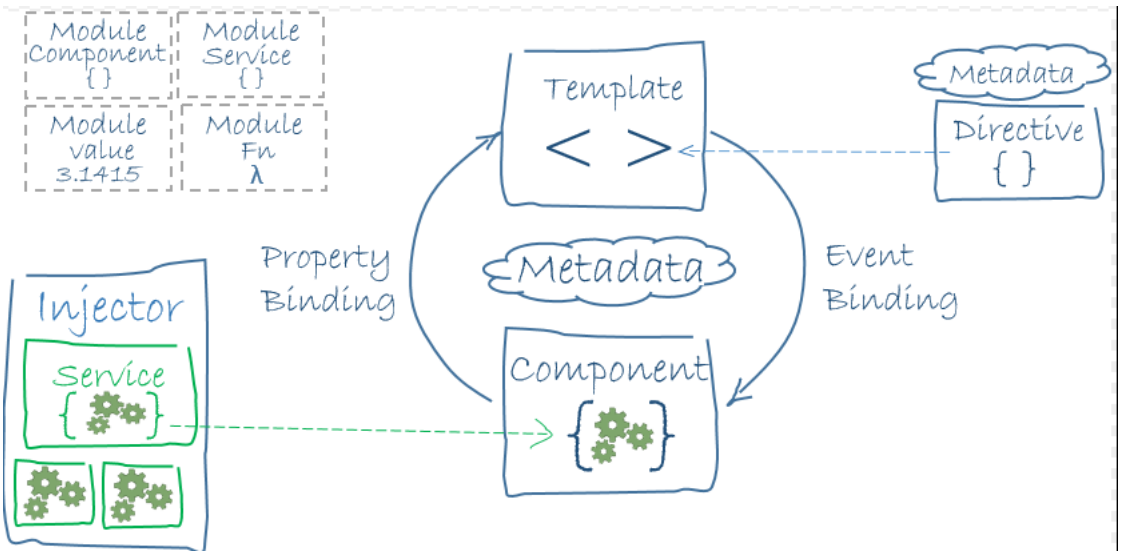

Рис. 2.4. Архітектура додатка на Angular

Основними елементами в розробці є компоненти, модулі, шаблони, метадані, байндінг даних, сервіси, директиви та ін'єкційні залежності.

З кожним роком Angular зростає у популярності серед інших інструментів в сегменті front-end. Станом на вересень 2020, кожного дня завантажується близько 1,5 мільйонів Angular/core, і цей показник подвоюється кожний наступний рік.

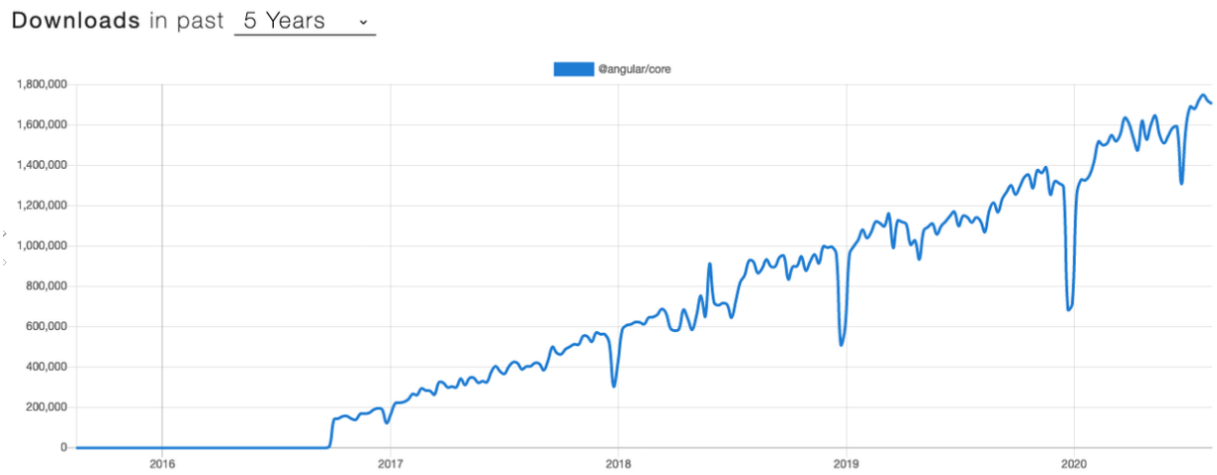

Рис. 2.5. Стистика завантаження Angular за останні 5 років

Технологія Angular використовується у веб-додатках таких компаній:

- Google (Gmail, Google Play, Google Translate, Google Ads, Youtube);
- Upwork;
- PayPay[en];
- Expedia[en]
- Adidas;
- Lego.

В рейтингу популярності фреймворків Angular займає 2 позицію після React.

2.3.5. MongoDB – документо-орієнтована система керування базами даних

Рис. 2.6. Лого системи керування базами данних MongoDB

MongoDB – система керування базами даних (СКБД), що є документо-орієнтована з відкритим вихідним кодом, та яка не потребує опису схеми таблиць. MongoDB займає нішу між масштабованими і швидкими системами, що оперують даними у форматі ключ-значення, і реляційними СКБД, зручними і функціональними у формуванні запитів. Код MongoDB написаний на мові C++ і розповсюджується в рамках AGPLv3 ліцензії.

Система керування базами даних MongoDB підтримує збереження документів в JSON форматі, має достатньо гнучку мову для формування запитів, має можливість створювати індекси для різних збережених атрибутів, надійно забезпечує зберігання великих бінарних об'єктів, підтримує журналювання операцій зі додавання та зміни даних в БД, може працювати

відповідно до парадигми Map/Reduce, підтримує побудову і реплікацію відмовостійких конфігурацій.

У MongoDB є вбудовані засоби із розподіл набору даних по серверах на основі певного ключа (забезпечення шардінгу), кобмінування якого з реплікацією даних можна спроектувати горизонтально масштабований кластер зберігання даних, в якому збій будь-якого вузла не позначається на роботі БД (єдина точка відмови), підтримується автоматичне відновлення після збою і перенесення навантажень з вузлів, що вийшли з ладу. Розширення кластера чи перетворення одного сервера на кластер здійснюється без зупинки роботи БД звичайним додаванням нових машин.

СКБД керує наборами JSON-подібних документів, які зберігаються в бінарному форматі BSON. Пошук і зберігання файлів в MongoDB відбувається завдяки виклику протоколу GridFS. MongoDB не є реляційною СКБД. Існує детальна та якісна документація, велике число прикладів і драйверів для популярних мов C#, Ruby, Java, Python, C++, PHP, Perl.

Під час випуску одразу було заявлено, що реліз MongoDB 1.0 готовий до використання у виробництві як зв'язки master/slave так і одиничний хост. Код цього релізу достатньо стабільний і перевірений в промисловій експлуатації. Існують рекомендації розгортати MongoDB мінімум на двох серверах використовуючи реплікацію Master/Slave. Це забезпечить наявність актуальних даних при виході з роботи однієї з СКБД. MongoDB — СКБД досить молода, і відтак у ній зустрічаються помилки але і з'являються нові можливості. Проекту MongoDB притаманний високий темп розробки адже проект пишуть не тільки волонтери, а й компанія професіоналів на повній зайнятості. Компанія-розробник надає платні хостинг, підтримку та консультації. Запити до MongoDB можуть витягати дані з вкладених масивів та об'єктів. Наприклад якщо в колекцію users вставлений такий об'єкт:

```
{
  "username" : "bob",
  "address" : {
    "street" : "123 Main Street",
    "city" : "Springfield",
    "state" : "NY"
  }
}
```

Можна запросити цей документ за допомогою:

```
> db.users.find({"address.state" : "NY"})
```

Можна також запитати один або безліч елементів масиву:

```
> db.food.insert({"fruit" : ["peach", "plum", "pear"]})
> db.food.find({"fruit" : "pear"})
```

2.3.6. Node.js – платформа для виконання мережевих застосунків, написаних мовою JavaScript.

Рис. 2.7. Лого платформи NodeJs

Node.js – платформа з відкритим кодом для виконання високонавантажених мережевих застосунків, написаних JavaScript. Засновником Node.js є Раян Дал. Раніше Javascript застосовувався для обробки даних в браузері, але після першого релізу node.js з'явилась можливість виконувати JavaScript-скрипти на сервері та надсилати користувачеві

результат їх виконання.Node.js перетворила JavaScript на інструмент загального використання з великою спільнотою розробників різних рівнів компетентності.

Node.js має такі властивості:

- асинхронна одно-поточна модель виконання запитів;
- неблокуючий ввід та вивід даних;
- модульна система CommonJS;
- движок JavaScript Google V8;

Для керування модулями використовується менеджер пакетів npm (node package manager).

Платформа Node.js застосовується для виконання високопродуктивних мережевих застосунків, спроектована мовою програмування JavaScript. Окрім роботи із серверними скриптами для веб-запитів, платформа також використовується для створення клієнтських програм. Платформа використовує розроблений компанією Google движок V8.

Обробка великої кількості паралельних запитів у Node.js забезпечується завдяки використанню асинхронної моделі запуску коду, яка заснована на обробці подій в неблокуючому режимі та визначенні обробників зворотніх викликів (callback). Мультиплексування з'єднань підтримується такими способами: epoll, kqueue, /dev/poll і select. Для мультиплексування з'єднань використовується бібліотека libuv, для створення пулу потоку (thread pool) задіюється бібліотека libeio, для виконання DNS-запитів у неблокуючому режимі інтегрований c-ares. Системні виклики, що спричиняють блокування, виконуються всередині пулу потоків і потім, як і обробники сигналів, надсилають результат виконання своєї роботи назад через неіменовані канали (pipe).

Платформа Node.js за своєю суттю схожа на фреймворки Perl AnyEvent, Ruby Event Machine і Python Twisted, але цикл обробки подій (event loop) у

Node.js інкапсульований (прихований) від розробника і схожий на обробку подій у веб застосунку, що працює у браузерному середовищі. При створенні програм для Node.js необхідно враховувати специфіку подієво-орієнтованого програмування, наприклад, замість виконання

```
var result = db.query ("select .. ");
```

з очікуванням завершення роботи і наступною обробкою результатів виконання коду, в Node.js використовує принцип асинхронного виконання, тобто код трансформується в

```
db.query ("select .. ", function (result) { /* обробка результату */ });
```

Також управління миттєво перейде до коду який слідує після виклику функції db.query, а результат запиту буде оброблений як тільки будуть оброблені всі дані в стеку. Жодна функція в Node.js не повинна безпосередньо виконувати синхронні, блокуючі операції вводу-виводу — для отримання даних з диска, від інших процесів або з мережі потрібна установка callback-обробника.

В Node.js підготовлена велика колекція модулів для розширення функціональності застосунків, які працюють на цій платформі. В цій колекції можна знайти реалізацію XMPP, HTTP, SMTP, DNS, POP3, IMAP, FTP клієнтів і серверів, модулі для інтеграції з різними вебфреймворками, обробники WebSocket і AJAX технологій, конектори до СКБД (MongoDB, PostgreSQL, SQLite, MySQL), XML-парсери, шаблонізатори, CSS-движки, реалізації систем авторизації і криптоалгоритмів.

Приклад програми, що починає роботу вебсервера та виводить в консоль повідомлення, і на кожен HTTP запит відповідає «Hello World» повідомленням:

```
var http = require('http'); // Завантажуємо модуль http

// Створюємо web-сервер і вказуємо функцію обробки запиту
var server = http.createServer(function (req, res) {
    console.log('Початок обробки запиту');
    // Передаємо код відповіді і заголовки
    res.writeHead(200, {
        'Content-Type': 'text/plain; charset=UTF-8'
    });
    res.end('Hello world!');
});

// Запускаємо web-сервер
server.listen(1991, "127.0.0.1", function () {
    console.log('Сервер запущено за адресою http://127.0.0.1:1991/');
});
```

FileSystem є одним з широко використовуваних стандартних модулів, який є вбудованим та містить операції роботи з файловою системою.

```
const fs = require("fs");
fs.readFile("test.txt", function(err, data) {
  if (err) throw err;
  console.log(data);
 });
```

Цей код завантажить текст з файла test.txt, що знаходиться в тій же директорії де й сам скрипт js. Однак результат буде повернутий в не читабельному для користувача, бінарному вигляді. Для цього потрібно виконати перетворення об'єкта data з бінарного на текстовий тип:

```
const fs = require("fs");
fs.readFile("test.txt",function(err,data) {
  if (err) throw err;
  data = String(data); // перетворення типів
  console.log(data);
 });
```

2.3.7. Express.js – Node.js фреймворк для реалізації серверної частини

Рис. 2.8. Лого фреймворка ExpressJS

Express.js, або просто Express – фреймворк для розробки серверної частини веб-застосунків для платформи Node.js, позиціонуючий себе як відкрите програмне забезпечення на ліцензійній основі MIT. Express спроєктований для створення API і веб-застосунків. Фактично є стандартним каркасом для платформи Node.js. Автором фреймворка вважається TJ Holowaychuk, він описує його як створений на основі написаного на мові Ruby каркаса Sinatra, маючи на увазі, що він мінімалістичний, але при цьому має багато плагінів.

