Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Даниша Олега Дмитровича*
(ПІБ)

академічної групи *122-20-3*
(шифр)

спеціальності *122 Комп'ютерні науки*
(код і назва спеціальності)

освітньої програми *Комп'ютерні науки*
(назва освітньої програми)

на тему: *Розробка вебдодатку для запису на оффлайн консультації з використанням технологій HTML, CSS, JavaScript, PHP*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи				
розділів:				
спеціальний	*доц. Ширін А.Л.*			
економічний	*доц. Касьяненко Л.В.*			
Рецензент				
Нормоконтролер	*доц. Гуліна І.Г.*			

Дніпро
2024

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

М.О. Алексєєв
(підпис) (прізвище, ініціали)

« » 2024 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *122-20-3* *Даниша Олега Дмитровича*
(група) (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка вебдодатку для запису на оффлайн консультації з використанням технологій HTML, CSS, JavaScript, PHP*

затверджена наказом ректора НТУ «ДП» від *23.05.2024* № *469-с*

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів проєктно-технологічної та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі.*** ***Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***01.06.2024 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***06.06.2024 р.***

Завдання видав _____ *доц. Ширін А.Л.*
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Даниш О.Д.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2024 р.*

Термін подання кваліфікаційної роботи до ЕК: *24.06.2024 р.*

РЕФЕРАТ

Пояснювальна записка: 66 с., 30 рис., 5 табл., 3 дод., 20 джерел.

Об'єкт розробки: вебдодаток для запису на оффлайн консультації.

Мета кваліфікаційної роботи: розробка вебдодатку для запису на оффлайн консультації з використанням HTML, CSS, JavaScript, PHP.

У вступі розглядається проблематика та завдання на кваліфікаційну роботу, мета та актуальність теми дослідження.

У першому розділі зазначаються загальні відомості з предметної галузі, ставиться мета та призначення додатку, вказується завдання на розробку програми та вимоги до програмного забезпечення.

У другому розділі описується призначення додатку, його можливості, зазначено технології та мови, що були застосовані для розробки. Крім цього, описано функціонал, принципи роботи, структура додатку та організація вхідних і вихідних даних.

В економічному розділі визначається складність розробки даного вебдодатку. Розраховується імовірний час та вартість на розробку, встановлення та підтримку додатку.

Практичне значення роботи полягає у розробці вебдодатку, де викладачі та студенти з НТУ "Дніпровська політехніка" можуть створювати та записуватися на оффлайн консультації.

Актуальність даного додатку обумовлена зростанням популярності онлайн сервісів та додатків які автоматизують рутинні завдання та допомагають економити час користувачів.

Список ключових слів: ВЕБДОДАТОК, БРАУЗЕР, КОНСУЛЬТАЦІЇ, ЗАПИС, СТУДЕНТ,УНІВЕРСИТЕТ , HTML, CSS, JAVASCRIPT, PHP, DB.

ABSTRACT

Explanatory note: 66 p., 30 figures, 5 tables, 3 app., 20 sources.

Object of development: a web application for making an appointment for offline consultations.

Purpose of the qualification work: development of a web application for offline consultations using HTML, CSS, JavaScript, PHP.

The introduction discusses the problems and tasks for the qualification work, the purpose and relevance of the research topic.

The first section provides general information on the subject area, sets the purpose and purpose of the application, specifies the task of developing the application and software requirements.

The second section describes the purpose of the application, its capabilities, technologies and languages used for development. In addition, the functionality, operating principles, application structure, and organization of input and output data are described.

The economic section determines the complexity of developing this web application. The probable time and cost of developing, installing, and maintaining the application are calculated.

The practical significance of the work lies in the development of a web application where teachers and students from NTU "Dnipro Polytechnic" can create and sign up for offline consultations.

The relevance of this application is due to the growing popularity of online services and applications that automate routine tasks and help save users' time.

List of keywords: WEB APPLICATION, BROWSER, CONSULTATIONS, RECORD, STUDENT, UNIVERSITY, HTML, CSS, JAVASCRIPT, PHP, DB.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

HTML – HyperText Markup Language.

HTTP – HyperText Transfer Protocol.

HTTPS – HyperText Transfer Protocol Secure.

CSS – Cascading Style Sheets.

JS – JavaScript.

SQL – Structured Query Language.

PHP – PHP: Hypertext Preprocessor.

MVC – Model View Controller.

БД – база даних.

ПК – персональний комп’ютер.

ОС – операційна система.

W3C – World Wide Web Consortium.

Regex – Regular Expression.

ЗМІСТ

РЕФЕРАТ 3
ABSTRACT 4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ 5
ВСТУП 7
РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАВДАННЯ .. 9
1.1. Загальні відомості з предметної галузі 9
1.2. Призначення розробки та галузь застосування 12
1.3. Підстави для розробки 12
1.4. Постановка завдання 13
1.5. Вимоги до програми або програмного виробу 14
1.5.1. Вимоги до функціональних характеристик 14
1.5.2. Вимоги до інформаційної безпеки 14
1.5.3. Вимоги до складу та параметрів технічних засобів 15
1.5.4. Вимоги до інформаційної та програмної сумісності 16
РОЗДІЛ 2 ПРОЄКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ 17
2.1. Функціональне призначення системи 17
2.2. Опис застосованих математичних методів 18
2.3. Опис використаних технологій та мов програмування 18
2.4. Опис структури системи та алгоритмів її функціонування 22
2.5. Обґрунтування та організація вхідних та вихідних даних програми 31
2.6. Опис розробленої системи 34
2.6.1. Використані технічні засоби 34
2.6.2. Використані програмні засоби 35
2.6.3. Виклик та завантаження програми 36
2.6.4. Опис інтерфейсу користувача 37
РОЗДІЛ 3 ЕКОНОМІЧНИЙ РОЗДІЛ 51
3.1. Розрахунок трудомісткості розробки програмного забезпечення 51
3.2. Розрахунок витрат на створення програмного забезпечення 55
ВИСНОВКИ 58
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 59
ДОДАТОК А. КОД ПРОГРАМИ 61
ДОДАТОК Б. Відгук керівника економічного розділу 65
ДОДАТОК В. ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ 66

ВСТУП

Сучасне людство наразі стикається зі збільшенням інформації та подій як в особистому житті так і в професійних сферах. Збільшення інтенсивності подій призводить до необхідності в жорсткому плануванні часу. Великі обсяги інформації майже неможливо продуктивно використовувати без планування свого часу. І в цьому на допомогу людині приходять органайзери.

Це можуть бути паперові нотатки, щоденники та календарі. Але найбільш ефективно вирішити ці задачі допомагають інформаційні технології, а саме завдяки високій швидкості обробки даних, можливості збереження майже безлімітних обсягів даних, а також найшвидшою можливістю пошуку необхідних даних.

Майже неможливо уявити наше життя сучасної людини без допомоги пошукових систем, електронних календарів, хмарних середовищ, локальних та глобальних баз даних. Ці всі інструменти дозволяють людині більш ефективно витрачати свій робочий та приватний час.

Ще первісна людина використовувала на скельні малюнки для передачі, зберігання та поширення інформації. З появою письменності та цифр, такими інструментами були книги, літописи, мапи, креслення, як зберігачі інформації, а передача інформації була виключно від людини до людини.

Пізніше до джерел розповсюдження інформації з розвитком технічного прогресу додалися: радіо, радіомовлення та телебачення, що дозволило передавати інформацію в текстовому, аудіо та відео форматах на великі відстані з більшою швидкістю.

Але найвагоміший прорив стався з розповсюдженням інтернету. На сьогодні, майже кожному користувачу доступні неосяжні обсяги інформації в будь-якому форматі. Сталось можливим зберігати та розповсюджувати як особисті спогади, моменти життя, так і мати майже необмежений доступ до інформації про інших користувачів, навіть якщо ви не знайомі. Та найбільша користь досягається у використанні ІТ-технологій у професійній діяльності

людини. Сучасні технології дають можливість економити дуже велику кількість, як людського ресурсу, так і робочого часу для обробки, пошуку та передачі інформації. Велика кількість програмного забезпечення спрямована на оптимізацію трудового часу, а саме замінює людину у багатьох обчислювальних процесах та зберігає час на пошук та збереження інформації.

До того ж неоціненну користь в сьогоднішньому суспільстві привносять так звані веборганайзери та онлайн-календарі. Завдяки таким системам-посередникам між замовником та надавачем послуги, процес замовлення послуг онлайн займає лічені секунди. Розробка однієї з таких систем є темою цієї кваліфікаційної роботи.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1. Загальні відомості з предметної галузі

Платформи для онлайн-запису клієнтів або онлайн-календарі – це вебплатформи або мобільні додатки, які дозволяють ефективно організовувати зустрічі між клієнтами та працівниками.

Працівникам вони дозволяють ефективно налаштувати свій графік прийому клієнтів та мінімізувати кількість таких дій як: узгодження дати й часу з клієнтами, ведення графіку зустрічей, планування свого робочого часу з урахуванням навантаження в окремий період часу, своєчасне нагадування щодо запланованих зустрічей, уникнення накладання графіків, а саме конфліктних ситуацій з клієнтами, збереження інформації щодо зустрічей з клієнтами в режиму «архіву» за попередній час, можливість статистичної обробки даних щодо кількості та цілей зустрічі з клієнтами.

Клієнтам, з іншого боку, вони дають зручний інструмент для вибору бажаного фахівця, зручної дати та часу, а також для відслідковування запланованих зустрічей з метою ефективного використання часу та гарантії проведення запланованих зустрічей.

З розвитком інформаційних технологій та сфери послуг у світі такі платформи стали набувати актуальності. Такий інструмент спрощує взаємодію між користувачами та дозволяє економити час на з’ясуванні організаційних питань. Ці платформи стають особливо корисними як в невеликих компаніях, в яких працює невелика кількість людей, у місцях, де на одного фахівця припадає велика кількість клієнтів, так і в транснаціональних корпораціях, в яких такі системи дозволяють не тільки ефективно використовувати робочий час та його планування, але й дозволяють керівництву контролювати ефективність окремих співробітників, або навіть підрозділів. Прикладами місць з великим потоком клієнтів можуть бути: лікарні, школи, університети, та інші заклади.

Розглянемо деякі популярні вебресурси, що пов’язані з онлайн-записом клієнтів і активно використовуються на сьогоднішній день.

SimplyBook.me – додаток онлайн-запису для будь-якого бізнесу у сфері послуг (рис 1.1).

Плюси SimplyBook.me:

– більше 30 доступних мов;
– вебсторінка може бути налаштована під клієнта;
– нагадування про заплановані записи;
– система лояльності для постійних клієнтів.

Мінуси SimplyBook.me:

– у безкоштовній версії доступно лише 50 записів на місяць;
– інтерфейс іноді не досить інтуїтивний.

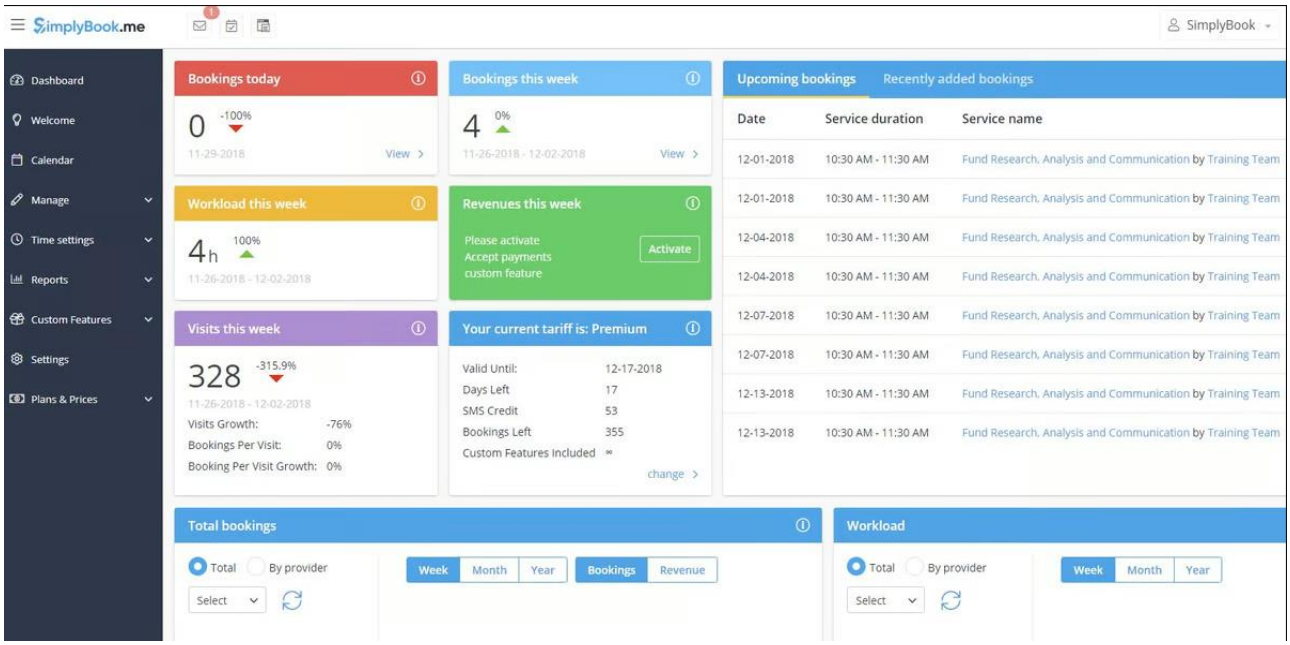

Рис. 1.1. Приклад інтерфейсу вебресурсу SimplyBook.me

Setmore – безкоштовна онлайн-платформа для планування та запису клієнтів (рис. 1.2).

