Київський національний торговельно-економічний університет

Кафедра комп'ютерних наук та інформаційних систем

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Розробка web додатку для обліку навчального процесу студентів»

Студента 4 курсу, 10 групи,
спеціальності
122 «Комп'ютерні науки»

_____
підпис студента

Іванченко
Сергій
Олександрович

Науковий керівник
кандидат фізико-математичних наук,

_____
підпис керівника

Філімонова Тетяна
Олегівна

Гарант освітньої програми
кандидат технічних наук, доцент

_____
підпис керівника

Демідов Павло
Георгійович

Київ 2021

Київський національний торговельно-економічний університет

Факультет інформаційних технологій
Кафедра комп'ютерних наук та інформаційних систем
Спеціальність 122 «Комп'ютерні науки»

Затверджую
Зав. кафедри _____Пурський О.І.
«18» грудня 2020р.

Завдання
на випускну кваліфікаційну роботу студенту

Іванченко Сергію Олександровичу
(прізвище, ім'я, по батькові)

1. Тема випускної кваліфікаційної роботи
«Розробка web додатку для обліку навчального процесу студентів»
Затверджена наказом ректора від *«15» грудня 2020 р. № 3780*
2. Строк здачі студентом закінченої роботи *31 травня 2021 року*
3. Цільова установка та вихідні дані до роботи
Мета роботи: розробка програмного засобу для обліку навчального процесу студентів
Об'єкт дослідження: процес проектування web додатку для обліку навчального процесу студентів
Предмет дослідження: засоби створення web додатку
4. Перелік графічного матеріалу_____
_____
_____

5. Консультанти по роботі із зазначенням розділів, за якими здійснюється консультування:

Розділ	Консультант (прізвище, ініціали)	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Філімонова Т.О.	22.12.2020 р.	22.12.2020 р.
2	Філімонова Т.О.	22.12.2020 р.	22.12.2020 р.
3	Філімонова Т.О.	22.12.2020 р.	22.12.2020 р.

6. Зміст випускної кваліфікаційної роботи (перелік питань за кожним розділом)

ВСТУП

РОЗДІЛ 1. Сучасні технології в розробці web додатків

1.1.Сутність поняття web-технологій

1.2.Огляд існуючих технологій розробки web додатків

1.3.Огляд сучасних мов для розробки web додатків

РОЗДІЛ 2. Структури веб-сервісу обліку успішності студентів

2.1.Загальна концепція

2.2.Структура front-end частини додатку

2.3.Логіка back-end частини додатку

РОЗДІЛ 3. Розробка програмного продукту

3.1.Розробка бази даних

3.2 Програмна реалізація front-end частини веб-сервісу обліку успішності студентів

3.3 Програмна реалізація back-end частини веб-сервісу обліку успішності студентів

ВИСНОВКИ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

7. Календарний план виконання роботи

№ Пор.	Назва етапів випускної кваліфікаційної роботи	Строк виконання етапів роботи	
		За планом	фактично
1	2	3	4
1	*Вибір теми випускної кваліфікаційної роботи*	*05.10.2020*	*05.10.2020*
2	*Розробка та затвердження завдання на випускну кваліфікаційну роботу*	*18.12.2020*	*18.12.2020*
3	*Вступ*	*03.02.2021*	03.02.2021
4	*РОЗДІЛ 1. Сучасні технології в розробці web додатків*	*26.02.2021*	26.02.2021
5	*РОЗДІЛ 2. Розробка моделі перевірки та оцінювання знань студентів*	*06.04.2021*	06.04.2021
6	*РОЗДІЛ 3. Інформаційна система перевірки та оцінювання знань студентів*	*12.05.2021*	12.05.2021
7	*Висновки*	*14.05.2021*	14.05.2021
8	*Здача випускної кваліфікаційної роботи на кафедру науковому керівнику*	*20.05.2021*	20.05.2021
9	*Попередній захист випускної кваліфікаційної роботи*	*26.05.2021*	
11	*Виправлення зауважень, зовнішнє рецензування випускної кваліфікаційної роботи*	*27.05.2021*	
12	*Представлення готової зшитої випускної кваліфікаційної роботи на кафедру*	*29.05.2021*	
13	*Публічний захист випускної кваліфікаційної роботи*	*За розкладом роботи ЕК*	

8. Дата видачі завдання «22» грудня 2020 р.

9. Керівник випускної кваліфікаційної роботи
Філімонова Т.О.
(прізвище, ініціали, підпис)

10. Гарант освітньої програми Демідов П.Г.
(прізвище, ініціали, підпис)

11. Завдання прийняв до виконання студент-дипломник
Іванченко С.О (
прізвище, ініціали, підпис)

12. Відгук керівника випускної кваліфікаційної роботи

_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____

Керівник випускної кваліфікаційної роботи
_____ 31.05.2021 р.
(підпис, дата)

13. Висновок про випускну кваліфікаційну роботу

_____
_____
_____
_____

Гарант освітньої програми_____Демідов П.Г.
(підпис, прізвище, ініціали)

Завідувач кафедри _____Пурський О.І.
(підпис, прізвище, ініціали)
«_____»_____2021 р.

АНОТАЦІЯ

Випускна кваліфікаційна робота присвячена проектуванню та розробці вебсервісу обліку навчального процесу студентів. Під час виконання бакалаврської дипломної роботи було проаналізовано основні методи та технології розробки web додатків, виявлено їх переваги та недоліки. Було розроблено структури back-end та front-end частин додатку. Було спроектовано базу даних, розроблено адміністративну та публічну частину вебсервісу. Для розробки веб-сервісу використано HTML, CSS, JavaScript, Python та SQLite.

КЛЮЧОВІ СЛОВА: ВЕБ-СЕРВІС, СЕРВЕР, БАЗА ДАННИХ, ФРЕЙМВОРК, ІНТЕРФЕЙС, ЗАПИТ.

ANNOTATION

The bachelor's thesis is devoted to the design and development of a web service for student performance accounting. During the bachelor's thesis the main methods and technologies of web application development were analyzed, their advantages and disadvantages were revealed. The back-end and front-end structures of the application parts have been developed. A database was designed, the administrative and public part of the web service was developed. HTML, CSS, JavaScript, Python and SQLite were used to develop the web service.

KEYWORDS: WEB SERVICE, SERVER, DATABASE, FRAMEWORK, INTERFACE, QUERY.

ЗМІСТ

ВСТУП

Зараз важко уявити собі будь-яку сферу діяльності людини без потреби в інформатизації. Ресторанний бізнес, торгівля, медицина – усі вони активно діджиталізуються. Освіта – галузь, яка потребує процесу інформатизації чи не найбільше.

Застосування інформаційних технологій в освіті допомагає викладачам витрачати менше часу на якісь формальні процеси та більш зосередитися на практичних та дієвих методах передачі знань студентам, також застосування інформаційних технологій в освіті сприяє підвищенню мотивації навчання студентів, економії навчального часу, а інтерактивність і наочність сприяє кращому уявленню, розумінню та засвоєнню навчального матеріалу. Впровадження інформаційних технологій в освітній процес дозволяє значно полегшити організацію безлічі внутрішньовузівських процесів, збір інформації про студентів, викладачів та іншого персоналу, систематизацію та аналіз результатів навчання.

Метою бакалаврської роботи є візуалізація процесів обліку даних, що супроводжують навчальний процес за допомогою використання вебтехнологій.

Об’єктом дослідження є веб-технології та процеси візуалізації та обробки даних в мережі Інтернет.

Методами дослідження є огляд існуючих web-технологій, можливостей та методів розробки, аналіз сучасних мов та технологій розробки. А саме:

- Системний метод
- Метод теорії бази данних
- Метод програмування

Предметом бакалаврської роботи є засоби та методи розробки веб-ресурсів та критерії їх варіантного аналізу.

Для досягнення поставленої мети необхідно вирішити такі **завдання**:

- Створити системи адміністрування та публічну частину веб-сервісу
- Розробити доступний та зручний інтерфейс, як для відвідувача сайту, так і для адміністратора
- Створити базу данних, у якій буде розміщуватися вся необхідна інформація

Актуальність даної дипломної роботи полягає в тому, що обробка, аналіз та збереження даних успішності студента, в закладах вищої освіти, відбуваються в різних не зв’язаних системно місцях.