Express застосовується в якості бекенд інструменту для програмного стека MEAN, разом з каркасом AngularJS для фронтенду і базою даних MongoDB. Права на управління проєктом були придбані компанією StrongLoop у червні 2014. Згодом, IBM у вересні 2015 придбала StrongLoop. У січні 2016 IBM заявила про намір помістити Express.js під управління інкубатора Node.js Foundation.

Приклад імпортування express в node-застосунок:

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

Ініціалізація застосунку:

```
const app = express();
```

Реєстрація функції зворотного виклику для GET-запиту і надання текстової відповіді:

```
app.get('/', (req,res)=>{
    res.send('Привіт, ми отримали ваш запит')
})
```

Зараз програму можна відкрити на http://localhost:2000.

2.3.8. Visual Studio Code – середовище для створення, редагування та завантаження програм

Рис. 2.9. Лого середовища VS Code

Visual Studio Code – середовище створення, редагування та налагодження програм і вебзастосунків. Visual Studio Code безкоштовний продукт і є доступним у версіях для платформ iOS, Windows і Linux.

Корпорація Microsoft представила Visual Studio Code у восени 2015 на конференції Build 2015. Visual Studio Code став першим кросплатформовим рішенням у лінійці Visual Studio.

За базу для Visual Studio Code взяли напрацювання вільного проєкту Atom, що розробляється компанією GitHub. VS Code є надбудовою над Atom Shell, що використовує браузерний движок Chromium і платформу Node.js.Про те, про використання напрацювань вільного проєкту Atom на сайті Visual Studio Code і в пресрелізі, і на офіційному блозі не зазначається.

Редактор містить навігацію по коду, вбудований налагоджувач, автодоповнення типових конструкцій і контекстної підказки, інструменти для роботи з Git і засоби рефакторингу. Продукт підтримує розробку для платформ Node.js і ASP.NET, і позиціює себе як легковагове рішення, що дозволяє використовувати інструменти без повного інтегрованого середовища розробки. Серед підтримуваних мов і технологій: C#, JavaScript, C++, F#, TypeScript, jade, PHP, Python, XML, Batch, DockerFile, Coffee Script, Java, HandleBars, R, Objective-C, PowerShell, Luna, Markdown, HTML, CSS, Visual Basic, LESS і SASS, Haxe, JSON.

2.3.9. Капмілятор коду в режимі реального часу Sandbox

Інтерпретація – це послідовний аналіз, обробка та виконання вихідного коду програми або запиту (на відміну від компіляції, де весь текст програми перед її початком аналізується та перекладається на машинний або байт-код без його виконання). Завдяки віртуалізації можна побачити результати введеного вами коду одним натисканням кнопки. Також віртуалізація також використовується в дослідницьких цілях: наприклад, є необхідність контролювати вплив новоствореної програми на систему або запускати дві різні версії програми одночасно. Або написати автономну програму, яка не залишитиме слідів у системі. Існує багато варіантів використання пісочниць. При цьому програма не диктує своїх умов у системі, а саме система керує та розподіляє ресурси.

Пісочниця – середовище, спеціально створене для безпечного виконання програмного забезпечення. Зазвичай це жорстко контрольований набір ресурсів для запуска клієнтської програми – наприклад, пам’яті на диску. Доступ до мережі, можливість комунікування з основною операційною системою або зчитування інформації з пристроїв введення часто або частково емулюються або сильно обмежуються. Пісочниця – це один з прикладів

віртуалізації. Простий інтерпретатор негайно аналізує та виконує програму покомандно (або через підрядника), коли її вихідний код надходить на вхід інтерпретатора. Перевагою такого підходу є негайний зворотний зв'язок. Недолік такого методу в тому що інтерпретатор виявляє помилки в коді програми, лише коли намагається виконати команду (або рядок) в якому є помилка.

Інтерпретатор компілюючого типу – це компіляторна система з, що перетворює вихідний код програми у проміжну форму, таку як байт-код або p-код, і самого інтерпретатора, який виконує отриманий проміжний код. Перевагою таких систем є більша швидкість виконання програми (за рахунок виносу аналізу вихідного коду в окремий, разовий прохід, і мінімізації цього аналізу в інтерпретаторі). До недоліків можна віднести більшу потребу в ресурсах та вимоги до коректності вихідного коду. Використовується такими мовами, як Java, PHP, Tcl, REXX, Perl (зберігаються результати парсинга вихідного коду), а також у різних СКБД. Крім того, вихідний код такого компілятора не обов'язково повинен бути у текстовому форматі або байт-коді, який розуміє тільки інтерпретатор, це може бути машинний код якоїсь існуючої апаратної платформи. Наприклад, віртуальні машини, такі як VMware, QEMU, Bochs, включають в собі інтерпретатори машинного коду, які належать до сімейства процесорів x86.

Деякі інтерпретатори (наприклад, для Python, BASIC, Lisp, Scheme, та інші) можуть працювати в діалоговому режимі або так званих циклах eval-print (читання-обчислення-друк). Таким чином, інтерпретатор читає готову мовну конструкцію (наприклад, s-expression в мові Lisp), виконує її, друкує результат, а потім переходить в режим очікування, поки користувач введе іншу конструкцію.

2.3.10. Технологія Drag-and-drop

Термін Drag-and-drop в перекладі з англійської буквально означає тягни-ікидай. Це спосіб оперування елементами інтерфейсу в інтерфейсі користувача (як текстового, так і графічного, де елементи інтерфейсу реалізовані за допомогою псевдографіки) за допомогою сенсорного екрану або миші.

Drag-and-drop здійснюється шляхом захопленням об’єкта лівою клавішою миші на екрані та та утримання і перетягування в необхідну область екрана де іздйснюється відпуск клавіші миші і видимими змінами інтерфейсу користувача. Що стосується вікон (які також мають можливість рухатися подібним чином), для них цей термін як правило не використовується.

Найпростішими прикладами drag-and-drop функціоналу є: переміщення об'єкта по області вікна, переміщення об'єкта з однієї панелі на іншу. В сучасних операційних системах drag-and-drop широко використовується і є одним з головних способів взаємодії користувача з комп'ютером в графічному інтерфейсі (рис.2.3).

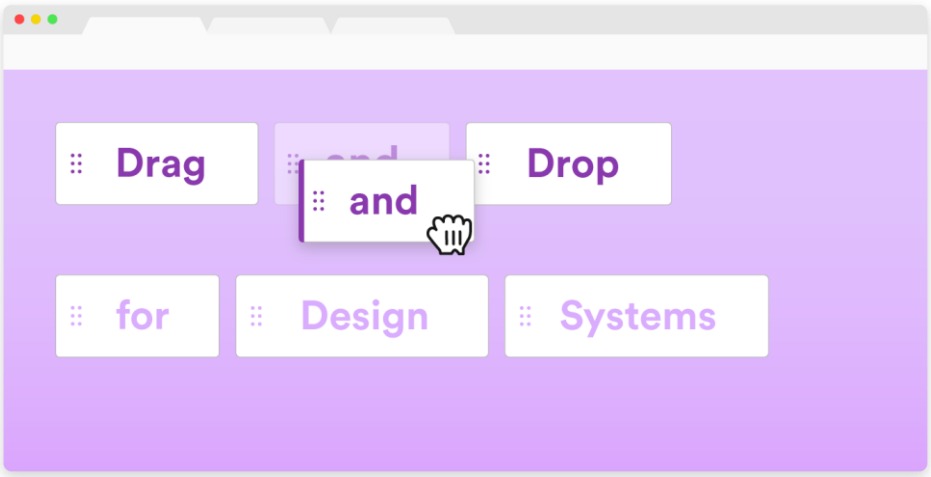

Рис. 2.10. Приклад роботи Drag-and-drop технології

Об'єктами для переміщення можуть бути такі елементи інтерфейсу: піктограми робочого столу, плаваючі панелі інструментів, ярлики на панелі завдань (починаючи з Windows XP), комірка DataGridView, текстові рядки та елементи TreeView та OLE. Об’єкти можуть переміщатися як в межах певної області так і в межах одного вікна, між різними вікнами та між різними панелями.

Функціонал виконання технології перетягування включають виконання таких кроків: Визначити елемент на якому буде прослуховуватись подія та присвоїти значення true атрибуту draggable елемента, який ми хочемо перенести (для детальної інформації дивитися The Draggable Attribute в документації). Визначає дані, які можна переміщати, адже вони можуть бути в різних форматах. Наприклад, текстові дані, що містять рядок тексту, можна переміщати, а інші ні, тощо (для детальної інформації дивитися Drag Data розділ в документації). Визначити зображення яке буде поруч з курсором миші на протязі всієї операції перетягування. Якщо для користувача не вказано зображення, буде згенерована картинка за замовчуванням, залежно від елемента, на якому натиснута кнопка миші (що означатє, що елемент збираються переносити).

2.4. Висновок за розділом 2.

Аналіз основних методологічних підходів показав, що найкращим рішенням яке можна імплементувати в систему буде комбінований підхід, який буде включати в себе особистісний, діяльнісний, індивідуальний та системний підходи, які в совокупності збільшать рівень навчальної мотивації та якісні показники засвоєння навальних матеріалів.

Також були обрані інструменти для реалізації задуманного проекту звертаючи увагу на декілька факторів, а смаме:

- якість зань з того чи іншого інструменту, що буде застосований при реалізації проектних рішень;
- популярність на ринку та спроможність знайти вже готові рішення на цій технології;
- швидкість реалізації проектних рішень за допомогою даного інструменту;
- перспективність даної технології, для подальшого розвитку проектну і імплементації найновіших функціональних рішень;

РОЗДІЛ 3

РЕАЛІЗАЦІЯ ПРОЕКТНИХ РІШЕНЬ.

3.1. Загальний опис розробленого продукту

Метод навчання програмування по свій суті являється односторінковим застосунком (SPA – single page application) в якому і застосовані основні методологічні підходи в навчанні та обрані наступні технології: Angular – для клієнтської частини, NestJs – для серверної та MongoDb – як база данних.

Веб-сервіс скаладається с двух модулів:

- модуль користувача;
- модуль адміністратора;

Модуль користувача реалізує, власне, сам метод навчання програмуванню. Для комфортнішої роботи з сервісом користувач може створити власний обліковий запис. Обліковий запис дає можливість коментувати, а також взаємодіяти з іншими користувачами системи.

Користувач має доступ до наступного функционалу:

- створювати обліковий запис;
- обирати мову програмування яку бажє вивчати;
- здійснювати навігацію по попередньо завантаженим навігаційним даним;
- коментувати кожний з трьох блоків (теорія, тести, практика);
- виконувати тести, що присутні в кожному розділі;
- виконувати завдання по кожному розділу;
- коментувати та оцінювати відповіді інших користувачів;

Даний модуль складається з таких основних блоків:

- Top-nav-bar – має лейбл веб-додатку, фуркціонал з вибору мови програмування та логіну чи створення облікового запису.

Рис. 3.1. Top-nav-bar

- Right-sidenav – має розділи по конктерній мові програмування.

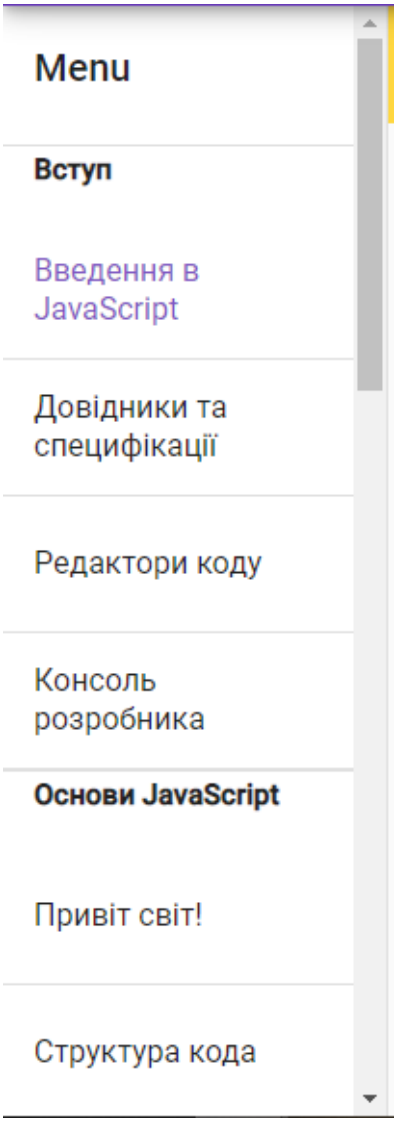

Рис. 3.2. Right-sidenav

- Toolbar – має три секції (теорія, тести, завдання) кожна з яких завантажує певний контент з конкретного розділу.

