Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка веб-застосунку для репетиторів «ВебМентор»

Виконав: студент IV курсу, групи СНс-42
спеціальності 122 Комп’ютерні науки
(шифр і назва спеціальності)

_____ Костюк Я. І.
(підпис) (прізвище та ініціали)

Керівник _____ Дмитроца Л. П.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Шимчук Г. В.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

Рецензент _____ Оробчук О. Р.
(підпис) (прізвище та ініціали)

Тернопіль
2024

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«__» _червня_ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

Студенту Костюку Ярославу Ігоровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка веб-застосунку для репетиторів «ВебМентор»

Керівник роботи Дмитроца Леся Павлівна, к.т.н., доцент кафедри КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «*29*» *квітня* 2024 року № *4/7-472*

2. Термін подання студентом завершеної роботи 24 червня 2024р.

3. Вихідні дані до роботи Літературні та інтернет джерела інформації щодо розробки веб-застосунку для репетиторів «ВебМентор»

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1. Аналіз предметної області та постановка завдання розробки веб-застосунку для репетиторів «ВебМетор». 1.1 Аналіз предметної області. 1.2 Задачі та складові веб-застосунку для репетиторів «ВебМентор». 1.3 Аналіз існуючих рішень-аналогів. 1.3.1 Notion. 1.3.2 Light 1.3.3 EasyWeak. 1.4 Вимоги до розроблюваного веб-застосунку для репетиторів «ВебМентор». 1.5 Висновки до першого розділу. 2. Проєктування та моделювання веб-застосунку для Репетиторів «Вебментор». 2.1 Обґрунтування вибору технологій розробки веб-застосунку для репетиторів «ВебМентор». 2.2 Вибір середовища розробки веб-застосунку для репетиторів «ВебМентор». 2.3 Аналіз акторів та варіантів використання веб-застосунку для репетиторів «ВебМентор». 2.4 Формування структури веб-застосунку для репетиторів «ВебМентор». 2.5 Моделювання бази даних веб-застосунку для репетиторів «ВебМентор». 2.5.1 Опис інформаційних сутностей та їх атрибути. 2.5.2 Формування зв'язків між таблицями. 2.6 Висновки до другого розділу. 3. Розробка та тестування веб-застосунку для репетиторів «ВебМентор». 3.1 Розробка інтерфейсу користувача веб-застосунку для репетиторів «ВебМентор». 3.2 Розробка серверної частини та логіки веб-застосунку для репетиторів. 3.2.1 Реєстрація та авторизація користувачів. 3.2.2 CRUD операції. 3.2.3 Пошук та фільтрація в таблицях. 3.2.4 Розрахунок середнього балу учня та підрахунок сум оплат. 3.3 Тестування функціональних можливостей веб-застосунку «ВебМентор». 3.4 Висновки до третього розділу 4. Безпека життєдіяльності, основи охорони праці. Висновки. Перелік джерел. Додатки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Титульна сторінка. 2. Мета, об'єкт та предмет дослідження. 3. Актуальність обраної теми. 4. Аналіз предметної області. 5. Огляд існуючих рішень. 6. Варіанти використання. 7. Структура веб-застосунку. 8. Інформаційні сутності. 9. ER-діаграма. 10. Інтерфейс користувача веб-застосунку. 11. Функції веб-застосунку. 12. Висновки

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	Сенчишин В.С., доцент кафедри МТ	12.06.2024	15.06.2024

7. Дата видачі завдання 29 січня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	30.01.2024	*Виконано*
2.	Підбір джерел про розробку веб-застосунків для	31.01.2024-03.02.2024	*Виконано*
	Репетиторів		
3.	Опрацювання джерел щодо розробки веб-застосунку	04.02.2024-06.02.2024	*Виконано*
	для репетитора «ВебМентор»		
4.	Виконання дослідження розробки веб-застосунків для	07.02.2024-11.02.2024	*Виконано*
	репетиторів. Розроблення веб-застосунку для		
	репетиторів «ВебМентор»		
5.	Оформлення розділу «Аналіз предметної області та	03.06.2024-05.06.2024	*Виконано*
	постановка завдання розробки веб-застосунку для		
	Репетиторів «ВебМентор»		
6.	Оформлення розділу «Проєктування та моделювання	06.06.2024-08.06.2024	*Виконано*
	веб-застосунку для репеторів «ВебМентор»		
6.	Оформлення розділу «Розробка та тестування веб-	09.06.2024-11.06.2024	*Виконано*
	застосунку для репетиторів «ВебМентор»		
7.	Виконання завдання до підрозділу «Безпека життєдіяльності»	12.06.2024-13.06.2024	*Виконано*
8.	Виконання завдання до підрозділу «Основи охорони	14.06.2024-15.06.2024	*Виконано*
	праці»		
9.	Оформлення кваліфікаційної роботи	16.06.2024-17.06.2024	*Виконано*
10.	Нормоконтроль	18.06.2024-19.06.2024	*Виконано*
11.	Перевірка на плагіат	20.06.2024	*Виконано*
12.	Попередній захист кваліфікаційної роботи	21.06.2024	*Виконано*
13.	Захист кваліфікаційної роботи	29.06.2024	

Студент _____ Костюк Я. І.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Дмитроца Л. П.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка веб-застосунку для репетиторів «ВебМентор» // Кваліфікаційна робота освітнього рівня «Бакалавр» // Костюк Ярослав Ігорович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СНс-42 // Тернопіль, 2024 // С. 64, рис. – 18, табл. – 1, презент. – 12, додат.– 5, бібліогр. – 48.

Ключові слова: веб-застосунок, бази даних, репетитор, учень, PHP, вебментор.

Кваліфікаційна робота присвячена розробці веб-застосунку для репетиторів «ВебМентор», як зручного та ефективного інструменту для автоматизації та оптимізації роботи репетиторів.

В першому розділі кваліфікаційної роботи проведено детальний аналіз предметної області. Досліджено наявні програмні рішення та їх основні функції. Проаналізовано основні вимоги до розроблюваного веб-застосунку.

В другому розділі кваліфікаційної роботи обґрунтовано вибір технологій розробки. Проаналізовано акторів системи та варіанти використання. Сформовано структуру веб-застосунку для репетиторів «ВебМентор». Описано інформаційні сутності та зв'язки між ними.

В третьому розділі кваліфікаційної роботи розглянуто етапи розробки веб-застосунку для репетиторів «ВебМентор», реалізовано інтерфейс користувача, серверну частину та проведено тестування функціональних можливостей.

ANNOTATION

Development of a Web Application for tutors «WebMentor» // Qualification work of the educational level «Bachelor» // Kostiuk Yaroslav // Ivan Pulyuy Ternopil National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Computer Sciences, group SNs-42 // Ternopil, 2024 // P. 64, fig. – 18, tabl. – 1, present. – 12, annexes. – 5, references - 48.

Keywords: web application, databases, tutor, student, PHP, webmentor.

The qualification work is devoted to the development of the web application for tutors "WebMentor", as a convenient and effective tool for automating and optimizing the work of tutors.

In the first section of the qualification work, a detailed analysis of the subject area was carried out. The available software solutions and their main functions have been studied. The main requirements for the developed web application were analyzed.

In the second section of the qualification work, the choice of development technologies is substantiated. System actors and usage options are analyzed. The structure of the web application for tutors «WebMentor» has been created. Information entities and connections between them are described.

In the third section of the qualification work, the stages of development of the web application for tutors «WebMentor» were considered, the user interface, the server part were implemented, and the functional capabilities were tested.

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ І ТЕРМІНІВ

AJAX (англ. Asynchronous JavaScript And XML) – підхід до побудови користувацьких інтерфейсів веб-застосунків, за яких веб-сторінка, не перезавантажуючись, у фоновому режимі надсилає запити на сервер і сама звідти довантажує потрібні користувачу дані.

CRUD (англ. Create read update delete) – 4 основні функції управління даними «створення, читання, оновлення і вилучення».

CSS (англ. Cascading Style Sheets) – спеціальна мова стилів, що використовується для опису зовнішнього вигляду сторінок.

HTML (англ. Hypertext Markup Language) – мова розмітки для створення веб-сайтів і веб-застосунків.

HTTP (англ. HyperText Transfer Protocol) – протокол прикладного рівня передачі даних/

JS (англ. JavaScript) – скриптова мова програмування.

MySQL – вільна система керування реляційними базами даних.

PDO (англ. PHP Data Objects) – розширення для PHP, що надає розробнику простий і універсальний інтерфейс для доступу до різних баз даних.

PHP – скриптова мова програмування.

SQL (англ. Structed Query Language) – декларативна мова програмування для реляційних БД.

UML (англ. Unified Modeling Language) – уніфікована мова моделювання.

URL (англ. Uniform Resource Locator) — стандартизована адреса певного ресурсу.

БД – база даних.

СКБД – система керуваннями базами даних.

ЦНС – центральна нервова система.

ЗМІСТ

ВСТУП

Актуальність теми. У сучасному глобалізованому світі неформальна освіта відіграє все більшу роль, і ринок репетиторства є важливою частиною цього процесу. Під репетиторством часто розуміють підготовку дітей до школи, вступних іспитів, незалежного тестування, розвитку професійних навичок та отримання нових знань.

Велика кількість учнів різного віку та щільний робочий графік ускладнюють ефективне здійснення діяльності репетиторів. Заплановані заняття часто переносяться або взагалі скасовуються, а якщо заняття все ж таки відбуваються, репетитори повинні враховувати багато додаткової інформації, наприклад, які теми будуть розглянуті, чи будуть тести, чи будуть перевірятися домашні завдання тощо.

Крім цього, важливо записувати підрахунок оплати занять, що також складно відслідковувати. Адже клієнти можуть оплачувати щодня, щомісяця або через певні проміжки часу. Тому розробка веб-застосунку для репетиторів «ВебМентор» є досить актуальним напрямком дослідження.

Мета і задачі дослідження. Метою даної кваліфікаційної роботи освітнього рівня «Бакалавр» є підвищення якості послуг репетиторів. Для досягнення поставленої мети потрібно виконати ряд завдань, зокрема:

– проаналізувати особливості роботи репетитора з урахуванням задач репетитора, які потребують реалізації у веб-застосунку;

– проаналізувати наявні рішення, та визначити основні аспекти їх функціональності;

– обґрунтувати вибір архітектури, середовища, технологій та засобів створення веб-застосунку;

– спроектувати базу даних для зберігання інформації;

– розробити застосунок з застосуванням обраних засобів і протестувати його роботу.

Практичне значення одержаних результатів.

Створений в результаті кваліфікаційної роботи веб-застосунок «ВебМентор» представляє собою набір інструментів для репетитора, якими він може відслідковувати, аналізувати, коригувати та організовувати свою роботу. Завдяки інтегрованим функціям, веб-застосунок дозволяє репетиторам ефективно взаємодіяти з даними про своїх учнів, та спрощує документальну діяльність.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ РОЗРОБКИ ВЕБ-ЗАСТОСУНКУ ДЛЯ РЕПЕТИТОРІВ «ВЕБМЕНТОР»

1.1 Аналіз предметної області

В сучасному світі освіта відіграє ключову роль. Вона є основою для розвитку особистості, кар'єри, а також для прогресу суспільства в цілому. В Україні, як і в багатьох інших країнах, вимоги до якості освіти постійно зростають. В результаті, багато учнів і батьків звертаються до послуг репетиторів, щоб підвищити рівень знань і навичок, необхідних для успішного навчання. Репетитор – це педагог, який надає індивідуальні заняття з певного предмету, допомагає засвоїти матеріал та підготуватися до іспитів. Також репетитори відіграють важливу роль у підготовці учнів до ЗНО (зовнішнє незалежне оцінювання) чи НМТ (національний мультипредметний тест). Воно визначає можливості для подальшого навчання в вищих навчальних закладах. Це досить тривалий та складний процес, який вимагає додаткових зусиль та ресурсів. Але репетиторство не закінчується лише на школярах чи студентах. Існує багато людей, особливо ті, які прагнуть до кар'єрного зросту, зміни діяльності чи просто особистого розвитку звертаються по допомогу до репетиторів. [1]

Автоматизація процесу роботи репетиторів є важливим аспектом сучасної освіти. Важливість надання цифрових послуг в освіті постійно зростає, оскільки вони забезпечують доступність та зручність навчання [2]. Інноваційні методи навчання, зокрема інтерактивні онлайн-заняття, використовуються для підвищення ефективності навчального процесу [3]. Цифрова трансформація освіти сприяє впровадженню нових технологій, що роблять навчання більш гнучким та доступним [4].Створення веб-застосунку «ВебМентор» дозволить репетиторам зосередитися на навчанні, замість витрачати багато часу на рутинну адміністративно-паперову роботу, дозволить покращити організацію занять та масштабувати свою діяльність. [5]

1.2 Задачі та складові веб-застосунку для репетиторів «ВебМентор»

Розробка веб-застосунку «ВебМентор» для репетиторів має великий потенціал у полегшенні їхньої роботи та покращенні якості навчання. Перш за все, реєстрація чи авторизація за допомогою електронної адреси та надійного паролю дозволяє гарантувати безпеку та конфіденційність даних. Однією з головних переваг є можливість ефективного управління робочим процесом. За допомогою цього веб-застосунку репетитор може легко зберігати інформацію про своїх учнів, включаючи контактні дані, рівень знань та інші важливі деталі. Крім того, він може планувати розклад занять та інші заходи, вести журнал занять та відстежувати прогрес кожного студента.

Оптимізація часу є ще однією важливою перевагою веб-застосунку «ВебМентор». Репетитори можуть ефективно організувати свій час, завдяки можливості створення календаря занять та журналу, що дозволяє уникнути збігів в графіку та забезпечує своєчасну підготовку до кожного заняття.

Для покращення якості навчання «ВебМентор» надає репетиторам зручні інструменти для відстеження прогресу учнів. Вони можуть аналізувати результати тестів та завдань, відстежувати успішність та виявляти слабкі моменти для подальшого удосконалення навчального процесу. Це дозволяє репетиторам адаптувати свій підхід до кожного учня та забезпечує індивідуальне навчання.

1.3 Аналіз існуючих рішень-аналогів

Для більш детального аналізу предметної області було проведено дослідження програмних засобів, які призначені для автоматизації роботи репетитора, ведення обліку, а також застосунки, які мають схожу функціональність. Результатом аналізу стали наступні додатки: Notion, Light, EasyWeek.

1.3.1 Notion

Notion – це універсальний інструмент для організації роботи, який дозволяє створювати сторінки, бази даних і співпрацювати з іншими. Це забезпечує чистий, відкритий простір для мислення, написання та планування. Notion безкоштовний для особистого користування, але існують обмеження на розмір завантажуваних файлів, кількість сторінок і кількість користувачів [6]. Розробником є відомий програміст Айван Чжао [9].

Інтерфейс додатку є досить сучасним та приємним у використанні (Див. рисунок 1.1). Проте для недосвідченого користувача потрібен час та деякі зусилля, щоб освоїти на навчитися використовувати застосунок[7].

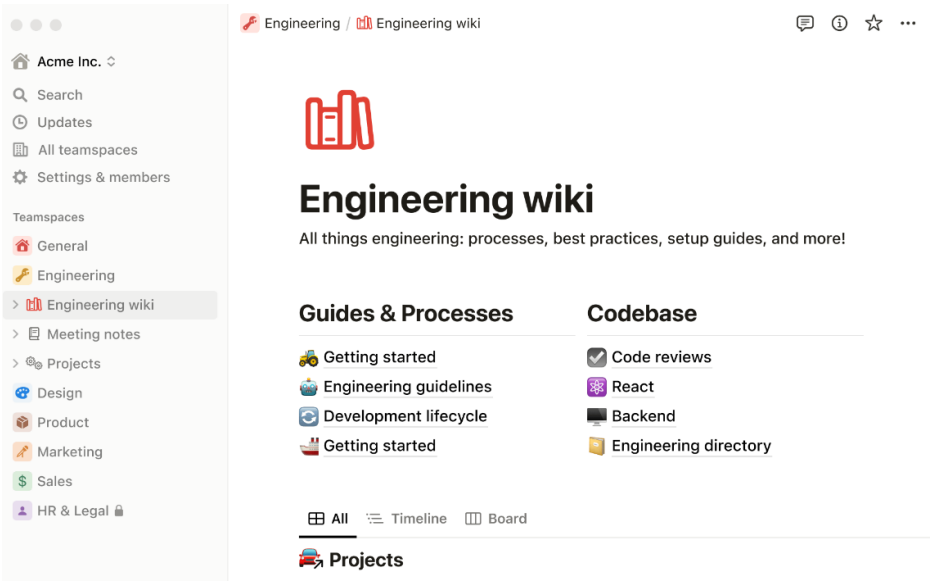

Рисунок 1.1 – Інтерфейс користувача додатку Notion [7]

Щодо інструментів для репетитора Notion пропонує наступне:

– Керування зустрічами. Інтегрований календар допомагає вчителям відстежувати всі зустрічі зі студентами.

– Відстеження інформації про студента. Надається можливість створити базу даних для відстеження прогресу студента.

– Керування курсом. Надається можливість створювати шаблони курсу та ділитися ними зі студентами.

– Відстеження оплат. Надається можливість відстежувати свої рахунки та платежі.

– Упорядкування нотаток. Надається можливість легко впорядкувати нотатки та матеріали курсу [8].

Однак, як і будь-яка інша програма, Notion має свої недоліки. Серед них виділяють:

– відсутність української мови інтерфейсу;

– обмеження використання ресурсів у безкоштовній версії;

– повільна швидкість роботи додатку [9].

1.3.2 Light

Light – це календар уроків репетитора, де можна відстежувати платежі та прогнозувати дохід. Репетитор може додати свої заняття зі студентами до календаря, відстежувати платежі, редагувати баланс клієнтів. Розробником є Кирило Мусейчук [10]

Інтерфейс додатку Light є простим і зручним для користувачів, зображений на рисунку 1.2.