Плюси Setmore:

– безкоштовна версія з достатнім базовим функціоналом;

– мобільний додаток з віртуальною касовою системою.

Мінуси Setmore:

– відсутність розширених функцій для продвинутого менеджменту;

– синхронізація календаря лише в один бік на безкоштовному тарифі.

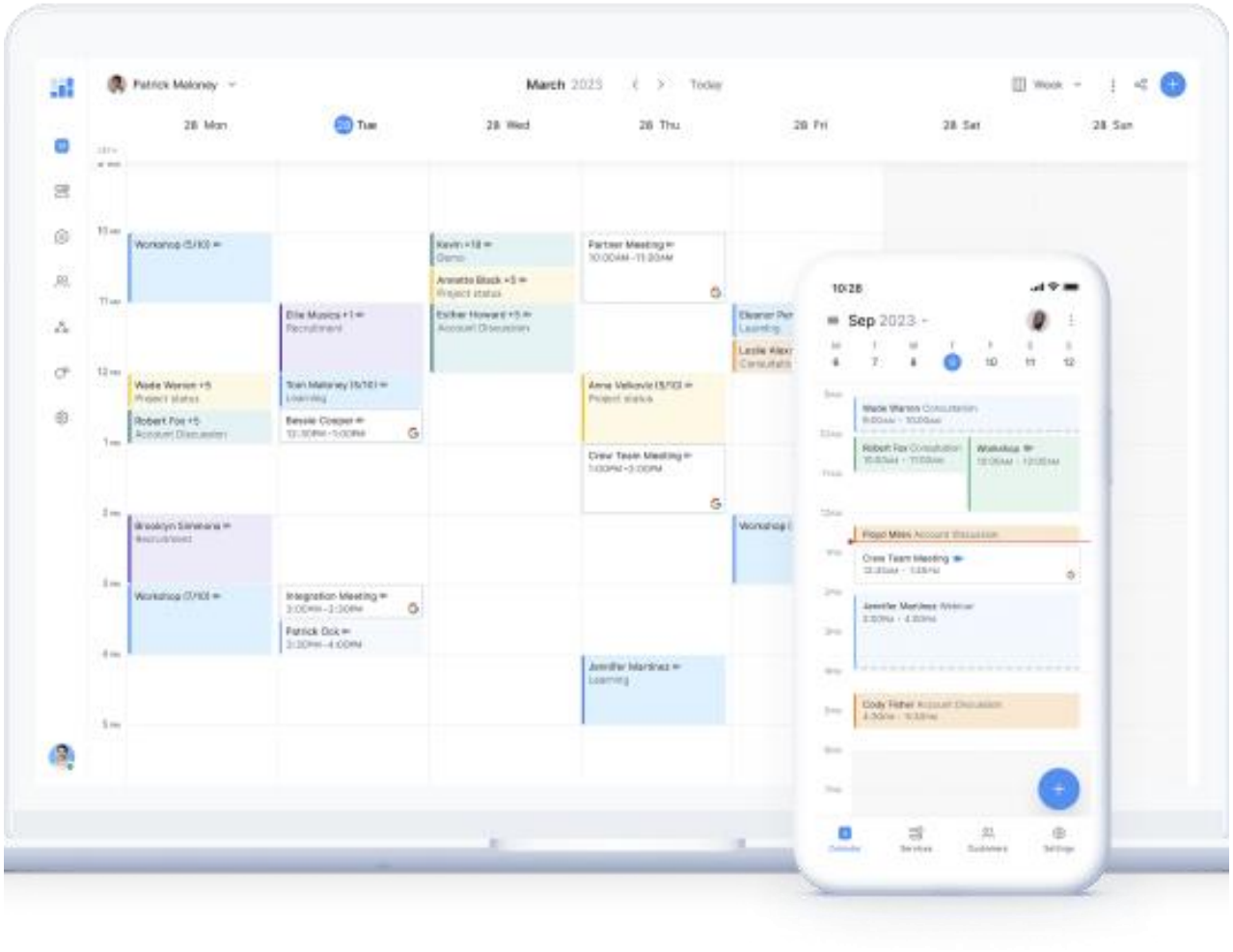

Рис. 1.2. Приклад інтерфейсу вебресурсу Setmore

Саме тому для цієї роботи було обрано темою «Розробка вебдодатку для запису на оффлайн консультації» для полегшення студентам процесу пошуку консультації, а викладачам організації їх робочого часу.

1.2. Призначення розробки та галузь застосування

В якості проекту для кваліфікаційної роботи розглядається вебдодаток для запису на оффлайн консультації. Метою цього проекту є проектування та розробка вебдодатку, який буде зручним у використанні та буде відповідати всім потребам користувачів. Данна розробка призначена для зручного та інтуїтивного використання, заощадження часу та автоматизації роботи викладачів.

Оскільки ця платформа розробляється саме для нашого університету НТУ "Дніпровська Політехніка", нам необхідно передбачити можливість реєстрації студентів та викладачів через корпоративну пошту університету та виключити можливість у сторонніх осіб доступу до персональних даних викладачів та студентів.

1.3. Підстави для розробки

Підставами для розробки та виконання кваліфікаційної роботи є:

− освітня програма 122 Комп’ютерні науки;

− навчальний план та графік навчального процесу;

− наказ ректора Національного технічного університету «Дніпровська політехніка» № 469-с від 23.05.2024 р;

− завдання на кваліфікаційну роботу на тему «Розробка вебдодатку для запису на оффлайн консультації з використанням технологій HTML, CSS, JavaScript, PHP».

1.4. Постановка завдання

Завдання: розробити вебдодаток для запису на оффлайн консультації з в використанням таких технологій: HTML, CSS, JS, PHP . Розробити БД . для збереження інформації про користувачів з використанням MySQL.

У таблиці 1.1 наведено можливості користувачів (об’єктів) при роботі з вебдодатком.

Таблиця 1.1

Можливості користувачів (об’єктів) при роботі з вебдодатком

Об’єкт	Можливості
Викладач	Реєстрація Вхід до системи
	Створення консультацій Зміна існуючих консультацій Видалення існуючих консультацій Перегляд створених консультацій
Студент	Реєстрація Вхід до системи Перегляд створених консультацій Пошук консультацій за предметом чи викладачем Запис на консультацію Скасування створеного запису
Гість (не авторизована особа)	Реєстрація Вхід до системи
Адміністратор	Вхід до системи Перегляд всіх створених консультацій Перегляд консультацій за діапазоном дат

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

На основі поставленого завдання розроблений продукт має відповідати таким вимогам:

- для верстки сторінки використовувати HTML та CSS;
- для серверної частини додатку використовувати мову PHP;
- для реалізації взаємодії на сайті використовувати JS;
- в якості БД використовувати MySQL;
- вебдодаток повинен бути розподілений на декілька сторінок, кожна з яких має свій функціонал та призначення;
- інформація про користувачів та консультації повинна зберігатися в БД.

Для досягнення поставленої мети вебдодаток має підтримувати виконання таких дій:

- реєстрація нових користувачів;аутентифікація та авторизація користувачів;
- навігація між сторінками;перегляд та пошук консультацій;
- редагування та видалення консультацій;
- запис та скасування запису на консультацію;
- перегляд профілів користувачів.

1.5.2. Вимоги до інформаційної безпеки

Інформаційна безпека є важливим питанням з яким стикаються розробники більшості програмних продуктів. Програми які не захищені від шахраїв та атак хакерів не матимуть довіри у клієнтів, адже мало хто захоче користуватися продуктом з ризиком того що важливі дані будуть втрачені або попадуть не в ті руки. Саме тому, щоб програма справно працювала, а конфіденційна інформація була захищена, розробникам слід приділяти велику увагу інформаційній безпеці. Вона охоплює комплекс заходів, які спрямовані на захист інформації від

несанкціонованого доступу, збереження цілісності даних та їх шифрування на випадок злому сторонніми особами.

Для захисту розробленого вебдодатку повинні бути впроваджені такі механізми:

Аутентифікація та авторизація – вебдодаток надає повний функціонал лише аутентифікованим користувачам, для того щоб переконатися що сторонні особи не мають доступу до даних користувачів та їх можливості обмежені наданими їм правами.

Обробка вхідних даних – додаток повинен ретельно оброблювати інформацію яку вводять користувачі за допомогою функцій та фільтрів. Це допомагає захиститися від атак типу SQL-ін’єкцій та XSS атак.

Шифрування даних – для захисту користувачів, дані в БД зберігаються в зашифрованому вигляді.

Знищення неактивних сесій – використовується для того щоб знищити сесію користувача, який не використовував сайт протягом певного часу, і тим самим виключити можливість повторного використання однієї сесії протягом великого проміжку часу.

1.5.3. Вимоги до складу та параметрів технічних засобів

Для доступу до вебдодатку знадобиться будь-який пристрій, що підтримує перегляд вебсторінок через браузер та має постійне підключення до інтернету. Таким пристроєм може бути: персональний комп’ютер, ноутбук, планшет чи смартфон. Для коректного відображення змісту сторінки на пристрої користувача, браузер повинен бути оновлений до останньої версії або його версія повинна бути не нижче вказаної рекомендованої:

- Chrome – версія 92;
- Firefox – версія 90;
- Edge – версія 92;
- Safari – версія 15.4;

– Opera – версія 78.

Рекомендовані характеристики пристроїв користувачів для роботи в браузері:

– процесор з тактовою частотою 2Ггц і більше;
– обсяг оперативної пам’яті не менше 4Гб;
– вільне місце на жорсткому диску 1Гб і більше.

1.5.4. Вимоги до інформаційної та програмної сумісності

Для ефективної роботи необхідно, щоб вебдодаток відповідав вимогам інформаційної та програмної сумісності. Дотримання цих вимог гарантує, що вебдодаток буде коректно працювати на будь-яких пристроях та зможе охопити широку аудиторію користувачів.

Дотримання вимог інформаційної сумісності передбачає підтримку та дотримання актуальних вебстандартів. Вебдодаток повинен керуватися такими стандартами як HTML5, CSS3 та JavaScript. Для передачі даних між клієнтом та сервером слід використовувати протокол HTTP або HTTPS.

Для програмної сумісності додаток повинен бути реалізований з використанням мов програмування JavaScript та PHP, мов розмітки HTML та CSS, а також з підтримкою БД MySQL. Для коректної роботи потрібно, щоб обчислювальна машина, на якій буде працювати вебдодаток, мала підтримку вказаних мов.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Функціональне призначення системи

В ході виконання кваліфікаційної роботи було розроблено вебдодаток для реєстрації студентів на оффлайн консультації, за допомогою простого і зрозумілого вебінтерфейсу.

Призначення розробленого додатку:

– надання студентам та викладачям зручного інструменту для організації їх часу у вигляді вебдодатку;

– надання адміністраторам можливості перегляду всіх консультацій для підведення підсумків їх ефективності та актуальності.

Для досягнення необхідного функціоналу вебдодатку, було реалізовано такі функції як:

– реєстрація нових користувачів, як студентів так і викладачів;

– авторизація та автентифікація користувачів;

– створення консультацій викладачями;

– корекція введених даних вже створеної консультації;

– перегляд створених консультація;

– видалення створених консультацій;

– пошук зручних консультацій;

– реєстрація на запропоновану консультацію;

– перегляд всіх записів на консультації;

– перегляд профілю користувача;

– перегляд консультацій за датами для адміністраторів.

Перш за все, цільовою аудиторію цього застосунку є студенти та викладачі навчальних закладів, а саме цей вебдодаток орієнтований на осіб, що працюють та навчаються в НТУ «Дніпровська політехніка».

2.2. Опис застосованих математичних методів

Розроблений додаток фокусується на збиранні, обробці та візуалізації даних, пов’язаних з консультаціями. Ці дані включають інформацію про осіб, що користуються вебдодатком, а також інформацію про самі консультації.

Обробка та візуалізація цих даних не потребує використання математичних формул. Отже, застосування математичних методів не є необхідним для реалізації основних функцій програми.

2.3. Опис використаних технологій та мов програмування

Створення будь-якого сайту чи вебдодатку починається з постановки цілей, мети та призначення цього продукту. Після чого для реалізації поставлених цілей необхідно визначитися зі списком тих технологій які доцільно буде використовувати для розробки програми. Вибір необхідних технологій є важливим питанням, адже використання актуальних або стандартизованих мов може бути ключовим критерієм при розробці і може вплинути на такі фактори як:

- час на розробку проекту;
- вартість розробки;
- масштабованість проекту;
- кількість спеціалістів, що працюють з цими технологіями;
- можливості для оновлення заходів безпеки.