Рис. 3.3. Toolbar

- Content-container – має конент який завантажується відповідно вибраній секції та розділу.

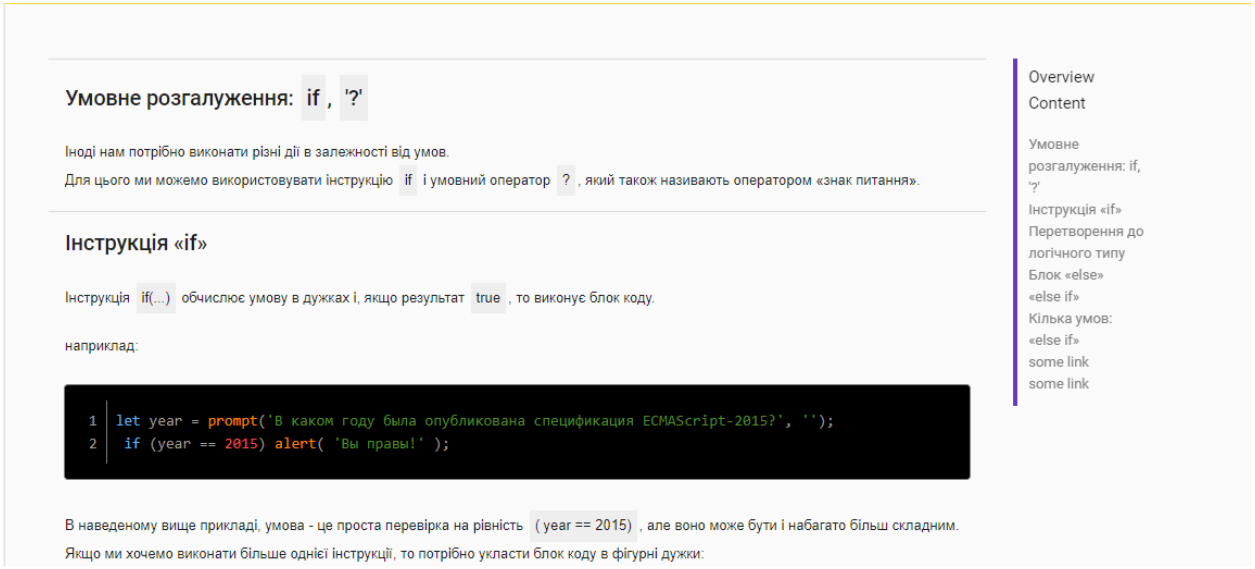

Рис. 3.4. Content-container (секція «Теорія»)

- Left-sidenav – має перелік всіх заголовків для зручнішої навігації по контенту.

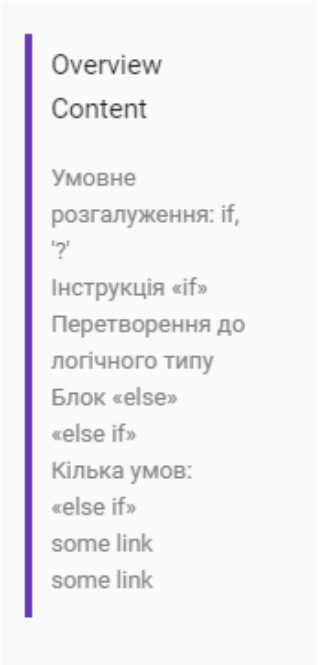

Рис. 3.5. Left-sidenav

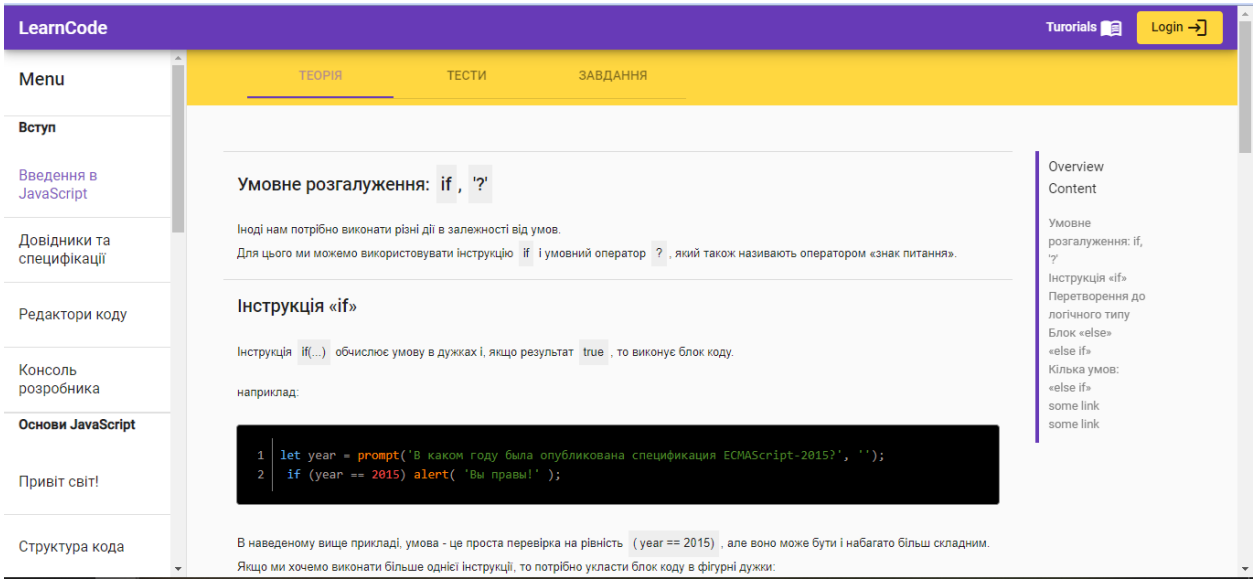

Рис. 3.5. Загальний вигляд веб-додатку.

Окремо треба розглянути кожну з трьох секцій. Всього їх три :

- теорія;
- тести;
- завдання;

Секція «Теорія» має на меті ознайомлення учня з теоретичною частиною матеріалу. Секція складається з контенту, коментарів та навігації. Контент поділяється на блоки, кожний з який починається заголовком. Блок може складатися з одного з двух типів контенту:

- Текстовий тип (текст, картинки, анімації, відео та їх стилізація) – використовує пакет Quill, який дає змогу створювати, форматувати та зберігати DOM в string типі, який потім зручно відображати на сторінці.

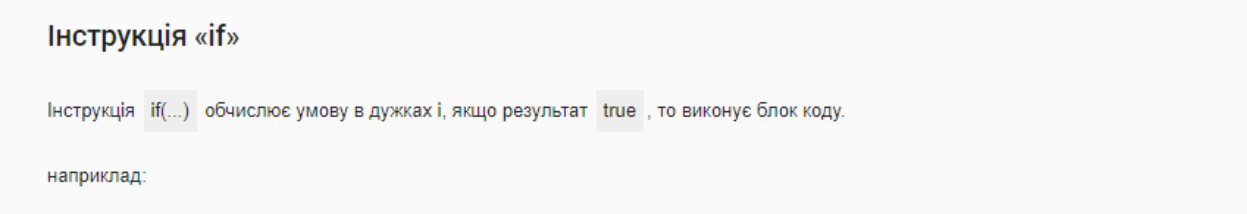

Рис. 3.6. Приклад текстового типу контента

- Кодовий тип – використовує пакет Angular Highlight.js, який надає можливість відображати код так як би це виглядало в сучасних середовищах розробки. Обливість такого типу полягає в тому, що контент можна копіювати, тобто він не являється картинкою.

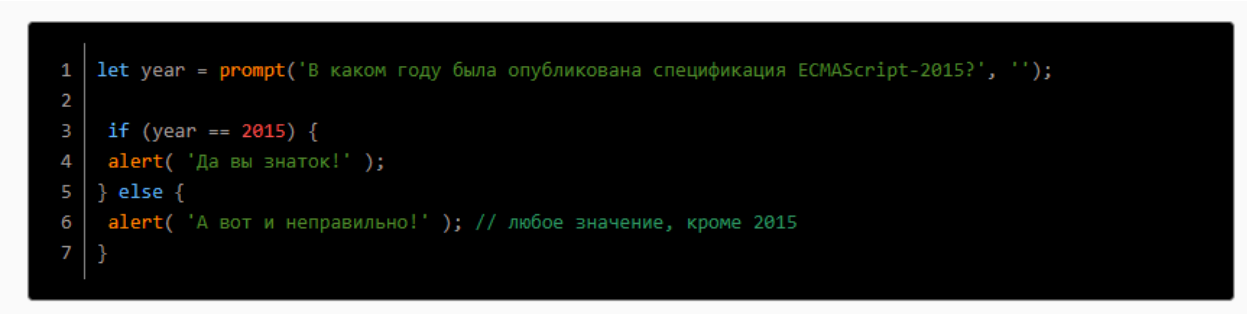

Рис. 3.6. Приклад кодового типу контента

Секція «Тести» має на меті показати учню, на якому рівні засвоєний теоретичний матеріал. Секція складається з тестів, статистики та коментарів. Тести розміщенні послідовно в mat-stepper елементі. Тест скаладається з блоку завдання та блоку варівантів відповідей. Блок завдання складається з

текстового (Quill - пакет) та кодового (ngx-highlightJs – пакет) типів. Блок варіантів відповідей скаладається з mat-radio-group елементу. Результат тесту можна побачити відразу після натискання на кнопку «перевірити відповідь».

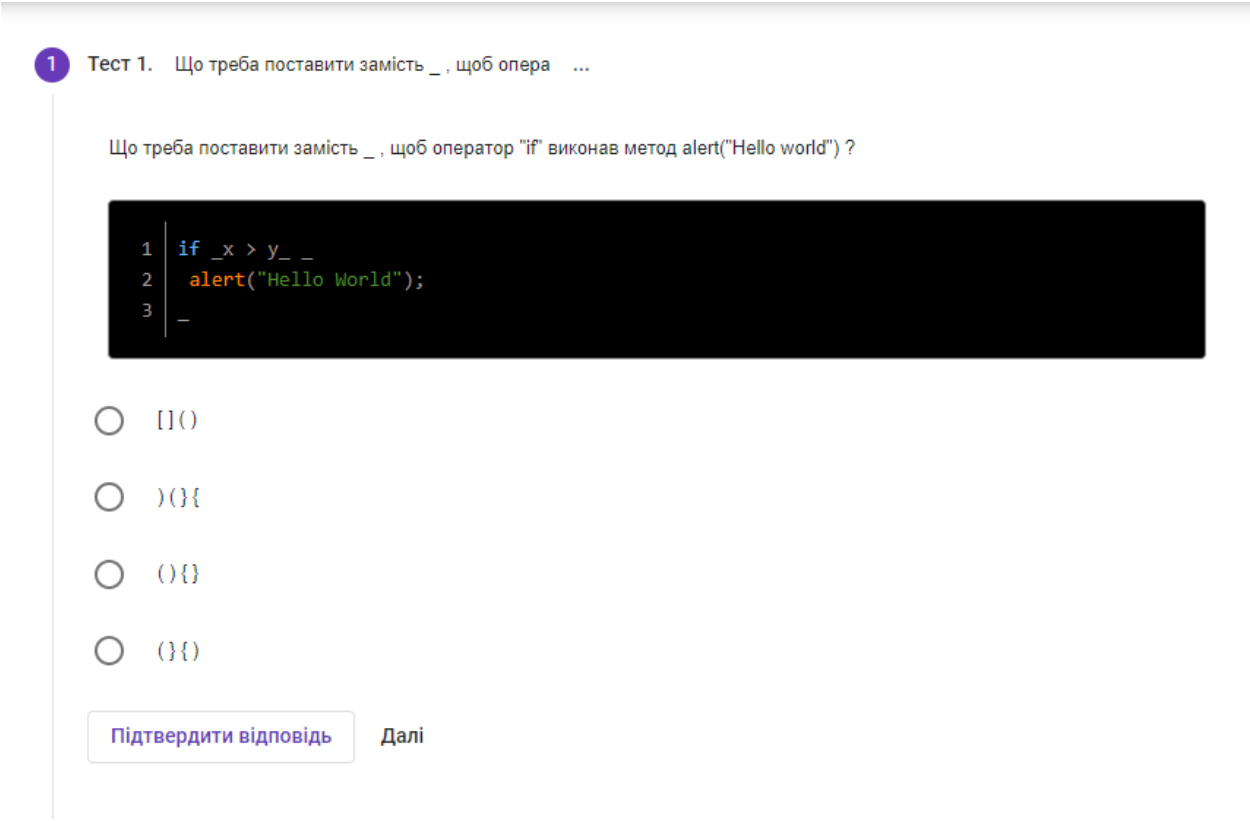

Рис. 3.7. Загальний вигляд тесту

Блок статистики дає можливість користувачеві бачити загальну кількість тестів, кількість правильних відповідей та неправильних.