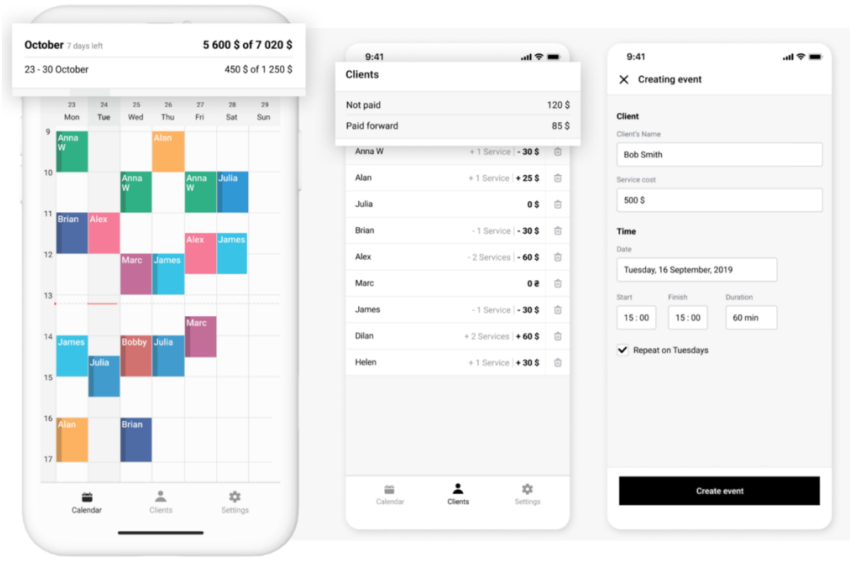

Рисунок 1.2 – Інтерфейс користувача додатку Light [10]

Light призначений для репетиторів, тренерів, психотерапевтів і всіх, хто працює з клієнтами за графіком. Найбільше він орієнтований на репетиторів, але Light чудово підходить і для багатьох інших професій [10]

До основних функцій додатку входить:

– Планування уроків. Репетитор може вести свій розклад у календарі, створеному спеціально для таких потреб. Щоб запланувати урок з клієнтом, вказується його ім’я, вартість та тривалість уроку. Повторювані події автоматично переносяться на кожен наступний тиждень. У календарі є можливість побачити весь свій графік на тиждень і швидко знайти вільний час для нових учнів.

– Прогнозування доходів. На основі графіка Light рахує, скільки грошей репетитор заробить за тиждень і місяць. Коли студенти внесли оплату за заняття, Light показує , скільки вже зароблено за минулий тиждень та місяць.

– Облік платежів. Коли заняття з учнем закінчується, додаток автоматично «списує» вартість уроку з його балансу. Також у календарі можна вказати, чи платний урок. Якщо учень заплатить наперед, то викладач може «поповнити» його баланс, і додаток автоматично розподілить ці гроші на його неоплачувану діяльність.

– Залишки клієнтів. Для кожного показує його баланс: скільки учень заплатив авансом, чи скільки він повинен заплатити. Щоб змінити баланс учня, слід позначити платіж у календарі або поповнити його вручну. Таким чином репетитор завжди побачить, скільки учень заборгував, або скільки уроків він заплатив наперед. Також на балансі учня відображається кількість уроків, оплачених наперед або пройдених, але ще не оплачених. [11]

На даний час, є деякі функції, які не реалізовані у додатку:

– Відсутність української мови для використання;

– Відсутність безкоштовної версії;

– Відсутність ведення обліку успішності. [11]

1.3.3 EasyWeek

EasyWeek – це система онлайн-запису та управління графіком. Вона допомагає вести розклад, відмічати оплати, редагувати баланс клієнтів та бронювати заняття. Додаток EasyWeek розроблений спеціально для репетиторів, тренерів, психотерапевтів та всіх, хто працює з клієнтами за розкладом. [12]

Інтерфейс даного застосунку є досить приємним та зручним у користуванні. Проте для недосвідченого користувача освоїти весь функціонал додатку буде складним процесом. Інтерфейс зображено на рисунку 1.3.

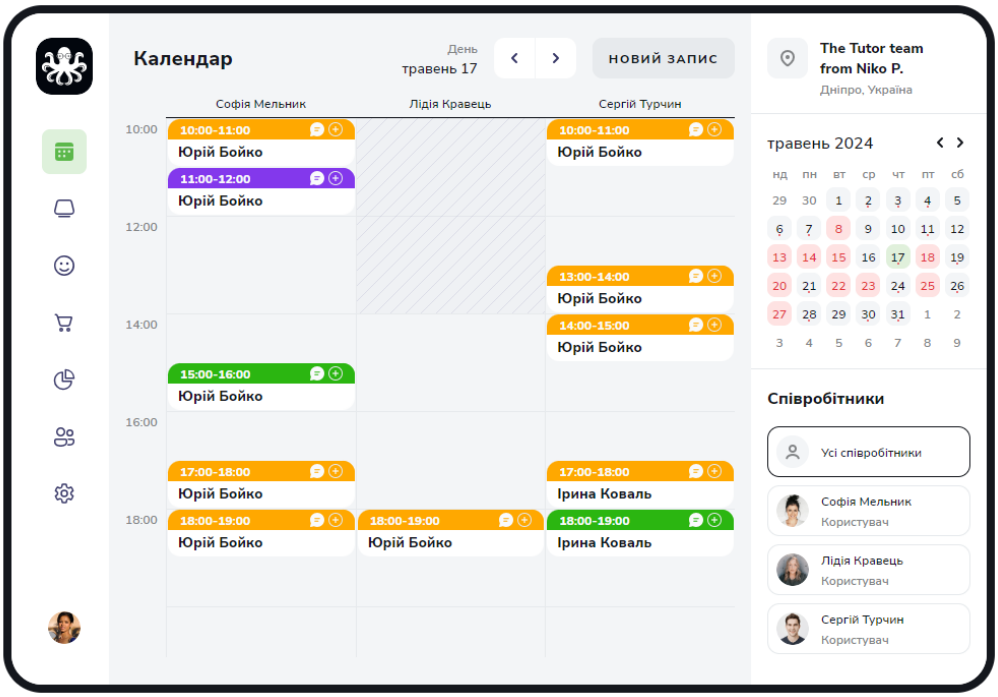

Рисунок 1.3 – Інтерфейс користувача додатку EasyWeek [12]

Серед переваг застосунку можна виділити:

– SMS, email та Push сповіщення;

– аналітика розкладу, прибутків та отримання платежів;

– просування за допомогою соціальних мереж та маркетингових інструментів;

– картка учня. [12]

Проте серед функціоналу можна виділити наступні недоліки:

– обмеження функціоналу безкоштовної версії;

– відсутність відстеження успішності учня та ведення його журналу;

– складність освоєння користувацького інтерфейсу. [12]

1.4 Вимоги до розроблюваного веб-застосунку для репетиторів «ВебМентор»

Провівши аналіз предметної області дослідження, детальний аналіз існуючих рішень та враховуючи висунуті завдання, важливо визначити вимоги до веб-застосунку «ВебМентор», які можна поділити на функціональні та нефункціональні. Вони будуть відігравати ключову роль у розробці.

Функціональні вимоги являють собою основні функціональні особливості та можливості веб-застосунку для досягнення поставленої мети [13]. Розроблюваний веб-застосунок повинен мати наступні функціональні можливості:

– можливість створення власного акаунту з іменем, електронною адресою та паролем;

– забезпечення відсутності дублікатів акаунтів користувачів;

– забезпечення захисту даних користувачів при авторизації в системі, перевіряючи вірність їх вводу та хешуванню паролю;

– забезпечення можливості додавання нових учнів, редагування наявної інформації, перегляд індивідуальної інформації та видалення неактивних учнів, а також можливість пошуку інформації про учня за прізвищем;

– можливість ведення журналу успішності учня, відвідувань занять та розрахунку середнього балу учня за трьома критеріями – домашня робота, робота на уроці та тестування;

– забезпечення ведення графіку проведення уроків та пошуку за назвою групи або прізвища учня (індивідуальні заняття) або за датою проведення;

– можливість запису конспектів уроків та власних корисних нотаток;

– можливість запису та планування подій, для ефективного використання часу;

– забезпечення обліку оплати проведених занять, пошуку за прізвищем учня та датою оплати, а також розрахунок заробітку як індивідуального учня та всіх разом за певний період часу.

Нефункціональні вимоги доповнюють функціональні, вказуючи як веб-застосунок повинен їх виконувати [13]. До таких вимог визначено:

– продуктивність;

– надійність;

– безпека;

– зручність.

Веб-застосунок повинен забезпечувати ефективне та швидке виконання запитів користувача, повинен працювати безперервно та надійно, повинен мати високий рівень захисту від атак та несанкціонованого доступу, повинен бути зручним у використанні та навігації для користувачів будь-якого рівня. Інтуїтивний інтерфейс користувача, проста навігація, зрозумілі повідомлення про помилки та інші аспекти, що полегшують використання системи.

Дані вимоги утворюють основу у забезпеченні якості та ефективності веб-застосунку «ВебМентор», а також в забезпеченні задоволення потреб користувачів та безпеки даних.

1.5 Висновки до першого розділу

У першому розділі кваліфікаційної роботи освітнього рівня «Бакалавр» проведено аналіз предметної області, що підкреслив важливість освіти в сучасному світі та роль репетиторства у підвищенні якості навчання. Зокрема, виявлено, що автоматизація процесу роботи репетиторів є ключовим аспектом сучасної освіти. З метою полегшення роботи репетиторів та покращення якості навчання пропонується розробка веб-застосунку «ВебМентор». Він має

потенціал забезпечити безпеку даних, оптимізувати час, а також надавати зручні інструменти для відстеження прогресу учнів.

Для кращого розуміння контексту було проведено аналіз існуючих рішень-аналогів, таких як Notion, Light та EasyWeek. Кожен з цих додатків має свої переваги та недоліки, що важливо врахувати під час розробки «ВебМентор».

З метою досягнення поставлених цілей у розробці веб-застосунку «ВебМентор» сформульовано функціональні та нефункціональні вимоги, які визначають основні функціональні можливості та вимоги до продуктивності, надійності, безпеки та зручності використання застосунку.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА МОДЕЛЮВАННЯ ВЕБ-ЗАСТОСУНКУ ДЛЯ РЕПЕТИТОРІВ «ВЕБМЕНТОР»

2.1 Обґрунтування вибору технологій розробки веб-застосунку для репетиторів «ВебМентор»

Веб-застосунок «ВебМентор» призначений для управління діяльністю репетитора, забезпечуючи ефективну організацію учнів, уроків, оплат та інших важливих аспектів роботи. Для розробки цього застосунку були обрані наступні технології: HTML, CSS, JavaScript, PHP та MySQL.

HTML є основою веб-застосунку, оскільки дозволяє структурувати вміст на веб-сторінці. HTML дозволяє створювати різні елементи інтерфейсу, включаючи форми, таблиці та інші компоненти, необхідні для взаємодії з користувачем.HTML – це стандартизована мова розмітки, яка забезпечує сумісність з усіма сучасними веб-браузерами.[14]

CSS є важливим інструментом у веб-розробці, який дозволяє відокремити структуру веб-сторінки від її стилізації. Це забезпечує більшу гнучкість та легкість у підтримці коду. CSS використовується для опису вигляду HTML-документів, визначаючи стиль елементів, таких як шрифти, кольори, відступи, розташування елементів на сторінці. Однією з ключових переваг CSS є можливість централізовано керувати стилями. Це означає, що зміни в одному файлі стилів можуть миттєво застосовуватися до всіх сторінок веб-застосунку, що значно спрощує процес оновлення дизайну. [15]

Сучасний CSS включає в себе потужні інструменти, такі як Flexbox, які полегшують створення складних макетів без необхідності використання JavaScript. Flexbox дозволяє легко розташовувати елементи в ряд або стовпець і автоматично коригувати їх розміри відповідно до доступного простору.[16]

Крім того, CSS підтримує анімації та переходи, які можуть значно покращити користувацький досвід. Анімації дозволяють плавно змінювати властивості елементів, додаючи до веб-сторінок динамічність та візуальну

привабливість. Переходи дозволяють створювати ефекти зміни стилів, такі як плавне зміна кольору або розміру, що робить взаємодію з веб-сторінкою більш інтуїтивною та естетично приємною. [17]

JavaScript є мовою програмування, що дозволяє реалізувати інтерактивність та динамічну поведінку на веб-сторінках. Веб-застосунок "ВебМентор" використовує JavaScript для реалізації таких функцій, як пошук, фільтрація даних та підрахунок оплат. Використання JavaScript дозволяє забезпечити швидку реакцію застосунку на дії користувача без необхідності перезавантаження сторінки, що значно покращує користувацький досвід. [18]

PHP є серверною мовою програмування, що використовується для створення динамічних веб-сторінок та взаємодії з базою даних. Використання PHP у "ВебМенторі" дозволяє реалізувати функціонал додавання, редагування, видалення та перегляду даних у таблицях, а також обробку форм та управління сесіями користувачів. PHP є дуже популярною мовою серед розробників завдяки своїй гнучкості, великій кількості бібліотек та активній спільноті підтримки. [19]

MySQL є системою управління реляційними базами даних, що використовується для зберігання та управління даними. У "ВебМенторі" MySQL використовується для зберігання даних учнів, журналу, календаря уроків, конспектів, подій та оплат. MySQL забезпечує надійне зберігання великих обсягів даних та швидкий доступ до них. Крім того, MySQL підтримує мову SQL, що дозволяє легко виконувати складні запити для фільтрації та підрахунку даних. [20]

2.2 Вибір середовища розробки веб-застосунку для репетиторів «ВебМентор»

Для реалізації всіх поставлених вимог і задач веб-застосунку «ВебМентор», враховуючи технології розробки, обрано Sublime Text. Це рішення ґрунтується на ряді переваг, які надає цей текстовий редактор, забезпечуючи ефективну та продуктивну роботу над проектом.

Sublime Text відомий своєю високою швидкодією та легкістю. Він швидко запускається і обробляє великі файли без затримок, що є критично важливим для роботи з великими обсягами коду та даних, зокрема при розробці складних веб-застосунків. Його мінімалістичний інтерфейс сприяє концентрації на коді, усуваючи зайві відволікання, що особливо важливо для розробників, які цінують ефективність і продуктивність. [21]

Інтерфейс Sublime Text є чистим і інтуїтивно зрозумілим, що полегшує його використання як для новачків, так і для досвідчених розробників. Він підтримує численні схеми кольорів та теми, які можна налаштувати під індивідуальні потреби, що робить роботу з редактором ще більш комфортною. Інтерфейс користувача даного середовища подано на рисунку 2.1.

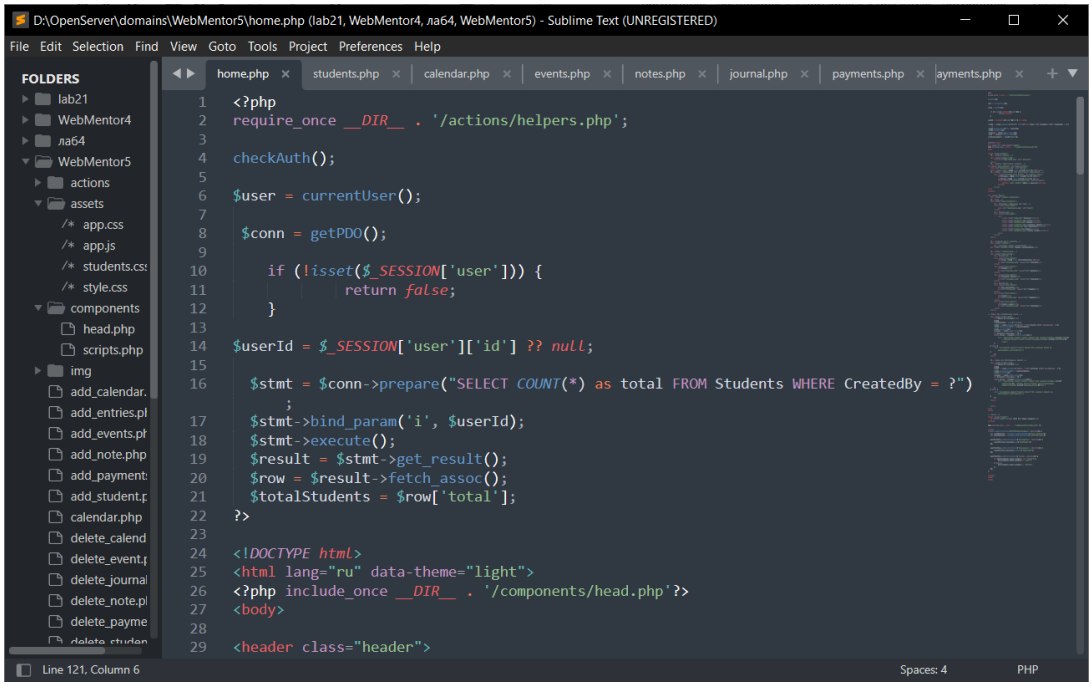

Рисунок 2.1 – Робоча область середовища Sublime Text

Ще однією важливою перевагою Sublime Text є його висока налаштовуваність. Існує безліч доступних плагінів, які можна легко встановлювати через Package Control, що дозволяє розширити функціональність редактора відповідно до специфічних потреб проекту. Це особливо корисно для інтеграції з іншими інструментами та технологіями, які використовуються у

вашому веб-застосунку, такими як PHP для серверної логіки та MySQL для управління базами даних. [21]

Отже, вибір Sublime Text для розробки веб-застосунку «ВебМентор» є обґрунтованим рішенням, яке забезпечує швидку та ефективну розробку, підтримуючи баланс між функціональністю та простотою використання.

2.3 Аналіз акторів та варіантів використання веб-застосунку для репетиторів «ВебМентор»

Важливим кроком у розробці є аналіз акторів та варіантів використання веб-застосунку для репетитора «ВебМентор». Це допоможе формалізувати функціональні вимоги до системи, відображаючи взаємодію між користувачами системи та сценаріями використання.

Актори – це дійові особи, які певним чином взаємодіють із системою або впливають на неї [22]. Дана система включає наступних акторів:

– незареєстрований користувач – особа, яка ще не має створеного акаунту, але може зареєструватись та взаємодіяти із системою;

– репетитор – зареєстрований користувач системи, який має акаунт у веб-застосунку та може використовувати функції системи, до яких відносять запис учнів, ведення журналу успішності, планування уроків та подій, облік оплат та інше.

Варіанти використання – це послідовність дій, за допомогою яких актор взаємодіє з застосунком для виконання певної дії та досягнення конкретної мети [23].

На основі функціональних вимог до веб-застосунку «ВебМентор», які описані у розділі 1, були визначені варіанти використання для акторів системи, тобто як вони будуть взаємодіяти з системою та які конкретні завдання вони виконуватимуть. Кожен варіант використання описує конкретний сценарій взаємодії користувача із системою, включаючи всі можливі шляхи, якими може піти користувач для виконання завдання. Для актора «незареєстрований

користувач» варіанти використання неведені в таблиці 2.1, для актора «репетитор» – у таблиці А.1.

Таблиця 2.1 – Варіанти використання для актора «незареєстрований користувач»

Актор	**Найменування**	**Формулювання**
Незареєстрований користувач	Реєстрація	Дозволяє даному актору створити новий акаунт у веб-застосунку

Переглянувши таблицю 2.1 стає очевидним, що актор «незареєстрований користувач» обмежений лише створенням акаунту. Це важливо зазначити, адже всі інші дії у веб-застосунку можливі лише після входу в особистий акаунт репетитора.

Для більш кращого розуміння взаємодії акторів з системою, наступний крок – це представлення UML діаграми варіантів використання. Діаграма прецедентів візуально відображає як актори системи взаємодіють із варіантами використання і будь-які зв’язки між ними [24]. UML – це набір інструментів для моделювання, який використовується для створення діаграм. Варіанти використання представлені позначенням овальної форми. Фігурки представляють акторів у процесі, а участь актора в системі моделюється лінією між актором і варіантом використання [25].