Саме тому, питання вибору технологій та мов програмування є важливим і може сприяти на результуючий вигляд та стан проекту.

Для початку, для створення вебсторінки необхідно створити її структуру, що буде слугувати «фундаментом» всієї сторінки. В подальшому цей «фундамент» ми зможемо прикрасити, додати йому функціоналу та взаємодії. Для створення основи сайту, за вебстандартами опублікованими W3C, можна обрати між HTML та XHTML.

HTML (HyperText Markup Language) – мова розмітки документів, яка використовується для створення структури та вмісту вебсторінки. Ця мова була побудована на основі правил SGML (Standard Generalized Markup Language).

XHTML (Extensible Hypertext Markup Language) – також мова розмітки документів, має таку саму виразну силу що й HTML, однак, на відміну від нього, ця мова була побудована на правилах XML (Extensible Markup Language).

Хоча ці мови і схожі між собою, XHTML має більш суворий синтаксис, що накладає на розмітку такі додаткові правила:

1. Теги та атрибути пишуться лише з малої літери.
2. Атрибути повинні бути записаними у лапках.
3. Не допускаються не закриті теги.
4. Чутливість до регістру.

HTML є більш простішим у вивчені і рекомендується для початківців, натомість XHTML рекомендується вже для досвідчених розробників, що вміють правильно структурувати та організовувати власний код.

Перевага XHTML полягає як раз в його сурових правилах, що полегшує читабельність коду, особливо для розробників, що працюють над великими проектами.

Обидві мови розмітки, на сьогоднішній день, підтримуються сучасними браузерами і коректно відображаються на будь-якому пристрої. Хоча XHTML була розроблена як покращена версія HTML, що мала в майбутньому замінити свого конкурента, але в результаті не досягла очікуваних результатів. Сьогодні HTML продовжує і надалі розвиватися і з виходом HTML5 все більше сайтів зосереджуються саме на використанні HTML.

Саме тому, для свого проекту було обрано за основу HTML, спираючись на ці критерії:

1. HTML продовжує розвиватися та пропонує нові та сучасні функції.
2. Кількість розробників на HTML набагато більше.
3. Проект для кваліфікаційної роботи не є громіздким та не потребує жорсткої розмітки тексту.

Далі, для оформлення зовнішнього вигляду сторінки необхідно обрати одну з таблиць стилів. Найпопулярнішими технологіями є CSS (Cascaded Style Sheets), SASS (Syntactically Awesome Style Sheets) та SCSS (Sassy CSS).

CSS – мова стилів, яка використовується для візуального оформлення вебсторінок. За допомогою тегів, класів та індексів, дозволяє надавати кожному елементу на сторінці свій унікальний вигляд.

На відміну від CSS, SASS та SCSS не є мовами стилів сторінки, а препроцесорами CSS, які відрізняються синтаксисом і потім компілюються в CSS.

Головними перевагами препроцесорів CSS є їх більш простий синтаксис, а також можливість використання змінних та математичних функцій, для створення складного та динамічного оформлення сторінки.

Не зважаючи на їх переваги, для проекту було обрано CSS, так як він не потребує додаткової компіляції коду, належить до стандарту W3C, та додаткові можливості препроцесорів CSS не є необхідними для коректної роботи вебдодатку.

Після того, як у вебдодатку будуть створені елементи взаємодії з користувачем такі як: кнопки, форми, поля та таблиці, можна почати обирати мову програмування яка буде відповідати за обробку запитів від користувачів та буде слугувати «мостом» між сторінкою та БД. Отже, нам необхідна BackEnd-мова, яка буде виконуватися на стороні сервера. Серед популярних рішень є такі мови програмування:

1. PHP. Ця мова програмування є справжнім ветераном у світі веброзробки, адже була створена ще в 1994 році і використовується по сьогоднішній день. Оскільки це інтерпретована мова, то для неї не потрібен компілятор, а також вона працює на всіх популярних операційних системах, таких як Linux , Windows, macOS. PHP легко вивчити, має ООП-функціонал, підтримує різні види БД, такі як MySQL, SQLite та інші.

2. Java. Відома та популярна мова програмування, яка також використовується і для серверної веброзробки. Головна перевага Java над

іншими мовами це те що її не потрібно кожен раз повторно компілювати. Достатньо один раз провести компіляцію коду і він буде однаково працювати на всіх машинах, що підтримують Java.

3. Ruby. Інтерпретована мова програмування загального характеру, що може буде використана для будь-якої задачі. Мова широко використовується веброзробниками по всьому світу і її часто рекомендують початківцям, адже її легко вивчити.

4. C#. Мова програмування, що вже протягом декількох років входить в топ-5 найпопулярніших мов програмування. Вона була розроблена корпорацією Майкрософт переважно для фреймворку .Net, та може використовуватися для серверної веброзробки. C# має великий набір бібліотек, які допомагають розробникам пришвидшити та спростити їх процес розробки.

5. Python. Відома на весь світ мова своїм простим синтаксисом та легкістю в її вивченні, однак не багато хто знав, що Python також часто використовується для створення вебсайтів. Головна перевага Python це велика кількість бібліотек, як стандартних так і написаних іншими розробниками. Також ця мова має прозорий та читабельний код, що робить її популярної серед розробників.

Спираючись на всі переваги та недоліки перелічених мов програмування, а також на мету та функціонал проекту, було обрано саме PHP для серверної веброзробки. PHP є простим і перевіреним рішенням для створення серверної частини сайту. Ця мова є популярною серед розробників, має інтеграцію з великою кількістю популярних БД, що дає простір для вибору, а також має необхідний вбудований функціонал для створення якісного та безпеченого вебсайту.

Також слід обрати другу мову програмування, яка буде створювати необхідний функціонал та динаміку для сайту. Такою мовою є JavaScript, що ідеально підходить для вирішення цих питань. Головною перевагою JavaScript в порівнянні з іншими мовами програмування є інтегрованість цієї мови безпосередньо в HTML та CSS, що робить її зручною для маніпуляцій з елементами вебсторінки без додаткових зусиль.

2.4. Опис структури системи та алгоритмів її функціонування

Структура системи розподілена по окремим папкам у проекті. Така структура проекту була обрана для організації робочого простору та розуміння в якій частині проекту зараз знаходиться розробник. Всі папки поділяються на три види:

1. Коренева папка. Ця папка містить у собі всю структуру проекту та має доступ до всіх інших папок. Крім цього, зазвичай при завантаженні сайту, першим завантажується сторінка з кореневої папки, отже вміст цієї папки виступає в ролі головної сторінки сайту, на яку при першому заході потрапляє користувач.

2. Папка зі сторінкою. Вони використовуються для розподілення різних сторінок вебсайту для кращої організації проекту. Всі файли які відносяться до однієї сторінки потрібно зберігати в одній папці з характерною назвою. Прикладом таких папок є «home» та «user-page».

3. Папка з утилітами. Ця папка слугує для всіх інших файлів, які не відносяться до двох вище названих папок. В нашому випадку ця папка використовується для таких файлів, які містять код, що буде використовуватися на декількох або на всіх сторінках сайту. Отже, для того, щоб не копіювати один і той самий функціонал в різні файли, було вирішено винести функції, які часто використовуються, в окрему папку.

Повна структура системи, включаючи всі папки, файли, алгоритми та зображення, подана на рис. 2.1.

В кореневій папці містяться 5 файлів і 3 папки. Файли в кореневій папці використовуються для завантаження першої вебсторінки, на яку потрапляє користувач після заходу на сайт. На цій сторінці користувачу доступні два функціонали, а саме авторизація та реєстрація. Коренева папка включає в себе такі файли:

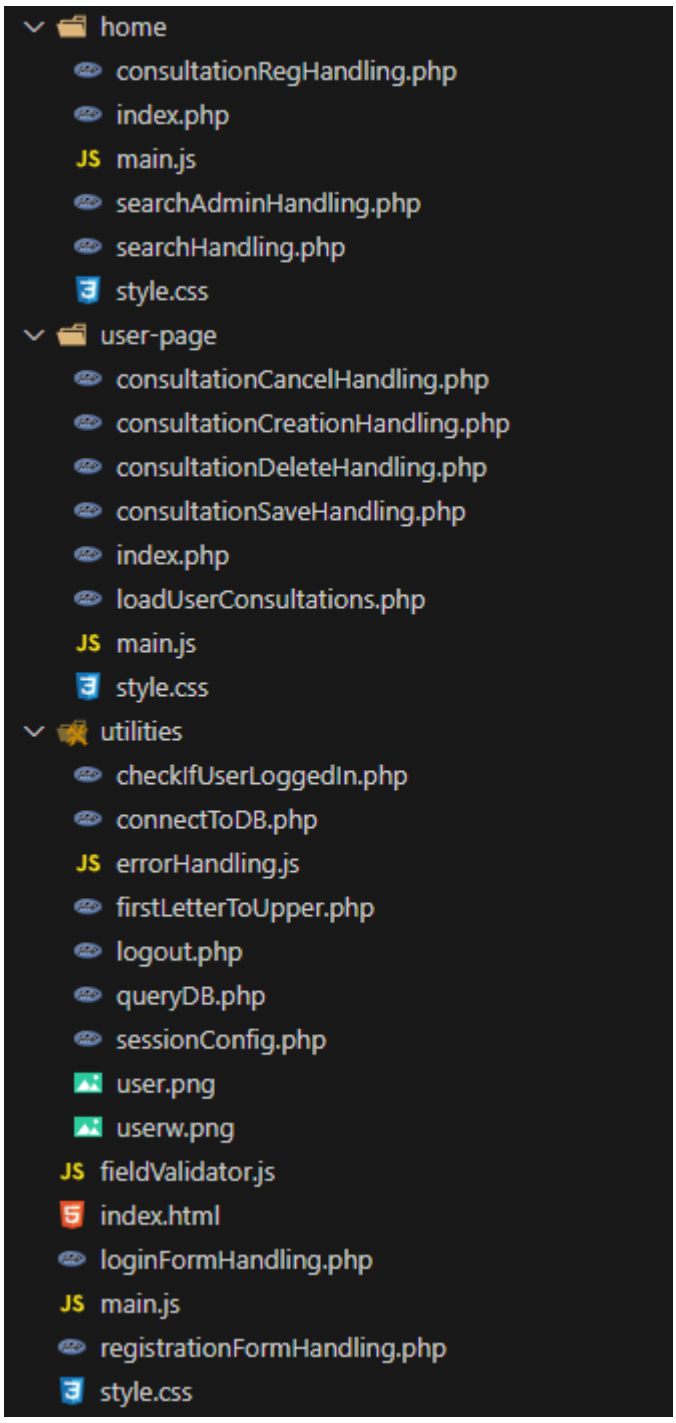

Рис. 2.1. Структура системи

– index.html – файл формату HTML, який містить структуру сторінки. Ця сторінка відповідає за реєстрацію нових користувачів та авторизацію вже зареєстрованих;

– style.css – файл формату CSS, що відповідає за опис ключових стилів сторінки index.html;

– main.js – файл формату JS, написаний на мові програмування JavaScript. Він відповідає за роботу всіх елементів, що можуть змінюватися динамічно. Наприклад, він відповідає за: кнопки для переключення між формою для авторизації та формою для реєстрації, зміну кількості полів при виборі між роллю користувача, відправку форми на сервер та перший крок перевірки введених користувачем даних;

– loginFormHamdling.php – файл написаний на мові програмування PHP. Він відповідає за обробку вхідних даних від користувача при його спробі авторизуватися у вебдодатку;

– registrationFormHamdling.php – файл написаний на мові програмування PHP. Він відповідає за обробку вхідних даних від користувача при спробі зареєструватися у вебдодатку.

В папці «home» знаходяться 6 файлів, які відповідають за структуру, вигляд та функціонал, що знаходяться на головній сторінці вебдодатку. На цю сторінку користувач потрапляє після вдалої авторизації на платформі. На цій сторінці користувачц стають доступні пошук та реєстрація на консультації. У папці містяться такі файли:

– index.php – файл написаний на мові PHP та HTML, який відповідає за структуру сторінки, а також за зміну вмісту сторінки в залежності від ролі авторизованого користувача. Наприклад, студент та викладач зможуть шукати тут консультації назвою предмету або за ПІБ викладачя, однак адміністраторам доступна додаткова функція для пошуку всіх консультацій за певний проміжок часу;

– style.css – файл CSS, що відповідає за опис ключових стилів сторінки index.php;

– main.js – файл написаний на мові JavaScript, що відповідає за коректну роботу головного меню сайту;

– searchHandling.php – файл написаний на мові PHP, що відповідає за оброку пошукових запитів від користувачів, а також повертає результати запиту на головну сторінку для подальшого його відображення;

– searchAdminHandling.php – файл написаний на мові PHP, має схожий функціонал з попереднім файлом, однак на відміну від нього він обробляє пошукові запити, що були надіслані адміністраторами вебдодатку;

– consultationRegHandling.php – файл написаний на мові PHP, що обробляє запит користувача на реєстрацію на обрану консультацію.