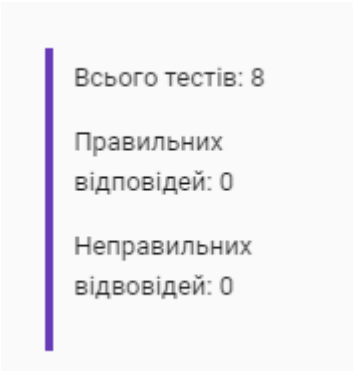

Рис. 3.8. Блок статистики

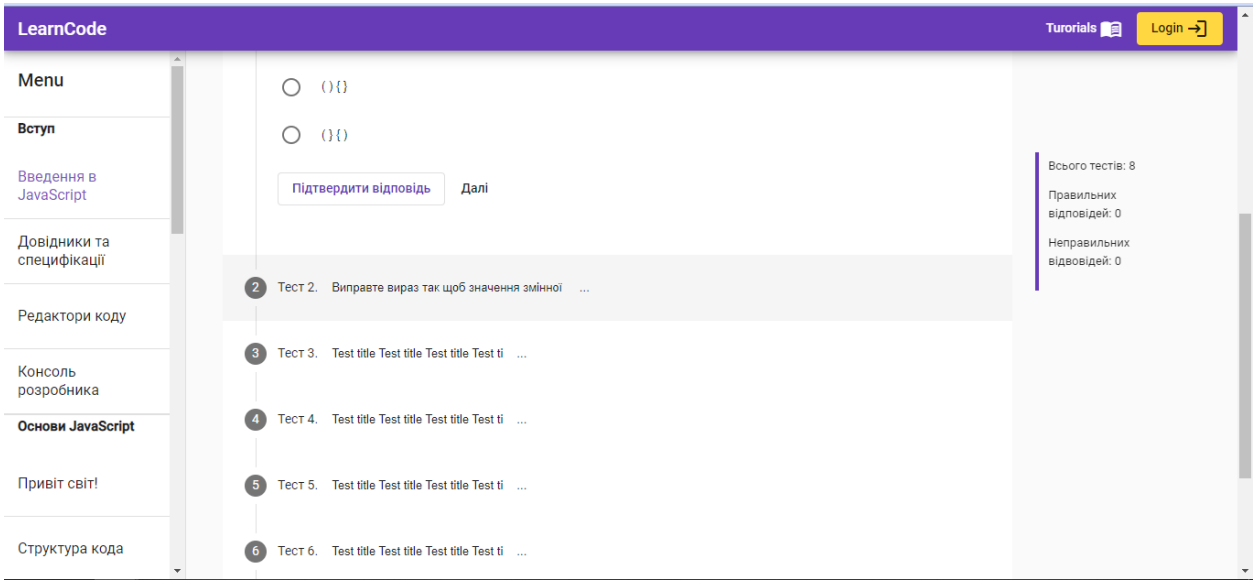

Рис. 3.9. Загальний вигляд секції «Тести»

Секція «Завдання» має на меті закріплення теоретичного матеріалу шляхом виконання практичних здавдань. Весь контекст секції розміщено в mat-stepper елементі. Кожен mat-stepper скаладається з mat-step елементів, які в свою чергу поділяється три підсекції.

- Завдання;
- Відповідь;
- Коментарі;

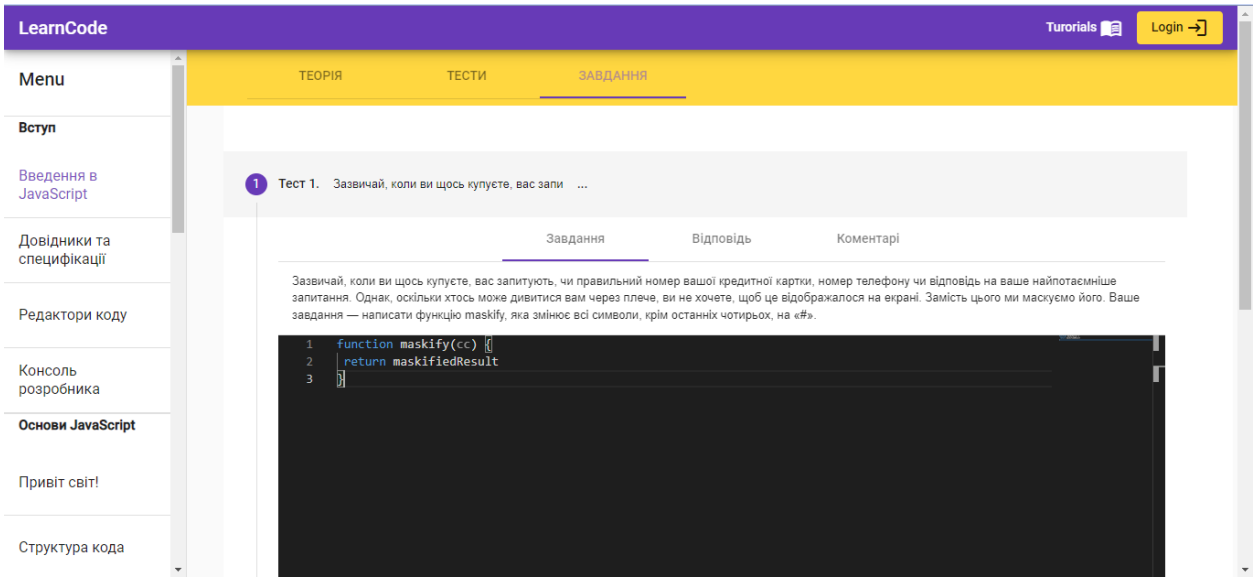

Рис. 3.10. Загальний вигляд секції «Завдання»

Підсекція «Завдання» складається з текстового (Quill - пакет) та кодового типів (Monaco Editor – пакету, який дає змогу писати код прямо на сторінці веб-додатку, так якби користувач писав код в сучасному середовищі розробки). Також в кожній підсекції присутні кнопки «Перевірити відповідь» та «Далі». «Перевірити відповідь» надсилає відповідь до завдання на сервер з подальшою обробкою і результатом відповіді, яка відображається під завданням. Кнопка «Далі» відкриває нове завдання в списку і закриває попереднє.

Підсекція «Відповідь» може мати текстове пояснення (Quill - пакет) та кодове (ngx-highlightJs – пакет)

Рис. 3.11. Загальний вигляд підсекції «Відповідь»

Підсекція «Коментарі» має функціонал додавання коментарів, та пошуку за ключевими словами. Також реалізована можливість відповіді на коментарі, репорту та оцінки (подобається/не подобається).

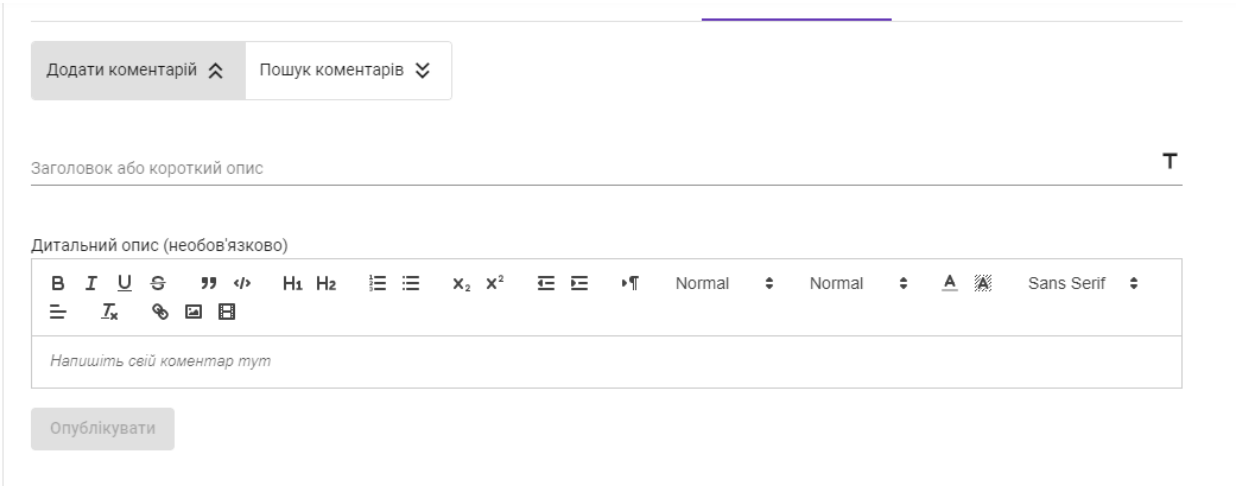

Рис. 3.12. Блок додавання коментарів

Рис. 3.13. Блок пошуку коментарів

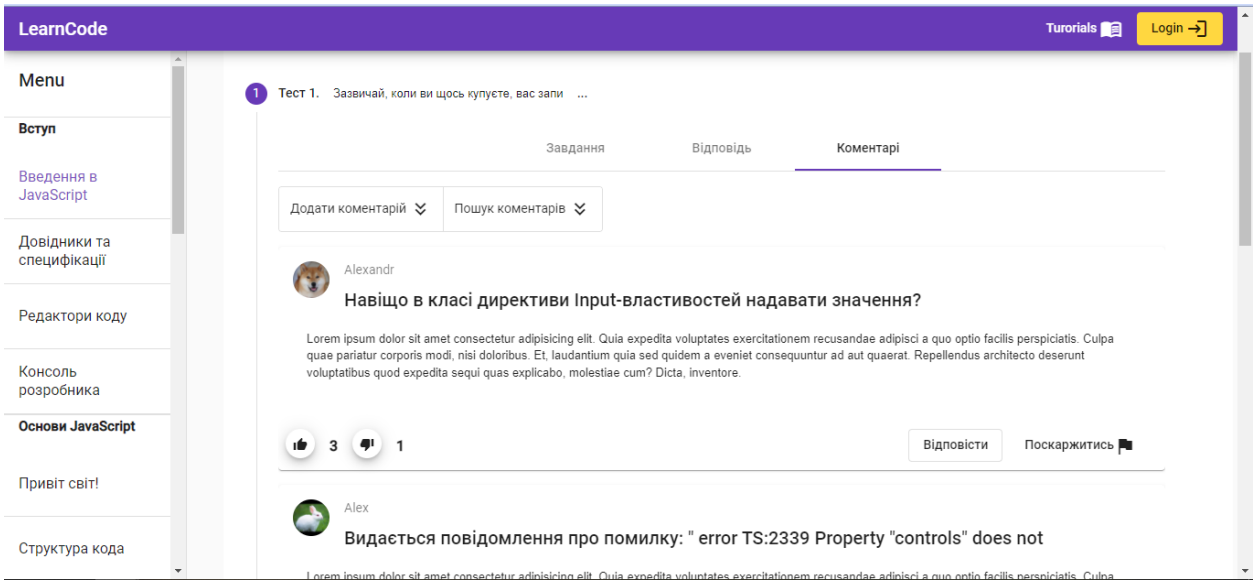

Рис. 3.14. Загальний вигляд підсекції «Коментарі»

Модуль адміністратора реалізує можливість маніпуляцій над контентом веб-додатку. Користувач з правами адміністратора має можливість створювати, редагувати, та видаляти контент. Загальний вигляд веб-додатку в модулі адміністратора майже такий самий як і в модулі користувача, але добавлені різні інструменти в виді кнопок, перемекачів модальних вікон тощо.

Після переходу за адресою /admin відкривається сторінка логіну, якщо аутентифікація не виконана раніше, чи токін не валідний.