РОЗДІЛ 1
Сучасні технології в розробці web додатків

1.1 Сутність поняття web-технологій

Веб-технологія – це сукупність методів та програмно-технічних засобів, інтегрованих з метою ефективного опрацювання веб-ресурсів, які знаходяться у веб-просторі (локальному або глобальному, наприклад, мережі Інтернет).

Отже, поняття веб-технології пов'язане з використанням веб-простору – WWW (англ. World Wide Web) — глобальний інформаційний простір, заснований на фізичній інфраструктурі Інтернету і протоколі передачі даних HTTP.

Веб-технології дають відповідь на питання:

Якими засобами можна опрацьовувати різні види інформації – текстову, графічну, звукові та іншу, що знаходиться в мережі?

Якби це питання стосувалось інформації на комп'ютері користувача (що не має підключення до мережі), то можна використовувати текстові, графічні, звукові, редактори. А для здійснення цих операцій в мережі достатньо використовувати браузер і набір відповідних веб-технологій, які функціонують в мережі на різних серверах.

Наприклад, для того, щоб ввести текст електронного листа не потрібно використовувати текстовий редактор, що встановлено на комп'ютері, адже в інтерфейсі електронної пошти є вбудований редактор, який дає змогу здійснювати набір тексту в браузері. Така технологія називається *веб-технологією*. Наприклад, всім відома соціальна мережа Facebook використовує різні веб-технології, які дають змогу завантажувати і переглядати фотографії, музичні файли, відео тощо.

Система Google пропонує сервіс Google-docs, який має текстовий веб-редактор для створення документів у форматі .doc, веб-редактор презентацій та електронних таблиць. Виходячи з цього, людині не потрібно

використовувати програмне забезпечення свого комп’ютера, достатньо відкрити веб-оглядач, ввести адресу системи Google та створити потрібний документ.

Будь-яку веб-технологію можна реалізувати засобами мережі – чи глобальної, чи локальної, в межах однієї аудиторії чи цілого корпусу. Розглянемо основні принципи функціонування мережі Інтернет.

Всесвітню мережу утворюють мільйони веб-серверів мережі Інтернет, розташованих по всьому світу.

Веб-сервер є програмою, що запускається на підключеному до мережі комп'ютері і використовує протокол HTTP для передачі даних.

У простому вигляді така програма отримує по мережі HTTP-запит на певний ресурс, знаходить відповідний файл на локальному жорсткому диску і відправляє його по мережі до того комп'ютера, який робив запит. Складніші веб-сервери здатні динамічно формувати ресурси у відповідь на HTTP-запит. Приклад такого веб-сервера можна назвати будь-яку пошукову систему, яка формує список веб-ресурсів на запит користувача.

В цілому можна зробити висновок, що всесвітня павутина (WWW) будується на основі великої кількості різноманітних технологій, які забезпечують ту чи іншу функцію в мережі Інтернет. Наприклад, для створення динамічних сторінок використовуються мову PHP, для поліпшення візуального сприйняття веб-простору – технологія CSS, яка дозволяє задавати єдині стилі оформлення для багатьох веб-сторінок одразу, тощо.[1]

1.2 Огляд існуючих технологій розробки web додатків

Першим і особливо важливим для великих і повнофункціональних сайтів рівнем є рівень сервера. Повністю невидимий для кінцевого користувача даний рівень являє собою ядро всього сайту. Типовими і особливо поширеними засобами для програмної реалізації цієї частини сайту

є мова програмування PHP і система управління базами даних MySQL. Мова PHP є скрипковою мовою програмування. Конкуренцію даному інструменту можуть скласти такі мови програмування як Python (включаючи фреймворки Django, TurboGears і web2py), Ruby (включаючи фреймворк Ruby on Rails), програмна платформа Node.js і технологія ASP.NET. При більш уважному розгляді можна помітити, що мови програмування PHP, Python і Ruby є мовами програмування загального призначення, а для роботи вимагають наявності інтерпретатора. У той же час, проект Node.js включає в себе цілу програмну платформу, в основі якої лежить рушій V8, що дозволяє використовувати для розробки серверної частини веб-додатків мову JavaScript, що перетворює дану вузькоспеціалізовану мову в мову загального призначення. Технологія ASP.NET, в свою чергу, включає в себе цілий набір засобів для реалізації вебдодатків і веб-сервісів та дозволяє використовувати для розробки будь-яку мову програмування з доступних на платформі .NET.

Одним із завдань функціонування серверної частини веб-додатки є підтримка веб-інтерфейсу додатку, що зводиться, в кінцевому рахунку, до динамічної генерації html-сторінок. Ні для кого не секрет, що мова HTML не є мовою програмування, а служить виключно для розмітки документа, а застосовується зазвичай в зв'язці з мовою опису зовнішнього вигляду CSS, що дозволяє забезпечити html-сторінці будь-який дизайн. Тому постає запитання, які існують засоби для створення інтерактивних веб-додатків. Відповіддю на це питання є огляд інструментів для реалізації клієнтської частини веб-додатки, представлений далі. Так званим монополістом в області розробки клієнтської частини веб додатків є мова JavaScript, створена з єдиною метою - надати статичним сторінкам інтерактивності. Варто відзначити, що завдяки вище згаданому проекту Node.js, існує і активно використовується можливість написання обох частин веб-програми (і серверної, і клієнтської) з використанням однієї мови програмування - JavaScript. Про плюси і мінуси даного мови сказано досить багато; саме наявність недоліків в JavaScript часто і призводить до створення альтернатив.

Важливо розуміти, що більшість мов, описуваних далі застосовують принцип трансляції коду в JavaScript. Найвідомішою альтернативою мови JavaScript є мова CoffeeScript. Основною метою його створення є спрощення використання JavaScript. Для отримання кінцевого коду, CoffeeScript використовує власний транслятор, що входить до переліку утиліт проекту Node.js. Варто відзначити, що синтаксис цієї мови насамперед відрізняється від кінцевого коду на JavaScript своєю стислістю і використанням досить оригінальних синтаксичних конструкцій (наприклад застосуванням постумови if). Для зберігання контенту сайту найкраще підходить база даних. З точки зору рушія база даних являє собою набір таблиць. Кожна таблиця - це сутність, в якій зберігаються однотипні дані. База даних має величезну кількість плюсів. По-перше, просте і швидке управління даними. Будь-яка сучасна база даних підтримує мову запитів SQL, за допомогою якого здійснюється вибірка, додавання, видалення і зміна даних в базі. По-друге, організація логічного зв'язку даних. Маючи логічний зв'язок між таблицями статей і авторів, ми можемо, наприклад, з легкістю дізнатися, скільки статей має конкретний автор. З використанням бази даних з легкістю вирішуються такі завдання як пошук по сайту, розбиття на сторінки, реєстрація і авторизація користувачів. З усіх цих плюсів слід, що база даних невід'ємна частина Web-сайтів, яка дозволяє швидко орієнтуватися по сайту і відбирати корисну інформацію.

Підводячи підсумок, слід зазначити, що розробка веб-додатків або вебсайтів часом є досить трудомістким завданням, у вирішенні якої задіяними виявляються цілі команди розробників. Можна сказати, що розробка вебпроектів є приклад розробки інформаційної системи. Разом зі зростаючим рівнем складності таких проектів, зростає і необхідність у виборі правильних інструментів для програмної реалізації. Представлений вище короткий огляд сучасних засобів розробки веб-додатків не претендує на право зватися вичерпним і є, в першу чергу, спробою формалізувати перелік найвідоміших і широко застосовуються в даній сфері розробки

інструментальних засобів.[2]