Для створення цієї діаграми було використано середовище проектування IBM Rational Software Architect (RSA). RSA пропонує широкий спектр інструментів для створення UML діаграм, забезпечуючи зручний інтерфейс та простоту у використанні. Це середовище дозволяє швидко і ефективно розробляти діаграми, що відображають складні взаємодії між компонентами системи, сприяючи глибшому розумінню її архітектури та функціональних можливостей [26].

На рисунку 2.2 представлено діаграму варіантів використання розроблюваного веб-застосунку для репетитора «ВебМентор».

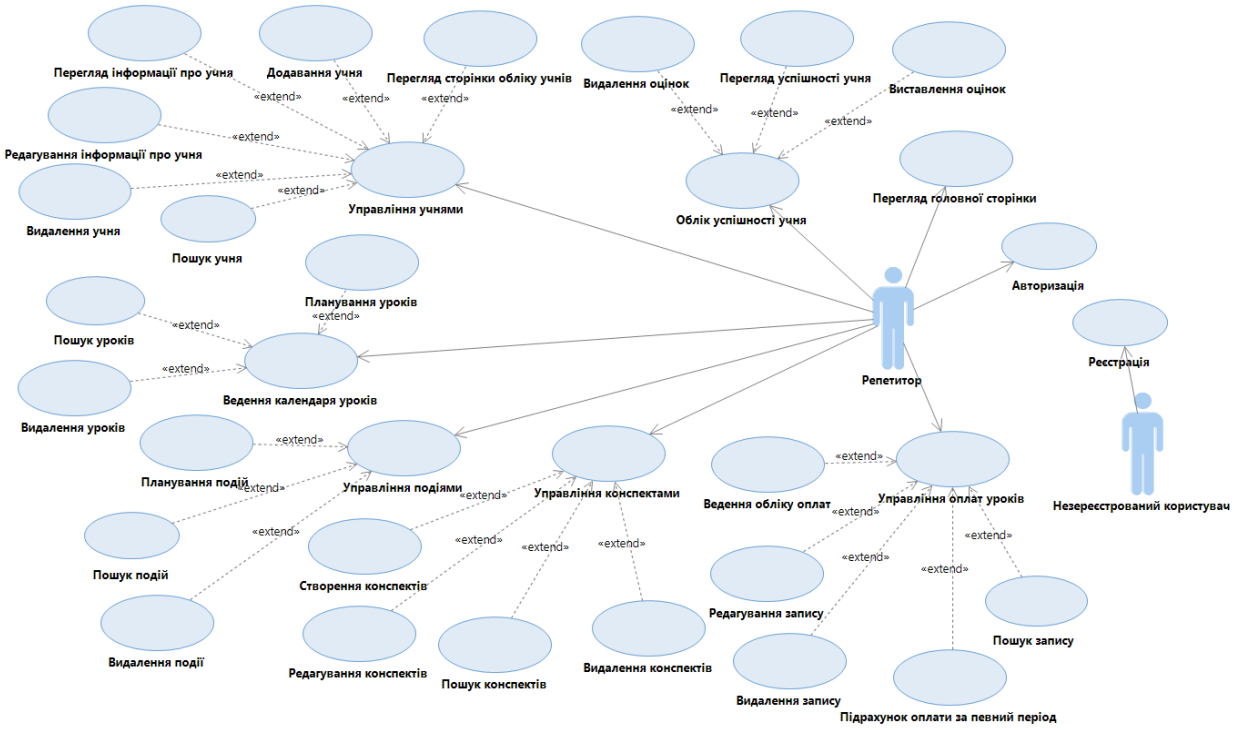

Рисунок 2.2 – Діаграма варіантів використання веб-застосунку для репетитора «ВебМентор»

Аналізуючи рисунок 2.2, можна побачити, що актор «репетитор» має можливість використовувати всі функції веб-застосунку.

2.4 Формування структури веб-застосунку для репетиторів «ВебМентор»

Провівши аналіз акторів та варіантів використання веб-застосунку для репетитора «ВебМентор», а також базуючись на сформованих у першому розділі вимогах до застосунку та обраних технологіях розробки, можна дійти висновку, що для даної розробки найбільш оптимальною є трьохрівнева архітектура [27]. Вона складається з наступних компонентів:

1. Презентаційний рівень (frontend) – це інтерфейс користувача та комунікаційний рівень веб-застосунку, де користувач взаємодіє із системою. Його ціль – відображати інформацію користувачу і збирати інформацію від користувача.

2. Рівень застосунку (backend) – серцевина веб-застосунку. На цьому рівні обробляється інформація зібрана на презентаційному рівні та взаємодіє з базою даних.

3. Рівень даних, який ще називають рівнем бази даних – це місце, де зберігається та керується інформація, яка обробляється веб-застосунком.

У веб-застосунку із трирівневою структурою увесь зв’язок проходить через backend. Frontend та рівень бази даних не можуть взаємодіяти між собою на пряму [28].

На рисунку 2.3 подано сформовану структуру веб-застосунку для репетитора «ВебМентор».

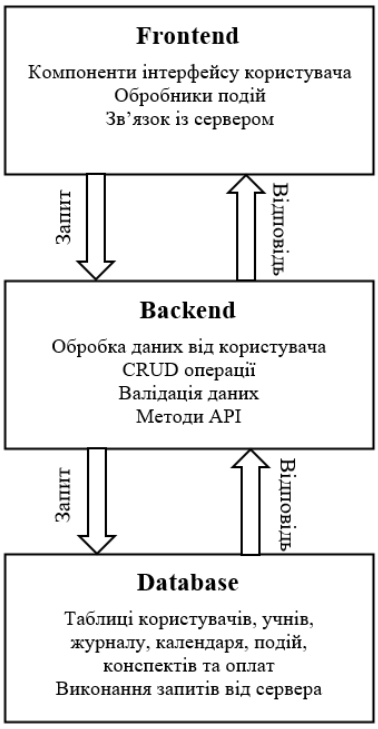

Рисунок 2.3 – Структура веб-застосунку «ВебМентор»

Frontend відповідає за інтерфейс користувача та взаємодію з ним. Використовується HTML для структурування та відображення веб-сторінок та їх контенту, CSS для стилізації зовнішнього вигляду веб-сторінок та їх елементів, JS для динамічної поведінки та взаємодії з користувачем [29]. Крім цього CSS використовує Pico – легкий і мінімалістичний фреймворк для стилізації HTML-елементів [30]. Також використовується AJAX для асинхронного обміну даними з сервером, дозволяючи оновлювати частини веб-сторінки без перезавантаження [31].

Backend забезпечує логіку та обробку даних за допомогою PHP, обробляючи запити від frontend та взаємодіючи з базою даних. Сюди входить реєстрація та авторизація користувачів, забезпечення безпеки даних [30]. PHP файли реалізують CRUD операції для маніпулювання даними в БД, це включає додавання, видалення, перегляд та редагування інформаційних таблиць учнів, журналу, календаря уроків та інших [33]. Backend також включає API, реалізоване на PHP, яке забезпечує взаємодію з frontend [34]. Також містяться конфігураційні файли для налаштування середовища та підключення до БД.

Рівень даних реалізований за допомогою MySQL та зберігає всі дані веб-застосунку. Скрипти для роботи з БД виконують запити на додавання, видалення, оновлення та отримання даних. Це забезпечує взаємодіяти з інформацією про учнів, записи про успішність учнів, план уроків, конспекти, організація подій, облік оплати занять.

2.5 Моделювання бази даних веб-застосунку для репетиторів «ВебМентор»

Моделювання бази даних є критичним етапом у розробці веб-застосунку для репетитора «ВебМентор». Структурована база даних забезпечує ефективне зберігання та доступ до даних, а також підтримує функціональність застосунку. Згідно обраних у розділі 2.1 технологій розробки, для реалізація БД обрано реляційну СКБД MySQL, яка працює на платформі веб-сервера Apache.

Apache HTTP Server – це один із найпопулярніших веб-серверів, який включає усі необхідні нефункціональні вимоги до веб-застосунку, а саме надійність, продуктивність, масштабованість тощо. [35].

2.5.1 Опис інформаційних сутностей та їх атрибути

Для моделювання БД веб-застосунку для репетитора «ВебМентор» було сформовано 7 інформаційних сутностей:

1. Users (користувачі) – зберігання інформації про користувачів системи.
2. Students (учні) – зберігання інформації про учнів певного репетитора.
3. Journal (журнал) – зберігання та відслідкування успішності учня методом запису оцінок по певних категоріях.
4. Calendar (календар уроків) – зберігання інформації щодо запланованих уроків з певними учнями.
5. Notes (конспекти) – зберігання інформації створених конспектів та корисних приміток репетитора.
6. Events (події) – зберігання інформації про заплановані справи репетитора, таких як зустрічі, перевірка домашнього завдання тощо.
7. Payments (оплата занять) – зберігання інформації про оплату учнями проведених уроків.

Відповідно до сформованих інформаційних сутностей проводимо опис їх атрибутів:

1. Users (репетитори):
   - ID (INT, PRIMARY KEY, AUTO_INCREMENT) – унікальний ідентифікатор користувача.
   - name (VARCHAR(255)) – ім'я користувача.
   - email (VARCHAR(255)) – електронна пошта користувача.
   - password (VARCHAR(255)) – пароль користувача.
2. Students (учні):

- ID (INT, PRIMARY KEY, AUTO_INCREMENT) – унікальний ідентифікатор учня.
- Lastname (VARCHAR (255)) – прізвище учня.
- Name (VARCHAR (255)) – ім'я учня.
- Fathername (VARCHAR (255)) – по-батькові учня.
- Class (VARCHAR (20)) – клас, в якому навчається учень.
- Grouptype (VARCHAR (50)) – група, до якої належить учень.
- Phone (VARCHAR (20)) – номер телефону учня.
- Address (VARCHAR (255)) – адреса проживання учня.
- Status (VARCHAR (50)) – статус учня.
- Parents (VARCHAR (255)) – імена батьків учня.
- ParentsPhone (VARCHAR (20)) – номер телефону батьків учня.
- CreatedBy (INT, FOREIGN KEY) – ідентифікатор користувача, який створив запис.

3. Journal (журнал):

- ID (INT, PRIMARY KEY, AUTO_INCREMENT) – унікальний ідентифікатор запису в журналі.
- Lastname (VARCHAR (255)) – прізвище учня.
- Name (VARCHAR (255)) – ім'я учня.
- Date (DATE) – дата запису.
- Homework (TEXT) – оцінка за домашню роботу.
- Classwork (TEXT) – оцінка за роботу на уроці.
- Test (TEXT) – оцінка за тест.
- CreatedBy (INT, FOREIGN KEY) – ідентифікатор користувача, який створив запис.

4. Calendar (календар уроків):

- ID (INT, PRIMARY KEY, AUTO_INCREMENT) – унікальний ідентифікатор запису в календарі.
- Date (DATE) – дата уроку.
- LessonTime (TIME) – час проведення уроку.

- Lastname_Group (VARCHAR(255)) – прізвище учня або група, для якої проводиться урок.
- Location (VARCHAR(255)) – місце проведення уроку.
- Notes (TEXT) – додаткові примітки.
- CreatedBy (INT, FOREIGN KEY) – ідентифікатор користувача, який створив запис.

5. Notes (конспекти):

- ID (INT, PRIMARY KEY, AUTO_INCREMENT) – унікальний ідентифікатор конспекту.
- Date (DATE) – дата створення конспекту.
- Head (VARCHAR(255)) – заголовок конспекту.
- Content (TEXT) – опис конспекту.
- Links (VARCHAR(255)) – посилання на додаткові матеріали.
- CreatedBy (INT, FOREIGN KEY) – ідентифікатор користувача, який створив запис.

6. Events (події):

- ID (INT, PRIMARY KEY, AUTO_INCREMENT) – унікальний ідентифікатор події.
- Date (DATE) – дата події.
- Time (TIME) – час проведення події.
- EventName (VARCHAR(255)) – назва події.
- Description (TEXT) – опис події.
- Location (VARCHAR(255)) – місце проведення події.
- CreatedBy (INT, FOREIGN KEY) – ідентифікатор користувача, який створив запис.

7. Payments (оплата занять):

- ID (INT, PRIMARY KEY, AUTO_INCREMENT) – унікальний ідентифікатор оплати.
- Date (DATE) – дата оплати.

- StudentsLastname (VARCHAR(255)) – прізвище учня, за якого здійснено оплату.
- Period (VARCHAR(50)) – період, за який здійснено оплату.
- Amount (FLOAT) – сума оплати.
- PaymentMethod (VARCHAR(50)) – спосіб оплати.
- Notes (TEXT) додаткові примітки щодо оплати.
- CreatedBy (INT, FOREIGN KEY) – ідентифікатор користувача, який створив запис.

Кожна таблиця БД веб-застосунку «ВебМентор», окрім таблиці «Users», має атрибут CreatedBy, що відповідає за зв'язок з репетитором, який створив ці дані. Це забезпечує можливість кожному репетитору мати доступ лише до своїх даних.

2.5.2 Формування зв’язків між таблицями

Було сформовано наступні зв’язки між таблицями:

- Users. Кожен репетитор має унікальний ідентифікатор і доступ до своїх даних. Взаємозв’язки з іншими таблицями реалізовані через зовнішні ключі CreatedBy.
- Students. Інформація про учнів використовується у таблиці «Журнал» через прізвище учня та у таблиці «Календар уроків» через групу.
- Journal. Записи в журналі пов'язані з учнями через прізвище та ім'я.
- Calendar. План уроків прив'язаний до групи учнів.
- Notes. Конспекти створюються репетиторами для власних потреб.
- Events. Організація подій створюється репетиторами для власних потреб, що може включати події, які пов’язані з учнями, але без зв’язків у БД.
- Payments. Записи про оплату пов'язані з учнями через прізвище учня.

Для наочності також створено ER-діаграму (діаграма сутностей та зв’язків), яка відображає структуру та взаємозв’язки між таблицями БД веб-застосунку «ВебМентор» [36]. ER-діаграму подано на рисунку 2.4.

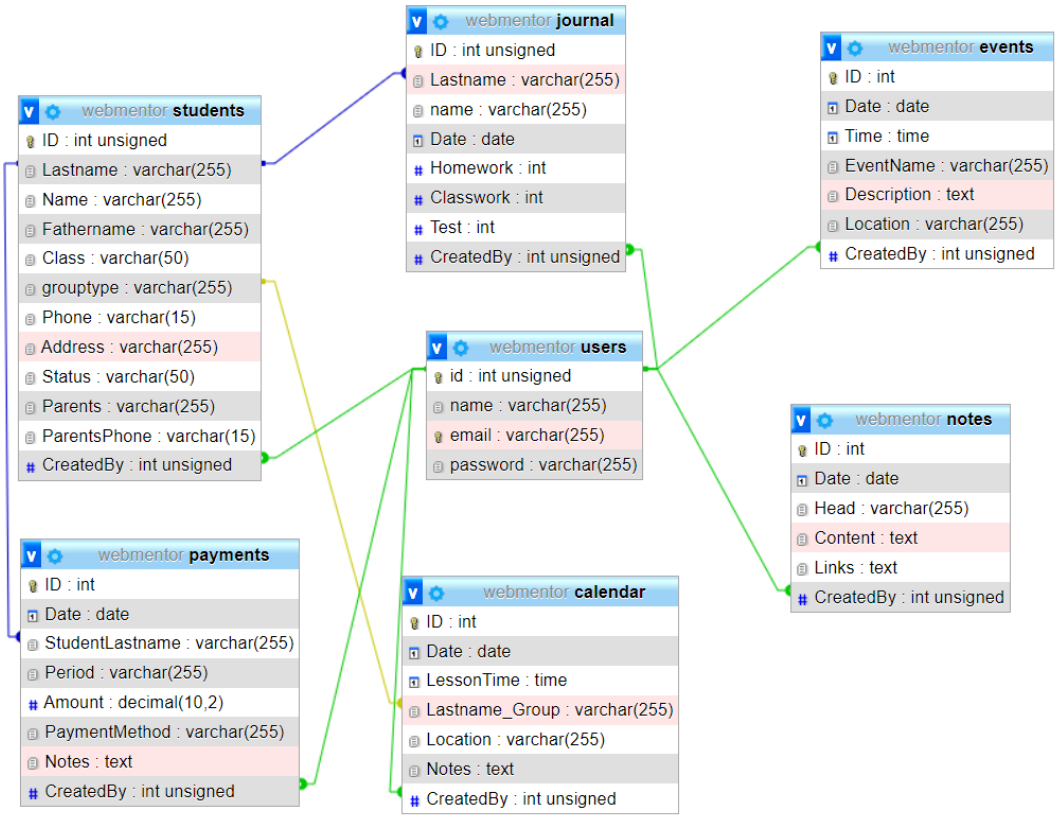

Рисунок 2.4 – ER-діаграма БД веб-застосунку «ВебМентор»

Як видно з рисунку 2.4, створена ER-діаграма дозволяє візуально представити інформаційні сутності та логічні зв’язки між ними.

2.6 Висновки до другого розділу

У другому розділі кваліфікаційної роботи розробки веб-застосунку для репетиторів «ВебМентор» було проведено проектування та моделювання веб-застосунку, який призначений для управління діяльністю репетитора. Обрані технології розробки – HTML, CSS, JavaScript, PHP та MySQL – вибрані на основі необхідності забезпечити ефективну організацію учнів, уроків, оплати та інших важливих аспектів роботи.

Для реалізації поставлених завдань використано Sublime Text як середовище розробки, що забезпечує швидку та ефективну розробку, зберігаючи баланс між функціональністю та простотою використання.

Аналіз акторів та варіантів використання виявив два основних актори системи – незареєстрований користувач та репетитор. На основі цього аналізу було сформовано варіанти використання для кожного актора, які послідовно визначають їхню взаємодію з веб-застосунком.

Структура веб-застосунку була сформована відповідно до архітектури трьох рівнів: frontend, backend та рівня даних. Frontend відповідає за інтерфейс користувача, backend забезпечує логіку та обробку даних, а рівень даних відповідає за зберігання та управління інформацією.

Моделювання бази даних було проведено з урахуванням інформаційних сутностей, атрибутів та зв'язків між таблицями. Сформована ER-діаграма дозволяє візуально представити інформаційні сутності та їхні взаємозв'язки, забезпечуючи ефективне зберігання та доступ до даних.

РОЗДІЛ 3. РОЗРОБКА ТА ТЕСТУВАННЯ ВЕБ-ЗАСТОСУНКУ ДЛЯ РЕПЕТИТОРІВ «ВЕБМЕНТОР»

3.1 Розробка інтерфейсу користувача веб-застосунку для репетиторів «ВебМентор»

Для розробки інтерфейсу користувача веб-застосунку «ВебМентор» використовуються наступні технології: HTML, CSS, Pico CSS, JS, AJAX.