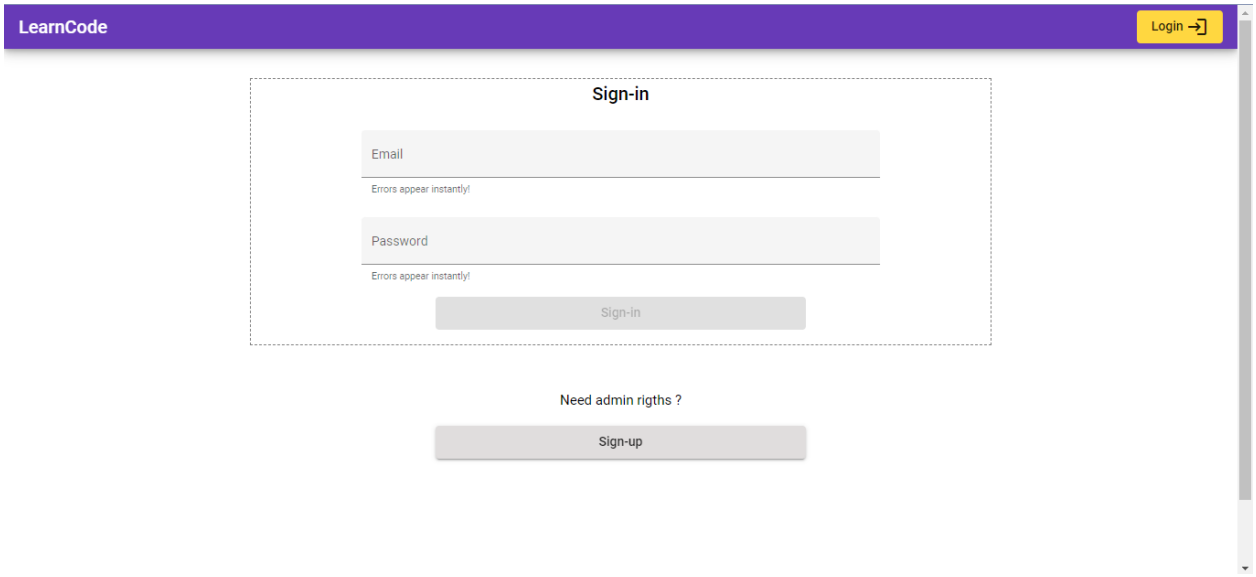

Рис. 3.15. Загальний вигляд сторінки логіну в модулі адміністратора

Після успішного логіну здійснюється автоматичний прехід на сторінку dashboard можна вибрати один з існуючих ресурсів чи створити новий.

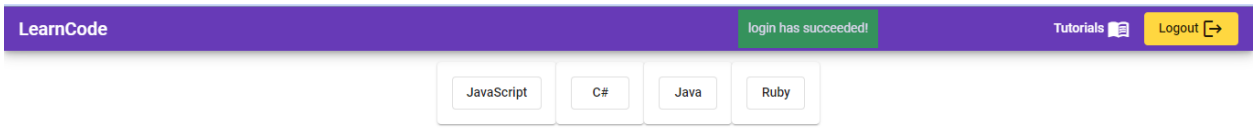

Рис. 3.16. приклад існуючих ресурсів на сторінці dashboard

Блок навігації по розділам в модулі адміністратора має наступні інструменти:

- Створення блоку розділів
- Створення розділів
- Зміна позицій блоків та розділів
- Редагування блоку розділів
- Редагування розділів
- Видалення блоку розділів
- Видалення розділів

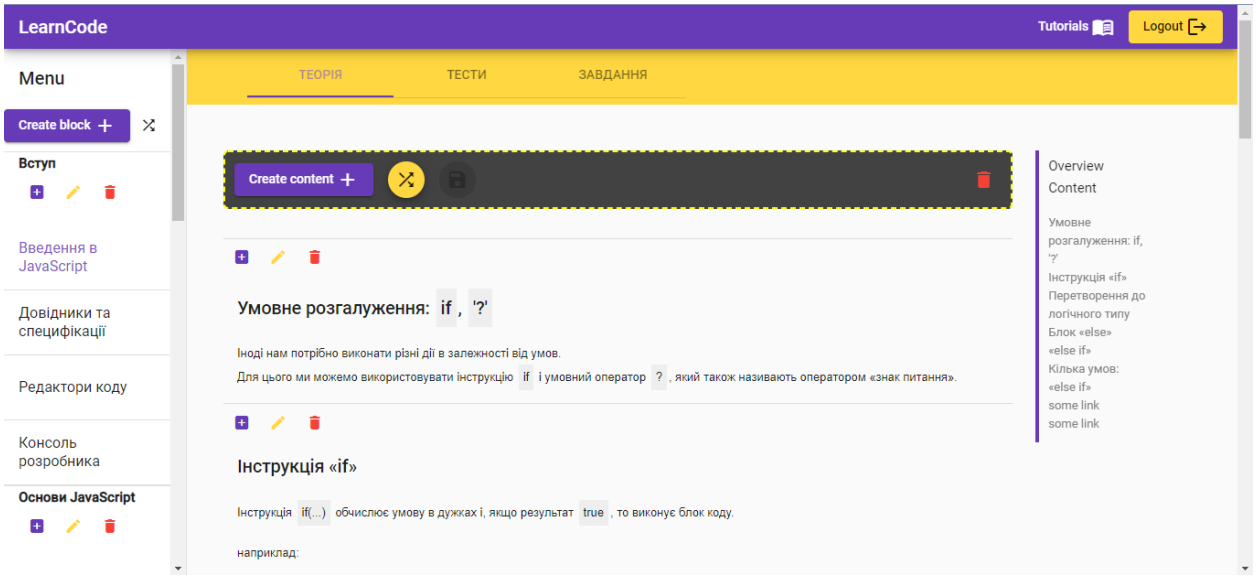

Рис. 3.17. Загальний вигляд веб-додатку (Admin module)

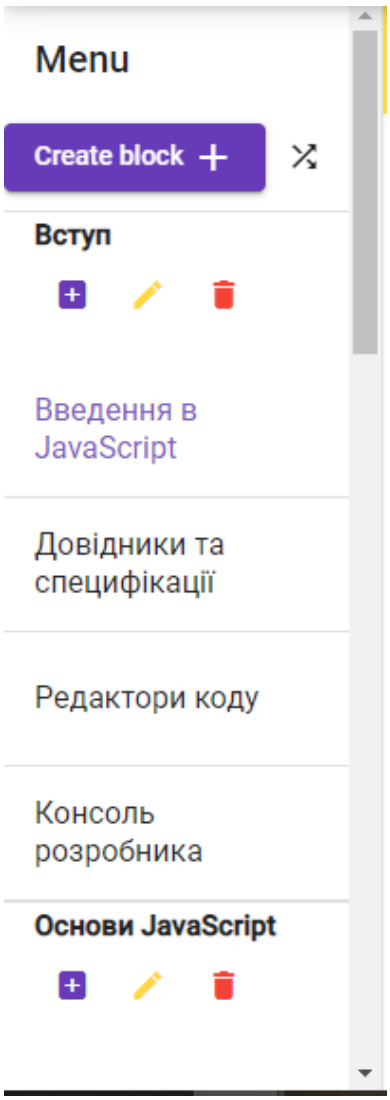

Рис. 3.18. Right-sidenav (Admin module)

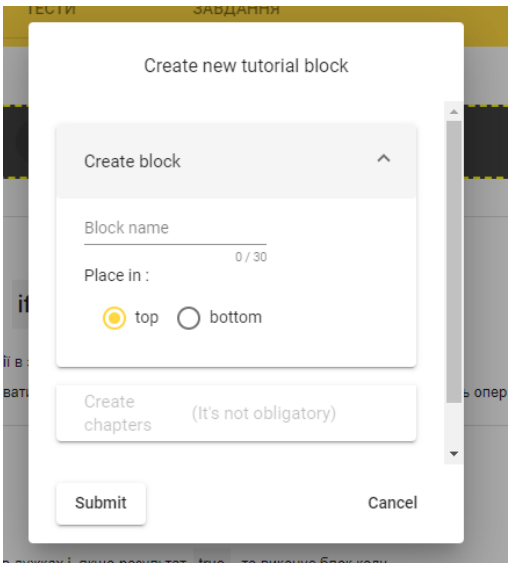

Рис. 3.19. Модальне вікно створення блоку розділів

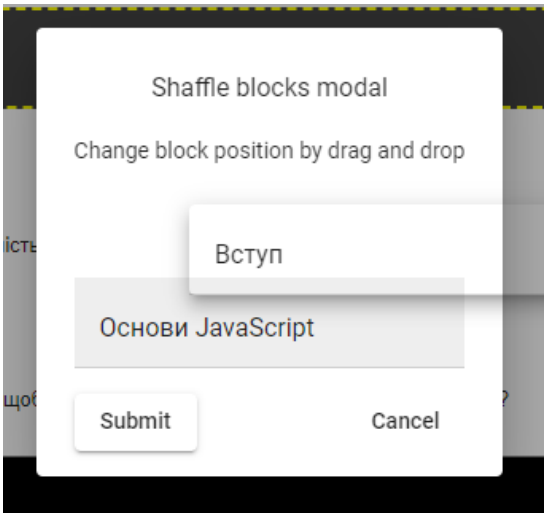

Рис. 3.20. Модальне вікно зміни позиції блоку розділів

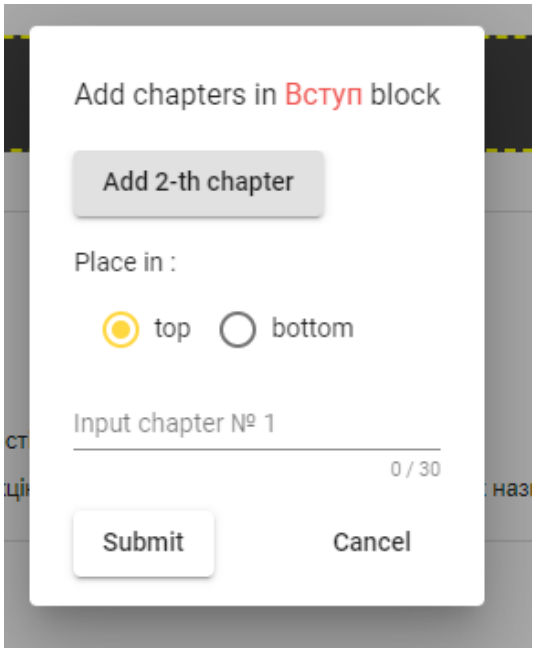

Рис. 3.21. Модальне вікно створення розділів

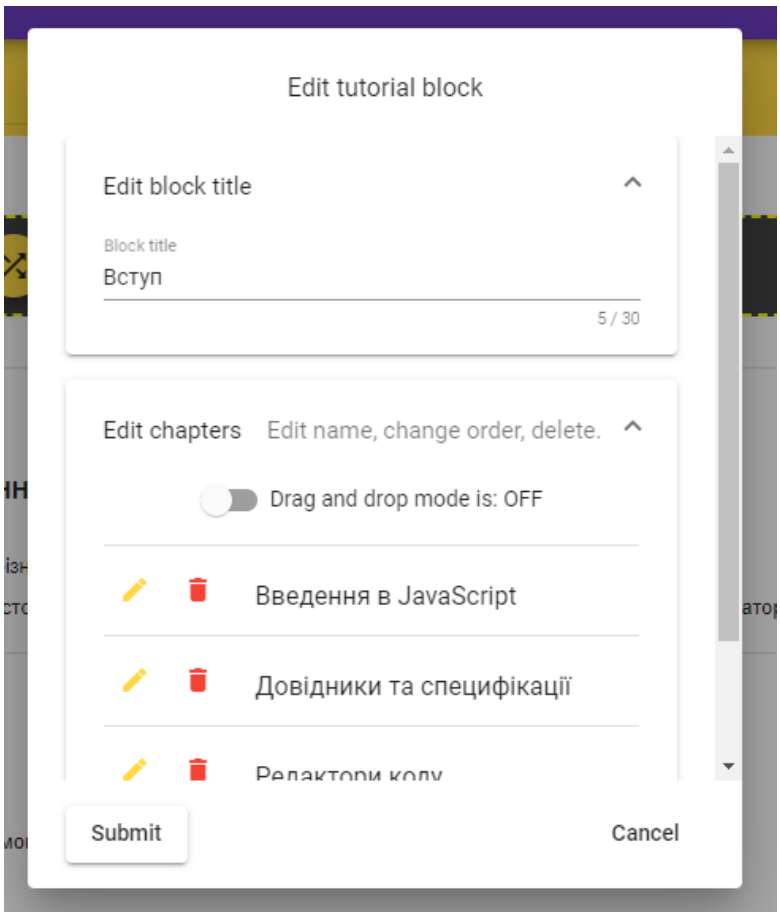

Рис. 3.22. Модальне вікно редагування блоку розділів та розділів

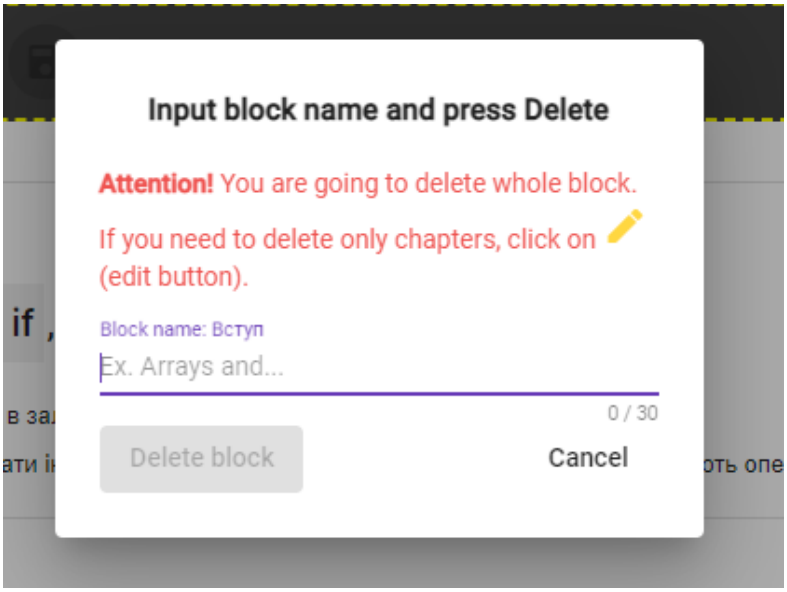

Рис. 3.23. Модальне вікно видалення блоку розділів

Секція «Теорія» має панель інструментів на початку контенту, а також додаткові інструменти створення, редагування та видалення в кожному блоці контенту, що починається заголовком. Після маніпуляцій з контентом, відкривається кнопка збереження змін на сторінці (іконка дискети).