1.3 Огляд сучасних мов для розробки web додатків

HTML — стандартна мова розмітки документів для Web, де всі Web сторінки створюються за допомогою HTML (або XHTML). Мова HTML інтерпретується браузером у вигляді документу, зручному для людини. HTML створювався в 1991-1992 роках як мова для обміну науковою та технічною документацією, яка зручна для людей, що не є спеціалістами з верстки.[3] Вона успішно мінімізує проблеми зі складністю SGML шляхом визначення невеликої кількості структурних та семантичних елементів (які розмічаються тегами), які використовуються для створення простих, але гарно оформлених документів. Також, крім спрощення структури документу, у HTML міститься підтримка гіпертексту.[4] Мультимедійні можливості були додані пізніше. Текстові документи, які містять код на мові HTML, обробляються спеціальними програмами, які відображають документ у форматованому вигляді. Такі програми, що називаються браузерами, забезпечують зручний графічний інтерфейс для взаємодії користувача із сервером — запит Webсторінок, їх відображення та відправлення введених користувачем даних на сервер. Від початку HTML був спроектований і створений як засіб структурування та форматування документів, без їх прив'язки до засобів відображення. Але сучасні застосування HTML далекі від його початкових задач — додані мультимедійні можливості, з'явилися засоби для створення складних графічних оформлень, додана можливості підключення плагінів та розширень. Для створення динамічних сторінок було розроблений цілий ряд технологій — JavaScript, Java Апплети, Adobe Flash, Microsoft Silverlight. Реалізації деяких з них інтегровані в браузери (JavaScript), для роботи з іншими потрібно підключати спеціальні плагіни (доступні безкоштовно на Web-сайтах розробників або поставляються разом з операційними системами чи браузерами). В середині 90х років розгорнулася боротьба між розробниками найбільш популярних (на той час)

браузерів — Netscape Navigator та Microsoft Internet Explorer за ринок інтернет-браузерів. Основний спосіб боротьби — розробка та впровадження нових технологій, що були не сумісні з іншими браузерами. В результаті навіть на сьогоднішній день не вдалося досягти повної сумісності між усіма браузерами, хоча їх розробники та консорціум W3C, який займається стандартизацією Web-технологій, докладають максимум зусиль для цього. [5] З іншого боку, в результаті цієї боротьби, з'явився ряд технологій, що займають ключову роль в розвитку сучасного Web, серед них — **JavaScript.**

JavaScript.

Як правило, ті хто приходять в веб-розробку та встигають швидко освоїти розмітку і опис зовнішнього вигляду, вирішують, що прийшла черга освоїти сценарії. Для машини клієнта, вони створюються на JavaScript і похідних. Характерна стандартизація ECMAScript, завдяки якій застосування стає легше.

Сьогодні з JavaScript можуть працювати навіть ті, хто не має навичок програмування. Довгий досвід роботи веб-розробників по всьому світу дозволив створити бібліотеки з готовими рішеннями: кліками, переходами, інтерактивними меню.

Однією з основних переваг JavaScript розробники називають виконання сценарію в браузері користувача. Таким чином знижується навантаження на сервер, і заощаджуються ресурси хоста.

Області використання мови досить великі:

- Створення веб-сторінок, які можуть змінюватися після завантаження документа
- Рішення локальних завдань
- Перевірка грамотності заповнення форм користувачем до їх пересилання на сервер

Різноманіття можливостей javascript обумовлює популярність мови. З

його допомогою можна:

- Вносити зміни на сторінку: працювати з тегами, міняти стилі, писати текст
- Реагувати на події (наприклад, клік миші) і виконувати певну функцію
- Виводити повідомлення, перевіряти коректність даних
- Завантажувати дані без перезавантаження сторінки і т.д.

JavaScript вже багато років займає лідируючі позиції по використовуванню. Незважаючи на «заточеність» під веб-додатки, його застосовують і в інших сферах. Наприклад, робота банківської системи, яка реалізована на JS, відрізняється швидкістю і стабільною роботою.[6]

PHP.

В основі лежить мова розмітки HTML. PHP - це мова сценаріїв загального призначення, вихідний код - відкритий. Синтаксис досить легко піддається освоєнню, має чимало спільних рис з C, Java і Perl. Головна перевага PHP полягає в тому, що з його допомогою розробники можуть оперативно створювати динамічно генеруємі веб-сторінки. При професійному володінні мовою, PHP можна використовувати і для виконання інших завдань.

PHP інтенсивно використовується в створенні веб-додатків, коли потрібно розвантажити призначений для користувача девайс, адже «гіпертекстовий процесор» працює на серверній стороні. Програма, написана на PHP, віддає користувачеві лише готовий результат. Також, до переваг PHP слід віднести безперервний розвиток. Остання версія PHP 7: 7.4 вийшла в червні 2019 року.[7]

Python.

Python - яскравий приклад мови програмування загального призначення, який здобув популярність серед веб-розробників. Все тому, що люди це, як правило, творчі. Веб-майстер шукає засіб, який допоможе йому створити якісний продукт з найменшими витратами.

Python не був настільки відомим, поки їм не зацікавилася компанія Google, яка і профінансувала подальший розвиток і популярність. Сьогодні безліч веб-сайтів створені на основі Python, наприклад, Instagram - одна з найпопулярніших соціальних мереж. У той же час, додатки для Android і iOS на ньому пишуть рідко. В цілому, це можливо при підключенні спеціальних бібліотек, але готовий продукт виходить важким і повільним. Разом зі стрімким розвитком продуктивності мобільних пристроїв, очікується, що Python отримає ще більшу популярність.[7]

Існує реалізація, яка адаптована спеціально під розробку веб-додатків. Так як сам по собі CPython (основна реалізація) досить повільний у виконанні, було придумано кілька легших варіантів:

Brython - написаний на JavaScript. Створений для того, щоб писати сценарії для браузера на Python. Вбудовувати так само легко, як і JavaScript: «type =" text / python »»;

QPython - варіант для Android. Поки що не тестується, але основними бібліотеками вже наділений. Це варіант для тих, хто хоче створювати Android-додатки, уникаючи тієї повільності, яка характерна для оригінальної реалізації.

Висновки

У результаті аналізу предметної області було проаналізовано, що в сучасному світі існує велика потреба в інтегруванні веб-технологій в кожну сферу життя людини. Також було виявлено, що засобів та методів реалізації існує безліч.

Було проаналізовано і проведено порівняльний аналіз засобів та мов розробки, за допомогою яких можна реалізувати веб-додаток для обліку успішності студентів.

Враховуючи всі недоліки та переваги проаналізованих аналогів, було визначено основні задачі розробки.

Проаналізовано основні методи розв’язання поставленої задачі. Було прийнято рішення обрати метод ручного написання веб-сайтів, використовуючи HTML, CSS, Java Script, Python, так як у розробці необхідно реалізувати функції, для вирішення яких найкраще підходить саме даний набір технологій.

РОЗДІЛ 2
Структури веб-сервісу обліку успішності студентів

2.1 Загальна концепція

Якщо розглядати майбутній продукт з логічної точки зору, то він повинен надавати можливість викладачу в більш зручному та швидкому режимі виставляти бали студентів, а останнім слідкувати за своєю успішністю з усіх дисциплін в одному місці.

Можемо зобразити структуру таким чином (рис. 2.1):

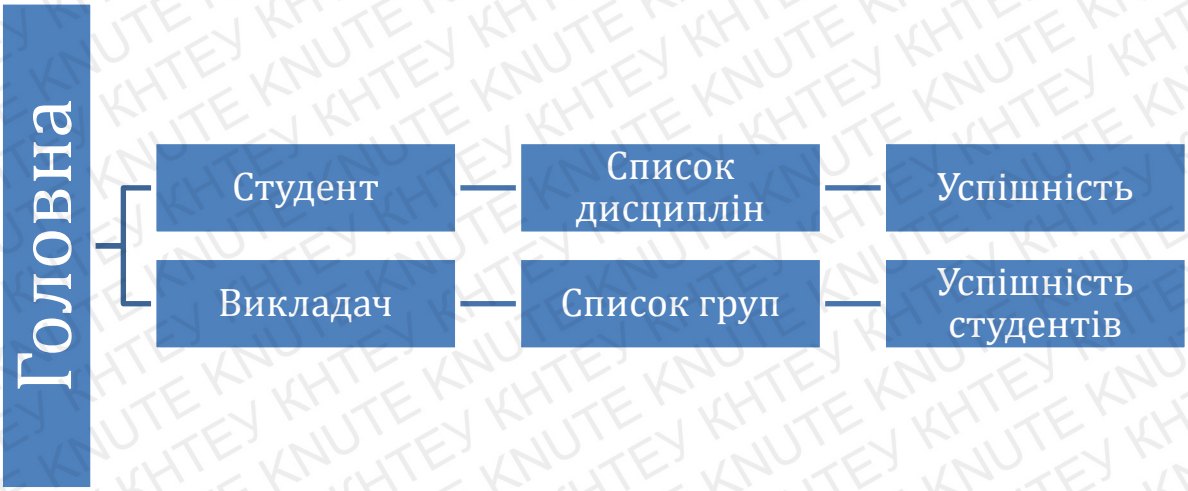

Рис. 2.1 Структура взаємодії користувачів і матеріалів

Тобто, на головній сторінці повинен міститися вибір студент-викладач, далі форма входу на сайт.