В папці «user-page» знаходяться 8 файлів, які відповідають за структуру, вигляд та функціонал сторінки з кабінетом користувача. На цій сторінці можна подивитися інформацію про себе, а також стає доступним додатковий функціонал для роботи з консультаціями. Студент може переглядати консультації на які він був зареєстрований, а викладач може переглядати та створювати власні консультації. В цій папці містяться такі файли:

– index.php – файл написаний на мові PHP та HTML, який відповідає за структуру сторінки, а також за зміну вмісту сторінки в залежності від ролі авторизованого користувача;

– style.css – файл CSS, що відповідає за опис ключових стилів сторінки index.php;

– main.js – файл написаний на мові JavaScript, що відповідає за роботу головного меню, кнопок для відображення додаткових елементів, а також для коректної роботи текстових полів;

– loadUserConsultations.php – файл написаний на мові PHP, що відповідає за процес завантаження з БД консультацій, які були створені викладачем або на які був зареєстрований учень;

– consultationCancelHandling.php – файл написаний на мові PHP, що відповідає за обробку запиту на скасування запису на консультацію від студента.

– consultationCreationHandling.php – файл написаний на мові PHP, що відповідає за процес обробки запиту на створення нової консультації;

– consultationDeleteHandling.php – файл написаний на мові PHP, що відповідає за видалення вже створеної консультації;

– consultationSaveHandling.php – файл написаний на мові PHP, що відповідає за обробку запиту на збереження змін, які були внесені викладачем до вже створеної консультації.

Остання папка «utilities» містить 7 файлів, які відповідають за різний функціонал сайту, який був винесений до окремих файлів через їх універсальне використання на всіх сторінках вебдодатку. В цю папку входять такі файли:

– checkIfUserLoggedIn.php – файл написаний на мові PHP, що містить алгоритм для перевірки авторизації користувача. В тому разі, якщо час дії сеансу користувача добіг кінця, або він намагався отримати доступ до інших сторінок сайту без попередньої авторизації, то його повертає на початкову сторінку;

– connectToDB.php – файл написаний на мові PHP, що відповідає за встановлення зв’язку з БД;

– errorHandling.js – файл написаний на мові JavaScript, що відповідає за обробку та зручне відображення всіх помилок, які могли статися при обробці запитів від користувача;

– firstLetterToUpper.php – файл написаний на мові PHP, який містить функцію, яка призначена для підняття першої літери з нижнього у верхній регістр. Використовується для коректного відображення тексту, що надійшов з БД;

– logout.php – файл написаний на мові PHP, що виконує операцію виходу користувача з його аккаунту та повертає на початкову сторінку;

– queryDB.php – файл написаний на мові PHP, що містить запити до БД у вигляді функцій, що дає змогу розробникам не створювати нові запити кожен раз, а звертатися вже до існуючих функцій. Прикладом функції, що створює запит до БД на основі дій користувача, може бути функція видалення консультації (рис. 2.2);

```php
function deleteConsultation($connection, $id_consultation)
{
    $stmt = mysqli_prepare($connection, "DELETE FROM consultation WHERE id_consultation = ?;");
    mysqli_stmt_bind_param($stmt, "i", $id_consultation);
    mysqli_stmt_execute($stmt);
    $answer = mysqli_stmt_affected_rows($stmt);
    return $answer;
}
```

Рис. 2.2. Функція створення запиту до БД

– sessionConfig.php – файл написаний на мові PHP, що містить початкові налаштування сесій на сайті, які роблять їх використання більш безпечними. Також, встановлює таймер на сесію, по закінченню якого, користувач автоматично виходить зі свого аккаунта і повертається на початкову сторінку. Такий підхід дозволяє підвищити безпеку сайту.

Вебдодаток був побудований використовуючи шаблон MVC. MVC (Model-View-Controller) це архітектурний шаблон, який використовується для розробки програм, додатків та вебзастосунків. Ключова ідея цього шаблону полягає в тому що він розподіляє логіку додатку на три окремі частини:

1. Model (Модель) – містить дані або бізнес логіку, зберігає та надає дані для їх подальшого використання. Користувач додатку не має доступу до моделі, а запит на отримання даних відбувається лише через Controller. В якості Model в цьому вебдодатку слугує БД.

2. View (Представлення) – відображає дані з Model, не містить жодної бізнес-логіки та використовується користувачами для перегляду результатів та створення запитів. Всі запити, що були створені надходять спочатку в Контролер. В якості View у цьому вебдодатку виступають HTML, CSS та JS файли.

3. Controller (Контролер) – слугує «мостом» між View та Model, обробляє та перевіряє запити користувача, оновлює дані в View та Model у відповідь на запит. В якості Controller у цьому вебдодатку виступають PHP файли, що оброблюють запити на стороні сервера.

Дотримання такої архітектури дозволяє полегшити розробку додатку і має такі переваги:

1. Окремі компоненти цієї моделі можуть повторно використовуватися у інших файлах застосунку

2. Полегшує тестування і виявлення багів

3. Дозволяє легко масштабувати додатки додаючи нові функції або змінюючи вже існуючі.

4. Дозволяє декільком розробникам працювати над одним проектом, розділяючи логіку на зони відповідальності.

За шаблоном MVC вебдодаток функціонує за таким алгоритмом:

1. Користувач через представлення View у вигляді вебсторінки робить запит до БД, використовуючи форму або відкриваючи нову сторінку.

2. Цей запит спочатку обробляється на стороні клієнта, а потім відправляється до необхідного Контролера.

3. Контролер, на стороні сервера, перевіряє коректність введеного користувачем запиту, після чого звертається з цим запитом до Моделі, в ролі якої виступає БД.

4. БД обробляє запит, знаходить відповідні дані і повертає їх назад до Контролера.

5. Контролер передає отриману відповідь до Представлення.

6. Представлення, маючи необхідні дані, відображає їх на вебсторінці користувача.

Цей алгоритм є універсальним і може застосовуватися для будь-якого етапу взаємодії користувача з додатком.

На прикладі авторизації користувача, розглянемо алгоритм роботи вебдодатку. Коли користувач заповнює поля та натискає на кнопку «Вхід» викликається функція «loginFieldCheck» (рис. 2.3). Ця функція працює на стороні клієнта і перевіряє:

1. Чи всі поля заповнені.

2. Чи вірна пошта була введена.

Після цього, якщо користувач помилився в якомусь з полів, то функція вкаже на помилку та виведе повідомлення. Якщо все було введено правильно, то форма буде відправлена і її перевіркою буде займатися алгоритм на стороні сервера.

```
function loginFieldsCheck()
{
    for (let i = 0; i < loginFields.length; i++) {
        if (!loginFields[i].value) {
          markFieldWithError("login", i);
          showErrorMessage("Поля не повинні бути пустими!");
          return false;
        }
    }
    if(!emailValidCheck(loginFields[0].value))
    {
        markFieldWithError("login", 0);
        showErrorMessage("Використовуйте лише корпоративну пошту @nmu.one !");
        return false;
    }
    return true;
}
```

Рис. 2.3. Клієнтська функція для перевірки полів форми авторизації

Після відправки форми, за перевірку даних приймається серверний алгоритм, який перевіряє дані від користувача за допомогою Regex (рис. 2.4). Цей алгоритм поетапно перевіряє:

1. Метод відправки даних
2. Підключення до БД
3. Наповненість полів
4. Правильність вказаної пошти

По завершенню перевірок, якщо отримані дані відповідають поставленим умовам, то далі викликається функція створення та обробки запиту.

```php
if($_SERVER["REQUEST_METHOD"] === "POST")
{
    try {
        require_once("utilities/connectToDB.php");
        require_once("utilities/queryDB.php");
    } catch (Exception $e) {
        header("Location: index.html?message=no_db_connection");
        die();
    }
    if(!isset($_POST["email"], $_POST["password"]))
    {
        header("Location: index.html?message=fields_empty");
        die();
    }
    if($_POST["email"] == $ADMIN_EMAIL && $_POST["password"] == $ADMIN_PASSWORD)
    {
        $_SESSION["id"] = 0;
        $_SESSION["occupation"] = "admin";
        $_SESSION["last_regeneration"] = time();
        header("Location: home/");
        die();
    }
    if(!filter_var($_POST["email"], FILTER_VALIDATE_EMAIL))
    {
        header("Location: index.html?message=not_valid_email");
        die();
    }
    $nmuEmailPattern = "/^([A-Za-z]+(\.[A-Za-z]+)+)@nmu\.one$/";
    if(preg_match($nmuEmailPattern, $_POST["email"]) == 0)
    {
        header("Location: index.html?message=wrong_email");
        die();
    }
    if(!isEmailAlreadyExist($connection, $_POST["email"]))
    {
        header("Location: index.html?message=email_dont_exist");
        die();
    }
    $userAuthenticationResult = userAuthentication($connection, $_POST["email"], $_POST["password"]);
    if($userAuthenticationResult["id"] == false)
    {
        header("Location: index.html?message=wrong_password");
        die();
    }
```

Рис. 2.4. Серверна функція для перевірки полів форми авторизації

Після завершення перевірки даних, функція «userAuthentication» створює запит до БД (рис 2.5). Отримавши відповідь від БД, функція приводить дані до зручного вигляду і повертає отримані дані клієнту.

```php
function userAuthentication($connection, $email, $password)
{
    $email = mb_strtolower($email);
    $request = "SELECT * FROM student WHERE email = ?;";
    $answer = mysqli_execute_query($connection, $request, [$email]);
    $studentTableAnswer = mysqli_fetch_array($answer);
    $request = "SELECT * FROM teacher WHERE email = ?;";
    $answer = mysqli_execute_query($connection, $request, [$email]);
    $teacherTableAnswer = mysqli_fetch_array($answer);
    if(password_verify($password, $studentTableAnswer["password"]))
    {
        return array("id" => $studentTableAnswer["id_student"], "occupation" => "student", "name" => $studentTableAnswer
        ["name"] , "surname" => $studentTableAnswer["surname"], "fathers_name" => $studentTableAnswer["fathers_name"], "email" =>
        $studentTableAnswer["email"], "st_group" => $studentTableAnswer["st_group"], "faculty" => $studentTableAnswer["faculty"]);
    }
    else if(password_verify($password, $teacherTableAnswer["password"]))
    {
        return array("id" => $teacherTableAnswer["id_teacher"], "occupation" => "teacher", "name" => $teacherTableAnswer
        ["name"] , "surname" => $teacherTableAnswer["surname"], "fathers_name" => $teacherTableAnswer["fathers_name"], "email" =>
        $teacherTableAnswer["email"],  "faculty" => $studentTableAnswer["faculty"]);
    }
    return array("id" => false);
}
```

Рис. 2.5. Функція створення та обробки запитів БД

Останній етап передбачає отримання даних клієнтом, а також їх подальше виведення на вебсторінці.

2.5. Обґрунтування та організація вхідних та вихідних даних програми

Під час взаємодії користувача з вебдодатком, постійно відбувається обмін даними між клієнтською та серверною частиною програми. Вхідними даними є будь-яка інформація або запит, яку вводить клієнт під час роботи з вебдодатком. Вихідними даними є відповідь БД, на запити клієнта, яка потім буде відображена на вебсторінці.

Для забезпечення безпеки додатку необхідно контролювати та перевіряти дані які отримуються від клієнта. Процес обробки вхідних даних складається з трьох етапів:

1. Перевірка запитів на стороні клієнта.
2. Перевірка запитів на стороні сервера.
3. Виконання запиту в БД.

В першу чергу відбувається перевірка даних на стороні клієнта. Це перший етап обробки, який передбачає, що за допомогою JavaScript, правильність даних перевіриться технічними можливостями клієнта. Це дозволяє економити ресурси

серверу і зменшити його навантаження. З іншого боку, такі перевірки не є безпечними, адже все що відбувається на стороні користувача, також може бути ним змінене. Це означає, що користувач може відключити частину коду та відправити небезпечні дані на сервер.

Щоб уникнути такої вразливості, було створено другий етап перевірки даних на стороні сервера. Перевіряючи дані на сервері, потрібно остаточно переконатися, що вони не містять загрози та були введені правильно. На відміну від першого етапу, до другого етапу перевірки користувач не має доступу, а отже він є більш безпечним.

Після того як дані були перевірені, в третьому етапі створюється запит, який потім буде оброблений БД.

Для зберігання даних на сервері використовується база даних на системі управління MySQL. Структура БД складається з 4 таблиць та зв'язків між ними, які зображені на рис. 2.6.