Рис. 3.24. Панель інструментів

Панель інструментів дає можливість створювати конент, змінювати послідовність розташування його елементів, а також видаляти весь контент в даній секції.

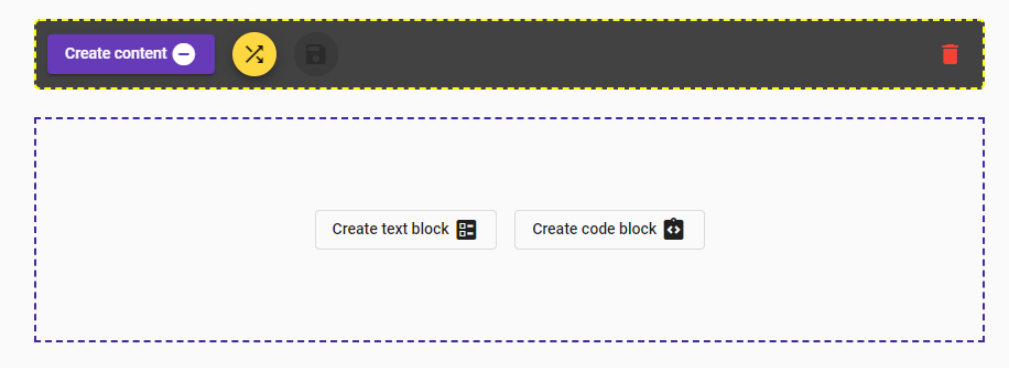

Рис. 3.25. створення контенту (секція «Теорія)

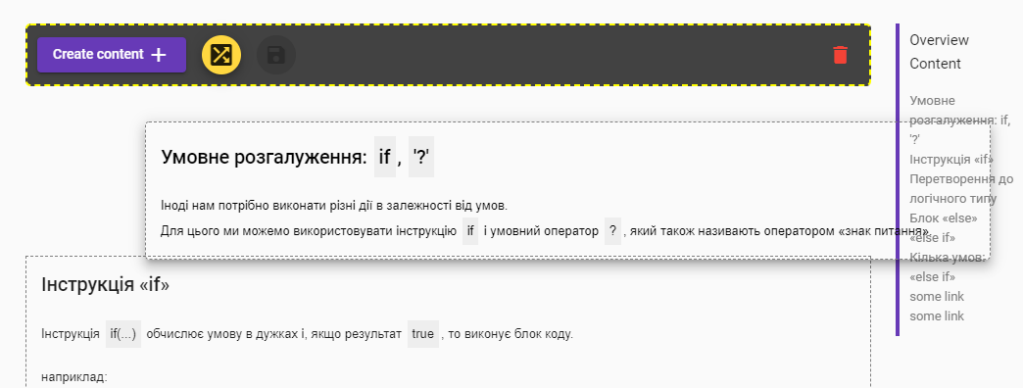

Рис. 3.26. зміна розташування елементів контенту (секція «Теорія) за допомогую технології «drag and drop»

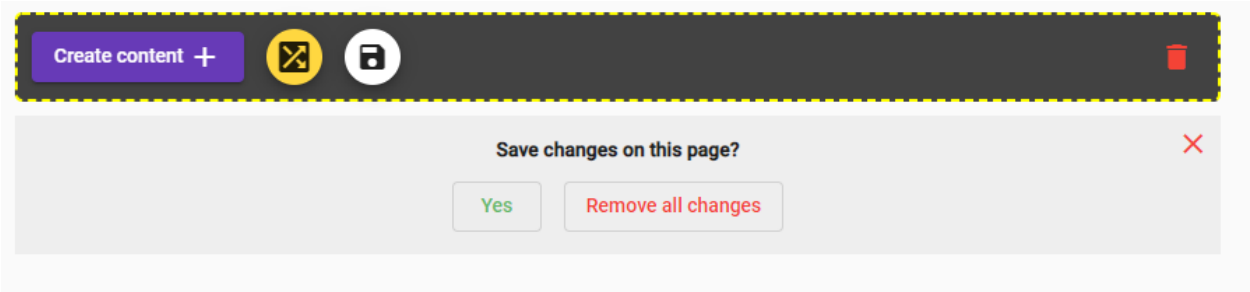

Рис. 3.27. Попередження про збереження контенту

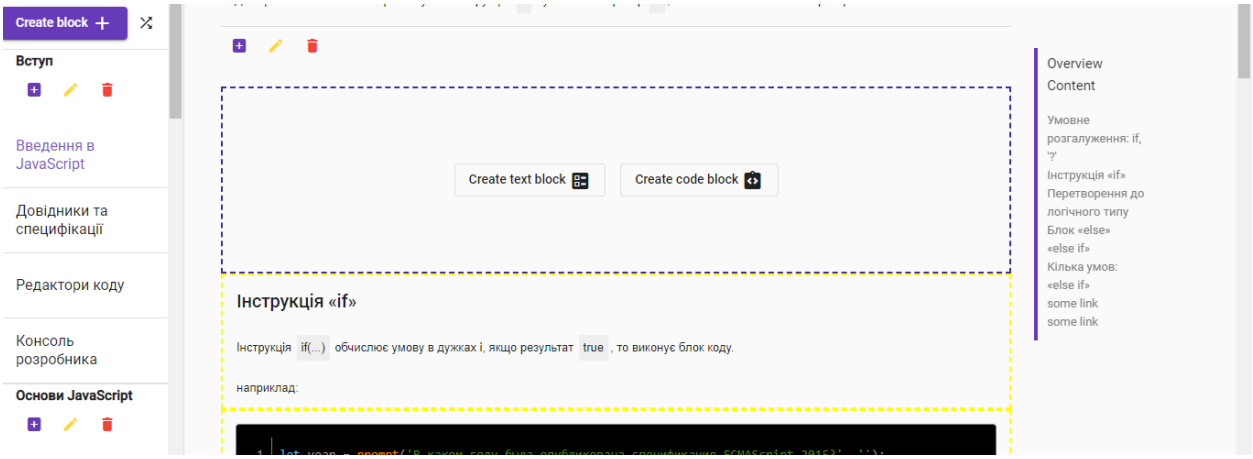

Рис. 3.28. Створення елементів контенту (секція «Теорія)

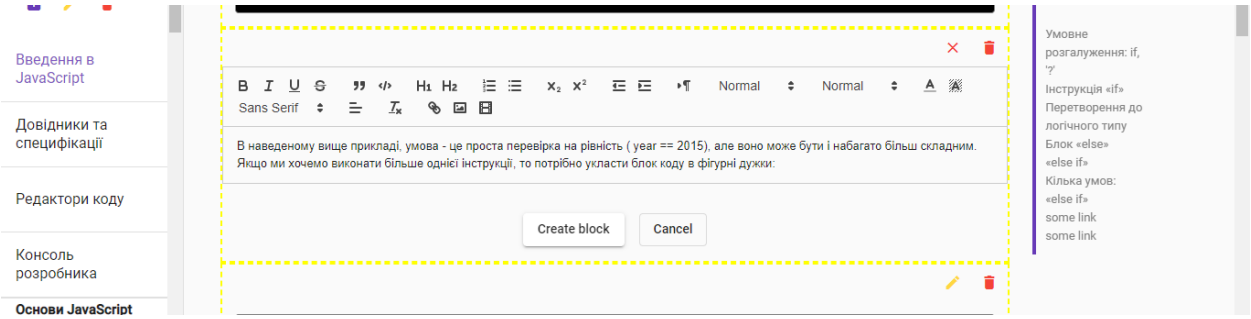

Рис. 3.29. Редагування текстового типу контента (секція «Теорія)

Рис. 3.30. Редагування кодового типу контента (секція «Теорія)

3.2. Реалізація клієнтської частини веб-додатку

Клієнтську частину реалізовано за допомогою Java Script фреймворку - Angular. Цей фрейморк обрано з наступних причин :

- Підтримка данної технології компанією Google;
- Компетентність розробника в даній технології;
- Можливість подальшого маштабування проекту (розширення існуючого функціоналу);
- Використання TypeScript – надає високий рівень підтримуваності коду.

Для реалізації проектних рішень обрано наступні сторонні ключові пакети:

- Angular Material;
- Quill
- Monaco Editor
- HighlightJs

Angular Material – це бібліотека компонентів інтерфейсу користувача (UI), яку розробники можуть використовувати у своїх проектах Angular, щоб прискорити розробку елегантних і послідовних інтерфейсів користувача. Angular Material пропонує багаторазові та красиві компоненти інтерфейсу

користувача, такі як картки, вводи, таблиці даних, інструменти для вибору дат та багато іншого.

Quill – це безкоштовний WYSIWYG-редактор із відкритим вихідним кодом, створений для сучасних веб-додатків. Завдяки модульній архітектурі та виразному API його можна повністю налаштувати під будь-які потреби. Він дозволяє створювати контент на веб-сторінках в виді DOM дерева, з яким можна легко працювати.

Monaco Editor – це редактор коду на основі веб-технологій, який схожий на VS Code. Ця бібліотека обробляє процес налаштування редактора monaco і надає виразний API для взаємодії з monaco з будь-якого середовища розробки.

Highlight.js – це підсвічувач синтаксису, написаний на JavaScript. Він працює як у браузері, так і на сервері. Він може працювати практично з будь-якою розміткою, не залежить від інших фреймворків і має автоматичне визначення мови.

Структура клієнтської частини включає в себе два головних модуля:

- App-module;
- Admin-module;

App-module має наступну структуру:

```
@NgModule({
  declarations: [
    AppComponent,
    MainLayoutComponent,
    TutorialPageComponent,
    TopNavComponent,
    MainNavComponent,
```

```
    TheorySectionComponent,
    TestsSectionComponent,
    PracticeSectionComponent,
    CommentsBlockComponent,
    PracticeTaskComponent,
    PracticeAnswerComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    BrowserAnimationsModule,
    SharedModule,
    MonacoEditorModule.forRoot()
  ],
  providers: [],
  bootstrap: [AppComponent]
})
```

Окремо треба розглянути AppRoutingModule який визначає роботу з навігацією по сторінкам веб-додатку.

```
const routes: Routes = [
  { path: '', component: MainLayoutComponent, children: [
    { path: '', redirectTo: 'tutorial/javaScript', pathMatch: 'full'},
    { path: 'tutorial/:language', component: TutorialPageComponent },
    ]
  },
  {
    path: 'admin',  loadChildren: () => import('./admin/admin.module').then(m => m.AdminModule)
  }
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
```

Якщо перейти за посиланням /admin то почнеться завантаження admin модуля який декларує наступні компоненти для роботи з контентом: AdminLayoutComponent, LoginPageComponent, AdminTutorialPageComponent, TopNavComponent, MainNavComponent, TheorySectionComponent TestsSectionComponent PracticeSectionComponent, CommentsBlockComponent, DashboardPageComponent, AlertComponent, AddSectionModalComponent, AddChapterModalComponent, EditSectionModalComponent, DeleteModalComponent, ShuffleSectionModalComponent, CreateContentComponent, ShuffleTestsModalComponent, CreateTestsComponent, CreateTasksComponent, PracticeTaskComponent, PracticeAnswerComponent, ShuffleTasksModalComponent, SaveConfirmationComponent.

Імпорти: CommonModule, RouterModule.forChild(routes), SharedModule.