Першим кроком для репетитора в системі є форма авторизації, яка подана на рисунку 3.1. Форма має поля для введення електронної пошти та пароля, а також кнопку для відправки даних. Також відображаються повідомлення про помилки, якщо вони є.

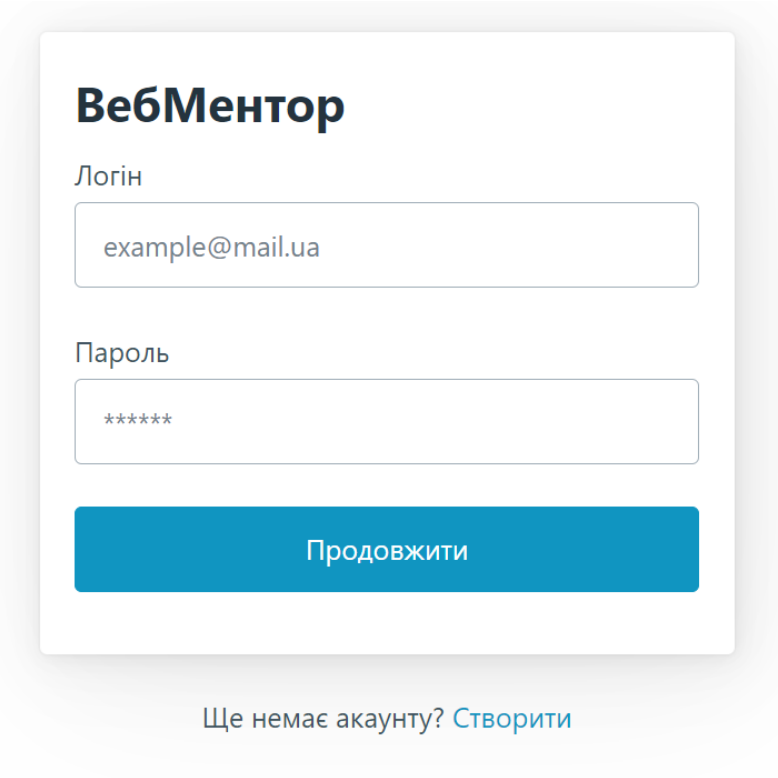

Рисунок 3.1 – Сторінка для авторизації користувача

Pico CSS додає базове оформлення, яке не потребує додаткових стилів, що спрощує створення акуратного і сучасного вигляду сторінки.

Якщо репетитор незареєстрований, він може перейти за посиланням «Створити», на якому у нас відкривається форма реєстрації нового користувача. Дана форма має таку ж саму структуру та стилі, як і форма авторизації, проте на ній розміщено більше полів введення, які включають ім’я, електронну адресу, пароль та підтвердження пароля. Також розміщено прапорець для підтвердження, що всі поля заповнені вірно.

Далі було розроблено основний інтерфейс користувача. Розглянемо його структуру, яка включає такі компоненти:

– шапка (header);
– бічне меню;
– основний вміст сторінки;
– футер.

Дана структура застосовується для усіх основних сторінок веб-застосунку для репетитора. Змінюється лише основний вміст відповідно самої сторінки. Інші елементи залишаються незмінними.

На рисунку 3.2 зображено домашню сторінку веб-застосунку.

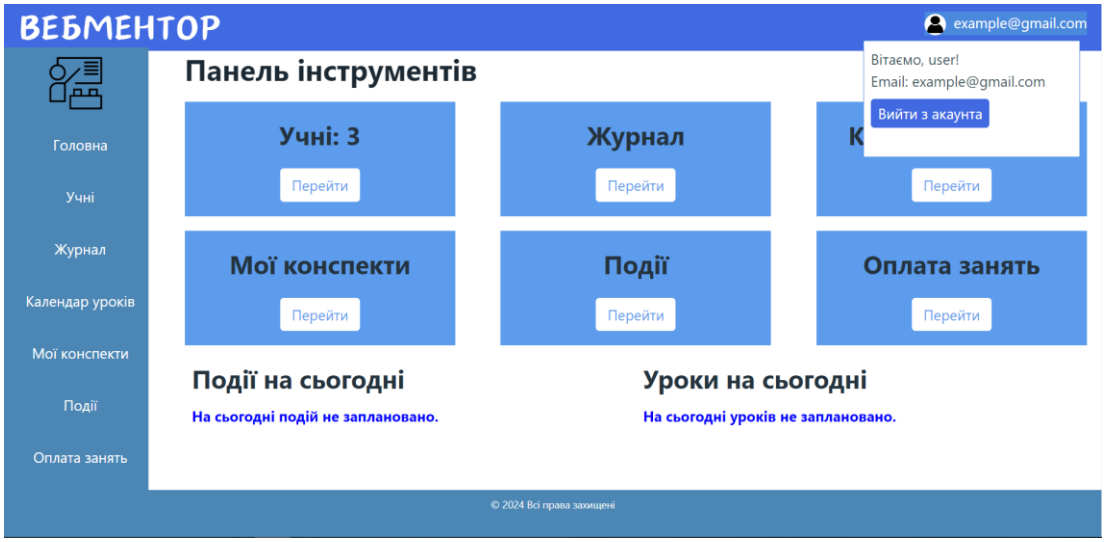

Рисунок 3.2 – Домашня сторінка веб-застосунку для репетитора

На даній формі задається вищеописана структура. Шапка сторінки включає логотип зліва та профіль користувача справа. При натисканні на

профіль користувача відкривається форма з інформацією про репетитора та можливістю виходу з акаунта. Прихована форма з інформацією про репетитора має атрибут class зі значенням profile-form та id зі значенням profile-form. Це дозволяє використовувати цей ідентифікатор для звернення до форми за допомогою JavaScript. Стилі CSS визначають її вигляд. Крім того, вона має початково прихований стан, що робить її невидимою для користувача. JavaScript використовується для керування відображенням форми при кліку на профіль користувача. Спочатку обробник події DOMContentLoaded встановлюється для виконання певних дій, коли сторінка повністю завантажена. Після цього обробник події додається до профілю користувача для відстеження подій миші. Коли користувач наводить мишу на профіль, клас highlight додається для зміни вигляду. При кліку на профіль відбувається перевірка видимості форми, яка змінюється з блокового на прихований та навпаки. Це досягається за допомогою зміни значення властивості display [37].

Меню сторінки прикріплене до лівого краю і містить пункти, що ведуть до інших сторінок веб-застосунку. Футер закріплений внизу сторінки та містить інформацію про авторські права.

Оскільки шапка, меню та футер будуть використовуватись у всіх сторінках веб-застосунку, вони були винесені у окремі файли і підключаються до сторінок за допомогою функції include_once. Таке рішення дозволяє уникнути повторення коду та забезпечує однаковий вигляд та функціонал для всіх сторінок проекту. Крім того, він полегшує зміни в дизайні або функціоналі, оскільки потрібно буде внести зміни лише у одному місці [38].

Основний вміст домашньої сторінки включає блоки з посиланнями, що відображають основні розділи, такі як студенти, журнал, календар уроків та інші. Ці блоки мають анімацію при наведенні, що збільшує їх розмір та додає тінь. Під блоками знаходиться контейнер для подій та уроків на сьогодні. Він використовує SQL-запити для отримання подій та уроків з БД і відображає їх. Якщо подій або уроків немає, відображається відповідне повідомлення.

На рисунку 3.3 зображено сторінку взаємодії з інформацією про учнів.

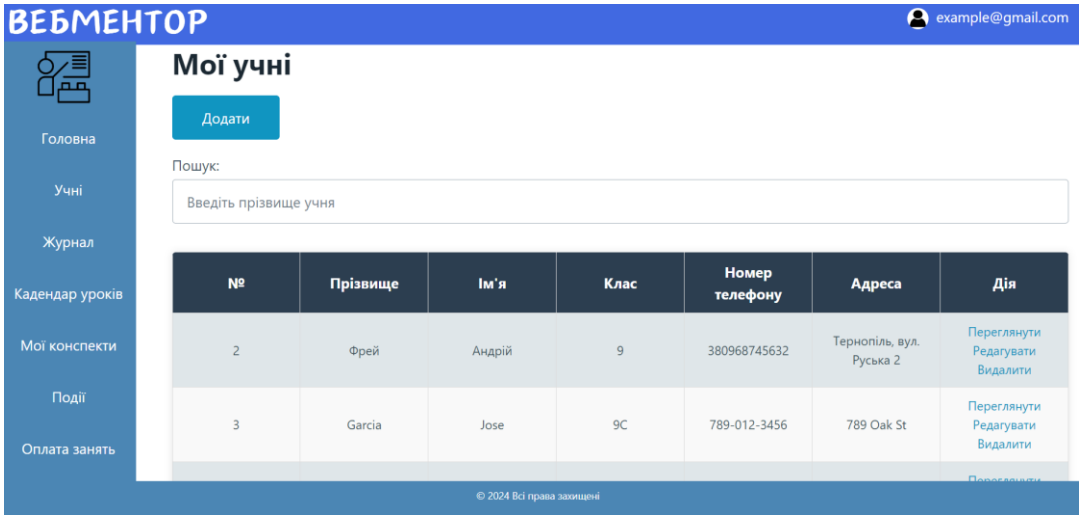

Рисунок 3.3 – Сторінка взаємодії з інформацією про учнів

На даній сторінці вгорі розміщено назву сторінки, під нею знаходиться кнопка, яка вміщує в собі посилання на сторінку додавання нового учня. Далі розміщено поле пошуку. Поле для введення тексту має подію input, яка відстежує введення тексту користувачем. Якщо поле пошуку порожнє, викликається функція getAllStudents(), яка отримує всіх учнів. Якщо введено текст, викликається функція searchStudents(searchTerm), яка виконує пошук учнів за прізвищем. getAllStudents() використовує XMLHttpRequest для відправки GET-запиту до сервера, щоб отримати всіх учнів. Отримані дані оновлюють вміст таблиці. searchStudents(searchTerm) використовує XMLHttpRequest для відправки GET-запиту з параметром пошуку до сервера, щоб отримати учнів, які відповідають введеному прізвищу [39].

Таблиця з учнями формується динамічно на основі даних з БД. Кожен рядок таблиці містить інформацію про конкретного учня. До кожного рядка таблиці застосовується останній стовпчик, який містить посилання на CRUD-операції для перегляду, редагування та видалення запису з таблиці.

Інші головні сторінки мають таку ж структуру і наповнення. Сторінки «Календар уроків» та «Події» ще містять форму для фільтрації даних за датою. На рисунку 3.4 зображено фільтрацію за датою.

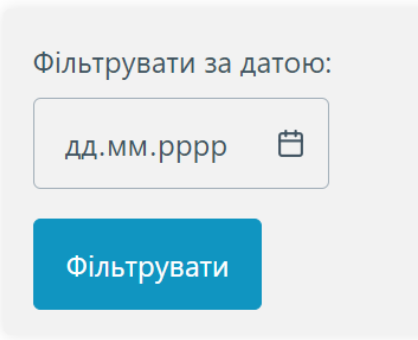

Рисунок 3.4 – Фільтрація даних за датою

Репетитор має можливість обрати дату на формі, яка йому потрібна. Коли репетитор натискає кнопку, змінна filterDate отримує значення, яке було введене в полі дати. Створюється новий об'єкт XMLHttpRequest, який відправляє GET-запит до сервера на адресу файлу filter_calendar_entries.php з параметром date, що дорівнює значенню filterDate. Коли сервер повертає відповідь (статус 200), вміст елемента з ідентифікатором students-table оновлюється отриманими даними, що містять відфільтровані записи.

Сторінки «Оплата занять» та «Журнал» мають в собі модальні вікна. На сторінці «Оплата занять», яка знаходиться на рисунку 3.5, модальне вікно відповідає за підрахунок суми оплат всіма учнями або одним індивідуально. На сторінці «Журнал» модальне вікно відповідає за розрахунок середнього балу учня за певний проміжок часу.

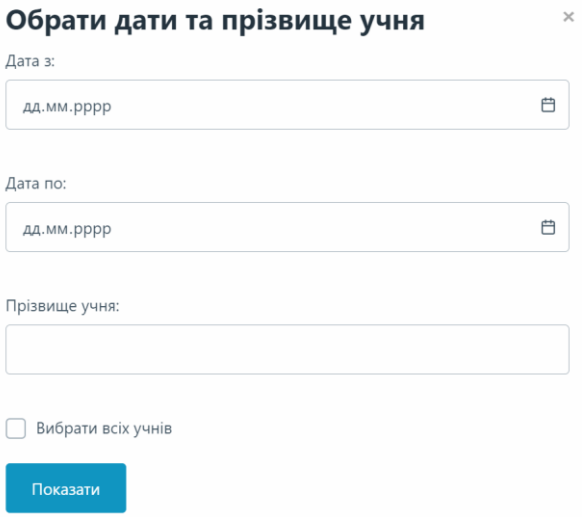

Рисунок 3.5 – Модальне вікно для підрахунку оплат учнів

Модальне вікно для оплат учнів працює наступним чином: Коли репетитор натискає кнопку «Розрахунок», модальне вікно стає видимим. Закрити вікно можна натиснувши на хрестик або клікнувши поза межами вікна. Всередині модального вікна знаходиться форма для вибору дат і прізвища учня. При натисканні кнопки «Показати» зчитуються введені дані. Якщо обраний чекбокс «Вибрати всіх учнів», відправляється запит до calculate_payments.php з відповідними параметрами, інакше перевіряється, чи введене прізвище учня. Якщо ні, виводиться повідомлення про помилку, інакше відправляється запит із введеними даними. Результати запиту відображаються у вікні.

Далі реалізовано сторінку додавання нового учня, яка зображена на рисунку 3.6.

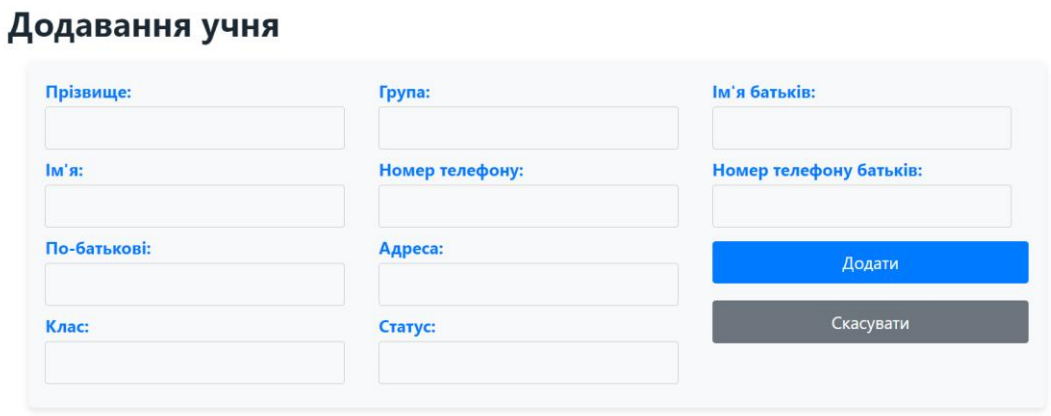

Рисунок 3.6 – Сторінка з формою додавання нового учня

HTML-форма розташована у центральній частині сторінки. Вона складається з декількох полів для введення інформації про учня і двох кнопок. CSS-стилі визначають вигляд форми, розташовуючи її по центру, з відступами, закругленими кутами, тінню, і розділяючи на три колонки. Також налаштовані стилі для міток, полів введення та кнопок, забезпечуючи приємний та зручний інтерфейс. Сторінки для додавання записів у інші таблиці мають таку ж структуру, але з назвами полів відповідно цим таблицям.

Створено також сторінку для редагування даних про наявного учня. Дана сторінка аналогічна структура та наповнення як і сторінки додавання, лише за

винятком того, що дані про учня вже наявні в полях вводу для здійснення редагування. Інші сторінки редагування записів точні такі ж, як і дана, лише з власними полями вводу.

Далі створено сторінку видалення запису з таблиці учнів, яка зображена на рисунку 3.7. Сторінки для видалення записів з інших таблиць мають аналогічну структуру та наповнення.

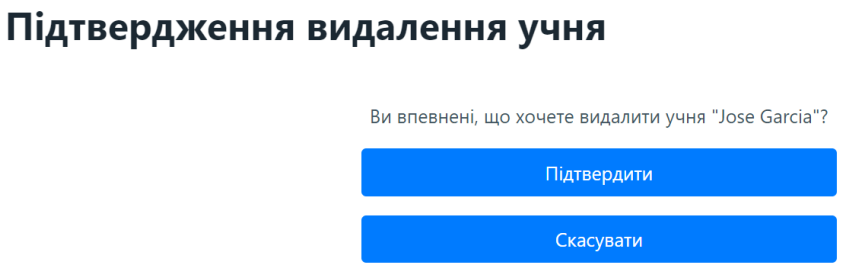

Рисунок 3.7 – Форма видалення запису з таблиці «Учні»

HTML-частина містить форму підтвердження видалення з кнопками «Підтвердити» і «Скасувати». CSS-стилі оформлюють форму, роблячи її привабливою і зручною для користувача.

3.2 Розробка серверної частини та логіки веб-застосунку для репетиторів «ВебМентор»

Наступним етапом створення веб-застосунку для репетитора «ВебМентор» є розробка серверної частини та логіки веб-застосунку. Як було описано у другому розділі, для цього використовується PHP. Для цього першим кроком є встановлення підключення до БД. Створено файл Config.php, в якому будуть встановленні параметри для підключення до БД локального сервера. Код даного файлу подано у лістингу 3.1.

Лістинг 3.1 – PHP-код конфігурації параметрів БД

```
<?php
$db_host = "localhost";
```

```
$db_user = "root";
$db_pass = "";
$db_name = "webmentor"; ?>
```

Далі було створено функцію getPDO, яка використовується для підключення до бази даних MySQL. У файлі Config.php містяться параметри підключення до бази даних, але сам по собі цей файл не встановлює з'єднання. Функція getPDO() виконує саме цю задачу, створюючи об'єкт mysqli та встановлюючи з'єднання до бази даних на основі параметрів, що знаходяться у файлі Config.php, код якого подано у лістингу 3.2.

Лістинг 3.2 – PHP-код функції getPDO()

```
function getPDO() {
    global $db_host, $db_user, $db_pass, $db_name;
    $conn = new mysqli($db_host, $db_user, $db_pass, $db_name);
    if ($conn->connect_error) {
        die("Connection failed: " . $conn->connect_error);
    }
    return $conn;
}
```

Функція getPDO() використовується для створення та повернення об'єкта mysqli, який можна використовувати для виконання запитів до бази даних. Вона відповідає за підключення до бази даних та повертає об'єкт, який можна використовувати для подальшої взаємодії з нею [40].

3.2.1 Реєстрація та авторизація репетиторів

Авторизація забезпечує безпеку доступу до системи, дозволяючи тільки зареєстрованим репетиторам входити в систему та керувати своїми даними.