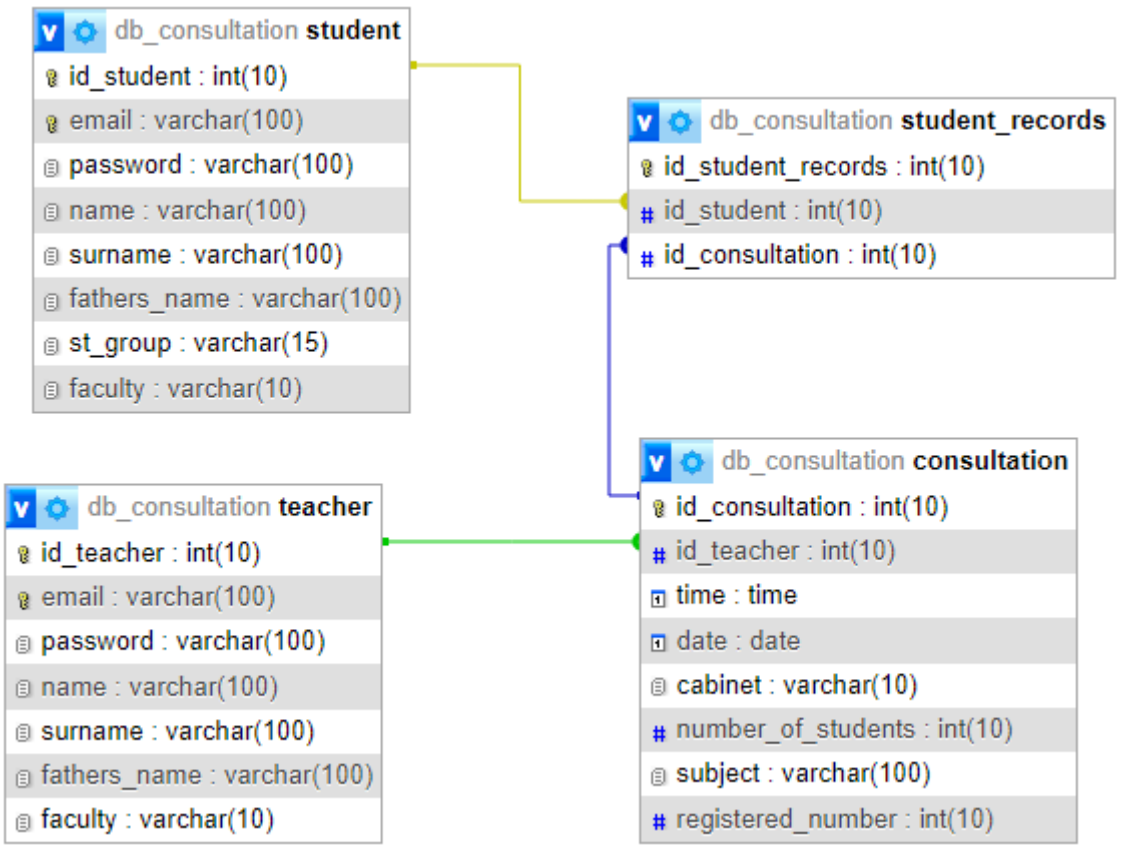

Рис. 2.6. Структура БД

Структура таблиць БД, складається з полів, які описані у таблицях 2.1-2.4.

Таблиця 2.1

Опис полів таблиці teacher

Назва поля	Тип даних	Ціль (призначення)	Обмеження
id_teacher	int	Унікальний ідентифікатор	primary index auto_increment max_length=10
email	varchar	Пошта	index max_length=100
password	varchar	Пароль	max_length=100
name	varchar	Ім’я	max_length=100
surname	varchar	Прізвище	max_length=100
fathers_name	varchar	По батькові	max_length=100
faculty	varchar	Факультет	max_length=10

Таблиця 2.2

Опис полів таблиці student

Назва поля	Тип даних	Ціль (призначення)	Обмеження
id_teacher	int	Унікальний ідентифікатор	primary index auto_increment max_length=10
email	varchar	Пошта	index max_length=100
password	varchar	Пароль	max_length=100
name	varchar	Ім’я	max_length=100
surname	varchar	Прізвище	max_length=100
fathers_name	varchar	По батькові	max_length=100
st_group	varchar	Група	max_length=15
faculty	varchar	Факультет	max_length=10

Таблиця 2.3

Опис полів таблиці consultation

Назва поля	Тип даних	Ціль (призначення)	Обмеження
id_consultation	int	Унікальний ідентифікатор	primary index auto_increment max_length=10
id_teacher	int	Зовнішній ключ id_teacher	index max_length=10
time	time	Час консультації	
date	date	Дата консультації	
cabinet	varchar	Місце проведення	max_length=100
number_of_students	int	Кількість місць	max_length=10
subject	varchar	Предмет	max_length=100
registered_number	int	Кількість зареєстрованих	max_length=10

Таблиця 2.4

Опис полів таблиці student_records

Назва поля	Тип даних	Ціль (призначення)	Обмеження
id_consultation	int	Унікальний ідентифікатор	primary index auto_increment max_length=10
id_student	int	Зовнішній ключ id_student	index max_length=10
id_consultation	int	Зовнішній ключ id_consultation	index max_length=10

2.6. Опис розробленої системи

2.6.1. Використані технічні засоби

Для доступу до вебдодатку можна використовувати ПК, ноутбук, смартфон, планшет або будь-який інший пристрій, який має доступ до інтернету і відповідає мінімальними характеристиками для коректної роботи браузеру.

Для розробки вебдодатку було використано ноутбук з такими характеристики системи:

– ОС: Windows 10;
– тип системи: 64 розрядна архітектура;
– процесор: Intel Core i5-8300H 2.30GHz;
– пам’ять (ОЗУ): об’єм 32 ГБ;
– відеокарта: NVIDIA GeForce GTX 1060;
– накопичувач: жорсткий диск об’ємом 1 ТБ;
– монітор: 1920x1080.

2.6.2. Використані програмні засоби

Під час створення платформи були використані наступні програмні засоби:

– Google Chrome – браузер, який використовувався для перегляду вебсторінок розробленого додатку.

– Microsoft Visual Studio Code – редактор коду, який використовувався для написання вебдодатку. Завдяки великій кількості розширень дозволяє писати код мовами HTML, CSS, JavaScript та PHP;

– XAMPP – багатоплатформна збірка для запуску свого вебсервера, що включає Apache, MySQL та інтерпретатор PHP.

Для бекенд розробки було використано мову програмування PHP.

База даних була створена за допомогою реляційної системи управління базами даних MySQL.

Для фронтенд розробки було використано такі технології:

– HTML;
– CSS;
– JavaScript.

2.6.3. Виклик та завантаження програми

Для завантаження вебдодатку необхідно спершу переконатися чи відповідає сервер необхідним умовам:

1. Вільне місце на жорсткому диску
2. Мінімальні технічні характеристики для роботи БД та вебдодатку
3. Підключення до інтернету
4. Встановлені необхідні технології та програми.

Необхідно переконатися, чи підтримує обраний сервер всі зазначені технології. Крім цього, необхідно встановити вебсервер, на якому буде запущений вебдодаток. Серед відповідних варінатів можна обрати Open Server або XAMPP. В цьому випадку, було вирішено, в якості вебсерверу обрати XAMPP тож всі подальші дії для завантаження програми будуть виконуватися в цьому середовищі.

Щоб завантажити вебдодаток необхідно виконати такі дії:

1. Всі файли які відносяться до вебдодатку необхідно помістити у папку htdocs за таким шляхом «%шлях до xampp%\xampp\htdocs\». З цієї папки буде завантажуватися програма, однак, за бажанням, цей шлях можна змінити в налаштуваннях XAMPP.

2. Після цього відкриваємо XAMPP Control Panel, інтерфейс якого показано на рис 2.7.

3. Далі в списку «Module» знаходимо необхідні нам модулі для роботи вебдодатку, а саме Apache та MySQL. Apache це HTTP вебсервер, який використовується для обробки запитів до вебсайтів та відображення вебсторінок користувачам. MySQL це система керування реляційними базами даних та відповідає за зберігання даних вебдодатка. Для старту роботи цих модулів у стовпці «Actions» необхідно натиснути кнопки «Start».

4. Після цих дій вебдодаток почне свою роботу і необхідно зайти на сайт щоб переконатися в його справності.

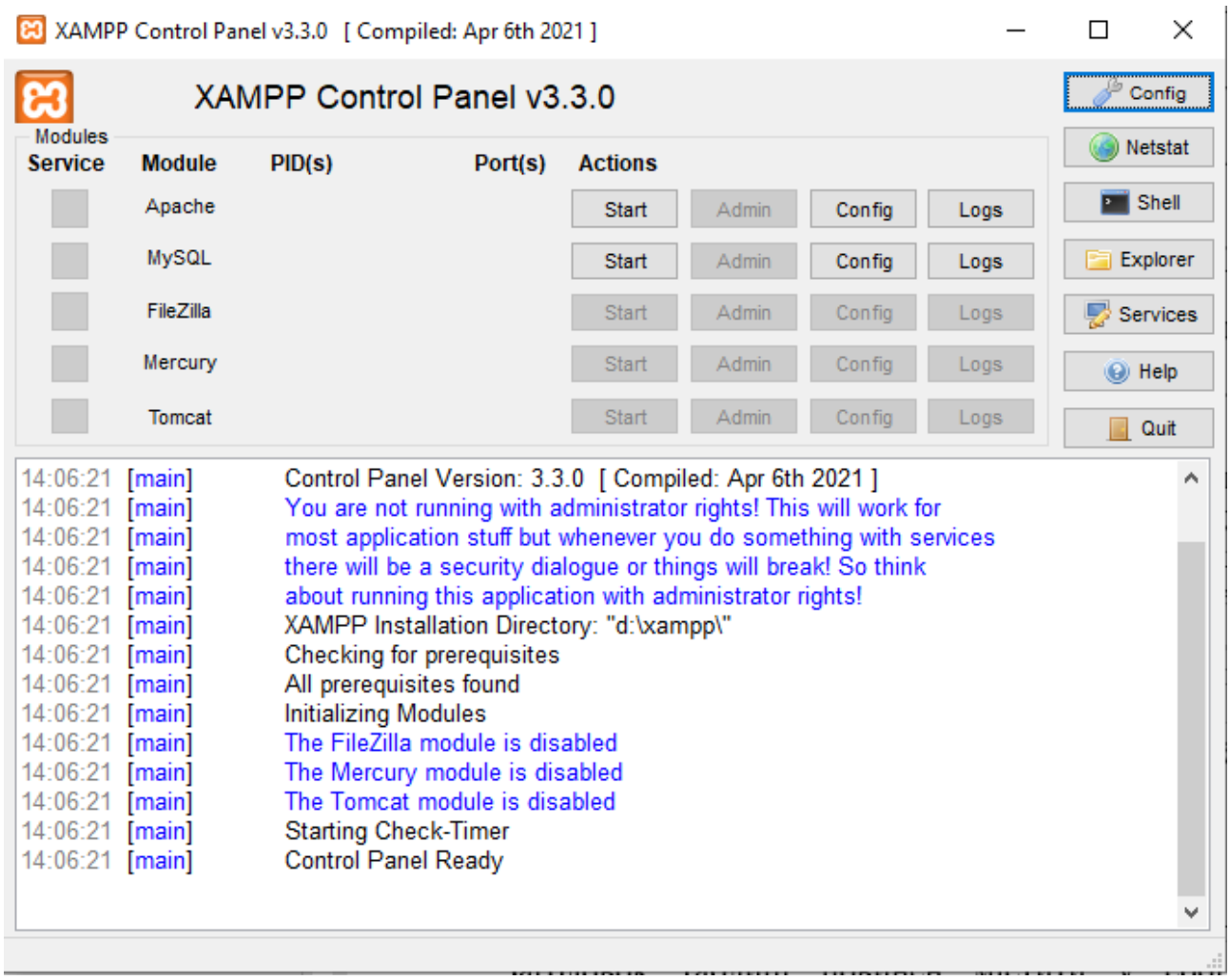

Рис. 2.7. Приклад інтерфейсу XAMPP Control Panel

Для виклику запущеного вебдодатку слід виконати такі дії:

1. Якщо тест програми проходить на тому ж пристрої, де вона запущена, то достатньо перейти в будь-якому браузері за локальною адресою. Це може бути адреси «127.0.0.1» або «localhost».

2. Для доступу з іншого пристрою, що знаходиться в одній мережі з сервером, необхідно дізнатися локальну адресу сервера і в браузері перейти за цією адресою.

2.6.4. Опис інтерфейсу користувача

Після того, коли додаток був запущений на вебсервері, користувач може відкрити сайт у новій вкладці браузеру. Вперше відкривши сторінку, користувач

потрапляє на початкову сторінку (рис 2.8). На цій сторінці можливо або зареєструвати новий аккаунт, або увійти у вже існуючий.