Провайдери: AlertService, AuthGuard, ContentBuilderService, ModalsService

Роути:

```
const routes: Routes = [
  { path: '', component: AdminLayoutComponent, children: [
    { path: '', redirectTo: '/admin/login', pathMatch: 'full'},
    { path: 'login', component: LoginPageComponent},
    { path: 'dashboard', component: DashboardPageComponent, canActivate: [AuthGuard]},
    { path: 'tutorial/:language', component: AdminTutorialPageComponent, canActivate: [AuthGuard]},
    ]
  },
];
```

Завдяки імплементації AuthGuard ми маємо змогу обмежити доступ до модуля адміністратора на клієнтській частині веб-додатку. Доступ до чього модуля з'являється тількі тоді коли користувач має валідну аутентифікацію під, яка надає права Адміністратора. AuthGuard має наступну реалізацію:

```
@Injectable()
export class AuthGuard implements CanActivate {

  constructor(private auth: AuthService, private router: Router) {}

  canActivate(
    route: ActivatedRouteSnapshot,
    state: RouterStateSnapshot): Observable<boolean | UrlTree> | Promise<boolean | UrlTree> | boolean | UrlTree {
    if (this.auth.isAuthenticated()) {
      return true;
    } else {
      this.router.navigate(['/admin', 'login'], {
        queryParams: {
          logingAgain: true,
        },
      });
      return false
    }
  }
}
```

Також для роботи AuthGuard реалізований AuthService, який виконую певну логіку з отриманням та обробкою аутентифікаційних даних, а саме токіну.

```
  get token(): string | null {
    if (new Date().getTime() > Number(localStorage.getItem('fb-token-exp'))) {
      this.logout();
      return null;
    } else {
      return localStorage.getItem('fb-token');
    }
  }

  login(user: User): Observable<any> {
    user.returnSecureToken = true;
    // do http request
    return of({idToken: 'someAdminToken', expiresIn: '3000'}).pipe(tap(this.setToken))
  }

  logout() {
    localStorage.clear();
  }

  isAuthenticated() {
    return !!this.token;
  }

  private setToken(response: AuthResponse) {
    const expDateInMs = new Date().getTime() + +response.expiresIn * 1000;
    localStorage.setItem('fb-token', response.idToken);
    localStorage.setItem('fb-token-exp', expDateInMs.toString());
  }
```

Ідея полягає в тому, щоб отримати токін згенерований на серверній частині веб-додатку так встановити його в localStarage. При кожному запиті до серверу в модулі адміністратора, буде виконуватись перевірка токіна на валідність і якщо токін не проходить по критеріям валідації, то здійснюється переадресація на сторінку логіна.

Також маємо проміжутній модуль SharedModule який має спільний функціонал, що розділяється між AppModule та AdminModule. Візьмем, наприклад, для розгляду пайп FilterCommentPipe, який може імплементуватися в різних модулях. Пайп має наступний вигляд:

```
@Pipe({
  name: 'filterComments'
})
export class FilterCommentsPipe implements PipeTransform {

  transform(comments: Comment[], searchValue: string = ''): any {
    if (searchValue === '') return comments

    return comments.filter(comment => {
      return comment.title.toLocaleLowerCase().includes(searchValue.toLocaleLowerCase()) || comment.content.
      toLocaleLowerCase().includes(searchValue.toLocaleLowerCase())
    })
  }
}
```

Завдання пайпу FilterCommentPipe – фільтрація масиву з елементами типу Comment по значенню яке пайп отрмує з інпут елементу.

```
<div *ngFor="let comment of comments | filterComments: searchValue">
  <mat-card class="example-card">
    <mat-card-header>
      <div
        mat-card-avatar
        class="example-header-image"
        style="background-image: url('{{ comment.userData.img }}');"
      ></div>
      <mat-card-subtitle>{{ comment.userData.nickName }}</mat-card-subtitle>
      <mat-card-title>{{ comment.title }}</mat-card-title>
```

```
</mat-card-header>

<mat-card-content>
  <quill-view-html [content]="comment.content"></quill-view-html>
</mat-card-content>
<mat-card-actions>
  <div class="btn-container flex">
    <div class="flex">
      <div class="btn-section">
        <button
          mat-mini-fab
          aria-label="Example icon button with a home icon"
        >
          <mat-icon>thumb_up</mat-icon>
        </button>
        <span class="statistic">{{ comment.likes }}</span>
      </div>
      <div class="btn-section">
        <button
          mat-mini-fab
          aria-label="Example icon button with a home icon"
        >
          <mat-icon>thumb_down</mat-icon>
        </button>
        <span class="statistic">{{ comment.dislikes }}</span>
      </div>
    </div>

    <div>
      <button mat-stroked-button>Відповісти</button>
```

```
        <button mat-button>
          <span>Поскаржитись</span><mat-icon>flag</mat-icon>
        </button>
      </div>
    </div>
  </mat-card-actions>
</mat-card>
</div>
```

3.3. Реалізація серверної частини веб- додатку

Серверну частину реалізовано за допомогою Java Script фреймворку - NestJs. Цей фрейморк обрано з наступних причин :

- Підхід, який використовує Angular (декоратори, модулі, сервіси, гварди тощо);
- Компетентність розробника в даній технології;
- Можливість подальшого маштабування проекту (розширення існуючого функціоналу);
- Використання TypeScript – надає високий рівень підтримуваності коду, комфортну розробку, та зменшує кількість багів на етапі реалізації проекту.

Функціонал серверної частини умовно, можна, поділити на наступні основні блоки:

- Блок по обробці запитів на весь контент веб-додатку;
- Блок по роботі з аутентифікацією та авторизацією користувачів;
- Блок загальних функціональних складових.

Блок по обробці запитів на весь контент веб-додатку відповідає за всі запити з клієнтської частини, які пов’язані з отриманням, обробкою, та

видаленням елементів контенту. Ендпоїнти в цьому блоці поділяються на захищенні, напівзахищенні та незахищенні. Захищенні ендпоїнти, потребують токін в тілі, якого буде поле role із значенням admin. Токін генерується під час проходження аутентифікації. Напівзахищенні також потребують токін в тілі, якого буде поле role із значенням user, але відмінність від захищенного ендпоїнта полягає в тому що role = user може отримати будь-який користувач за допомогою створення облікового запису в модулі користувача клієнтської частини, тоді як role = admin надається тільки додаткових умовах, які реалізуються на рівні самої інформаційної системи. Незахищенні ендпоїнти не потребують аутентифікаційних даних (токіна) для здійснення запита на сервер. Зазвичай незахищенні ендпоїнти використовуються в методі запиту Get для отримання лише контенту веб-додатку, але не редагування та видалення.

Реалізація функціоналу захисту ендпоїнтів, завдяки фреймворку NestJs виглядає достатньо лаконічно та просто. Захист ендпоїнтів реалізується головним чином завдяки декораторам, гвардам та стратегіям, які по свої суті являються провайдерами зареєстрованами на рівні AuthModule.

Для прикладу розглянемо nav-data ендпоїнт цього блоку. Nav-data ендпоїнт призначений для різних маніпуляцій з навігаційним контентом певного навчального ресурсу. Запити на цей ендпоїнт мають три рівня доступи: захищений (потребує права адміністратора), напівзахищений (потребує права користувача), незахищений (не потребує ніяких прав).

Структуру контролера nav-data має наступний вигляд:

```
@Controller('nav-data')
export class NavDataController {
  constructor(private readonly navDataService: NavDataService) {}
  @Get(':tutorialId')
  public async getAllNavData(
    @Param('tutorialId') tutorialId: string
  ) {
    return this.navDataService.getAllNavData(tutorialId)
  }
  @UseGuards(JwtAuthGuard)
  @Post()
  @UsePipes(ValidationPipe)
  public async createNavData(
    @Body() body: CreateNavDataDto,
    @Request() req: RequestFromJWT
  ) {
    return this.navDataService.createNavData(body.tutorialId, body.data, req.user.role)
  }
  @UseGuards(JwtAuthGuard)
  @Put()
  @UsePipes(ValidationPipe)
  updateNavData(@Body() body: UpdateBlockDto, @Request() req: RequestFromJWT) {
    return this.navDataService.update(body, req.user.role);
  }
}
```

Метод getAllNavData дозволений для користувачів з будь-якими правами доступу, а методи createNavData та updateNavData дозволені тільки для користувачів з правами доступу рівня admin.

Розглянемо реалізацію методів createNavData та updateNavData в NavDataService детальніше. NavDataService інджектує в своєму конструкторі сервіси: SectionService, ChapterService та TutorialNamesService, завдяки яким реалізується делегування частини функціоналу іншим компонентам, що збільшує якість коду та його підрумуваність.

```
@Injectable()
export class NavDataService {
  constructor(
    private sectionService: SectionService,
    private chaterService: ChapterService,
    private tutorialNamesService: TutorialNamesService
  ) {}
```

В NavDataService реалізовано три методи для взаємодії з бізнес логікою додатку.

getAllNavData – метод, який призначений для отримання всіх навігаційних данних по переданому в параметрах запиту ідентифакатора навчального ресурсу. Його реалізація виглядає наступним чином:

```
public async getAllNavData(tutorialId: string): Promise<INavData[]> {
  await this.tutorialNamesService.findById(tutorialId)

  const allSections = await this.sectionService.getAll(tutorialId)
  const chaptersInAllSectionsProm = allSections.map((section) => {
    const chaptersInOneSection: Promise<IChapter[]> = this.chaterService.getAll(section.sectionId)
    return chaptersInOneSection
  })
  const chaptersInAllSections = await Promise.allSettled(chaptersInAllSectionsProm)
  return chaptersInAllSections.map<INavData>((currentValue, currentIndex) => {
    if (currentValue.status === 'fulfilled') {
      const navDataItem: INavData = {
        section: allSections[currentIndex],
        chapters: currentValue.value
      }
      return navDataItem
    } else {
      return { section: allSections[currentIndex], chapters: [] }
    }
  })
}
```

Метод приймає в параметри ідентифакатор навчального ресурсу за яким буде здійснений пошук всіх навігаційних даних за допомогою методу getAll інджектованого сервісу SectionService. Далі здійснюється перебір кожної секції за індентифікатором якої отримуємо проміс масиву з елементами типу IChapter. Далі виконується очікування виконання всіх промісів в масиві за допомогою метода проміса allSettled, який дозволяє отримати результати всіх виконаних промісів масиву незалежно від успішності. Далі виконується сортування всіх розділів та повертається в контролер результат виконання методу getAllNavData в виді масиву елементів з типами INavData.

createNavData – метод, який призначений для створення навігаційних данних в навчальному ресурсі ідентифікатор якого приймається в параметрамх методу. Його реалізація виглядає наступним чином:

```
public async createNavData(tutorialId: string, createBlockDto: CreateBlockDto, role: string): Promise<INavData> {
  const sectionOnCreate: ISectionOnCreate = {tutorialId, sectionTitle: createBlockDto.section.sectionTitle,
  position: createBlockDto.section.position}
  const newSection = await this.sectionService.create(role, sectionOnCreate)
  if (!newSection) return

  const chaptersInSectionProm: Promise<IChapter>[] = createBlockDto.chapters.map<Promise<IChapter>>((chapter) => {
    const chapterOnCreate: IChapterOnCreate = {sectionId: newSection.sectionId, chapterTitle: chapter.
    chapterTitle, position: chapter.position}
    return this.chaterService.create(role, chapterOnCreate)
  })
  const newChapters = await Promise.all(chaptersInSectionProm)
  if (!newChapters) return

  const navDataItem: INavData = {
    section: newSection,
    chapters: newChapters
  }
  return navDataItem;
}
```

Метод приймає індентифікатор навчального ресурсу в якому будуть створені дані, що передаються наступним параметром в цей метод, а також третім параметром метод приймає роль, що отримана за допомогою JwtStrategy через токін що передається в параметрах запиту.

updateNavData– метод, який призначений для оновлення навігаційних данних в навчальному ресурсі ідентифікатор якого приймається в параметрамх методу. Його реалізація виглядає наступним чином:

```
public async updateNavData(updateDto: UpdateBlockDto, role: string): Promise<INavData> {
  if (role !== Role.Admin) {
    throw new InappropriateRoleError(403);
  }
  const updatedSection: UpdateSectionDto = await this.sectionService.update(updateDto.section.tutorialId,
  updateDto.section)

  let updatedChapters!: UpdateChapterDto[]
  if (updateDto.chapters.length) {
    updatedChapters = await this.chaterService.update(updateDto.chapters)
  } else {
    updatedChapters = []
  }

  const updatedNavData: INavData = {
    section: updatedSection,
    chapters: updatedChapters
  }
  return updatedNavData
}
```

Метод приймає дані які будуть обновлені, та роль за якою здійснюється перевірка чи користувач має права на оновлення конентну. Головною складністю тут є те, що оновлювати дані потрібно не тільки тих що приходять в методі запиту, але й інших залежних від них. Наприклад якщо змінити

позицію в розташувані елементів то треба змінювати позицію всіх елементів, які присутні також в цьому блоці. Для цього реалізований метод в сервісі SectionService – update. Виглядає він наступним чином:

```
public async update(
   tutorialId: string,
   updateDto: UpdateSectionDto,
 ): Promise<UpdateSectionDto> {
   let allSections = await this.getAll(tutorialId);
   let isChangedSectionPosition = false;
   updateDto.sectionId = updateDto.sectionId.toString()

   allSections.forEach((section) => {
     if (section.sectionId.valueOf() === updateDto.sectionId &&
         section.position !== updateDto.position) {
       isChangedSectionPosition = true;
     }
   });

   if (isChangedSectionPosition) {
     allSections = allSections.filter((section) =>
         section.sectionId.valueOf() !== updateDto.sectionId)
     const firstPartOfAllNavData = allSections.slice(0, updateDto.position - 1);
     const secondPartOfAllNavData = allSections.slice(updateDto.position - 1,
         allSections.length);

     const sectionsDueToPositions = [...firstPartOfAllNavData, updateDto,
       ...secondPartOfAllNavData];
     const renewedSectionsPositions = sectionsDueToPositions.map((section, i) =>
       {
```

```
        return {
          ...section,
          position: i + 1
        }
      });
    const updatedSectionsProm: Promise<UpdateSectionDto>[] =
        renewedSectionsPositions.map((section) => {
      return this.updateById(section.sectionId, section);
    });
    const updatedSections: UpdateSectionDto[] = await
        Promise.all(updatedSectionsProm)
    if (updatedSections) {
      return updatedSections.find((section) =>
        section.sectionId.valueOf() === updateDto.sectionId)
    } else {
      throw new UpdateError();
    }
  } else {
    return this.updateById(updateDto.sectionId, updateDto);
  }
}
```

Блок по роботі з аутентифікацією та авторизацією користувачів відповідає за безпеку даних системи, та даних особистих даних користувачів.