Процес входу репетитора включає перевірку введених облікових даних (електронна пошта та пароль) на відповідність даним у базі даних. Якщо дані коректні, користувач авторизується та отримує доступ до свого акаунту.

Спочатку у файлі Index.php (див. додаток Б), де розміщена сама форма введення електронної адреси та паролю, вказано шлях до її обробника, а саме

файлу Login.php, який відповідає за обробку введених користувачем даних. Атрибут method=«post» вказує, що дані форми будуть відправлені методом POST, що забезпечує більшу безпеку порівняно з методом GET, оскільки дані не відображаються у URL [41].

У файлі Login.php (див. додаток Б) спочатку створені змінні, які отримають дані з полів форми за допомогою методу POST. Далі перевіряється чи електронна адреса користувача не пуста та правильно введена. Використовується функція filter_var для перевірки, чи є отримане значення дійсною електронною адресою та фільтр filter_validate_email, який повертає true, якщо значення є правильною електронною адресою, і false в іншому випадку [42]. Далі застосовуються 3 функції – setOldValue() (зберігає поточне значення поля під ключем, щоб після перезавантаження значення залишилося у полі), setValidationError() (повідомлення щодо помилки валідації для поля електронної адреси) та setMessage() (загальне повідомлення про помилку). Далі створена функція findUser(), яка шукає користувача за вказаною електронною адресою у БД. Значення передається в змінну, яка в свою чергу перевіряється. Якщо користувач не знайдений працює функція, яка виводить повідомлення про помилку. Далі перевіряється правильність вводу пароля, для цього використовується функція password_verify, яка приймає в якості аргументів пароль введений користувачем та збережений в БД хеш пароля [43]. Якщо значення паролів не співпадають, виводиться відповідне повідомлення про неправильний пароль. Якщо всі перевірки пройдено, користувач авторизується шляхом збереження його ID в сесії. Після цього користувача перенаправляють на домашню сторінку (home.php).

Процес реєстрації репетитора включає збір інформації про нового репетитора, її валідацію та збереження в БД. Репетитор заповнює форму, вказуючи своє ім'я, електронну пошту та пароль. Пароль хешується перед збереженням для забезпечення безпеки.

Таким самим чином, як і процес авторизації, у файлі Register.php (див. додаток Б) знаходиться форма з полями для заповнення в якій вказаний шлях до обробника Registerback.php методом POST.

У файлі Registerback.php, який подано у додатку Б, дані також спочатку отримуються у змінні за допомогою методу POST. Після чого система перевіряє чи поля не порожні, чи паролі співпадають та чи правильний формат електронної пошти за допомогою вищезгаданих функцій. Далі відбувається з'єднання з БД. Підготовлюється запит для вставки даних в таблицю користувачів, використовуючи методи query, prepare та stmt [44]. Відбувається хешування паролю перед його збереженням. Після цього використовується метод bind_param для прив'язки значень у підготовленому запиті. Метод execute виконує запит з даними параметрами. Якщо в ході запиту виникла помилка, виконується функція die(), яка виводить відповідне повідомлення та зупиняє виконання скрипта. Метод close зупиняє запит [45].

Відповідно, після успішної авторизації репетитор потрапляє на початкову головну сторінку веб-застосунку Home.php (див. додаток Б). У цьому файлі відбувається перевірка авторизації репетитора за допомогою функції checkAuth(). Також створюється змінна, яка приймає значення з функції currentUser() для подальшої взаємодії. Вихід з системи відбувається за кнопкою «Вийти з акаунта» – викликається обробник подій, який вказує шлях до файлу Logout.php методом POST.

У файлі Logout.php (див. додаток Б) перевіряється чи скрипт був викликаний методом POST, якщо так, спрацьовує функція logout(), яка закінчує поточну сесію та перенаправляє користувача на сторінку авторизації.

Функції getPDO(), setValidationError(), hasValidationError(), validationErrorAttr(), validationErrorMessage(), setOldValue(), old(), setMessage(), hasMessage(), getMessage(), findUser(), currentUser(), logout(), checkAuth(), checkGuest(), які згадувалися вище винесені в окремий файл Helpers.php. Даний файл знаходиться у додатку Б. Файл підключається та виконується за допомогою конструкції require_once [46].

3.2.2 CRUD операції

CRUD операції дозволяють репетиторам додавати, переглядати, редагувати та видаляти записи в базі даних. Ці операції є основою для управління даними учнів, журналом успішності, календарем уроків, конспектами, подіями та оплатами. Дані операції пояснюються на прикладі таблиці Students.

Першим кроком розглянемо процес перегляду даних. Даний фрагмент коду подано в лістингу 3.3.

Лістинг 3.3 – PHP-код вибірки інформації з таблиці Students

```
<?php
$conn = getPDO();
if (!isset($_SESSION['user'])) {
    return false;
}
$userId = $_SESSION['user']['id'] ?? null;
$stmt = $conn->prepare("SELECT * FROM Students WHERE CreatedBy =
?");
$stmt->bind_param('i', $userId);
$stmt->execute();
$result = $stmt->get_result();

while ($row = $result->fetch_assoc()) { ?>
    <tr>
        <td><?php echo $row['ID']; ?></td>
        <td><?php echo $row['Lastname']; ?></td>
        <td><?php echo $row['Name']; ?></td>
        <td><?php echo $row['Class']; ?></td>
        <td><?php echo $row['Phone']; ?></td>
        <td><?php echo $row['Address']; ?></td>
        <td>
            <a  href="viewstudent.php?id=<?php  echo  $row['ID'];
?>">Переглянути</a><br>
            <a  href="edit_student.php?id=<?php  echo  $row['ID'];
?>">Редагувати</a><br>
             <a href="delete_student.php?id=<?php echo $row['ID'];
?>">Видалити</a>
        </td>
</tr>
<?php}?>
```

Відбувається з’єднання з БД за допомогою функції getPDO(). Далі перевіряється умова чи існує сесійна змінна. Якщо ні, функція повертає значення

false, що вказує на те, що немає інформації про поточного користувача. Далі відбувається звернення до сесії, що встановлює ідентифікатор користувача у змінну userId. Після цього готується SQL-запит для вибірки усіх записів з таблиці Students, де значення стовпця CreatedBy дорівнює значенню змінної userId. Застосовується прив'язка параметра до підготовленого SQL-запиту. В даному випадку тип параметра – це ціле число, і значення – це значення змінної userId. Виконується підготовлений запит. Отримується результат запиту у вигляді об'єкта mysqli_result. Використовується цикл while для виведення даних, отриманих з запиту, у вигляді HTML-таблиці. Кожен рядок таблиці відповідає одному рядку з результатів запиту. Кожен стовпець таблиці відповідає одному полю в рядку результатів запиту. Код PHP вбудовується у HTML, щоб вивести значення кожного поля у відповідному стовпці таблиці. Також вставленні посилання в останній стовпець кожного рядка таблиці посилання для перегляду, редагування та видалення учня з таблиці. Посилання приймають параметр id зі значенням ID поточного учня. Це посилання передає ідентифікатор студента на сторінку, яка буде відкриватись.

Лістинги файлів відображення інших таблиць БД подано у додатку В.

Перегляд даних одного учня (viewstudent.php) відбувається таким самим чином як і перегляд усіх учнів, за вийнятком того, що в цьому файлі параметр учня ID отримується методом GET, тобто через URL [41]. Задається умова, яка це перевіряє, відповідно якщо параметр відсутній, користувач відправляється на сторінку із списком учнів. Лістинг файлу viewstudent.php подано у додатку В. Реалізація перегляду таблиць журналу, календаря уроків, конспектів, подій та обліку здійснено таким самим чином як і учнів.

Наступним кроком створено логіку додавання учня add_student.php. Фрагмент коду подано у лістингу 3.4.

Лістинг 3.4 – PHP-код додавання нового учня у таблицю Students

```
<?php
require_once __DIR__ . '/actions/helpers.php';
checkAuth();
```

```
$user = currentUser();
if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['add']))
{
    $lastname = $_POST['Lastname'];
    $name = $_POST['Name'];
    $fathername = $_POST['Fathername'];
    $class = $_POST['Class'];
    $grouptype = $_POST['grouptype'];
    $phone = $_POST['Phone'];
    $address = $_POST['Address'];
    $status = $_POST['Status'];
    $parents = $_POST['Parents'];
    $parentsPhone = $_POST['ParentsPhone'];
    $conn = getPDO();
    $stmt = $conn->prepare("INSERT INTO Students (Lastname, Name,
Fathername, Class, grouptype, Phone, Address, Status, Parents,
ParentsPhone, CreatedBy) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)");
    $stmt->bind_param('sssssssssi', $lastname, $name, $fathername,
$class, $grouptype, $phone, $address, $status, $parents,
$parentsPhone, $user['id']);
    $stmt->execute();
    header("Location: students.php");
    exit;
}?>
```

Відбувається підключення файлу Helpers.php з необхідними допоміжними функціями. Функція checkAuth() перевіряє чи користувач авторизований після чого отримується інформація про нього. Перевіряється умова, чи метод запиту – POST, і чи була натиснутий кнопка додавання учня. Наступні рядки отримують дані, які були введені користувачем у формі додавання нового учня. Наступні рядки готують та виконують запит до бази даних для додавання нового учня за допомогою методів prepare, bind_param та execute. Дані з форми передаються як параметри запиту, а також зберігається ідентифікатор користувача, який додав учня. Далі відбувається перенаправлення користувача на сторінку зі списком учнів після успішного додавання нового учня. Функція exit забезпечує те, що подальший виконавчий код не буде виконуватися після перенаправлення.

Реалізація файлів для додавання записів до інших таблиць здійснена таким же чином.

Далі створено логіку редагування наявних даних про учня – edit_student.php. Оновлення даних відбувається на основі опису попередніх

файлів, адже ми спочатку утворюємо вибірку даних про конкретного учня за допомогою унікального ідентифікатора ID методом GET. Після чого отримуємо дані методом форми POST та створюємо запит на оновлення (update) переліченими в описі попередніх файлів функціями та методами. Лістинг даного коду подано у додатку В. Інші файли на оновлення даних усіх таблиць БД реалізовано таким же чином.

Наступним кроком створено логіку видалення даних про учня – delete_student.php. Видалення даних відбувається також на прикладі оновлення. Спочатку отримується вибірка даних про конкретного учня допомогою унікального ідентифікатора ID методом GET. Після чого отримуємо дані методом форми POST та створюємо запит на видалення (delete) переліченими в описі попередніх файлів функціями та методами. Лістинг даного коду подано у додатку В. Інші файли на видалення даних з усіх таблиць реалізовано таким же чином

3.2.3 Пошук та фільтрація в таблицях

Пошук та фільтрація в таблицях БД веб-застосунку «ВебМентор» є невід’ємною частиною ефективної роботи репетитора. Це дозволяє легко отримати необхідний запис серед усієї таблиці.

У таблицях БД пошук відбувається за прізвищем учня, за винятком таблиці подій та конспектів, де пошук відбувається за назвою.

Логіка роботи пошуку розглядається на прикладі таблиці Students. Цей процес розділяється на 2 файли – search_student.php (для відображення результату пошуку) та get_all_students.php (для повернення таблиці у початковий стан). У файлі search_student.php відбувається перевірка аутентифікації користувача, отримання його ідентифікатора, перевірка параметра search, а також підготовка, прив’язка параметрів та виконання запиту. У запиті вибірка відбувається за прізвищем та параметром CreatedBy, що передбачає доступ до інформації лише поточному користувачу. У файлі search_student.php також

відбувається вибірка, але лише за одним параметром – CreatedBy, що повертає таблицю до початкового стану.

Лістинг даних файлів представлені у додатку Д. Файли пошуку для інших таблиць БД реалізовані таким же чином.

Фільтрація даних по даті застосовується лише у таблицях Календар уроків та Події. Розглянемо процес на прикладі таблиці Calendar, а саме файлу filter_calendar_entries.php. Фрагмент коду подано у лістингу 3.5.

Лістинг 3.5 – PHP-код фільтрації даних за датою у таблиці Calendar

```
if (isset($_GET['date'])) {
    $filterDate = $_GET['date'];
    $filterDate = date("Y-m-d", strtotime($filterDate));
    $stmt = $conn->prepare("SELECT * FROM calendar WHERE CreatedBy
= ? AND Date = ?");
    $stmt->bind_param('is', $userId, $filterDate);
    $stmt->execute();
    $result = $stmt->get_result();
} else {
    $stmt = $conn->prepare("SELECT * FROM calendar WHERE CreatedBy
= ?");
    $stmt->bind_param('i', $userId);
    $stmt->execute();
    $result = $stmt->get_result();
}
```

У цьому коді виконується перевірка, якщо у GET-запиті існує параметр date, виконується запит для вибірки записів календаря за вказаною датою для поточного користувача. Дата очищується від можливих небажаних символів та перетворюється у формат Y-m-d за допомогою strtotime() та date() [46]. Якщо параметра date немає, виконується запит для вибірки всіх записів календаря для поточного користувача. У запиті використовуються підготовлені вирази для захисту від SQL-ін'єкцій. Метод bind_param(); прив'язує параметри до запиту. Метод execute() виконує підготовлений запит. Метод get_result() отримує результат виконання запиту.

Лістинг коду фільтрації за датою таблиці подій (filter_events.php) подано у додатку Д.

3.2.4 Розрахунок середнього балу учня та підрахунок сум оплат

Розрахунок середнього балу учня проводиться за результатами його домашніх завдань, роботи на уроці та тестів для можливості відслідкування результатів успішності за певний період. Середній бал обчислюється як середнє арифметичне значень. Лістинг даного коду знаходиться у додатку Е.

Розглянемо логіку даного коду детальніше. Створюється підключення до БД за допомогою функції getPDO(). Отримуємо дані з форми через метод GET: початкову дату (startDate), кінцеву дату (endDate) та прізвище студента (studentLastName). Відбувається підготовка, прив'язка параметрів, виконання запиту та отримання результатів методами, які описані у попередніх розділах. Метод fetch_assoc() отримує рядок результату у вигляді асоціативного масиву [37]. Далі обчислюється середні бали та виводяться результати на екран.

Підрахунок оплат дозволяє вести облік оплат за заняття як для всіх учнів, так і для конкретного учня за певний період часу. Це допомагає тримати під контролем фінансові аспекти діяльності репетитора. Лістинг коду розрахунку оплат подано у додатку Е.

Даний код працює наступним чином. Створюється підключення до БД за допомогою функції getPDO(). Отримуються дані з форми через метод GET: початкова дата (startDate), кінцева дата (endDate), прізвище учня (studentLastName) та прапорець вибору всіх учнів (selectAll). Встановлюється умова, яка перевіряє, чи вказані початкова та кінцева дати. Якщо дати не вказані, виводить повідомлення про помилку. Якщо прапорець вибору всіх учнів встановлено, готується SQL-запит для отримання суми оплати для всіх учнів за вказаний період. Якщо вказане прізвище учня, готується SQL-запит для отримання суми оплати для конкретного учня за прізвищем за вказаний період. Якщо ні одне з умов не виконане, виводиться повідомлення про помилку, якщо прізвище учня не вказано і прапорець вибору всіх учнів не встановлено. Далі відбувається підготовка, прив'язка параметрів, виконання запиту та отримання

результатів методами, які описані у попередніх розділах та виводяться загальні суми оплат.

3.3 Тестування функціональних можливостей веб-застосунку для репетиторів «ВебМентор»

Першим кроком проведемо реєстрацію нового репетитора. Для цього було заповнено всі необхідні поля, обрано чекбокс та натиснуто кнопку підтвердження. Форма реєстрації зображено на рисунку 3.8.

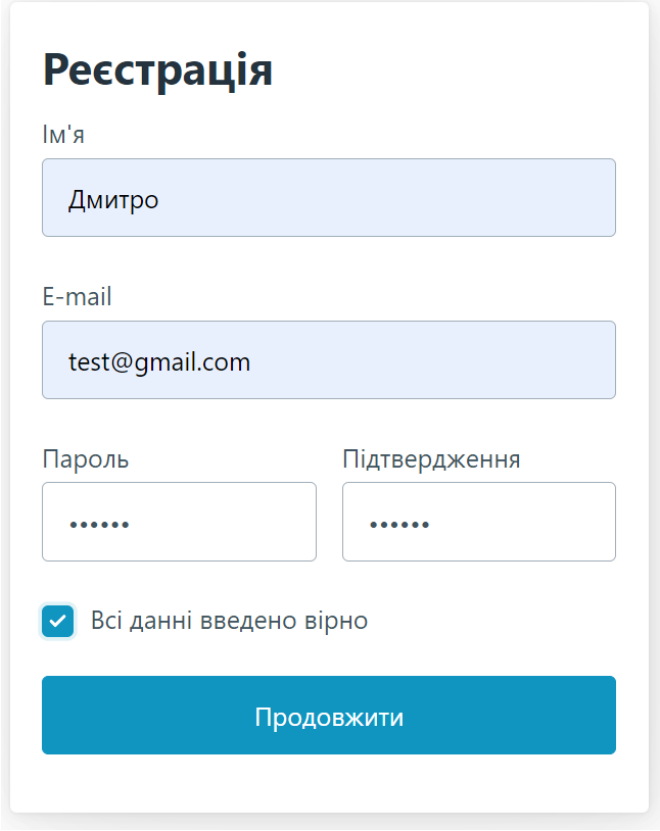

Рисунок 3.8 – Реєстрація нового користувача

Після цього авторизуємось у системі та перейдемо на головну сторінку веб-застосунку.

Підтвердити успішність реєстрації ми можемо за допомогою форми інформації про поточного репетитора у шапці сторінки, яка зображена на рисунку 3.9.

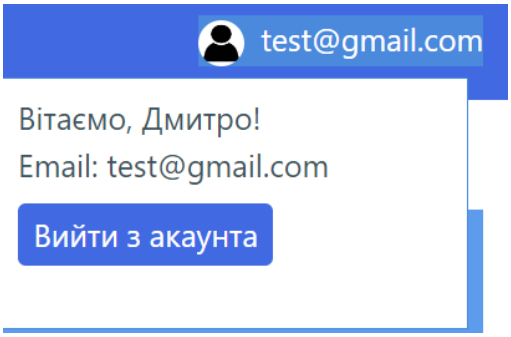

Рисунок 3.9 – Підтвердження авторизації користувача

Далі переходимо до сторінки учнів та додаємо нового учня. Після додавання інформація про нього з'являється у відповідній таблиці. Такі ж операції проводимо з усіма іншими таблицями. Додавши кілька записів проводимо пошук учня в таблиці. Приклад пошуку зображено на рисунку 3.10.