При введенні валідних даних студент повинен побачити перед собою список: список дисциплін з подальшим переходом на сторінку успішності.

При введенні валідних даних викладач повинен побачити вибір груп в

яких він проводить заняття.

В кінцевому результаті продукт має представляти собою веб-додаток, у якому викладач може:

- Виставляти оцінки за різні види робіт
- Виставляти модульні бали
- Виставляти екзаменаційні/підсумкові бали

Студент використовуючи цей додаток має можливість:

- Бачити свої бали за певні види робіт
- Моніторити свою успішність протягом періоду навчання

2.2 Структура front-end частини додатку

Структура інтерфейсу веб-сервісу використовується при розробці його дизайну. Структура інтерфейсу показує найбільш важливі елементи інтерфейсу користувача, їх положення і взаємозв'язок між сторінками сайту. Структуру інтерфейсу сторінок сайту зображають в чорно-білому кольорі найбільш важливі елементи інтерфейсу, такі як заголовок і нижній колонтитул сайту, форма контактів, навігація тощо.

Розробка структури інтерфейсу веб-сайту – це процес, який може значно скоротити час, необхідний для проектування та розробки, усуваючи потенційні візуальні відволікаючі фактори і фокусуючи увагу розробників проекту на базовій функціональності.

Інтерфейс веб-сайту – сукупність способів і засобів, за допомогою яких користувач взаємодіє з будь-якою веб-сторінкою. Макет сайту – це схема сторінок, на якій розташовуються графічні і текстові елементи. Іншими словами, макет сайту це каркас, на якому формується дизайн і здійснюється наповнення сторінок.

Розробка інтерфейсу сайту – це створення не тільки дизайну сторінок, але і її структури. Сюди можна віднести панель навігації, блоки, розділи, розмітку сторінки, шапку сайту тощо. Іншими словами всі графічні та текстові елементи, які присутні на сторінках веб-сервісу.

Основними перевагами розробки структури інтерфейсу перед створенням сайту:

- раціональний розподіл часу і зосередження саме на тому, для чого призначена кожна сторінка;
- видалення зайвих елементів, які можуть виявитися непотрібними для майбутнього веб-сайту;
- зниження ймовірності збільшення обсягу роботи по розробці дизайну;
- отримання чіткого уявлення про те, що потрібно зробити дизайнеру.

Вибір колірної гамми відіграє важливу роль при створенні сайту. Від неї безпосередньо залежить його сприйняття відвідувачами – сприйняття сайту в цілому, психологічний і фізіологічний стан людини, зручність читання тощо.[8] Тому, при створенні сайту дуже важливо враховувати:

– особливості фірмового стилю, на підставі якого розробляється сайт;

– фізіологічні та психологічні особливості сприйняття колірної гамми користувачами.

Колірна гамма повинна бути побудована так, щоб не порушувати основні закономірності впливу кольорів на психологічну реакцію організму.[9] Вона має відповідати стійким гармонійним поєднанням кольорів. Існують певні закономірності для вибору колірної гами:

– кольори повинні відповідати фірмовому стилю компанії;

– комбінація кольорів має відповідати цільовій аудиторії даного сайту;

– не варто використовувати велику кількість кольорів;

– кольори повинні гармоніювати між собою;

– потрібно забезпечити контраст між фоном і звичайним текстом

Для розробки веб-сервісу обліку успішності студентів було обрано два основних кольори: фіолетовий та блакитний (рис. 2.2).

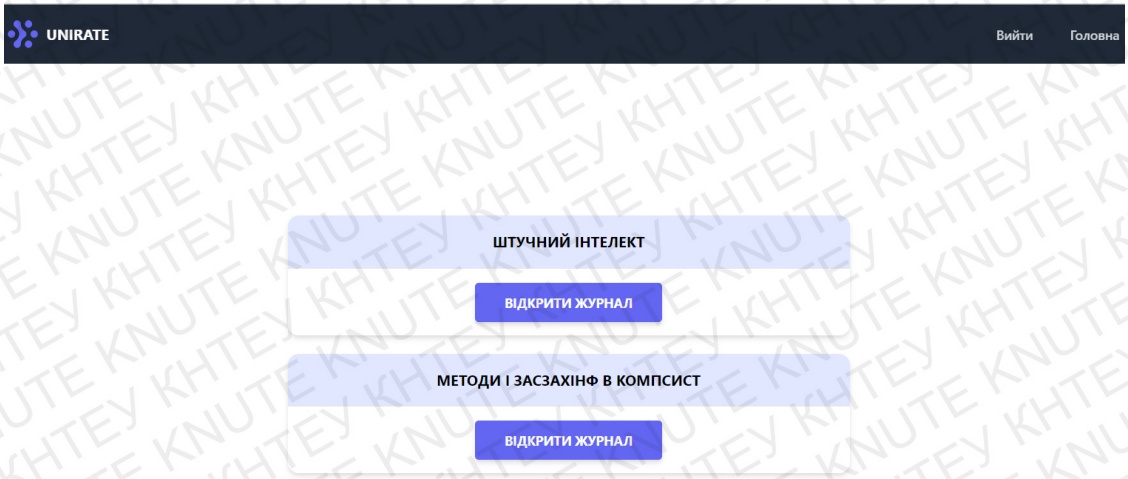

Рис. 2.2 Головна сторінка сайту при вході студента

Фіолетовий колір символізує вищий розум, мудрість, зрілість, сприяє натхненню.

Блакитний – один з найпопулярніших кольорів, що використовується в Інтернеті. Відноситься до категорії «безпечних» кольорів, який подобається більшості відвідувачів. Блакитний колір асоціюється з такими поняттями, як мир, спокій, надійність, довіра, чесність, чистота, ясність.

Рис. 2.3 Вигляд журнала від залогіненого студента

При виборі одного з предметів переходимо на сторінку з назвою предмета, списку студентів групи, та балів по датах їх виставлення, модуля, екзамена (рис. 2.3.).

Зверху сторінки бачимо кнопки Вийти (розлогінитися) та Головна (перехід на сторінку з вибором предметів).

Структура фронт-енд частини додатку можна зобразити діаграмою(рис. 2.4).