Один із способів аутентифікації в інформаційній системі полягає у попередній ідентифікації на основі користувацького ідентифікатора («логіна», від англ. login — реєстраційного імені користувача) і пароля — певної конфіденційної інформації, знання якої передбачає володіння певним

ресурсом в мережі. Отримавши введений користувачем логін і пароль, комп'ютер порівнює їх зі значенням, яке зберігається в спеціальній захищеній базі даних і, у випадку успішної автентифікації проводить авторизацію з подальшим допуском користувача до роботи в системі.

Авторизація — керування рівнями та засобами доступу до певного захищеного ресурсу, як у фізичному розумінні (доступ до кімнати готелю за карткою), так і в галузі цифрових технологій (наприклад, автоматизована система контролю доступу) та ресурсів системи залежно від ідентифікатора і пароля користувача або надання певних повноважень (особі, програмі) на виконання деяких дій у системі обробки даних.

В даній інформаційній системі реалізований ролевий підхід авторизації, який надає окремі повноваження для взаємодії з елементами контенту та користувачами системи. Роль призначається залежно від облікового запису користувача, та надається в аутентифікаційних даних (токіна) в зашифрованому виді.

Для прикладу розглянемо реалізацію деяких методів сервісу аутентифікації. Сервіс аутентифікації має наступні методи:

- login;
- createUser (signup - контролер);
- updateUser;
- deleteUser;
- validateEmail;

Метод login призначений для аутентифікації користувача в системі і генерування токіна з подальшою передачею його на клієнтську частину веб-додатку. Метод login має наступний вигляд:

```
public async login(user: UserDocument): Promise<FormatedUser> {
  const access_token = this.hashService.generateAccessToken(user.email, user.userName, user._id, user.role)
  return {
    access_token,
    userId: user._id,
    email: user.email,
    userName: user.userName
  };
}
```

Метод createUser призначений для створення облікового запису користувача. Його особливість полягає в тому, що окрім іншого він конвертує пароль користувача в хеш-кодування, для забеспечення безпечного збереження особистих даних користувачів системи. Метод createUser має наступний вигляд:

```
public async createUser(createUserDto: CreateUserDto): Promise<FormatedUser> {
  const isAvailable = await this.validateEmail(createUserDto.email);
  if (isAvailable) {
    throw new EmailExistsError(400);
  }
  createUserDto.password = await this.hashService.bcryptPassword(createUserDto.password);
  const user: UserDocument = await this.usersService.create(createUserDto)
  const access_token = this.hashService.generateAccessToken(user.email, user.userName, user._id, user.role)
  return {
    access_token,
    userId: user._id,
    email: user.email,
    userName: user.userName
  };
}
```

Також можемо розглянути стратегії аутентифікації які використовуються в методах AuthService.

JwtStrategy – визначає, по яким параметрам буде здійснюватись перевірка та шифрація токіна. Шифрація здійснюється за допомогою певного алгоритму, який надається passport-jwt пакетом. Для шифрації необхідно передавати в методи JwtStrategy деякі дані, наприклад: дані із аутентифікації, змінну яка містить дані про середовище на, на якому виконується робота системи, тощо.

JwtStrategy має наступний вигляд:

```
@Injectable()
export class JwtStrategy extends PassportStrategy(Strategy) {
  constructor() {
    super({
      jwtFromRequest: ExtractJwt.fromAuthHeaderAsBearerToken(),
      ignoreExpiration: false,
      secretOrKey: jwtConstants.secret,
    });
  }

  async validate(payload: Payload): Promise<ReturnedJWT> {
    return { userId: payload.sub, email: payload.email, userName: payload.userName, role: payload.role };
  }
}
```

Конструктор даного класу приймає параметри для класу з якого наслідується в цілях конфігурації стратегії. Метод validate приймає в параметри об’єкт даних розпарсеного токіна, що надійшов з клієнтської частини, та надає можливість для різних додаткових маніпуляцій, які потребують цих даних.

Блок загальних функціональних складових призначений для збереження частини функціоналу серверної частини веб-додатка, яка розподіляється між різними частинами системи. Завдяки цьому блоку реалізується один із підходів в програмуванні – перевикористання функціоналу та неповторюванність.

Можемо взяти для прикладу кастомний функціонал пайпу, який реалізований за допомогою пакетів class-validator та class-transformer та призначений для валідації даних, що приходять із клієнтської частини.

```
@Injectable()
export class ValidationPipe implements PipeTransform {
  async transform(value: any, { metatype }: ArgumentMetadata) {
    if (!metatype || !this.toValidate(metatype)) {
      return value;
    }

    const object = plainToClass(metatype, value);
    const errors: ValidationError[] = await validate(object, { skipMissingProperties: true });

    if (errors.length > 0) {
      Logger.error(`${Messages.validationFailed}\n`, JSON.stringify(errors));
      throw new BadRequestException(errors);
    }

    return object;
  }

  private toValidate(metatype: any): boolean {
    const types = [String, Boolean, Number, Array, Object];
    return !types.find((type) => metatype === type);
  }
}
```

ValidationPipe використовується декоратором @UsePipes(), який надається необхідним контролерам. Наприклад:

```
@Controller('auth')
export class AuthController {
  constructor(private readonly authService: AuthService) {}

  @UseGuards(LocalAuthGuard)
  @Post('login')
  @UsePipes(ValidationPipe)
  async login(@Body() body: LoginDto, @Request() req: any) {
    return this.authService.login(req.user);
  }
```

Умови валідації встановлюються в класах, які описують тип даних, що буде очікувати отримати контролер при запиті по певному шляху. На поле об’єкта, що очікує контролер встановлюється декоратори валідації, наприклад: @IsString(), @IsNumber(), @IsNotEmpty(), @IsDefined().

3.4. Висновок за розділом 3

Метод навчання мовам програмування реалізований застосунком (SPA – single page application) в якому і застосовані основні методологічні підходи в навчанні та обрані настпуні технології: Angular – для клієнтської частини, NestJs – для серверної та MongoDb – як база данних.

Веб-додаток було вирішено реалізовувати в виді (SPA – single page application) в якому імплементуються основні методологічні підходи в навчанні. Клієнтську частину вирішено реалізовавути за допомогую фронт-ент технології Angular яка по своїй суті являється фрейморком мови програмування Java Script.

Технологію Angular обрано з наступних причин:

- Підтримка данної технології компанією Google;
- Компетентність розробника в даній технології;
- Можливість подальшого маштабування проекту (розширення існуючого функціоналу);
- Використання TypeScript – надає високий рівень підтримуваності коду.

Серверну частину вирішено реалізовувати за допомогую бек-енд технології NestJs, який використовує функціонал фрейморку ExpressJs, але також добавляє типізацію (TypeScript), та деякі нові патерни проектування.

Технологію NestJs обрано з наступних причин:

- Підхід, який використовує Angular (декоратори, модулі, сервіси, гварди тощо);
- Компетентність розробника в даній технології;
- Можливість подальшого маштабування проекту (розширення існуючого функціоналу);

- Використання TypeScript – надає високий рівень підтримуваності коду, комфортну розробку, та зменшує кількість багів на етапі реалізації проекту.

В якості системи керування базами данних обрано MongoDb, який з одної сторони забезпечає всі потреби веб-додатку в збережені, обробки та видаленні даних, а з іншої, є відносно, простою в налаштуванні та користуванні.

Сукупність обраних інструментів забезпечує надійність інформаційної системи, а також додатстній рівень комфорту при реалізації та розширені існуючого функціоналу.

ВИСНОВКИ

Для полегшення процесу навчання програмуванню реалізований метод навчання програмуванню за допомогою веб-технологій. Додаток використовує mobile-first підхід, який визначає, що додаток в першу чергу буде використовуватись з мобільніх пристрої. Додаток призначений для вивчення мов програмування, який використовує певні методологічні підходи в навчанні реалізовані за допомогую веб-технологій. В додатку наведені базові конструкції та елементи які присутні в усіх сучасних мовах програмування, що відрізняє його зі всіх схожих додатків. У магістерській роботі виконаний аналіз та програмна реалізація веб-додатку для вивчення мов прорамування.Також були задіяні сучасні, основні технології: Angular, NestJs, MongoDb, Angular Material, Sandbox та інші. Середа розробки – Visual Code. Sandbox інтерпретатор це середовище програмування. В якому можливо реалізовувати невеликі задачі з програмування. Реалізований веб-додаток дає моєливість застосовувати знання відразу на практиці, без переходу на сторонні інформаційні ресурси.

Магістерська робота складається з трьох розділів. У першому розділі наведений порівняльний аналіз різних відомих систем для вивченя мов програмування. Більшість з них мають схожий функціонал, але вони оринтовані чи на малих дітей чи на людей вже добре знаючих хочаб базові єлементи та правила програмування. У другому розділі виконаний аналіз потрібної інформації для реалізації додатку. Надані класифікація та робота окремих методів. Визначені їх переваги та недоліки. Надані алгоритми методів Drag-and-drop технології та толкування роботи Sandbox. У третьому розділі роботи виконано проектування програмного забезпечення. Цей метод навчання програмування може бути одним з елементів вдосконалення освітнього процесу майбутнії спеціалистів з ІТ. Реалізація зручного інтерфейсу веб-застосування зробить процес навчання більш простим та зрозумілим для початківців програмістів.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ

1. Ресурс Udacity для вивчення програмування [Електронний ресурс] – Режим доступу: https://www.udacity.com/.
2. Додаток Swift Playgrounds для вивчення мови програмування Swift [Електронний ресурс] – Режим доступу: https://www.apple.com/swift/playgrounds/.
3. Ресурс Codecademy для вивчення програмування [Електронний ресурс] – Режим доступу: https://www.codecademy.com/.
4. Додаток Lrn для вивчення мов програмування та гіпер-розмітки [Електронний ресурс] – Режим доступу: https://itunes.apple.com/us/app/lrn-learnto-code-in-html-css-javascript-ruby-python/id1019622677?mt=8.
5. Додаток Tynker для вивчення алгоритмізації та програмування [Електронний ресурс] – Режим доступу: https://www.tynker.com/.
6. Ресурс Coursera для онлайн навчання [Електронний ресурс] – Режим доступу: https://ru.coursera.org/.
7. Ресурс Khan Academy для онлайн навчання [Електронний ресурс] – Режим доступу: https://ru.khanacademy.org/.
8. Ресурс EdX для онлайн навчання [Електронний ресурс] – Режим доступу: https://www.edx.org/.
9. Ресурс CodeCombat для онлайн навчання [Електронний ресурс] – Режим доступу: https://codecombat.com/.
10. Ресурс CodeWars для онлайн навчання [Електронний ресурс] – Режим доступу: https://www.codewars.com/.
11. Документация для разработчика [Електронний ресурс] – Режим доступу: http://habrahabr.ru/Developer_Documentation.
12. Angular documentation – Википедия [Електронний ресурс] – Режим доступу: https://angular.io/docs.

13. Angular Material – https://material.angular.io/components/categories– Режим доступу: https://material.angular.io/.
14. Do you know what a REST API is? [Електронний ресурс] – Режим доступу: https://www.sitepoint.com/developers-rest-api/.
15. NestJs – Википедия [Електронний ресурс] – Режим доступу: https://docs.nestjs.com/.
16. Mongoose – Википедия [Електронний ресурс] – Режим доступу: https://mongoosejs.com/docs/guide.html.
17. AtlasMongoDb – [Електронний ресурс] – Режим доступу: https://account.mongodb.com/.
18. Visual Studio – Википедия [Електронний ресурс] – Режим доступу: https://code.visualstudio.com/.