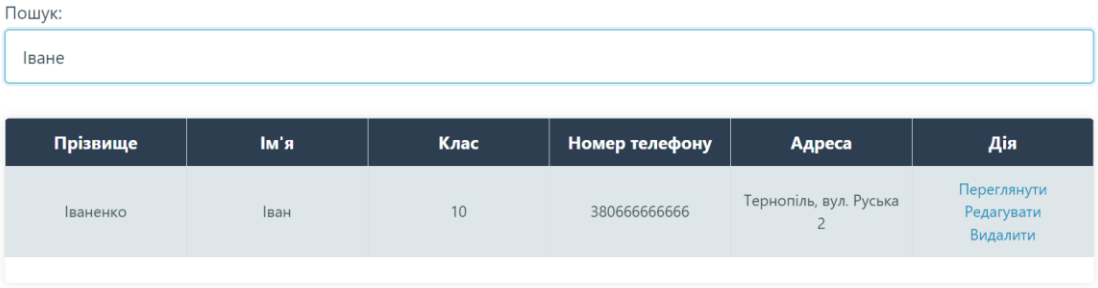

Рисунок 3.10 – Пошук учня у таблиці «Учні»

Проводимо тестування фільтрації записів на сторінках «Календар уроків» та «Події» по даті уроків та подій. Тестування успішне. Також відповідні записи про уроки та події виводяться на домашню сторінку під інформаційними блоками.

Проводимо тестування форми відслідкування успішності учня. Заповнюємо всі важливі поля та натискаємо кнопку «Показати». Приклад розрахунку зображено на рисунку 3.11).

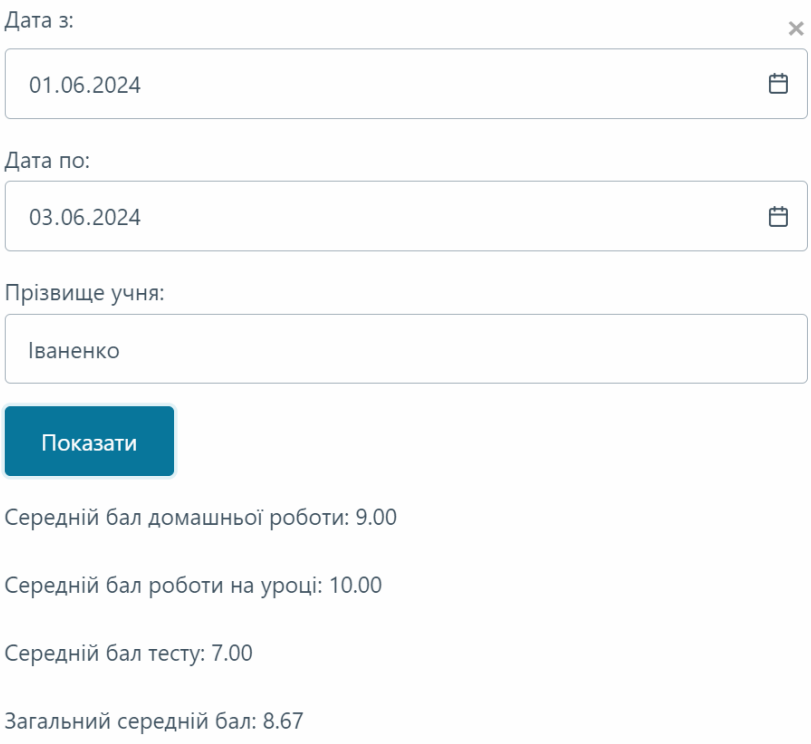

Рисунок 3.11 – Тестування вікна розрахунку успішності учня

Наступним кроком проводимо тестування розрахунку оплат учнів за певний період на сторінці «Оплата занять». Обираємо одного учня, після цього ставимо чекбокс на «Вибрати всіх учнів». Тестування успішне.

Останнім кроком проводимо тестування перегляду запису з таблиці, редагування та видалення. Усі функції працюють коректно.

Отже, можна зробити висновок, що усі функціональні можливості веб-застосунку для репетитора «ВебМентор» працюють належним чином.

3.4 Висновки до третього розділу

В розділі 3 кваліфікаційної роботи освітнього рівня «Бакалавр» було проведено розробку і тестування веб-застосунку «ВебМентор» для репетитора.

Перш за все, було розроблено інтерфейс користувача, що включає форму авторизації та основний інтерфейс з шапкою, бічним меню, основним вмістом сторінки та футером. Ця структура застосовується для усіх основних сторінок веб-застосунку. Реалізовано також сторінки для додавання, редагування та

видалення учнів, а також модальні вікна для підрахунку суми оплат та середнього балу учня.

Наступним етапом була розробка серверної частини та логіки веб-застосунку з використанням PHP. Було створено функцію для підключення до бази даних, реалізовано процеси авторизації та реєстрації користувачів, а також CRUD операції для управління даними в БД. Пошук та фільтрація в таблицях БД також були реалізовані.

Останнім етапом було тестування функціональних можливостей веб-застосунку. Тестування було зосереджено на перевірці всіх функцій, включаючи авторизацію, реєстрацію, додавання, редагування та видалення даних, а також пошук та фільтрацію в таблицях БД. Тестування дозволило перевірити працездатність та надійність веб-застосунку перед впровадженням в експлуатацію.

РОЗДІЛ 4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Роль центральної нервової системи в трудовій діяльності людини

В процесі трудової діяльності на працездатність людини впливають, зокрема, її фізичні зусилля та нервове напруження.

Фізичні зусилля виникають під час праці фізичної, коли переважає робота м'язової системи. Нервове напруження створюється під час праці розумової, коли основне навантаження падає на центральну нервову систему, її вищі відділи.

Поділ праці на фізичну і розумову є умовним, оскільки будь-яка діяльність людини не може здійснюватися без участі вищих відділів центральної нервової системи, так само як будь-яка розумова діяльність не може відбуватися без участі м'язової системи.

У трудовій діяльності людини на сучасному виробництві переважають функції управління, контролю, спостереження, які пов'язані з розумовою діяльністю. Фізичні зусилля виникають при рухах, пов'язаних з управлінням пультами машин, механізмів, переміщенням тіла у просторі та підтриманням певної робочої пози.

Тому в організмі людини особливу роль відіграє нервова система і м'язи. Нервова система пов'язує між собою, об'єднує (інтегрує) різні частини багатокліткового організму. Функції нервової системи універсальні і надзвичайно важливі. В основі діяльності нервової системи лежить універсальний фізіологічний процес розповсюдження збудження - нервовий імпульс. Нервовий імпульс, розповсюджуючись по поверхні нервової клітини і її відростків, виконує в нервовій системі функцію носія інформації. Передача інформації здійснюється з допомогою зміни або послідовності нервових імпульсів в часі, або розподілу між різними клітинами. Це так зване просторово-часове кодування. Переважною формою нервової діяльності є рефлекси - закономірні реакції, що виникають у відповідь на подразнення рецепторів

змінами у навколишньому чи внутрішньому середовищі організму. Нервова система мас відділи: спинний мозок, вегетативну систему, головний мозок, які об'єднані в центральну нервову систему. Інтегративна діяльність ЦНС зводиться до спів підпорядкування і об'єднання всіх функціональних елементів організму в цілісну систему.

Розрізняють три фази зміни функціонального стану ЦНС під час роботи:

– перша фаза інерційного гальмування на початку роботи, яка відповідає періоду втягування в роботу тривалістю від кількох хвилин до кількох десятків хвилин;

– друга фаза робочого збудження, яка відповідає стійкій працездатності, а її тривалість залежить від важкості виконуваної роботи;

– третій стан вторинного охоронного гальмування, який виникає під час роботи внаслідок розвитку втоми. Загальноприйнята теорія втоми пов'язує її розвиток із станом ЦНС.

Втома призводить до розвитку охоронного гальмування в корі головного мозку, яке запобігає розвитку виснаження організму.

Втома – це стан, що спричинюється інтенсивною і тривалою роботою, характеризується тимчасовим зменшенням працездатності, виражається зниженням кількості та якості роботи і погіршенням координації робочих функцій. Основою втоми є виснаження фізіологічних ресурсів органу, який працює чи усієї функціональної системи. Виробничі шкідливості можуть прискорювати та поглиблювати процес гальмування.

З утомою тісно пов'язаний стан перенапруження (перевтоми), який є граничним станом організму між нормою і патологією.

Боротьба з втомою і перенапруженням має проводитись комплексно, включаючи технічні, організаційні, гігієнічні та психофізіологічні заходи. Зокрема, правильний режим праці та відпочинку, механізація та автоматизація операцій, належні санітарно-гігієнічні умови тощо. [47]

4.2 Естетичне оформлення робочого місця оператора ПК

Робоче місце – це зона простору, що оснащена необхідним устаткуванням, де відбувається трудова діяльність одного працівника чи групи працівників.

Раціональне планування робочого місця має забезпечувати: найкраще розміщення знарядь і предметів праці, не допускати загального дискомфорту, зменшувати втомлюваність працівника, підвищувати його продуктивність праці. Площа робочого місця має бути такою, щоб працівник не робив зайвих рухів і не відчував незручності під час виконання роботи. Важливо мати також можливість змінити робочу позу, тобто положення корпуса, рук, ніг. Проте доцільно виключати або мінімізувати всі фізіологічно неприродні і незручні положення тіла.

Конструкція робочого столу має забезпечувати можливе розташування навчального обладнання, конструкція робочого стільця (крісла) – підтримання раціональної робочої пози під час виконання основних робочих операцій, створювати умови для зміни пози. Майстер виробничого навчання повинен відрегулювати висоту та кут нахилу сидіння і спинки відповідно до зросту і віку учня. Сидіння, спинка та підлокітники стільця мають м'яке, неслизьке, повітропроникне покриття.

Проведені дослідження показують, що при раціональній організації робочих місць продуктивність праці зростає знати на 15-25%.

Вимоги до конструкції меблів (робочий стіл, стілець (крісло), розташованих на робочих місцях учнів, які навчаються у навчальних лабораторіях з професії оператор комп’ютерного набору, визначаються вимогами ДСТУ 7299:2013. Відповідно до ДСТУ 7299:2013 екран ПК слід розташовувати на оптимальній відстані від очей учня, але не ближче 0,4 м залежно від розміру екрана монітора.

Для зручності зорового спостереження площина екрана ПК має бути перпендикулярна лінії зору, при цьому має бути передбачена можливість

переміщення монітора у вертикальній площині під кутом ± 30 град. (справа наліво).

Гігієнічні вимоги визначають умови життєдіяльності і працездатності людини у процесі взаємодії з технікою і середовищем; показниками є рівень освітлення, температура, вологість, шум, вібрація, токсичність, загазованість тощо.

Антропометричні вимоги визначають відповідність конструкцій техніки антропометричним характеристикам людини (зріст, розміри тіла та окремі рухові ланки). Показниками є раціональна робоча поза, оптимальні зони досягнення, раціональні трудові рухи.

Фізіологічні та психофізіологічні вимоги визначають відповідність техніки і середовища можливостям працівника щодо сприйняття, переробки інформації, прийняття і реалізації рішень.

Робоча поза – це основне положення працівника у просторі: зручна робоча поза має забезпечувати стійкість положення корпуса, ніг, рук, голови працівника під час роботи, мінімальні затрати енергії та максимальну результативність праці.

Найпоширенішими у процесі праці є пози сидячи і стоячи. Проектуючи робоче місце, потрібно враховувати, що при виконанні роботи з фізичним навантаженням бажана поза стоячи, а при малих зусиллях – сидячи.

Робоча поза стоячи втомлює людину більше, ніж сидяча. Вона вимагає на 10 % більше енергії, спричиняє підвищення артеріального і венозного тиску крові, розширення вен на ногах, пошкодження ступень, викривлення хребта.

Під час роботи сидячи нижня частина корпуса розслаблена, а основне статичне навантаження припадає на м'язи шиї, спини, таза, стегон. Неправильна сидяча поза може викликати застій крові в ногах, а якщо виконується великий обсяг роботи для пальців рук – запалення суглобів.

Організація робочого місця користувача комп'ютера повинна забезпечувати відповідність усіх елементів робочого місця та їх взаємного розташування ергономічним вимогам.

Виконуючи практичні завдання щодо використання робочої пози, потрібно: зменшувати величину статичних напружень; розподіляти статичні напруження; передбачати можливість змін пози під час роботи. [48]

4.3 Висновки до четвертого розділу

У четвертому розділі кваліфікаційної роботи освітнього рівня «Бакалавр» досліджено важливі питання щодо ролі центральної нервової системи в трудовій діяльності людини та естетичного оформлення робочого місця. Згідно з розглянутими даними, фізичні зусилля та нервове напруження є невід'ємною частиною трудової діяльності, проте на сучасних виробництвах переважають функції управління, контролю та спостереження, пов'язані з розумовою діяльністю. Центральна нервова система виконує ключову роль у забезпеченні цих функцій, і важливо зберігати її оптимальний стан під час праці.

Раціональне планування та організація робочого місця можуть зменшити втомленість працівника та підвищити продуктивність праці. Ергономічне розташування обладнання та можливість змінити робочу позу під час виконання роботи грають ключову роль у забезпеченні комфорту та збереженні здоров'я працівника.

Таким чином, важливо враховувати не лише фізичні аспекти трудової діяльності, але й психологічні та ергономічні чинники для забезпечення безпеки та ефективності працівників.

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи освітнього рівня «Бакалавр» було розроблено веб-застосунок для репетиторів «ВебМентор». Він створює ефективні та зручні умови для діяльності репетиторів, дозволяючи полегшити адміністративну роботи, економити час та зосередитись на навчанні учнів.

У першому розділі кваліфікаційної роботи освітнього рівня «Бакалавр»:

- проведено детальний аналіз предметної області, підкреслюючи важливість освіти в сучасному світі та роль репетиторства у підвищенні якості навчання;
- розглянуто ключові аспекти автоматизації процесів репетиторства та виявлено потребу в ефективних інструментах;
- висвітлено результати аналізу існуючих аналогів, що сприяло уточненню вимог до розробки веб-застосунку «ВебМентор»;
- проаналізовано функціональні та нефункціональні вимоги до веб-застосунку, які визначили основні функціональні можливості та вимоги до продуктивності, надійності, безпеки та зручності використання застосунку.

У другому розділі кваліфікаційної роботи:

- досліджено основні аспекти проектування та моделювання веб-застосунку «ВебМентор» для управління діяльністю репетитора;
- обґрунтовано вибір технологій розробки на основі потреб у забезпеченні ефективної організації учнів, уроків, оплати та інших аспектів;
- сформовано архітектуру та структуру веб-застосунку «ВебМентор» на основі аналізу акторів та варіантів використання.
- проведено опис інформаціних сутностей, їх атрибутів та зв’язків між ними.

У третьому розділі кваліфікаційної роботи:

– розроблено і реалізовано веб-застосунок «ВебМентор», який включає інтерфейс користувача, серверну частину та функціональності для управління діяльністю репетитора;

– спроєктовано і реалізовано функціональність для забезпечення безпеки даних та зручності використання застосунку;

– протестовано всі функціональні можливості веб-застосунку для перевірки їхньої працездатності та надійності.

У розділі «Безпека життєдіяльності, основи охорони праці»:

– висвітлено важливі аспекти забезпечення безпеки, комфорту та ефективності працівників через увагу до фізичних, психологічних та ергономічних чинників;

– підкреслено роль центральної нервової системи та естетичного оформлення робочого місця у підтримці оптимального стану працівника та підвищенні продуктивності.

ПЕРЕЛІК ДЖЕРЕЛ

1. Зростання ринку репетиторства – вирок системі освіти в Україні. Освіта нова. URL: https://osvitanova.com.ua/posts/4194-zrostannia-rynku-repetytorstva-vyrok-systemi-osvity-v-ukraini (дата звернення 04.02.2024).

2. Струтинська І. В. Ключові фактори, що сприяють розвитку цифрової економіки / І. В. Струтинська, Л. П. Дмитроца, Г. В. Козбур, У. І. Дмитрук // Тези доповідей міжнародної науково-практичної конференції «Цифрова економіка як фактор інноваційного розвитку суспільства», 11 листопада 2020 року. – Тернопіль: ТНТУ, 2020. – С. 43–45. – (Теоретичні та прикладні аспекти розвитку цифрової економіки).

3. Шведа Н. М. Важливість інноваційних методів навчання для студентів-іноземців / Наталія Шведа // V Міжнародна науково-методична конференція актуальні питання організації навчання іноземних студентів в Україні присвячена 60-річчю ТНТУ імені Івана Пулюя, 14-16 жовтня 2020 року. – Тернопіль: ТНТУ, 2020. – С. 65–67. – (Організація навчального процесу для студентів- іноземців, теоретичні та прикладні аспекти).

4. Лесів В. М. Інструменти цифрової трансформації малого та середнього бізнесу в країнах ЄС та Україні / В. М. Лесів, Л. П. Дмитроца // Тези доповідей міжнародної науково-практичної конференції «Цифрова економіка як фактор інноваційного розвитку суспільства», 11 листопада 2020 року. – Тернопіль. : ТНТУ, 2020. – С. 120. – (Міжнародні інтеграційні процеси в умовах цифрової трансформації бізнесу-науки-освіти-влади).

5. Автоматизація процесів за допомогою роботизації. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Автоматизація_процесів_за_допомогою_роботизації (дата звернення 04.02.2024)).

6. What is Notion? Notion. URL: https://www.notion.so/help/guides/what-is-notion (дата звернення 04.02.2024)

7. Notion. Notion. URL: https://www.notion.so/ (дата звернення 04.02.2024).

8. Notion: The Best Platform for Private Tutoring. Notion4Teachers. URL: https://www.notion4teachers.com/blog/notion-for-private-tutors (дата звернення 04.02.2024).

9. Філановський О. Notion, зручна програма для управління завданнями. ICOOLA: Фабрика відновлених телефонів. URL: https://icoola.ua/blog/programa-notion-na-iphone/ (дата звернення: 04.02.2024).

10. Museychuk K. Light Tutoring. App Store. URL: https://apps.apple.com/ua/app/light-tutoring/id1535036589?l=uk (дата звернення: 04.02.2024).

11. Kyrylo M. Light Tutoring - Apps on Google Play. Android Apps on Google Play. URL: https://play.google.com/store/apps/details?id=com.thelight.app&pli=1 (дата звернення: 04.02.2024).

12. Оптимізуйте ваш графік репетитора з нашим ПЗ. Програма онлайн-запису та CRM управління бізнесом – EasyWeek. URL: https://easyweek.com.ua/solutions/tutor (дата звернення 04.02.2024).

13. Функціональні та нефункціональні вимоги - Visure Solutions. Visure Solutions. URL: https://visuresolutions.com/uk/requirements-management-traceability-guide/functional-vs-non-functional-requirements/ (дата звернення 04.02.2024).

14. Інструменти та ресурси для початківців у веб-розробці - Web crafting code. Web crafting code. URL: https://webcraftingcode.com/uk/pochatok-roboty/instrumenty-ta-resursy-dlia-pochatkivtsiv-u-veb-rozrobtsi/ (дата звернення 04.02.2024).