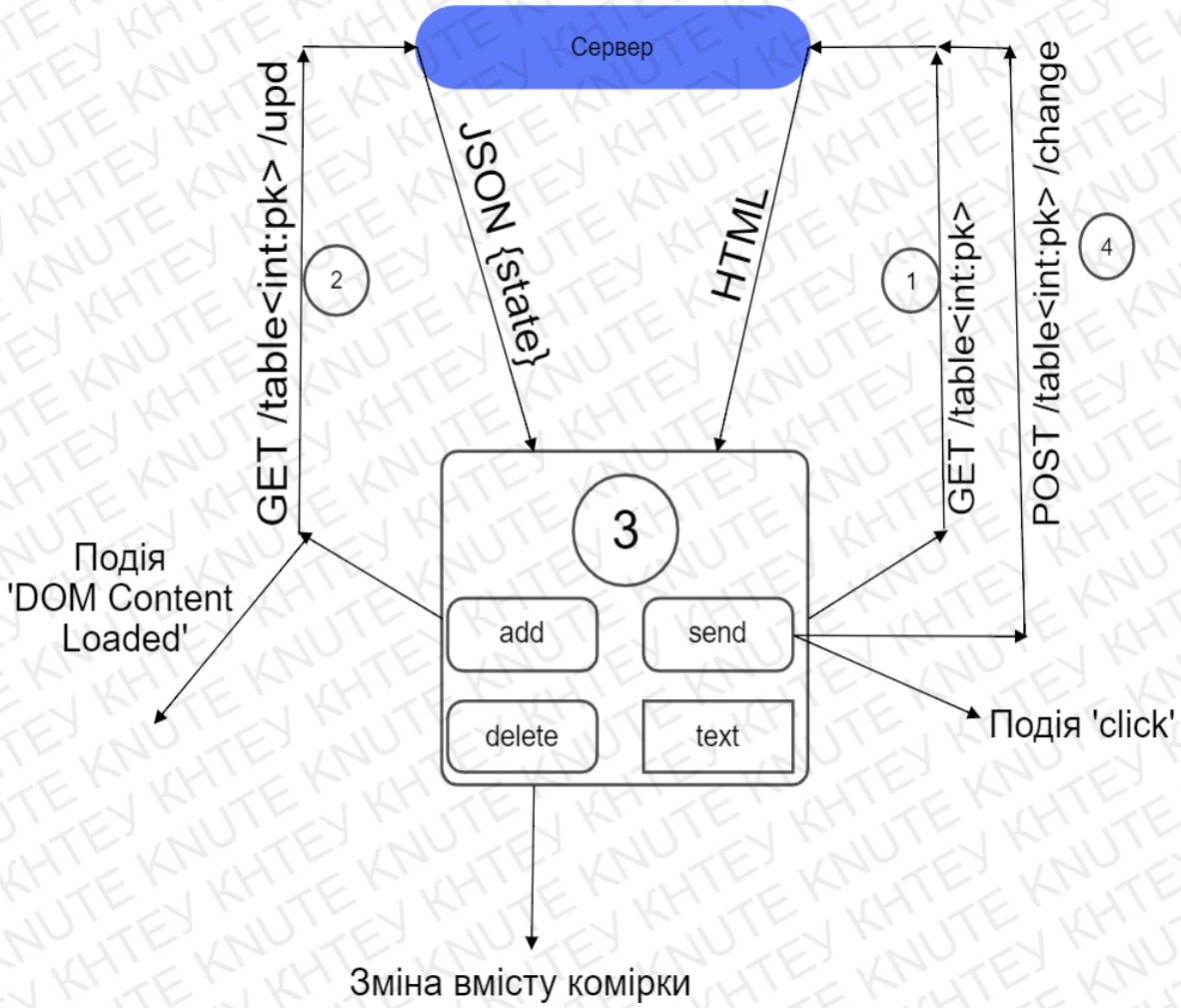

Рис. 2.4 Структура front-end

2.3 Логіка back-end частини додатку

У роботі для створення серверної частини використана мова Python, фреймворк Django 3.2.2. Логічні зв’язки між даними зображено на рис. 2.5.

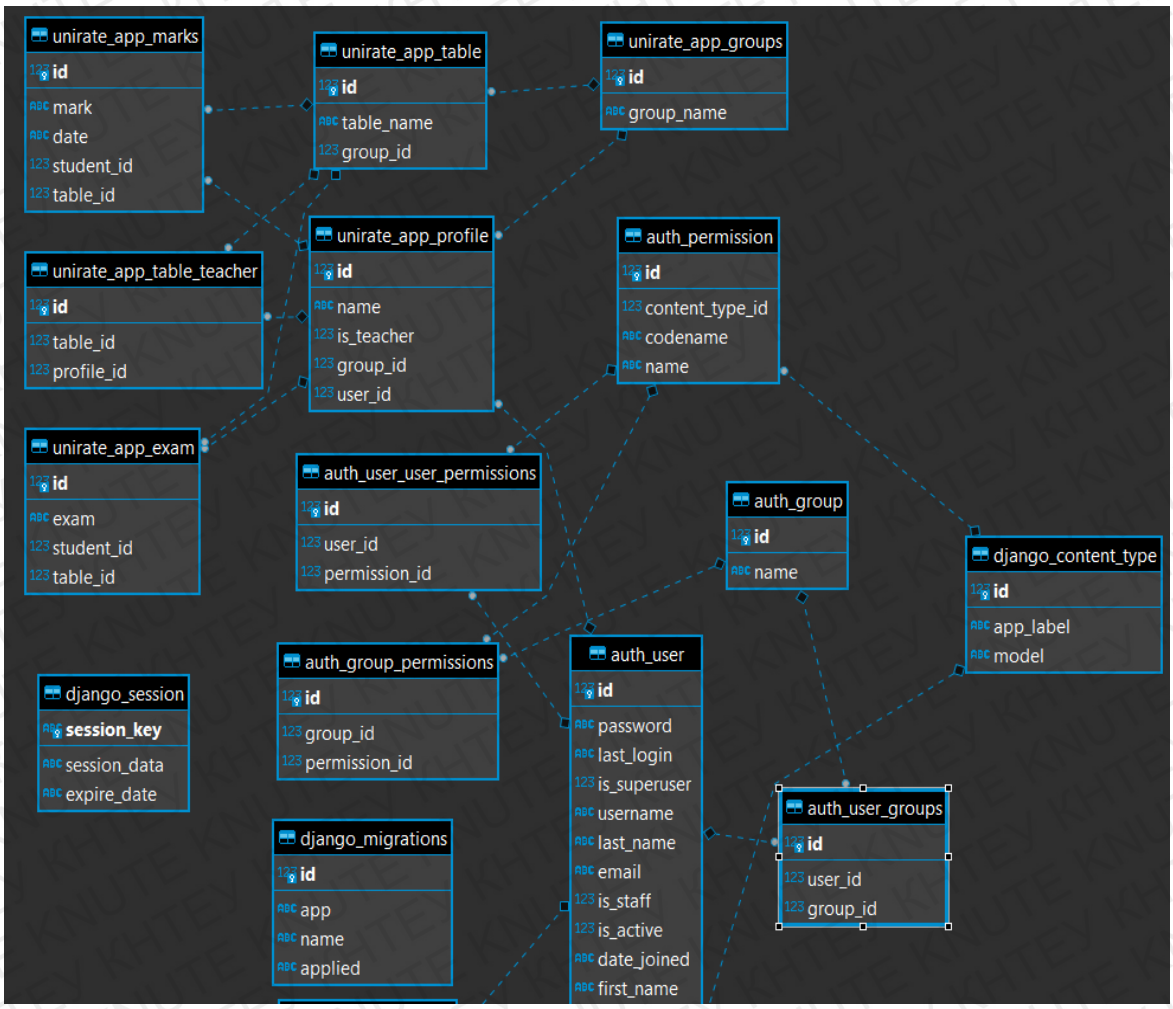

Рис. 2.5 Логічні зв’язки в базі даних

Основні переваги мови Python описані в розділі 1.

Django - фреймворк для веб-додатків на мові Python. Один з основних принципів фреймворка - DRY (don’t repeat yourself). Веб-системи на Django будуються з одного або декількох додатків, які рекомендується робити відчужуваними.[10] Це одне з помітних архітектурних відмінностей цього фреймворка від деяких інших (наприклад, Ruby on Rails). Також, на відміну від багатьох інших фреймворків, обробники URL в Django конфігуруються явно (за допомогою регулярних виразів), а не автоматично задаються зі структури контролерів.

Django проектувався для роботи під управлінням Apache (з модулем mod_python) і з використанням PostgreSQL в якості бази даних. В даний час, крім PostgreSQL, Django може працювати з іншими СУБД: MySQL

(MariaDB), SQLite, Microsoft SQL Server, DB2, Firebird, SQL Anywhere і Oracle.

У нашому випадку використовуємо SQLite.

SQLite — полегшена реляційна система керування базами даних.

Особливістю SQLite є те, що вона не використовує парадигму клієнт-сервер, тобто рушій SQLite не є окремим процесом, з яким взаємодіє застосунок, а надає бібліотеку, з якою програма компілюється і рушій стає складовою частиною програми. Таким чином, як протокол обміну використовуються виклики функцій (API) бібліотеки SQLite. Такий підхід зменшує накладні витрати, час відгуку і спрощує програму. SQLite зберігає всю базу даних (включаючи визначення, таблиці, індекси і дані) в єдиному стандартному файлі на тому комп'ютері, на якому виконується застосунок. Простота реалізації досягається за рахунок того, що перед початком виконання транзакції весь файл, що зберігає базу даних, блокується; ACID-функції досягаються зокрема за рахунок створення файлу-журналу.[11]

Кілька процесів або потоків можуть одночасно без жодних проблем читати дані з однієї бази. Запис в базу можна здійснити тільки в тому випадку, коли жодних інших запитів у цей час не обслуговується; інакше спроба запису закінчується невдачею, і в програму повертається код помилки. Іншим варіантом розвитку подій є автоматичне повторення спроб запису протягом заданого інтервалу часу.