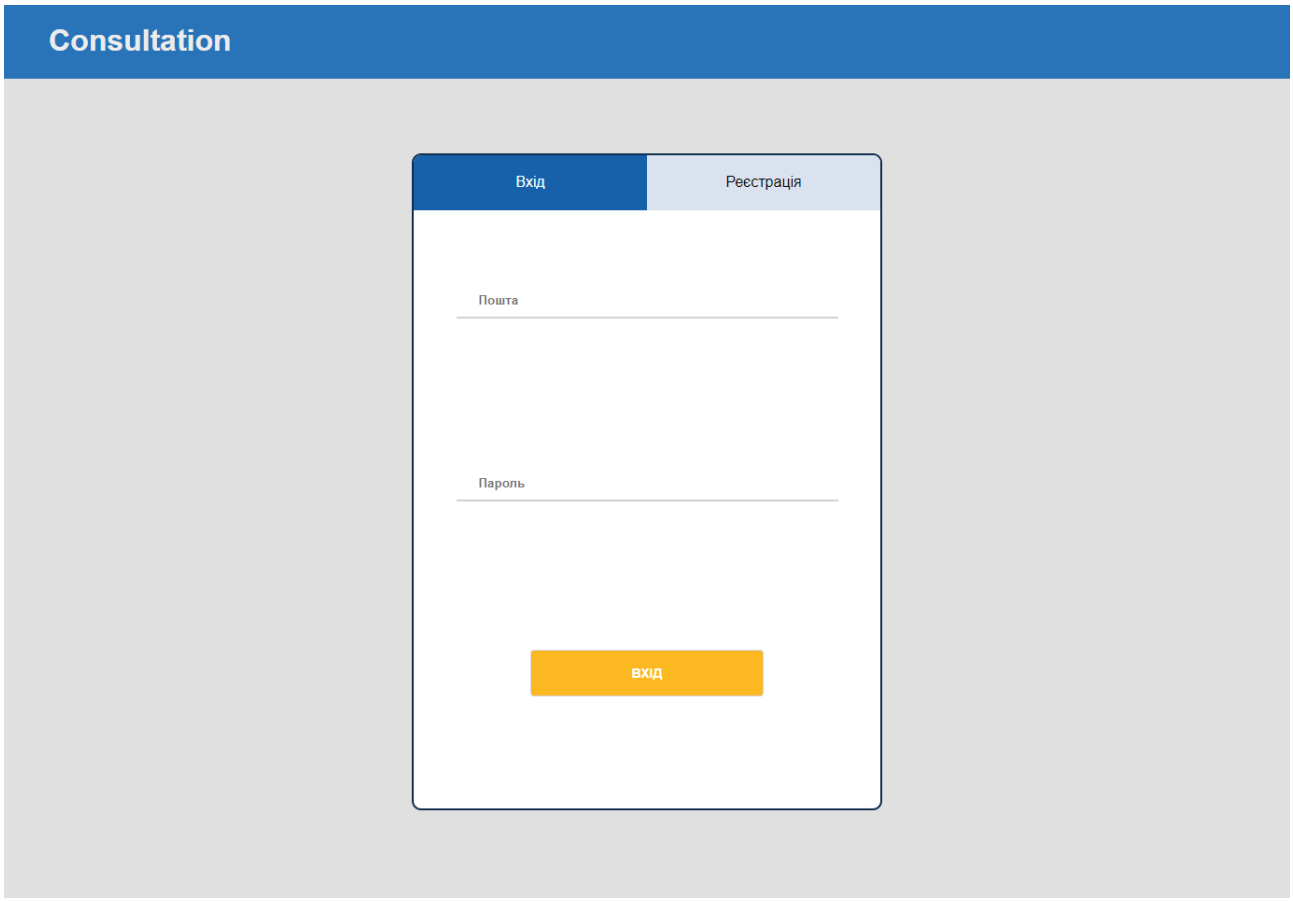

Рис. 2.8. Початкова сторінка

Для того щоб перемикатися між формами для авторизації та реєстрації використовуються кнопки «Вхід» та «Реєстрація» у верхній панелі форми (рис 2.9).

Рис. 2.9. Кнопки «Вхід» та «Реєстрація»

Якщо користувач бажає зареєструватися у вебдодатку, він переходить на другу форму «Реєстрація», де йому необхідно заповнити всі поля (рис 2.10). Для

відправки форми необхідно такі поля: пошта, прізвище, ім’я, по батькові, факультат, пароль та групу лише для студентів. Крім цього, необхідно обрати роль студента або викладача.

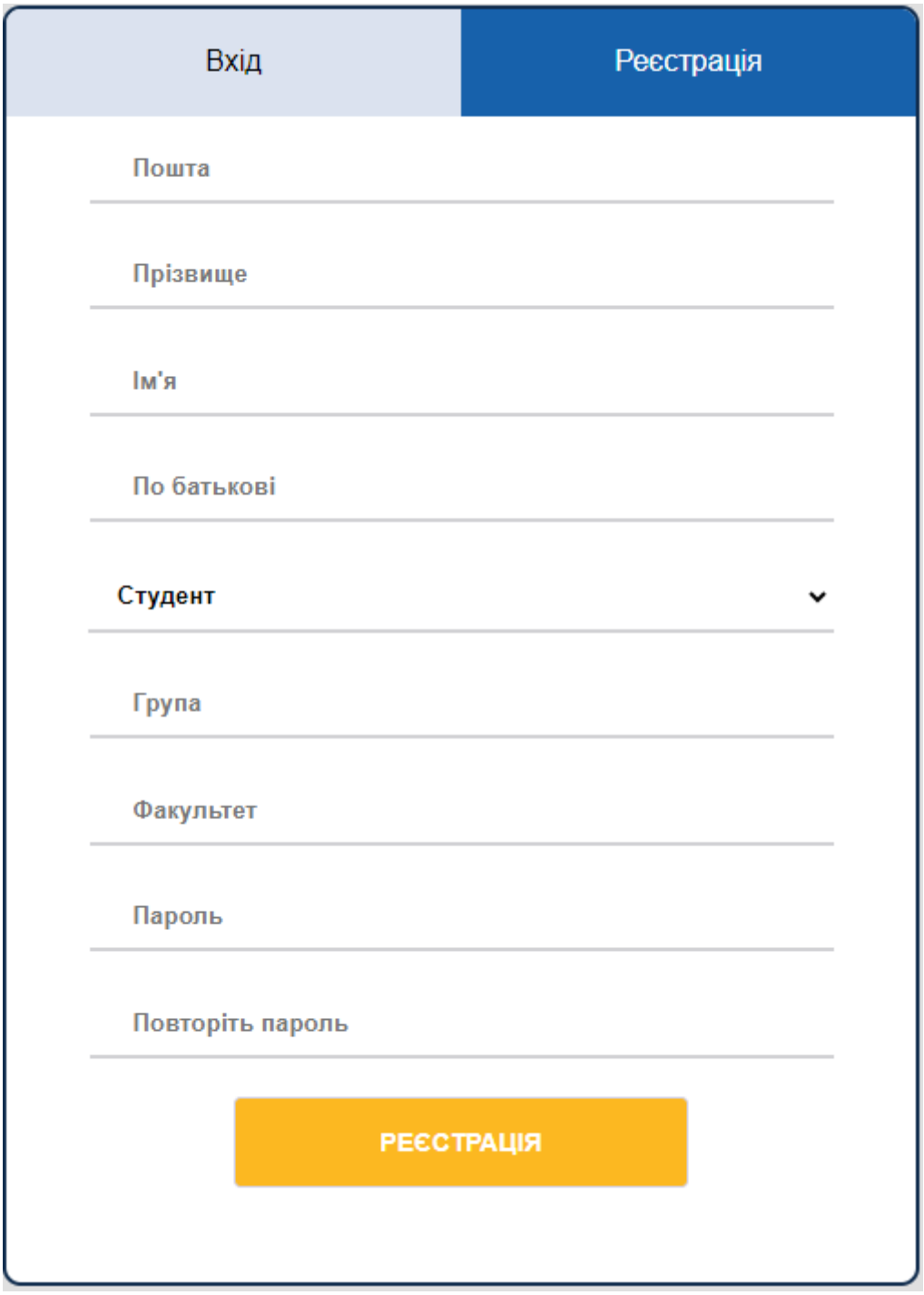

Рис. 2.10. Форма реєстрації

Після заповнення форми необхідно натиснути на жовту кнопку «Реєстрація», що знаходиться внизу форми. Якщо була допущена помилка при

заповнені одного з полів, користувач побачить відповідне повідомлення, а також те поле буде помічене червоним кольором (рис 2.11).

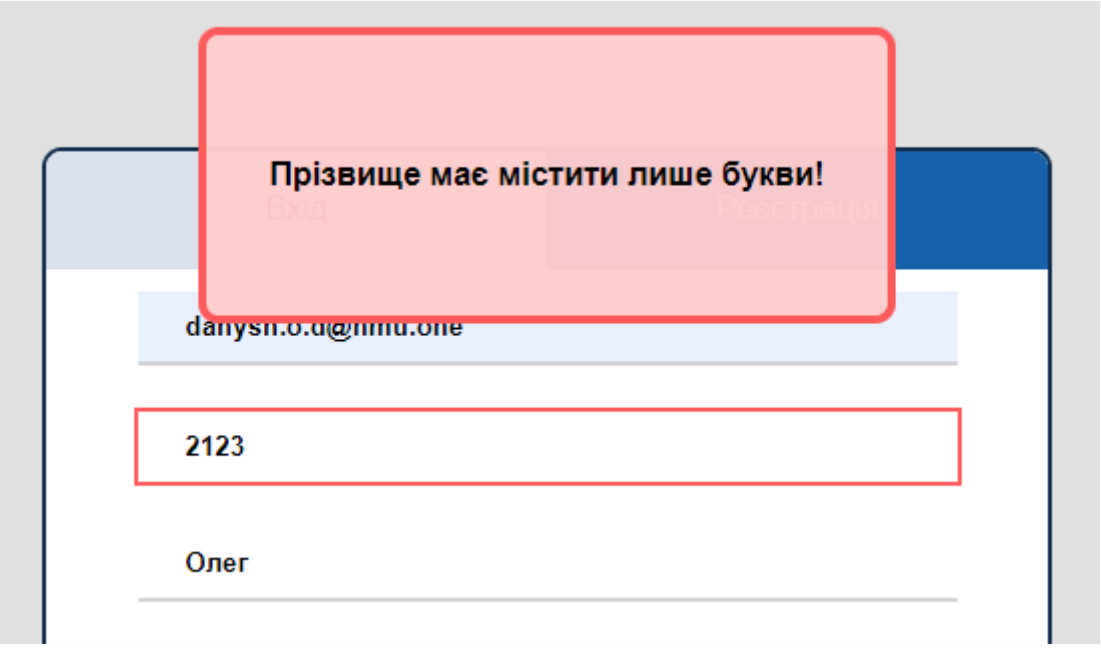

Рис. 2.11. Повідомлення про помилку під час реєстрації

Після вдалої реєстрації користувачу показується відповідне повідомлення, а також його повертають на початкову сторінку для подальшої авторизації (рис 2.12).

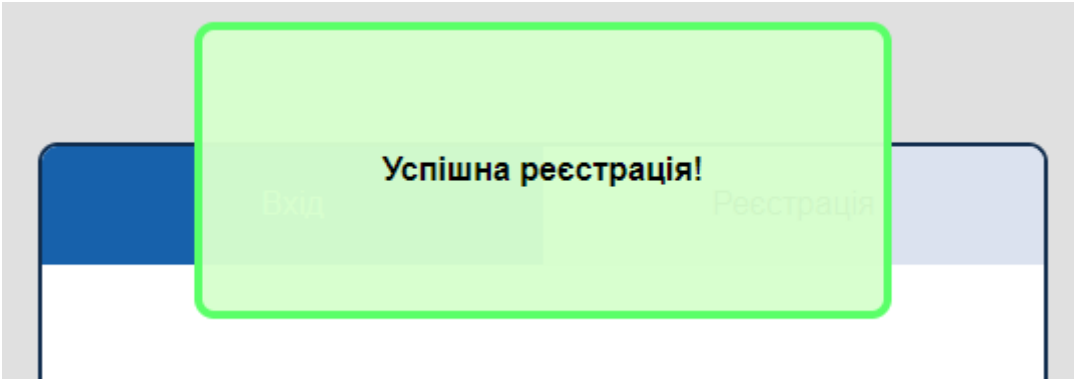

Рис. 2.12. Повідомлення про успішну реєстрацію

На формі авторизації користувачу пропонується ввести свою пошту та пароль у відповідні поля (рис 2.13). Натиснувши на жовту кнопку «Вхід» внизу форми, буде відправлено запит на авторизацію.