15. Getting started with CSS-Learn web development. MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Learn/CSS/First_steps/Getting_started (дата звернення 04.02.2024).

16. CSS Flexbox (Flexible Box). W3Schools Online Web Tutorials. URL: https://www.w3schools.com/csS/css3_flexbox.asp (дата звернення 04.02.2024).

17. Buievych A. Як створити анімації CSS (з прикладами). freeCodeCamp.org. URL: https://www.freecodecamp.org/ukrainian/news/yak-stvoryty-animatsiyu-css-z-prykladamy/ (дата звернення 04.02.2024).

18. Вступ до JavaScript. Сучасний підручник з JavaScript. URL: https://uk.javascript.info/intro (дата звернення 04.02.2024).

19. FREEhost.UA. Що таке PHP. URL: https://freehost.com.ua/ukr/faq/wiki/chto-takoe-php/ (дата звернення 04.02.2024).

20. Суслов Р. Система управління базами даних MySQL. Lemon School. URL: https://lemon.school/blog/systema-upravlinnya-bazamy-danyh-mysql (дата звернення 04.02.2024).

21. Sublime Text vs VSCode - Which Editor is Better? - Ropstam Solutions Inc. Ropstam Solutions Inc. URL: https://www.ropstam.com/visual-studio-code-vs-sublime-text/ (дата звернення 04.02.2024).

22. Актор (UML) – Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Актор_(UML) (дата звернення 04.02.2024).

23. Що таке Use Case та для чого вони потрібні. Онлайн-курси від компанії QATestLab. URL: https://training.qatestlab.com/blog/technical-articles/what-is-a-use-case-and-what-are-they-for/ (дата звернення 04.02.2024).

24. Maxym Z. Варіанти використання та сценарії (Use Cases and Scenarios). Maxym Zosym. URL: https://www.maxzosim.com/use-cases-and-scenarios/ (дата звернення 05.02.2024).

25. UML Use Case Diagram Tutorial. Lucidchart. URL: https://www.lucidchart.com/pages/uml-use-case-diagram (дата звернення 05.02.2024).

26. Rational Software Architect 9.6.1. IBM in Deutschland, Österreich und der Schweiz. URL: https://www.ibm.com/docs/en/rational-soft-arch/9.6.1?topic=designer-rational-software-architect-product-overview (дата звернення 05.02.2024).

27. Триярусна архітектура – Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Триярусна_архітектура (дата звернення 05.02.2024).

28. What Is Three-Tier Architecture? | IBM. IBM in Deutschland, Österreich und der Schweiz. URL: https://www.ibm.com/topics/three-tier-architecture (дата звернення 05.02.2024).

29. Що таке фронтенд: етапи розробки та технології. Wezom. – Київ, Україна. URL: https://wezom.com.ua/ua/blog/chto-takoe-front-end-razrabotka (дата звернення 05.02.2024).

30. Larroche L. Pico CSS. Pico CSS. URL: https://picocss.com/docs/mission (дата звернення 05.02.2024).

31. AJAX Tutorial: What AJAX Is and How to Use it. freeCodeCamp.org. URL: https://www.freecodecamp.org/news/ajax-tutorial/ (дата звернення 05.02.2024).

32. Zobenko E. PHP-backend Roadmap. DEV Community. URL: https://dev.to/he110/php-backend-roadmap-2j1a (дата звернення 05.02.2024).

33. PHP MySQL CRUD Application - Tutorial Republic. Tutorial Republic - Online Web Development Tutorials. URL: https://www.tutorialrepublic.com/php-tutorial/php-mysql-crud-application.php (дата звернення 05.02.2024).

34. How To Use an API with PHP & cURL (PHP API Tutorial). Rapid Blog. URL: https://rapidapi.com/blog/how-to-use-an-api-with-php/ (дата звернення 05.02.2024).

35. FREEhost.UA. Apachi – що це. URL: https://freehost.com.ua/ukr/faq/wiki/apache-chto-eto/ (дата звернення 05.02.2024).

36. Як створити ER-діаграму для бази даних. ShallBD: Your Trading Navigator in Forex and Binary Options. URL: https://shallbd.com/uk/diznaitesia-iak-krok-za-krokom-stvoriti-er-diagramu-dlia-bazi-danikh/ (дата звернення 05.02.2024).

37. JavaScript DOMContentLoaded Event. JavaScript Tutorial. URL: https://www.javascripttutorial.net/javascript-dom/javascript-domcontentloaded/ (дата звернення 06.02.2024).

38. PHP: include_once - Manual. PHP: Hypertext Preprocessor. URL: https://www.php.net/manual/en/function.include-once.php (дата звернення 06.02.2024).

39. XMLHttpRequest – Web APIs. MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest (дата звернення 06.02.2024).

40. PHP MySQLi Functions. W3Schools Online Web Tutorials. URL: https://www.w3schools.com/PHP/php_ref_mysqli.asp (дата звернення 06.02.2024).

41. GET y POST en PHP. Desarrollo web en PHP y SEO - Diego Lázaro. URL: https://diego.com.es/get-y-post-en-php (дата звернення 06.02.2024).

42. Sebhastian N. PHP validate email address (with code examples). Sebhastian. URL: https://sebhastian.com/php-validate-email/ (дата звернення 06.02.2024).

43. PHP password_verify() Function. Online Tutorials. | Tutorialspoint. URL: https://www.tutorialspoint.com/php/php_function_password_verify.htm (дата звернення 06.02.2024).

44. PHP MySQL: Insert Data Into a Table. MySQL Tutorial. URL: https://www.mysqltutorial.org/php-mysql/php-insert-data-into-mysql-table/ (дата звернення 06.02.2024).

45. Why include __DIR__ in the require_once?. Stack Overflow. URL: https://stackoverflow.com/questions/32444572/why-include-dir-in-the-require-once (дата звернення 06.02.2024).

46. Formating date string with strtotime and date. Stack Overflow. URL: https://stackoverflow.com/questions/24094571/formating-date-string-with-strtotime-and-date (дата звернення 06.02.2024).

47. Копил В. В. Роль центральної нервової системи в трудовій діяльності працівників на об'єктах галузі за фахом // Охорона праці в галузі освіти: теоретичні і технологічні аспекти : матеріали Всеукраїнської науково-практичної конференції (7–8 квітня 2014 р.). – Полтава : ПНПУ. – С. 243-246.

48. Геврик Є. О. Охорона праці / Є. О. Геврик . – К.: Ельга, Ніка-Центр, 2003 – 280 с.

ДОДАТКИ

Додаток А

Таблиця 1 – Варіанти використання для актора «репетитор»

Актор	**Найменування**	**Формулювання**
Репетитор	Авторизація	Дозволяє увійти у акаунт веб-застосунку
	Перегляд головної сторінки	Дозволяє переглянути головну сторінку веб-застосунку
	Перегляд сторінки обліку учнів	Дозволяє переглянути сторінку з інформацією про учнів
	Пошук учня за прізвищем	Дозволяє здійснити пошук учня в таблиці за його прізвищем
	Додавання учня	Дозволяє додати нового учня
	Перегляд інформації про учня	Дозволяє переглянути повну інформацію про конкретного учня
	Редагування інформації про учня	Дозволяє змінити дані про учня
	Видалення учня	Дозволяє видалити учня
	Виставлення оцінок для учня	Дозволяє ставити оцінки учню за критеріями
	Перегляд успішності учня	Дозволяє переглянути інформацію щодо оцінок учня
	Розрахунок успішності за певний період	Дозволяє переглянути середній бал успішності за певний проміжок часу
	Видалення оцінок	Дозволяє видалити оцінку

Продовження таблиці А.1.2

Зареєстрований користувач	Планування уроків	Дозволяє створити запис про наступний урок, за потреби редагувати запис
	Пошук уроку	Дозволяє знайти запис про урок за назвою групи чи прізвищем учня або за датою проведення
	Видалення уроку	Дозволяє видалити запис про урок
	Планування подій	Дозволяє створити запис про подію, за потреби редагувати запис
	Пошук події	Дозволяє знайти подію за назвою або датою
	Видалення події	Дозволяє видалити запис про подію
	Створення конспектів	Дозволяє створити власний конспект
	Перегляд конспектів	Дозволяє переглянути конспект
	Редагування конспектів	Дозволяє редагувати конспект
	Видалення конспектів	Дозволяє видалити конспект
	Ведення обліку оплат уроків	Дозволяє записувати оплати учнів за проведені уроки
	Редагування запису про оплату	Дозволяє редагувати запис про оплату
	Видалення запису про оплату	Дозволяє видалити запис про оплату за потреби

Продовження таблиці А.1

Зареєстрований користувач	Пошук запису оплати	Дозволяє знайти запис у таблиці за прізвищем учня або за датою
	Підрахунок оплати	Дозволяє підрахувати суму заробітку за певний період часу загалом або за конкретним учнем
	Вихід	Дозволяє вийти із свого акаунту

Додаток Б

Програмний код авторизації та реєстрації веб-застосунку для репетитора «ВебМентор»

Index.php:

```
<?php
require_once __DIR__ . '/actions/helpers.php';
checkGuest();
?>
<!DOCTYPE html>
<html lang="en" data-theme="light">
<?php include_once __DIR__ . '/components/head.php'?>
<body>
<form class="card" action="/actions/login.php" method="post">
    <h2>ВебМентор</h2>
    <?php if(hasMessage('error')): ?>
        <div class="notice error"><?php echo getMessage('error')
?></div>
    <?php endif; ?>
    <label for="email">
        Логін
        <input
            type="text"
            id="email"
            name="email"
            placeholder="example@mail.ua"
            value="<?php echo old('email') ?>"
            <?php echo validationErrorAttr('email'); ?>
        >
        <?php if(hasValidationError('email')): ?>
            <small><?php echo validationErrorMessage('email');
?></small>
        <?php endif; ?>
    </label>
    <label for="password">
        Пароль
        <input
            type="password"
            id="password"
            name="password"
            placeholder="******"
        >
    </label>
    <button
        type="submit"
        id="submit"
    >Продовжити</button>
</form>
<p>Ще немає акаунту? <a href="/register.php">Створити</a></p>
<?php include_once __DIR__ . '/components/scripts.php' ?>
</body> </html>
```

Login.php:

```
<?php
require_once __DIR__ . '/helpers.php';
```

```
$email = $_POST['email'] ?? null;
$password = $_POST['password'] ?? null;
if (empty($email) || !filter_var($email, FILTER_VALIDATE_EMAIL)) {
    setOldValue('email', $email);
    setValidationError('email', 'Неправильний формат електронної
адреси');
    setMessage('error', 'Помилка');
    redirect('/index.php');
}
$user = findUser($email);
if (!$user) {
    setMessage('error', "Корстувач $email не знайдений");
    redirect('/index.php');
}
if (!password_verify($password, $user['password'])) {
    setMessage('error', 'Неправильний пароль');
    redirect('/index.php');
}
$_SESSION['user']['id'] = $user['id'];
redirect('/home.php');
```

Register.php:

```
<?php
require_once __DIR__ . '/actions/helpers.php';
checkGuest();
?>
<!DOCTYPE html>
<html lang="en" data-theme="light">
<?php include_once __DIR__ . '/components/head.php'?>
<body>
<form class="card" action="/actions/registerback.php" method="post"
enctype="multipart/form-data">
    <h2>Реєстрація</h2>
    <label for="name">
        Ім'я
        <input
            type="text"
            id="name"
            name="name"
            placeholder="Введіть Ваше ім'я"
            value="<?php echo old('name') ?>"
            <?php echo validationErrorAttr('name'); ?>
        >
        <?php if(hasValidationError('name')): ?>
            <small><?php    echo    validationErrorMessage('name');
?></small>
        <?php endif; ?>
    </label>
    <label for="email">
        E-mail
        <input
            type="text"
            id="email"
            name="email"
            placeholder="example@mail.ua"
            value="<?php echo old('email') ?>"
```

```
            <?php echo validationErrorAttr('email'); ?>
        >
        <?php if(hasValidationError('email')): ?>
            <small><?php   echo   validationErrorMessage('email');
?></small>
        <?php endif; ?>
    </label>
    <div class="grid">
        <label for="password">
            Пароль
            <input
                type="password"
                id="password"
                name="password"
                placeholder="******"
                <?php echo validationErrorAttr('password'); ?>
            >
            <?php if(hasValidationError('password')): ?>
                <small><?php                                    echo
validationErrorMessage('password'); ?></small>
            <?php endif; ?>
        </label>
        <label for="password_confirmation">
            Підтвердження
            <input
                type="password"
                id="password_confirmation"
                name="password_confirmation"
                placeholder="******"
            >
        </label>
    </div>
    <fieldset>
        <label for="terms">
            <input
                type="checkbox"
                id="terms"
                name="terms"
            >
            Всі данні введено вірно
        </label>
    </fieldset>
    <button
        type="submit"
        id="submit"
        disabled
    >Продовжити</button>
</form>
<p>У мене вже є <a href="/index.php">акаунт</a></p>
<?php include_once __DIR__ . '/components/scripts.php' ?>
</body>
</html>
```

Registerback.php:

```
<?php
require_once __DIR__ . '/helpers.php';
```

```
$name = $_POST['name'] ?? null;
$email = $_POST['email'] ?? null;
$password = $_POST['password'] ?? null;
$passwordConfirmation = $_POST['password_confirmation'] ?? null;
if (empty($name)) {
    setValidationError('name', 'Невірне ім\'я');
}
if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
    setValidationError('email', 'Вказано неправильну пошту');
}
if (empty($password)) {
    setValidationError('password', 'Пароль порожній');
}
if ($password !== $passwordConfirmation) {
    setValidationError('password', 'Паролі не співпадають');
}
if (!empty($_SESSION['validation'])) {
    setOldValue('name', $name);
    setOldValue('email', $email);
    redirect('/register.php');
}
$conn = new mysqli($db_host, $db_user, $db_pass, $db_name);
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
$query = "INSERT INTO users (name, email, password) VALUES (?, ?,
?)";
$stmt = $conn->prepare($query);
if (!$stmt) {
    die("Помилка підготовки запиту: " . $conn->error);
}
$passwordHash = password_hash($password, PASSWORD_DEFAULT);
$stmt->bind_param("sss", $name, $email, $passwordHash);
if (!$stmt->execute()) {
    die("Помилка виконання запиту: " . $stmt->error);
}
$stmt->close();
$conn->close();
redirect('/index.php');
?>
```

Home.php:

```
<?php
require_once __DIR__ . '/actions/helpers.php';
checkAuth();
$user = currentUser();
$conn = getPDO();
if (!isset($_SESSION['user'])) {
    return false;
}
$userId = $_SESSION['user']['id'] ?? null;
$stmt = $conn->prepare("SELECT COUNT(*) as total FROM Students WHERE
CreatedBy = ?");
$stmt->bind_param('i', $userId);
$stmt->execute();
$result = $stmt->get_result();
$row = $result->fetch_assoc();
```

```
$totalStudents = $row['total'];
?>

<!DOCTYPE html>
<html lang="ru" data-theme="light">
<?php include_once __DIR__ . '/components/head.php'?>
<body>
            <div class="today-info">
                <!-- Форма для відображення подій -->
                <div class="event-form">
                    <h2>Події на сьогодні</h2>
                    <?php
                    $currentDate = date('Y-m-d');
                    $stmt = $conn->prepare("SELECT * FROM Events
WHERE Date(Date) = ?");
                    $stmt->bind_param('s', $currentDate);
                    $stmt->execute();
                    $result = $stmt->get_result();
                    if ($result->num_rows > 0) {
                        while ($row = $result->fetch_assoc()) {
                            echo "<p><strong style=\"color:
blue;\">Час події:</strong> {$row['Time']}, <strong style=\"color:
blue;\">Назва:</strong> {$row['EventName']}</p>";
                        }
                    } else {
                        echo "<p><strong style=\"color: blue;\">На
сьогодні подій не заплановано.</strong></p>";
                    }
                    ?>
                </div>
                <div class="lesson-form">
                    <h2>Уроки на сьогодні</h2>
                    <?php
                    $stmt = $conn->prepare("SELECT * FROM Calendar
WHERE Date(Date) = ?");
                    $stmt->bind_param('s', $currentDate);
                    $stmt->execute();
                    $result = $stmt->get_result();
                    if ($result->num_rows > 0) {
                        while ($row = $result->fetch_assoc()) {
                            echo "<p><strong style=\"color:
blue;\">Час уроку:</strong> {$row['LessonTime']}, <strong
style=\"color: blue;\">Прізвище/Група:</strong>
{$row['Lastname_Group']}</p>";
                        }
                    } else {
                        echo "<p><strong style=\"color: blue;\">На
сьогодні уроків не заплановано.</strong></p>";
                    }
                    ?>
                </div>
            </div>
        </div>
    </div>
</main>
<?php include_once __DIR__ . '/components/scripts.php' ?>
```

```
</body>
</html>
```

Logout.php:

```
<?php
require_once __DIR__ . '/helpers.php';
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    logout();
}
redirect('/index.php');
```

Helpers.php:

```
<?php
session_start();
include 'config.php';
function getPDO() {
    global $db_host, $db_user, $db_pass, $db_name;
    $conn = new mysqli($db_host, $db_user, $db_pass, $db_name);
    if ($conn->connect_error) {
        die("Connection failed: " . $conn->connect_error);
    }
    return $conn;
}
function redirect(string $path)
{
    header("Location: $path");
    die();
}
function setValidationError(string $fieldName, string $message):
void
{
    $_SESSION['validation'][$fieldName] = $message;
}
function hasValidationError(string $fieldName): bool
{
    return isset($_SESSION['validation'][$fieldName]);
}
function validationErrorAttr(string $fieldName): string
{
    return isset($_SESSION['validation'][$fieldName]) ? 'aria-
invalid="true"' : '';
}
function validationErrorMessage(string $fieldName): string
{
    $message = $_SESSION['validation'][$fieldName] ?? '';
    unset($_SESSION['validation'][$fieldName]);
    return $message;
}
function setOldValue(string $key, $value): void
{
    $_SESSION['old'][$key] = $value;
}

function old(string $key)
{
    $value = $_SESSION['old'][$key] ?? '';
    unset($_SESSION['old'][$key]);
```

```
    return $value;
}
function setMessage(string $key, string $message): void
{
    $_SESSION['message'][$key] = $message;
}
function hasMessage(string $key): bool
{
    return isset($_SESSION['message'][$key]);
}
function getMessage(string $key): string
{
    $message = $_SESSION['message'][$key] ?? '';
    unset($_SESSION['message'][$key]);
    return $message;
}
function findUser(string $email)
{
    $conn = getPDO();
    if (!$conn) {
        die("Помилка отримання об'єкта PDO");
    }
    $stmt = $conn->prepare("SELECT * FROM users WHERE email = ?");
    $stmt->bind_param('s', $email);
    $stmt->execute();
    $result = $stmt->get_result()->fetch_assoc();
    if ($result) {
        return $result;
    } else {
        return false;
    }
}
function currentUser()
{
    $conn = getPDO();
    if (!isset($_SESSION['user'])) {
        return false;
    }
    $userId = $_SESSION['user']['id'] ?? null;
    $stmt = $conn->prepare("SELECT * FROM users WHERE id = ?");
    $stmt->bind_param('i', $userId);
    $stmt->execute();
    $result = $stmt->get_result()->fetch_assoc();
    if ($result) {
        return $result;
    } else {
        return false;
    }
}
function logout(): void
{
    unset($_SESSION['user']['id']);
    redirect('/');
}
function checkAuth(): void
```

```
{
    if (!isset($_SESSION['user']['id'])) {
        redirect('/');
    }
}
function checkGuest(): void
{
    if (isset($_SESSION['user']['id'])) {
        redirect('/home.php');
    }
}
```