У комплекті постачання йде також функціональна клієнтська частина у вигляді виконуваного файлу sqlite3, за допомогою якого демонструється реалізація функцій основної бібліотеки. Клієнтська частина працює з командного рядка, і дозволяє звертатися до файлу БД на основі типових функцій ОС.[11]

Висновки

В ході розробки другого розділу було спроектовано логічну та фізичну

структури веб-сайту. Також розроблено структуру інтерфейсу веб-сервісу, що зображає найбільш важливі елементи інтерфейсу користувача, їх положення і взаємозв'язок між сторінками сайту.

Спроектовано структурну схему бази даних. Логічна схема бази даних, що включає таблиці і зв'язки між ними, є основою оптимізації реляційної бази даних. Схема бази даних може забезпечити фундамент для оптимальної продуктивності бази даних.

Дизайн веб-сервісу розроблено з урахуванням тематики і загальної концепції проекту.Колірна гамма підібрана, враховуючи особливості фірмового стилю та фізіологічні та психологічні особливості сприйняття кольорів користувачами. В якості основних кольорів використано синій (#008cf0) та фіолетовий (#333333).

РОЗДІЛ 3
Розробка програмного продукту

3.1 Розробка бази даних

На основі проведеного у першому розділі аналізу методів розв'язання поставленої задачі було вибрано ручний метод написання веб-сайтів.

На сьогоднішній день у розробників існує широкий вибір щодо того, яку мову чи технологію використовувати для створення сайту. У нашому випадку для back-end частини використано мову Python та фреймворк Django.

Структура сайту складається з двох директорій, файлу manage.py, через який здійснюється все виконання команд Django та бази даних sqlite.

Перша директорія – unirate. Це коренева директорія в якій містяться 5 файлів, серед яких основні:

1. Settings.py – файл, в якому знаходяться всі підключення налаштувань а також підключаються всі директорії з логікою сайту
2. Urls.py – початок всього руху запитів. За замовчуванням тут підключений лише шлях до нашої адмін панелі, але ми можемо додати початок шляху до кожного нашого додатку. Так ми даємо знати Django, до якого додатку звертатися при запитах.

Друга директорія – unirate_app – це наш перший і єдиний застосунок. Тут знаходяться такі основні файли:

1. Models.py – файл в якому ми прописуємо всю логіку для нашої бази даних, а саме, створюємо таблиці та поля, де зберігатимемо всю інформацію
2. Urls.py – файл-продовження подібного файлу в першій директорії, але тепер він після адресу запиту містить посилення на кінцеві точки обробки запитів у файлі views.py.
3. View.py – файл, в якому прописана вся логіка обробки кожного із діючих адресів запитів.

Функції в цьому файлі поділені на 3 частини. Перша частина містить 3 функції для роботи авторизації: відрисовування сторінки, авторизація користувача та функція виходу з аккаунта. Друга частина містить лише одну функцію відрисовування сторінки з вибором груп для викладача та предметів для студентів. Також ця функція для отримання потрібних даних відправляє запити в базу даних. Третя частина містить 4 функції: перша – отримує всі дані з бази даних та пакує їх в словник, щоб потім передати іншим функція, друга – відрисовує таблицю, користуючись словником з першої, а третя відправляє даний словник у форматі json для подальшої роботи з ним на фронт-енді, остання, четверта функція потрібна для внесення змін з таблиці в базу даних.

Також в другій директорії міститься дві папки: template та static. В папці з темплейтами містяться всі html файли, які підключаються як модулі до файлу base.html. В цих файлах дані розміщуються на потрібних місцях за допомогою вбудованої в Django бібліотеки jinja. В папці статиків містять всі статичні файли: картинки, css, js.

3.2 Програмна реалізація frond-end частини веб-сервісу обліку успішності студентів

У даному розділі більш детально розглянемо програмну реалізацію додатку.

Обробник на кореневий елемент перевіряє чи завантажився весь html сторінки.

Якщо завантажився, спрацьовує функція, яка передана другим елементом.

Далі в змінні записуються переходи на потрібні вузли дом-дерева, тобто на елементи, окремо залишаємо переходи на логін, таблицю, лоадери і підказку через querySelector і ID елементів (рис.3.1)

```
document.addEventListener('DOMContentLoaded', () => {
    const loginForm = document.getElementById('login-form');
    const editableTable = document.getElementById('editableTable');
    const loader = document.getElementById('loader');
    const tip = document.querySelector('.js-tip');
```

Рис. 3.1 Функція переходів на потрібні вузли

Далі створюємо функцію для роботи мобільного меню.

```
(function(){
    const headerToggler = document.getElementById('mobile-toggler');
    const headerMenu = document.getElementById('mobile-menu');
    const headerClose = document.getElementById('close-icon');
    const headerOpen = document.getElementById('open-icon');
```

Створюємо переключення класів елементів. Якщо елемент був схований – показати, і навпаки.

```
const toggleHidden = (...elements) => {
    elements.forEach(element => {
      element.classList.toggle('hidden');
      element.classList.toggle('block');
    });
    }
```

Якщо на сторінці є форма авторизації, то отримуємо дані з форми. Задаємо метод запиту, режим запиту, заголовки запиту, перетворюємо об’єкт з даними форми на JSON. Показуємо лоадер, відправляємо запит. Коли запит отриманий, перетворюємо його на звичайний об’єкт. Приховуємо лоадер. Якщо все добре – переправляємо користувача на список таблиць, інакше показуємо помилку з текстом 'Неправильний логін або пароль!'.

```
if(loginForm){
    (function(){
      loginForm.addEventListener('submit', (event) => {
        event.preventDefault();
```

```
            const formData = {
                email: event.currentTarget
                    .querySelector('#email-address').value,
                password: event.currentTarget
                    .querySelector('#password').value
            };
            const reqParams = {
                credentials: 'include',
                method: 'POST',
                mode: 'same-origin',
                headers: { 'Content-Type': 'application/json' },
                body: JSON.stringify(formData)
            };

            loader.classList.remove('hidden');
            fetch('/auth/', reqParams).then(data => data.json())
            .then(json => {
                loader.classList.add('hidden');
                if(json) {
                    document.location.href = '/'
                }else {
                    alert('Неправильний логін або пароль!');
                }
            });
        });
    })();
}
```

Якщо на сторінці є таблиця для викладача, функція відправляє запит на отримання state - об'єкта з даними всієї таблиці, його ми будемо оновлювати, та змінювати в ході взаємодії користувача з таблицею. Задаємо параметри запиту. Якщо запит потребує відправки тіла запиту, додаємо тіло. Повертаємо отриману від сервера відповідь, показуємо лоадер, повертаємо наш state.

```
if(editableTable) {
    (function(){
        const sendReq = (url, method, body = false) => {
            const reqParams = {
```

```
        method: method,
        headers: {
          'Content-Type': 'application/json'
        },
      };

      if(body){
        reqParams.body = body;
      }

      return fetch(url, reqParams)
        .then(data => data.json()).then(json => {
          loader.classList.add('hidden');
          return json;
        });
    }
```

Отримуємо ID таблиці з пошукового рядка(Рис.3.2.)

```
const tableId = window.location.pathname.split('/')[2];
```

Рис.3.2 Отримання ID таблиці

Відправляємо запит (Рис.3.3)

```
sendReq(`/table/${tableId}/upd`, 'GET')
```

Рис.3.3. Відправлення запиту

Коли запит виконується, переходимо до наступного блоку, де є кнопка додати стовпчик, відправити, видалити стовпчик, поле для вводу заголовка стовпчика і лічильник для назви нового стовпчика.

```
const editableCells = document.querySelectorAll('.js-editable-cell');
const addButton = document.getElementById('addButton');
const submitButton = document.getElementById('submitButton');
const deleteButton = document.getElementById('deleteButton');
const deleteInput = document.getElementById('deleteInput');
let cnt = 1;
```

Далі функція, що додає обробники для клітинок таблиці, приймає в себе список вузлів з клітинками, обробник кліку по клітинці, обробник зміни фокусу, обробник натискання на клавіші.

```
const addCellListeners = (nodeList) => {
        nodeList.forEach(cell => {
          const input = cell.querySelector('input');

          cell.addEventListener('click', (e) => {
            cellClick(cell);
            refreshTip(cell);
          });

          input.addEventListener('blur', (e) => {
            inputBlur(cell);
            clearTip();
          });

          input.addEventListener('keyup', (e) => {
            if(e.key === 'Enter') {
              input.blur();
            }
          });
        });
    };
```