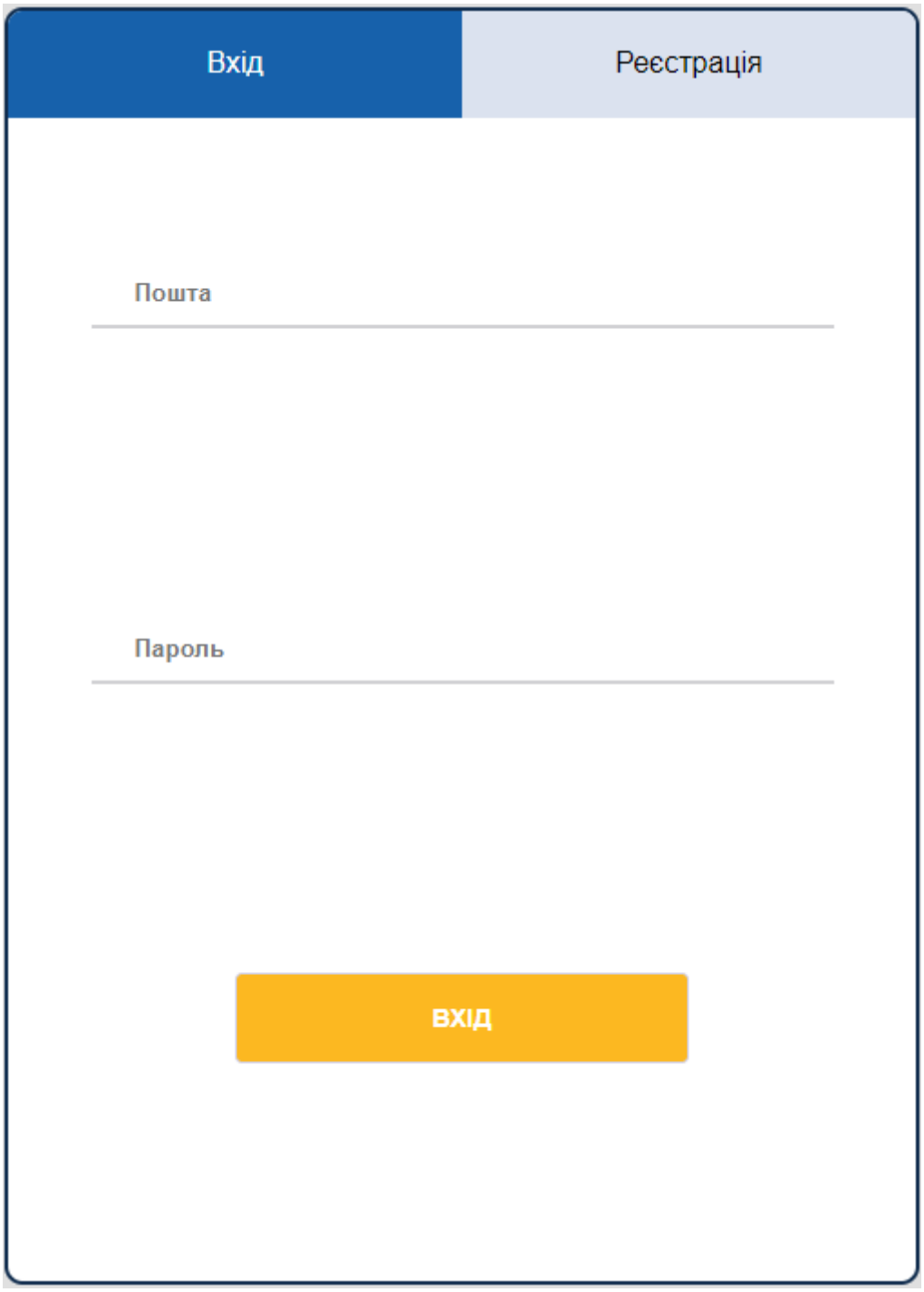

Рис. 2.13. Форма для авторизації

Якщо користувач помилився при введені пошти або пароля, система відобразить відповідну помилку (рис 2.14).

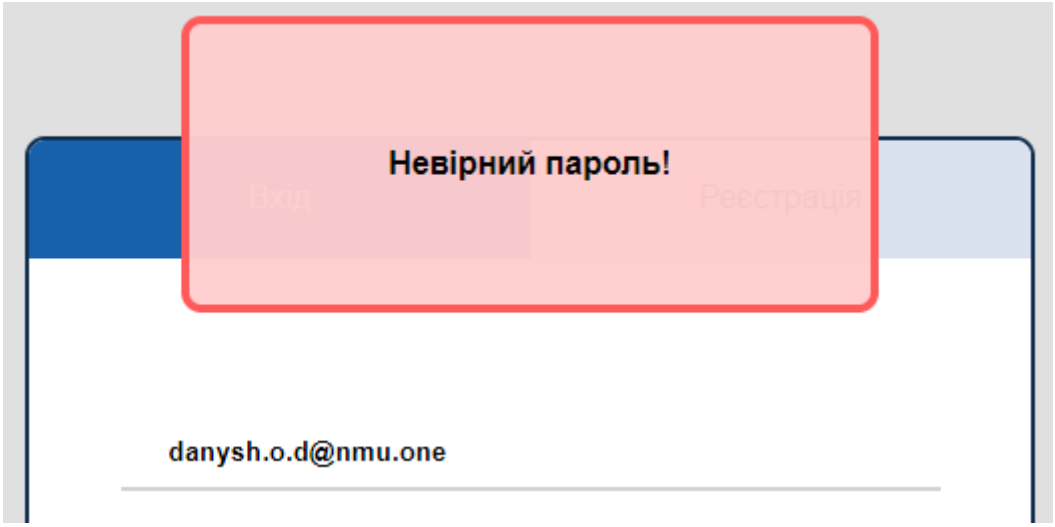

Рис. 2.14. Повідомлення про помилку під час авторизації

Після успішної авторизації студент потрапляє на головну сторінку сайту (рис 2.15). На цій сторінці доступний пошук бажаних консультацій, а також меню користувача для навігації по сайту.

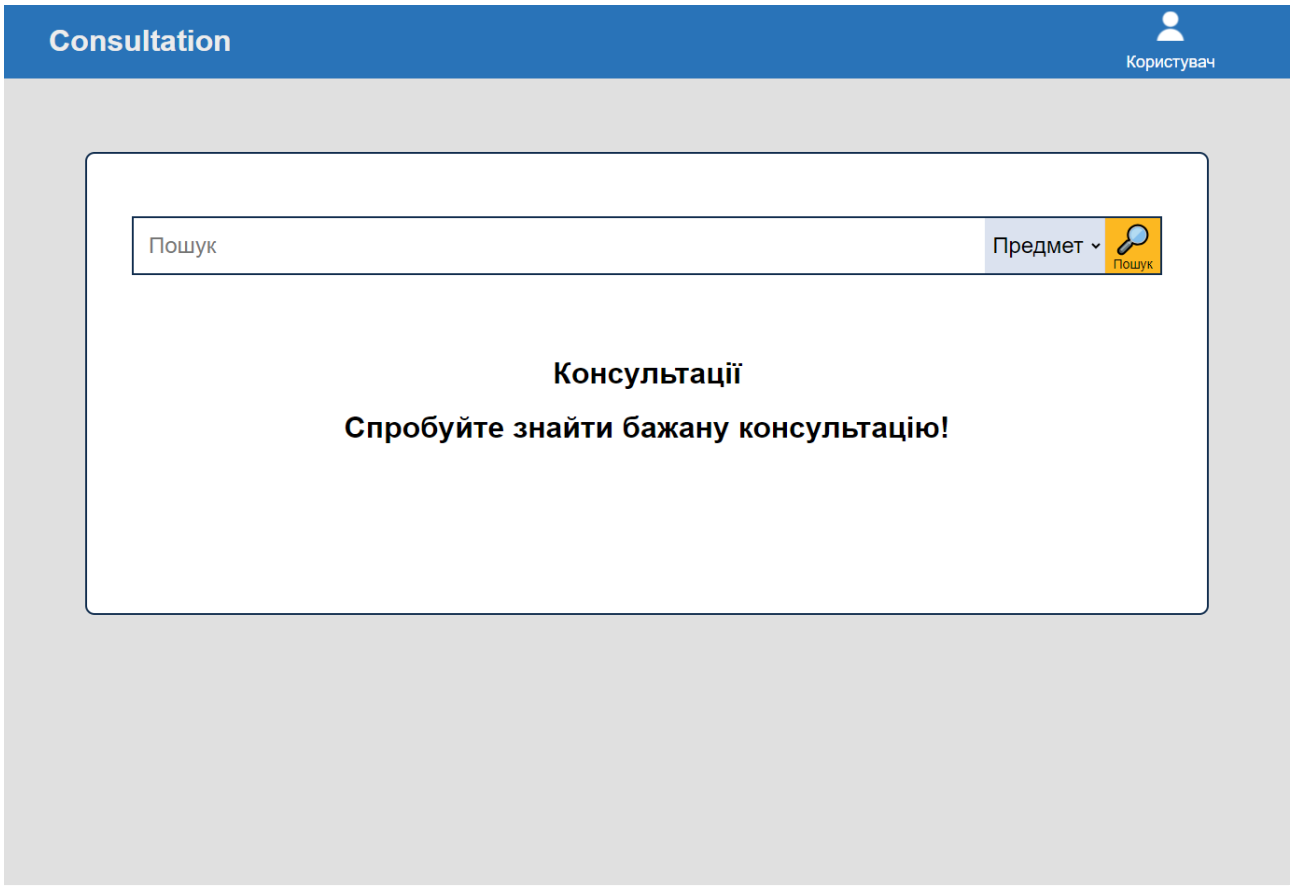

Рис. 2.15. Головна сторінка

Натиснувши на кнопку «Користувач» йому відкриється список з трьох можливих дій (рис 2.16). Перші дві кнопки використовуються в ролі навігаторів по сайту і дозволяють переходити між вкладками «Головна» та «Мій кабінет». Третя кнопка «Вийти» завершає сесію користувача, а також повертає на початкову сторінку для подальшої авторизації.

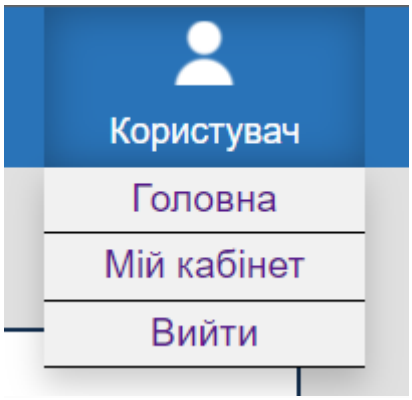

Рис. 2.16. Бокове меню користувача

Для пошуку консультації, необхідно заповнити відповідне поле на головній сторінці (рис 2.17). У списку можна обрати пункт «Предмет» для пошуку за бажаним предметом, або пункт «Викладач» для пошуку за ПІБ викладачя. Після цього для відправки запиту необхідно натиснути на кнопку «Пошук».

Рис. 2.17. Поле для пошуку консультацій

Після вдалого запиту на пошук консультації на сторінці відобразиться таблиця з інформацією про знайдені консультації (рис. 2.18). Студент може

переглянути знайдені варіанти і при бажанні може до них доєднатися, натиснувши на кнопку «Доєднатися» в розділі «Дії».

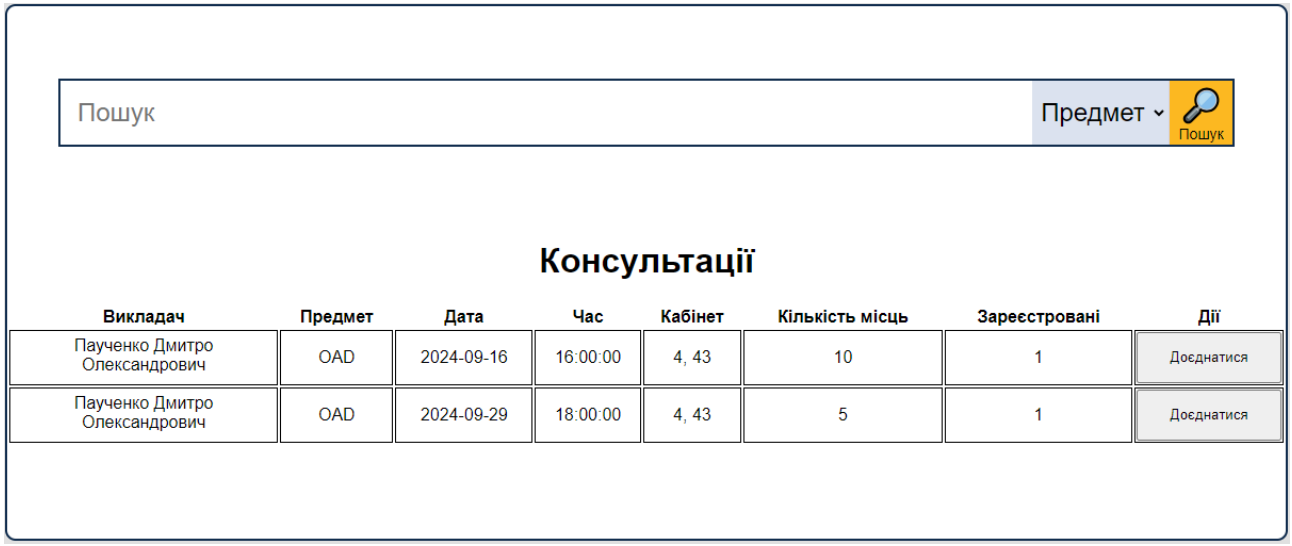

Рис. 2.18. Результат пошуку консультації

Якщо при реєстрації на консультацію станеться помилка, то на сторінці відобразиться відповідне повідомлення з помилкою (рис 2.19). Такими помилками можуть бути: брак місць на консультації, студент вже був зареєстрований на цю консультацію, помилки в роботі БД.