Додаток В

Програмний код CRUD операцій веб-застосунку для репетитора «ВебМентор»

Journal.php:

```
<?php
require_once __DIR__ . '/actions/helpers.php';
checkAuth();
$user = currentUser();
?>
<?php
$conn = getPDO();
if (!isset($_SESSION['user'])) {
    return false;
}
$userId = $_SESSION['user']['id'] ?? null;
$stmt = $conn->prepare("SELECT * FROM Journal WHERE CreatedBy = ?");
$stmt->bind_param('i', $userId);
$stmt->execute();
$result = $stmt->get_result();
while ($row = $result->fetch_assoc()) {
    // Обчислюємо середній бал
    $averageGrade = ($row['Homework'] + $row['Classwork'] +
$row['Test']) / 3;
?>
<tr>
    <td><?php echo $row['ID']; ?></td>
    <td><?php echo $row['Lastname']; ?></td>
    <td><?php echo $row['name']; ?></td>
    <td><?php echo $row['Date']; ?></td>
    <td><?php echo $row['Homework']; ?></td>
    <td><?php echo $row['Classwork']; ?></td>
    <td><?php echo $row['Test']; ?></td>
    <td><?php echo number_format($averageGrade, 2); ?></td>
    <td>
        <a href="viewjournal.php?id=<?php echo $row['ID'];
?>">Переглянути</a><br>
        <a href="edit_journal.php?id=<?php echo $row['ID'];
?>">Редагувати</a><br>
        <a href="delete_journal.php?id=<?php echo $row['ID'];
?>">Видалити</a>
    </td>
</tr>
<?php } ?>
```

Calendar.php:

```
<?php
require_once __DIR__ . '/actions/helpers.php';
checkAuth();
$user = currentUser();
?>
<?php
$conn = getPDO();

if (!isset($_SESSION['user'])) {
```

```
    return false;
}

$userId = $_SESSION['user']['id'] ?? null;

$stmt = $conn->prepare("SELECT * FROM calendar WHERE CreatedBy =
?");
$stmt->bind_param('i', $userId);
$stmt->execute();
$result = $stmt->get_result();

while ($row = $result->fetch_assoc()) { ?>
    <tr>
        <td><?php echo $row['ID']; ?></td>
        <td><?php echo $row['Date']; ?></td>
        <td><?php echo $row['LessonTime']; ?></td>
        <td><?php  echo  $row['Lastname_Group']  ??  $row['Group'];
?></td>
        <td><?php echo $row['Location']; ?></td>
        <td><?php echo $row['Notes']; ?></td>
        <td>
            <a  href="view_calendar.php?id=<?php  echo  $row['ID'];
?>">Переглянути</a><br>
            <a  href="edit_calendar.php?id=<?php  echo  $row['ID'];
?>">Редагувати</a><br>
            <a href="delete_calendar.php?id=<?php echo $row['ID'];
?>">Видалити</a>
        </td>
    </tr>
<?php } ?>
```

Notes.php:

```
<?php
require_once __DIR__ . '/actions/helpers.php';
checkAuth();
$user = currentUser();
?>
<?php
$conn = getPDO();

if (!isset($_SESSION['user'])) {
    return false;
}

$userId = $_SESSION['user']['id'] ?? null;

$stmt = $conn->prepare("SELECT * FROM notes WHERE CreatedBy = ?");
$stmt->bind_param('i', $userId);
$stmt->execute();
$result = $stmt->get_result();

while ($row = $result->fetch_assoc()) { ?>
    <tr>
        <td><?php echo $row['ID']; ?></td>
        <td><?php echo $row['Date']; ?></td>
        <td><?php echo $row['Head']; ?></td>
```

```
        <td><?php echo $row['Links']; ?></td>
        <td>
            <a   href="view_note.php?id=<?php   echo   $row['ID'];
?>">Переглянути</a><br>
            <a   href="edit_note.php?id=<?php   echo   $row['ID'];
?>">Редагувати</a><br>
            <a   href="delete_note.php?id=<?php   echo   $row['ID'];
?>">Видалити</a>
        </td>
    </tr>
<?php } ?>
```

Events.php:

```
<?php
require_once __DIR__ . '/actions/helpers.php';
checkAuth();
$user = currentUser();
?>
<?php
$conn = getPDO();

if (!isset($_SESSION['user'])) {
    return false;
}

$userId = $_SESSION['user']['id'] ?? null;

$stmt = $conn->prepare("SELECT * FROM events WHERE CreatedBy = ?");
$stmt->bind_param('i', $userId);
$stmt->execute();
$result = $stmt->get_result();

while ($row = $result->fetch_assoc()) { ?>
    <tr>
        <td><?php echo $row['ID']; ?></td>
        <td><?php echo $row['Date']; ?></td>
        <td><?php echo $row['Time']; ?></td>
        <td><?php echo $row['EventName']; ?></td>
        <td><?php echo $row['Location']; ?></td>
        <td>
            <a   href="view_events.php?id=<?php   echo   $row['ID'];
?>">Переглянути</a><br>
            <a   href="edit_event.php?id=<?php   echo   $row['ID'];
?>">Редагувати</a><br>
            <a   href="delete_event.php?id=<?php   echo   $row['ID'];
?>">Видалити</a>
        </td>
    </tr>
<?php } ?>
```

Payments.php:

```
<?php
require_once __DIR__ . '/actions/helpers.php';
checkAuth();
$user = currentUser();
?>
<?php
```

```
$conn = getPDO();

if (!isset($_SESSION['user'])) {
    return false;
}

$userId = $_SESSION['user']['id'] ?? null;

$stmt = $conn->prepare("SELECT * FROM Payments WHERE CreatedBy =
?");
$stmt->bind_param('i', $userId);
$stmt->execute();
$result = $stmt->get_result();

while ($row = $result->fetch_assoc()) { ?>
    <tr>
        <td><?php echo $row['ID']; ?></td>
        <td><?php echo $row['Date']; ?></td>
        <td><?php echo $row['StudentLastname']; ?></td>
        <td><?php echo $row['Period']; ?></td>
        <td><?php echo $row['Amount']; ?></td>
        <td><?php echo $row['PaymentMethod']; ?></td>
        <td><?php echo $row['Notes']; ?></td>
        <td>
            <a  href="edit_payments.php?id=<?php  echo  $row['ID'];
?>">Редагувати</a><br>
            <a href="delete_payments.php?id=<?php echo $row['ID'];
?>">Видалити</a>
        </td>
    </tr>
<?php } ?>
```

Viewstudent.php

```
<?php
require_once __DIR__ . '/actions/helpers.php';
checkAuth();
$user = currentUser();
if (!isset($_GET['id'])) {
    // Якщо параметр id відсутній, перенаправляємо користувача на
сторінку з учнями
    header("Location: students.php");
    exit;
}
$studentId = $_GET['id'];
$conn = getPDO();
$stmt = $conn->prepare("SELECT * FROM Students WHERE id = ? AND
CreatedBy = ?");
$stmt->bind_param('ii', $studentId, $_SESSION['user']['id']);
$result = $stmt->get_result();
if ($result->num_rows === 0) {
    // Якщо учень не знайдений або не належить поточному користувачу,
перенаправляємо користувача на сторінку з учнями
    header("Location: students.php");
    exit;
}
$student = $result->fetch_assoc();
```

```
?>
```

Edit_student.php:

```
<?php
require_once __DIR__ . '/actions/helpers.php';
checkAuth();
$user = currentUser();
if (!isset($_GET['id'])) {
    // Якщо параметр id відсутній, перенаправляємо користувача на
сторінку з учнями
    header("Location: students.php");
    exit;
}
$studentId = $_GET['id'];
$conn = getPDO();
$stmt = $conn->prepare("SELECT * FROM Students WHERE id = ? AND
CreatedBy = ?");
$stmt->bind_param('ii', $studentId, $_SESSION['user']['id']);
$stmt->execute();
$result = $stmt->get_result();
if ($result->num_rows === 0) {
    // Якщо учень не знайдений або не належить поточному користувачу,
перенаправляємо користувача на сторінку з учнями
    header("Location: students.php");
    exit;
}
$student = $result->fetch_assoc();
if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['edit']))
{
    // Отримання даних з форми
    $lastname = $_POST['Lastname'];
    $name = $_POST['Name'];
    $fathername = $_POST['Fathername'];
    $class = $_POST['Class'];
    $grouptype = $_POST['grouptype'];
    $phone = $_POST['Phone'];
    $address = $_POST['Address'];
    $status = $_POST['Status'];
    $parents = $_POST['Parents'];
    $parentsPhone = $_POST['ParentsPhone'];
    $stmt = $conn->prepare("UPDATE Students SET Lastname = ?, Name
= ?, Fathername = ?, Class = ?, grouptype = ?, Phone = ?, Address =
?, Status = ?, Parents = ?, ParentsPhone = ? WHERE ID = ?");
    $stmt->bind_param('sssssssssi', $lastname, $name, $fathername,
$class,   $grouptype,   $phone,   $address,   $status,   $parents,
$parentsPhone, $studentId);
    $stmt->execute();
    header("Location: students.php");
    exit;
}
?>
```

Delete_student.php:

```
<?php
require_once __DIR__ . '/actions/helpers.php';
checkAuth();
$user = currentUser();
```

```
if (!isset($_GET['id'])) {
    // Якщо параметр id відсутній, перенаправляємо користувача на
сторінку з учнями
    header("Location: students.php");
    exit;
}
$studentId = $_GET['id'];
$conn = getPDO();
$stmt = $conn->prepare("SELECT * FROM Students WHERE id = ?");
$stmt->bind_param('i', $studentId);
$stmt->execute();
$result = $stmt->get_result();
if ($result->num_rows === 0) {
    header("Location: students.php");
    exit;
}
$student = $result->fetch_assoc();
if       ($_SERVER['REQUEST_METHOD']       ===       'POST'       &&
isset($_POST['confirm'])) {
    // Видалення учня з бази даних
    $stmt = $conn->prepare("DELETE FROM Students WHERE id = ?");
    $stmt->bind_param('i', $studentId);
    $stmt->execute();
    header("Location: students.php");
    exit;
}
?>
```

Додаток Д

Програмний код пошуку та фільтрації в таблицях веб-застосунку для репетитора «ВебМентор»

Search_student.php:

```
<?php
require_once __DIR__ . '/helpers.php';
$conn = getPDO();
if (!isset($_SESSION['user'])) {
    exit();
}
$userId = $_SESSION['user']['id'] ?? null;
if (isset($_GET['search'])) {
    $searchTerm = $_GET['search'];
    $stmt = $conn->prepare("SELECT * FROM Students WHERE CreatedBy
= ? AND Lastname LIKE ?");
    $searchTerm = "%$searchTerm%";
    $stmt->bind_param('is', $userId, $searchTerm);
    $stmt->execute();
    $result = $stmt->get_result();
} else {
    $stmt = $conn->prepare("SELECT * FROM Students WHERE CreatedBy
= ?");
    $stmt->bind_param('i', $userId);
    $stmt->execute();
    $result = $stmt->get_result();
}
echo "<thead>";
echo "<tr>";
echo "<th>Прізвище</th>";
echo "<th>Ім'я</th>";
echo "<th>Клас</th>";
echo "<th>Номер телефону</th>";
echo "<th>Адреса</th>";
echo "<th>Дія</th>";
echo "</tr>";
echo "</thead>";
while ($row = $result->fetch_assoc()) { ?>
    <tr>
        <td><?php echo $row['Lastname']; ?></td>
        <td><?php echo $row['Name']; ?></td>
        <td><?php echo $row['Class']; ?></td>
        <td><?php echo $row['Phone']; ?></td>
        <td><?php echo $row['Address']; ?></td>
        <td>
            <a  href="view_student.php?id=<?php  echo  $row['id'];
?>">Переглянути</a><br>
            <a  href="edit_student.php?id=<?php  echo  $row['id'];
?>">Редагувати</a><br>
            <a  href="delete_student.php?id=<?php  echo  $row['id'];
?>">Видалити</a>
        </td>
    </tr>
<?php } ?>
```

Get_all_students.php:

```
<?php
require_once __DIR__ . '/helpers.php';
$conn = getPDO();
if (!isset($_SESSION['user'])) {
    return false;
}
$userId = $_SESSION['user']['id'] ?? null;
$stmt = $conn->prepare("SELECT * FROM Students WHERE CreatedBy =
?");
$stmt->bind_param('i', $userId);
$stmt->execute();
$result = $stmt->get_result();
echo "<thead>";
echo "<tr>";
echo "<th>Прізвище</th>";
echo "<th>Ім'я</th>";
echo "<th>Клас</th>";
echo "<th>Номер телефону</th>";
echo "<th>Адреса</th>";
echo "<th>Дія</th>";
echo "</tr>";
echo "</thead>";
while ($row = $result->fetch_assoc()) { ?>
    <tr>
        <td><?php echo $row['Lastname']; ?></td>
        <td><?php echo $row['Name']; ?></td>
        <td><?php echo $row['Class']; ?></td>
        <td><?php echo $row['Phone']; ?></td>
        <td><?php echo $row['Address']; ?></td>
<td>
                            <a href="view_student.php?id=<?php echo
$row['id']; ?>">Переглянути</a><br>
                            <a href="edit_student.php?id=<?php echo
$row['id']; ?>">Редагувати</a><br>
                            <a    href="delete_student.php?id=<?php
echo $row['id']; ?>">Видалити</a>
                        </td>
    </tr>
<?php } ?>
```

Filter_events.php:

```
<?php
require_once __DIR__ . '/helpers.php';
$conn = getPDO();
if (!isset($_SESSION['user'])) {
    exit();
}
$userId = $_SESSION['user']['id'] ?? null;
if (isset($_GET['date'])) {
    $filterDate = $_GET['date'];
    $filterDate = date("Y-m-d", strtotime($filterDate));
    $stmt = $conn->prepare("SELECT * FROM events WHERE CreatedBy =
? AND Date = ?");
    $stmt->bind_param('is', $userId, $filterDate);
    $stmt->execute();
```

```
    $result = $stmt->get_result();
} else {
    $stmt = $conn->prepare("SELECT * FROM events WHERE CreatedBy =
?");
    $stmt->bind_param('i', $userId);
    $stmt->execute();
    $result = $stmt->get_result();
}
echo "<thead>";
echo "<tr>";
echo "<th>ID</th>";
echo "<th>Дата</th>";
echo "<th>Час події</th>";
echo "<th>Назва події</th>";
echo "<th>Опис події</th>";
echo "<th>Місце проведення</th>";
echo "<th>Дія</th>";
echo "</tr>";
echo "</thead>";
while ($row = $result->fetch_assoc()) { ?>
    <tr>
        <td><?php echo $row['ID']; ?></td>
        <td><?php echo $row['Date']; ?></td>
        <td><?php echo $row['Time']; ?></td>
        <td><?php echo $row['EventName']; ?></td>
        <td><?php echo $row['Description']; ?></td>
        <td><?php echo $row['Location']; ?></td>
        <td>
            <a   href="view_event.php?id=<?php   echo   $row['ID'];
?>">Переглянути</a><br>
            <a   href="edit_event.php?id=<?php   echo   $row['ID'];
?>">Редагувати</a><br>
            <a  href="delete_event.php?id=<?php  echo  $row['ID'];
?>">Видалити</a>
        </td>
    </tr>
<?php } ?>
```

Додаток Е

Програмний код розрахунку середнього балу учня та підрахунку сум оплат занять веб-застосунку для репетитора «ВебМентор»

Calculate_average_grade.php:

```
<?php
require_once __DIR__ . '/helpers.php';
$conn = getPDO();
$startDate = $_GET['startDate'];
$endDate = $_GET['endDate'];
$studentLastName = $_GET['studentLastName'];
$stmt = $conn->prepare("SELECT AVG(Homework) AS HomeworkAverage, AVG(Classwork) AS ClassworkAverage, AVG(Test) AS TestAverage FROM Journal WHERE Date BETWEEN ? AND ? AND Lastname = ?");
$stmt->bind_param('sss', $startDate, $endDate, $studentLastName);
$stmt->execute();
$result = $stmt->get_result();
$row = $result->fetch_assoc();
$totalAverageGrade = ($row['HomeworkAverage'] + $row['ClassworkAverage'] + $row['TestAverage']) / 3;
echo "<p>Середній бал домашньої роботи: " . number_format($row['HomeworkAverage'], 2) . "</p>";
echo "<p>Середній бал роботи на уроці: " . number_format($row['ClassworkAverage'], 2) . "</p>";
echo "<p>Середній бал тесту: " . number_format($row['TestAverage'], 2) . "</p>";
echo "<p>Загальний середній бал: " . number_format($totalAverageGrade, 2) . "</p>";
?>
```

Calculate_payments.php:

```
<?php
require_once __DIR__ . '/helpers.php';
$conn = getPDO();
$startDate = $_GET['startDate'] ?? null;
$endDate = $_GET['endDate'] ?? null;
$studentLastName = $_GET['studentLastName'] ?? null;
$selectAll = $_GET['selectAll'] ?? false;
if ($startDate && $endDate) {
    $conn = getPDO();
    if ($selectAll) {
        $stmt = $conn->prepare("SELECT SUM(Amount) AS totalAmount FROM Payments WHERE Date BETWEEN ? AND ?");
        $stmt->bind_param('ss', $startDate, $endDate);
    } elseif ($studentLastName) {
        // Виконати запит для отримання суми оплати за вказаним прізвищем учня та періодом
        $stmt = $conn->prepare("SELECT SUM(Amount) AS totalAmount FROM Payments WHERE Date BETWEEN ? AND ? AND StudentLastname = ?");
        $stmt->bind_param('sss', $startDate, $endDate, $studentLastName);
    } else {
        echo "Будь ласка, введіть прізвище учня або виберіть 'Вибрати всіх учнів'";
        exit;
```

```
    }
    $stmt->execute();
    $result = $stmt->get_result();
    $row = $result->fetch_assoc();
    $totalAmount = $row['totalAmount'] ?? 0;
    echo "Загальна сума оплати: $totalAmount";
} else {
    echo "Будь ласка, введіть коректні дати";
}
?>
```