Додаємо логіку кліку по картинці

```
const cellClick = (cell) => {
        const textContainer = cell.querySelector('span');
        const input = cell.querySelector('input');

        input.classList.remove('hidden');
        textContainer.classList.add('hidden');
        input.focus();
    };
```

Додаємо утилітарну функцію для отримання номера стовпчика.

```
const getCol = (cell) => {
        const tableRow = cell.parentElement;
        return [...tableRow.children].indexOf(cell) - 1;
    };
```

Додаємо утилітарну функцію для отримання студента за ім’ям

```
const getStudent = (rowHeader) => {
        const name = rowHeader.getAttribute('data-name');
        return state.students.find(
            student => student.name === name
        );
    };
```

Додаємо утилітарну функцію для підрахування модуля, де ми перебираємо оцінки, перетворюємо рядок на число. Якщо спробувати перетворити на число рядок, що містить не лише числа, ми отримаємо NaN, отже, якщо не NaN – сумуємо. Повертаємо модуль.

```
const countModule = (marks) => {
        let sum = 0, mark = null;
        for(let i = 0; i < marks.length; i++) {
            mark = +marks[i];

            if(!isNaN(mark)) {
                sum += mark;
            }
        }
        return sum;
    };
```

Створюємо логіку роботи для поля вводу, де маємо поточне значення, тип клітинки(екзамен, дата, просто оцінка), клітинку з ім’ям студента, об’єкт даних студента. Залежно від типу клітинки, якщо дані в полі змінилися, оновлюємо оцінки студента, екзаменаційну оцінку студента, або заголовок/дату. Потім ховаємо поле вводу.

```
const inputBlur = (cell) => {
        const input = cell.querySelector('input');
```

```
            const currentValue = cell.getAttribute('data-val');
            const type = cell.getAttribute('data-type');
            const rowHeader = cell.parentElement.children[0];
            const student = getStudent(rowHeader);

            if(currentValue !== input.value) {
               const col = getCol(cell);
               switch(type) {
                  case 'marks':
                     student.marks[col] = input.value;
                     student.module = countModule(student.marks);
                     refreshBody();
                  break;
                  case 'exam':
                     student.exam = input.value;
                     refreshBody();
                  break;
                  case 'lessons':
                     state.lessons[col] = input.value;
                     refreshHead();
                  break;
                  default:break;
               }
            }
input.classList.add('hidden');
         };
```

Створюємо функції для оновлення верхівки та тіла таблиці(Рис.3.4.)

```
const refreshHead = () => {
                  const root = document.getElementById('tableHead');
                  const lessons = state.lessons;

                  let html = `<tr class="flex w-auto"><th class="border border-
purple-800 flex `+
                  `items-center justify-center sticky left-0 w-36 bg-indigo-
700">Ім'я/Дата</th>`;

                  lessons.forEach(lesson => {
                      html += `<th data-type="lessons" class="js-editable-
cell w-28 border border-purple-800 `+
                      `h-24 flex items-center justify-center bg-indigo-400">
                          <span>${lesson}</span>
```

```
                        <input value="${lesson}" class="hidden text-
center outline-none w-28 h-24 `+
                        `bg-indigo-400 border border-purple-800" type="text">
                    </th>`;
                });

                html += `<th data-type="module" class="w-28 border border-
purple-800 `+
                `h-24 flex items-center justify-center bg-indigo-400">
                <span>Модуль</span></th><th data-type="exam" class="w-
28 border border-purple-800 `+
                `h-24 flex items-center justify-center bg-indigo-
400"><span>Екзамен</span></th>`;

                root.innerHTML = html + '</tr>';
                addCellListeners(root.querySelectorAll('.js-editable-cell'));
            }

            const refreshBody = () => {
                const root = document.getElementById('tableBody');
                const students = state.students;

                let html = '', row;

                students.forEach(student => {
                    row = `<tr class="flex w-auto"><th data-
name="${student.name}" class="border border-purple-800 `+
                    `flex items-center justify-center sticky left-0 w-36 bg-
indigo-400">${student.name}</th>`;

                    student.marks.forEach(mark => {
                        row += `<td data-type="marks" class="js-editable-
cell w-28 border border-purple-800 h-24 `+
                        `flex items-center justify-
center"><span>${mark}</span><input value="${mark}" class="hidden `+
                        `text-center outline-none w-28 h-24 border border-
purple-800" type="text"></td>`;
                    });

                    row += `<td data-type="module" class="w-28 border border-
purple-800 h-24 flex items-center `+
                    `justify-center"><span>${student.module}</span><td data-
type="exam" class="js-editable-cell `+
                    `w-28 border border-purple-800 h-24 flex items-
center justify-center"><span>${student.exam}</span>`+
                    `<input value="${student.exam}" class="hidden text-
center outline-none w-28 h-24 border border-purple-800" type="text"></td>`

                    html += row + '</tr>';
                });
```

```
            root.innerHTML = html;
            addCellListeners(root.querySelectorAll('.js-editable-cell'));
        }
```

Рис 3.4 Функція оновлення верхівки та тіла таблиці

Додаємо логіку для кліків на кнопки(додати, відправити, видалити). Наприклад, кнопка додати.

```
addButton.addEventListener('click', () => {
        state.lessons.push('Дата' + cnt);
        cnt++;

        for(let i = 0; i < state.students.length; i++) {
          state.students[i].marks.push('');
        }

        refreshHead();
        refreshBody();
      });
```

3.3 Програмна реалізація back-end частини веб-сервісу обліку успішності студентів

У даному розділі більш детально розглянемо програмну реалізації back-end частини нашого додатку(яка згадувалася у розділі 3.1).

Розглянемо директорію unirate_app, тобто саме наш додаток.

Розпочнемо з models.py, де ми маємо 5 таблиць.

Таблицю для даних про групи, яка містить колонку з назвами груп

```
class Groups(models.Model):
  group_name = models.CharField(max_length=200)
```

Таблиця для даних про користувачів, де є колонка зі значенням id користувача в auth_user, колонку з іменем користувача, колонку з id групи, яка відноситься до таблиці Groups, та колонку з булевим значенням, де True – викладач, False – студент

```
class Profile(models.Model):
```

```
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    name = models.CharField(max_length=200)
    group = models.ForeignKey(Groups, blank=True, null=True, on_delete=models.
CASCADE)
    is_teacher = models.BooleanField(default=False)
```

Таблиця для даних про предмети, де міститься колонка з назвою, колонка зі значенням id викладача, що відноситься до таблиці Profile, і створює свою таблицю, колонка з id групи, що відноситься до таблиці Groups

```
class Table(models.Model):
    table_name = models.CharField(max_length=200)
    teacher = models.ManyToManyField(Profile)
    group = models.ForeignKey(Groups, on_delete=models.CASCADE)
```

Таблиця для даних про оцінки, де міститься колонка з назвою, колонка зі значенням id викладача, що відноситься до таблиці Profile, колонка з назвою заняття,
колонка зі значенням id таблиці, відноситься до таблиці Table

```
class Marks(models.Model):
    mark = models.CharField(max_length=200)
    student = models.ForeignKey(Profile, on_delete=models.CASCADE)
    date = models.CharField(max_length=200)
    table = models.ForeignKey(Table, on_delete=models.CASCADE)
```

Таблиця для даних про екзамен, де міститься колонка з назвою, колонка зі значенням id викладача, що відноситься до таблиці Profile, колонка зі значенням id таблиці, відноситься до таблиці Table.

```
class Exam(models.Model):
    exam = models.CharField(max_length=200)
    student = models.ForeignKey(Profile, on_delete=models.CASCADE)
    table = models.ForeignKey(Table, on_delete=models.CASCADE)
```

Urls.py – початок всього руху запитів, містить посилання на кінцеві точки обробки запитів у файлі views.py.

Містить рендер сторінки з формою логіна, аутентифікацію користувача, рендер сторінки зі списком груп/предметів, вихід користувача зі свого акаунта, рендер таблиці з даними по предмету, запит на внесення змін в базу даних, запит на отримання даних для таблиці у форматі JSON.

```
urlpatterns = [
    path('login/', views.render_login),
    path('auth/', views.auth_user),
    path('', views.render_menu),
    path('logout/', views.logout_user),
    path('table/<int:pk>/', views.render_tables),
    path('table/<int:pk>/change/', views.change_table),
    path('table/<int:pk>/upd', views.send_table)
]
```

Більш детально розглянемо файл views.py.

Тут міститься функція для рендеру сторінки логіну, де проходить перевірка аутентифікації користувача. Якщо користувач аутентифікований, редіректимо його на сторінку з меню. Якщо користувач не аутентифікований, редіректимо на сторінку з логіном

```
def render_login(request):
    if request.user.is_authenticated:
        return HttpResponseRedirect('/')
    else:
        return render(request, 'unirate_app/index.html')
```

Функція для обробки запиту та аутентифікації користувача в акаунт. Вона перевіряє чи метод запиту POST. Якщо так, то отримуємо дані з тіла запиту, отримуємо email по ключу зі словника auth_data, отримуємо password по ключу зі словника auth_data. Далі аутентифікуємо користувача, перевіряємо, чи аутентифікація пройшла успішно та профіль користувача активний.

```
def auth_user(request):
    if request.method == 'POST':
```

```
    auth_data = json.loads(request.body)
    email = auth_data['email']
    password = auth_data['password']

    user = authenticate(username=email, password=password)
    if user is not None and user.is_active:
      login(request, user)
      return HttpResponse('1')
    else:
      return HttpResponse('0')
```

Функція для обробки запиту та виходу з акаунта(якщо користувач виходить з профіля, повертаємо його на сторінку логіну)

```
def logout_user(request):
  logout(request)
  return HttpResponseRedirect('/login/')
```

Функція рендера сторінки для вибору групи/предмета. У ній ми отримуємо id користувача, якщо користувач – вчитель, тоді фільтруємо дані, отримуємо дані з назвами груп, змінюємо id групи на її назву, рендеримо сторінку зі списком груп для вчителя; якщо користувач – студент, тоді отримуємо id групи студента, по id групи отримуємо список предметів, рендеримо сторінку студента зі списком предметів.

```
@login_required
def render_menu(request): # Отрисовка меню
  c_user = request.user.profile.id
  tables = Table.objects.values()
  if request.user.profile.is_teacher:
    tables = Table.objects.filter(teacher=c_user).values()
    groups_name = Groups.objects.values_list('group_name')
    for e in tables:
      e['group_id'] = groups_name.filter(id=e['group_id']).first()[0]
    return render(request, 'unirate_app/groups.html', {'tables':tables})
  else:
    group_id = Profile.objects.filter(user=c_user).values_list().first()[3]
    tables = Table.objects.filter(group=group_id)
    return render(request, 'unirate_app/subjects.html', {'tables':tables})
```

Функція для заповнення словника балами, іменами студентів та додатковою інформацією. Вона створює пустий словник, отримує дані, фільтрує їх. Далі заповнює список занять, додає назви предметів та груп.

```
def get_tables(pk):
    marks = {}

    marks_set = Marks.objects.filter(table=pk).order_by('id').values()
    table_filter = Table.objects.filter(id=pk)
    group_id = table_filter.values_list('group').first()[0]
    profile_set = Profile.objects.filter(group_id=group_id).values()

    lessons_list = [e['date'] for e in marks_set]
    if len(lessons_list) == 0:
        lessons_list.append('Дата')
    marks['lessons'] = []
    for e in lessons_list:
        if e not in marks['lessons']:
            marks['lessons'].append(e)

    marks['students'] = []
    marks['title'] = table_filter.values_list('table_name').first()[0]
    marks['group'] =
Groups.objects.filter(id=group_id).values_list('group_name').first()[0]
    students_id = set([e['id'] for e in profile_set])
```

Функція для рендеру таблиці. Вона перевіряє чи користувач залогінений,викликає функцію get_tables та передає їй id таблиці. Якщо користувач – викладач, рендерить таблицю, яку можна змінювати для викладача; якщо користувач – студент, рендерить таблицю, яку не можна змінювати для студента.

```
@login_required
def render_tables(request, pk):
    marks = get_tables(pk)

    if request.user.profile.is_teacher:
        return render(request, 'unirate_app/editable.html', marks)
    else:
```

```
    return render(request, 'unirate_app/simple-table.html', marks)
```

Функція для обробки запиту та внесення змін в базу даних. Вона отримує дані з моделі, заповнює список заняттями. Перевіряє, чи метод запиту POST, отримує дані з тіла запиту. Для кожного студента в даних запиту створює змінну для імені, отримує id студента для імені, створює змінну лічильника. Далі додає значення для екзаменів та занять, якщо ці значення відсутні в базі даних, зберігає інформацію в базі даних та збільшує значення лічильника. У разі успіху, відправляє 1

```
def change_table(request, pk):
  marks_set = Marks.objects.values()
  lessons_count = set([e['date'] for e in marks_set])

  if request.method == 'POST':
    data = json.loads(request.body)

    for student in data['students']:
      student_name = student['name']
      student_id =
Profile.objects.filter(name=student_name).values_list().first()[0]

      counter = 0

      try:
        exam = Exam.objects.get(table=pk, student_id=student_id)
      except:
        exam = Exam()
        exam.exam = student['exam']
        exam.student_id = student_id
        exam.table_id = pk
        exam.save()
      else:
        exam.exam = student['exam']
        exam.save()


      for lesson in lessons_count:
        if lesson not in data['lessons']:
          marks = Marks.objects.get(table=pk, student_id=student_id,
date=lesson)
```

```
                marks.delete()

            for lesson in data['lessons']:
                try:
                    marks = Marks.objects.get(table=pk, student_id=student_id,
date=lesson)
                except:
                    marks = Marks()
                    marks.mark = student['marks'][counter]
                    marks.student_id = student_id
                    marks.date = lesson
                    marks.table_id = pk
                    marks.save()
                else:
                    marks.mark = student['marks'][counter]
                    marks.save()

                counter += 1
    return HttpResponse('1')
```

Функція для обробки запиту та відправки даних у форматі JSON, яка викликає функцію get_tables, передає їй id таблиці та дає відповідь у форматі JSON.

```
def send_table(request, pk):
    marks = get_tables(pk)
    return JsonResponse(marks)
```

ВИСНОВКИ

У випускній кваліфікаційній роботі було розроблено веб-сервіс для обліку успішності студентів.

Було проведено аналіз завдання та вибір методів його вирішення. Проведено ознайомлення з сучасними технологіями веб-розробки, огляд сучасних сучасних мов веб-розробки. Обрано найбільш доцільні методи та технології розробки.

Було проведено проектування структури веб-сервісу. Враховуючи проаналізовані способи та можливий вигляд майбутнього додатку, спроектовано дизайн веб-сервісу. Розроблено логічну та фізичну структуру сайту окремо для back-end і front-end частин додатку. Розроблено базу данних.

Наступним етапом була програмна реалізація роботи. Для розробки бази данних використано SQLite. Для створення back-end частини використано Python та фреймворк Django. Для розробки front-end частини використано HTML, CSS, JavaScript. В кінцевому результаті було розроблено веб-сервіс для обліку навчального процесу студентів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. https://studfile.net/preview/1624161
2. АНАЛІЗ СУЧАСНИХ ЗАСОБІВ ВЕБ-РОЗРОБКИ / [Андріїв І.В. https://www.sworld.com.ua/konfer49/44.pdf]
3. https://www.uzhnu.edu.ua/en/infocentre/get/5215
4. https://cad.kpi.ua/attachments/093_2016d_Yakymets.pdf
5. https://studwood.ru/1052752/informatika/zagalni_vidomosti_tehnologiyi
6. https://webformyself.com/yazyki-programmirovaniya-dlya-veb-razrabotki/
7. https://www.castcom.ru/publications/web/kakie_yazyki_programmirovaniya_ispolzuyutsya_pri_sozdanii_sajtov.html
8. http://inmad.vntu.edu.ua/portal/static/F679CC32-7EC2-49A1-9034-EB0559D47085.pdf
9. https://shag.com.ua/1-etapi-proektuvannya-web-sajtiv-6.html?page=6
10. https://web-creator.ru/articles/django
11. https://uk.wikipedia.org/wiki/SQLite