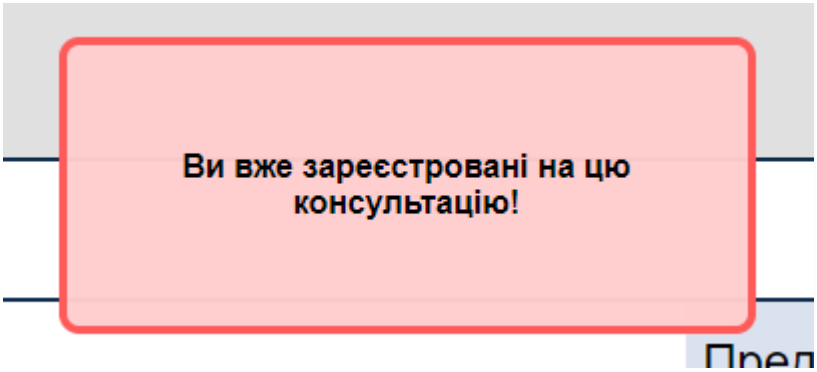

Рис. 2.19. Повідомлення про помилку під час реєстрації на консультацію

Якщо студент вдало зареєструвався на консультацію, він може перейти на сторінку «Мій кабінет», де побачить інформацію про себе, а також інформацію про зареєстровані консультації (рис 2.20). Крім цього, зі свого кабінету студент

може скасувати запис на консультацію, натиснувши на відповідну кнопку «Скасувати» біля запису.

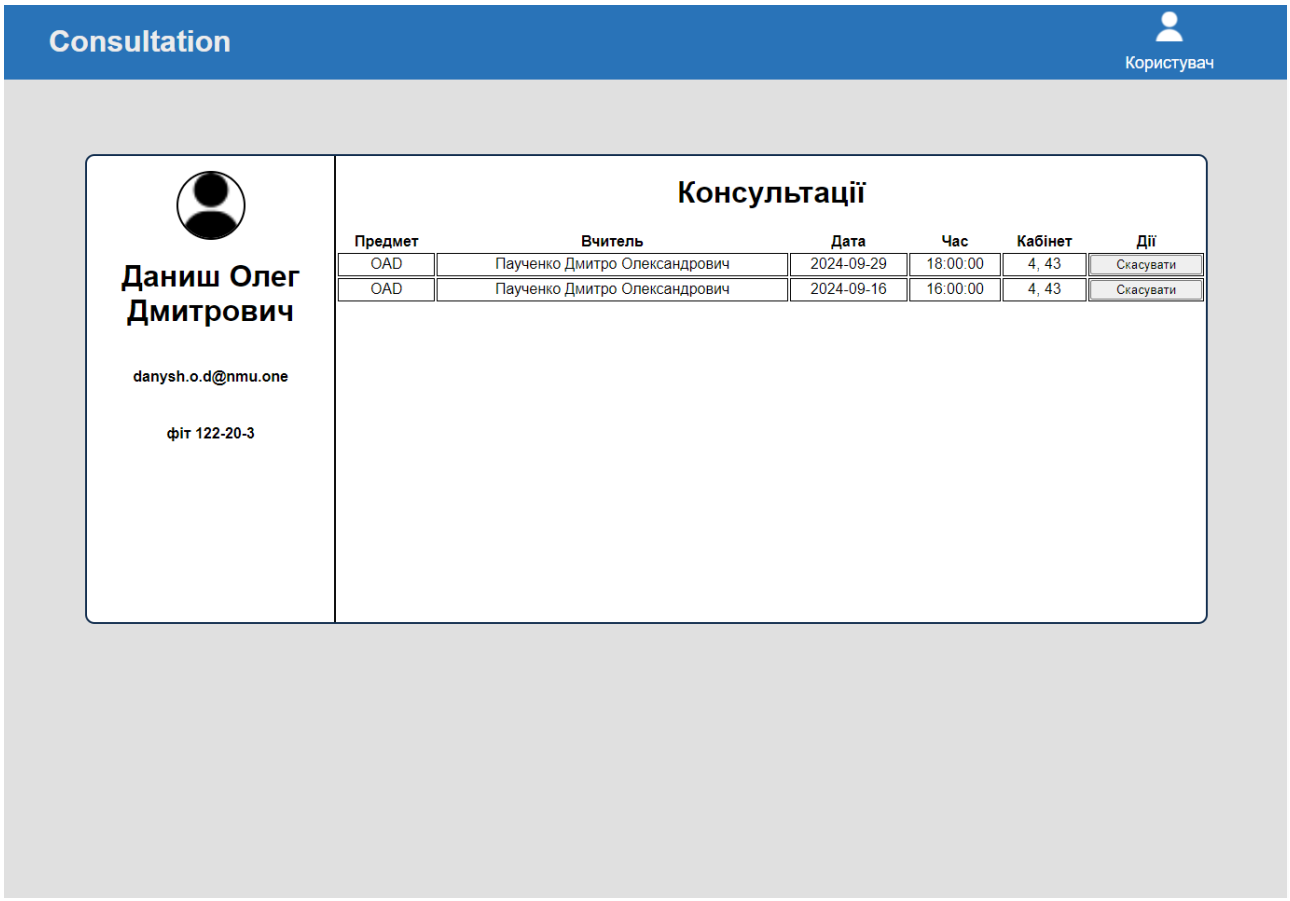

Рис. 2.20. Сторінка кабінету студента

Сторінка кабінету викладачя, за структурою схожа на кабінет студента (рис 2.21). Однак, в розділі «Консультації» викладачю показуються ті консультації, які були ним створені, а також присутня додаткова кнопка «Створити» для створення нової консультації.

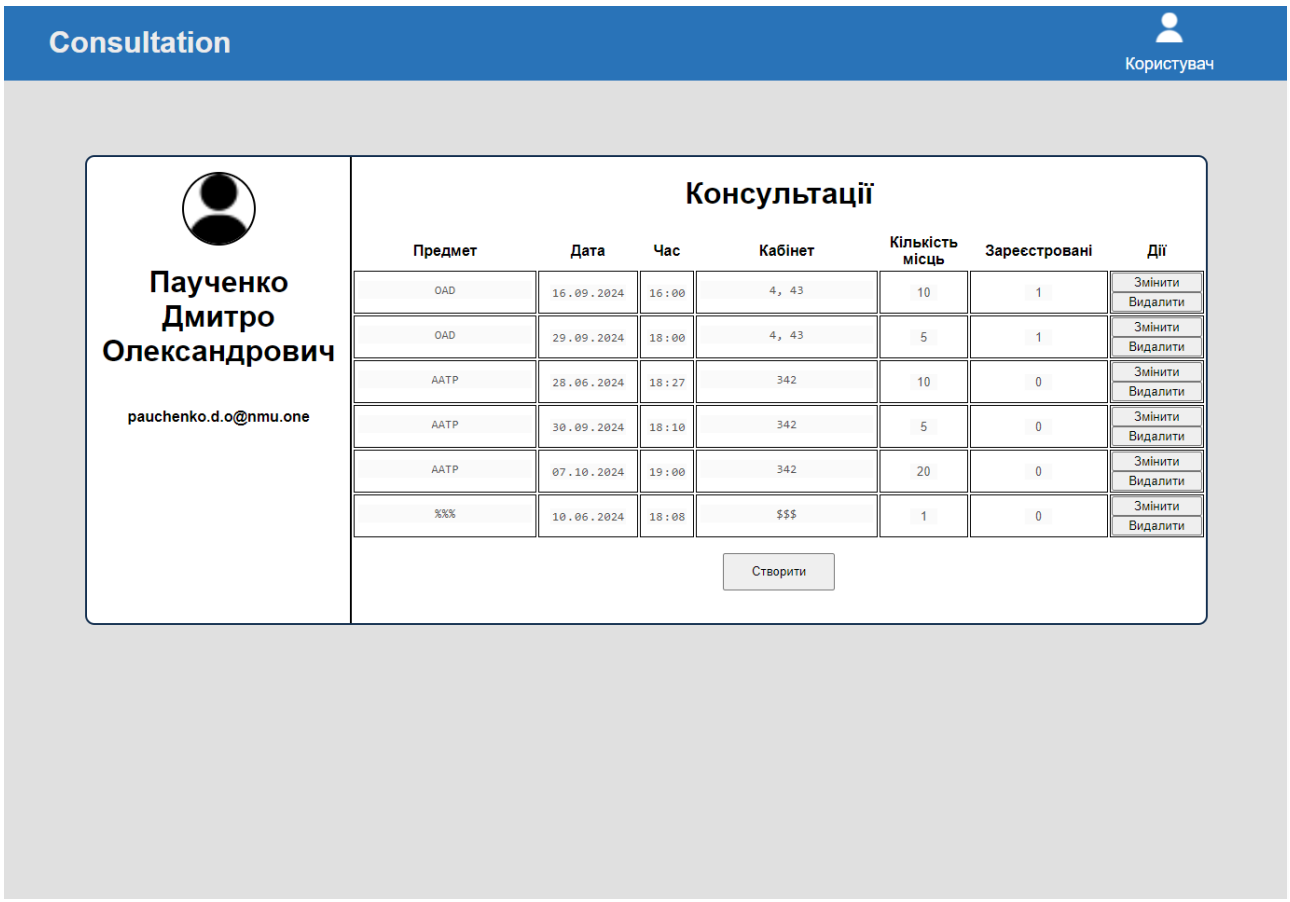

Рис. 2.21. Сторінка кабінету викладача

Викладачю, також, доступні додаткові дії з консультаціями. Наприклад, натиснувши на кнопку «Видалити», можна видалити непотрібну консультацію. Після натискання на цю кнопку, з’явиться вікно для підтвердження дії (рис 2.22).

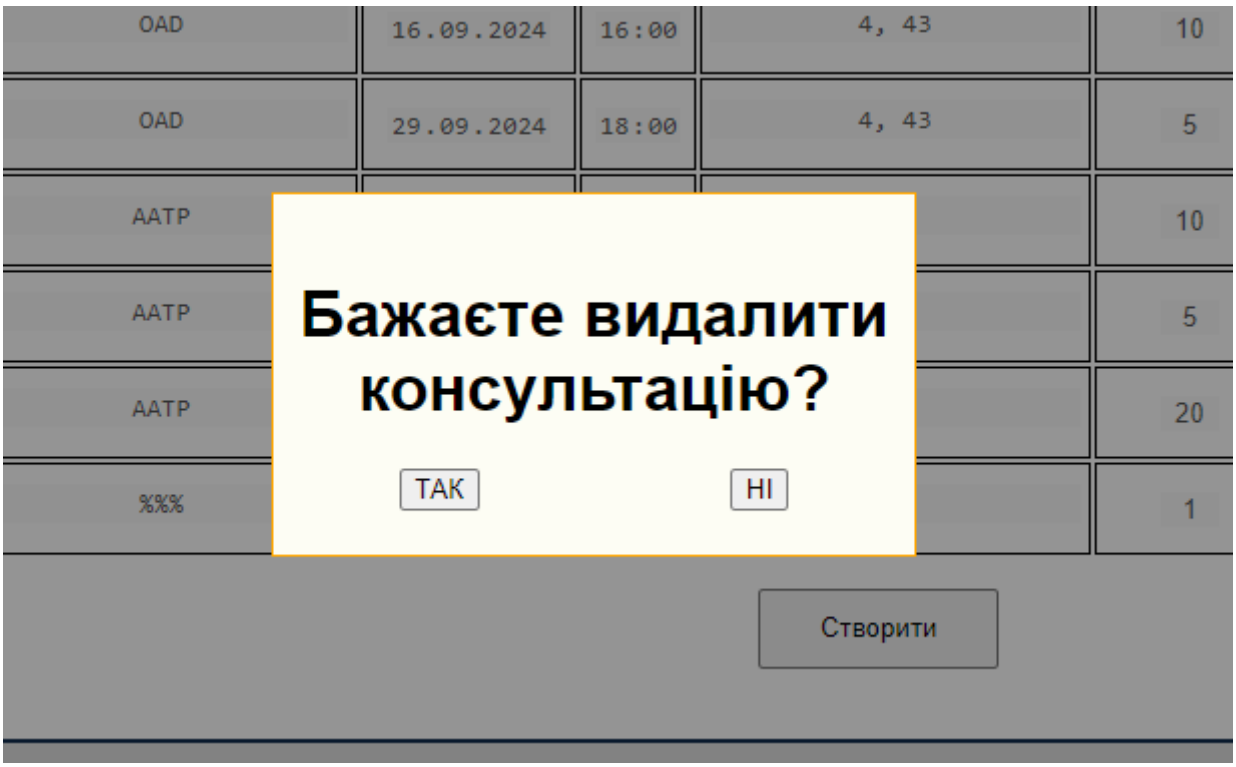

Рис. 2.22. Вікно для підтвердження видалення консультації

Якщо викладач погоджується, він натискає на кнопку «ТАК» і, в разі успішного видалення, на сторінку виведеться відповідне повідомлення, а видалена консультація пропаде зі списку (рис 2.23).

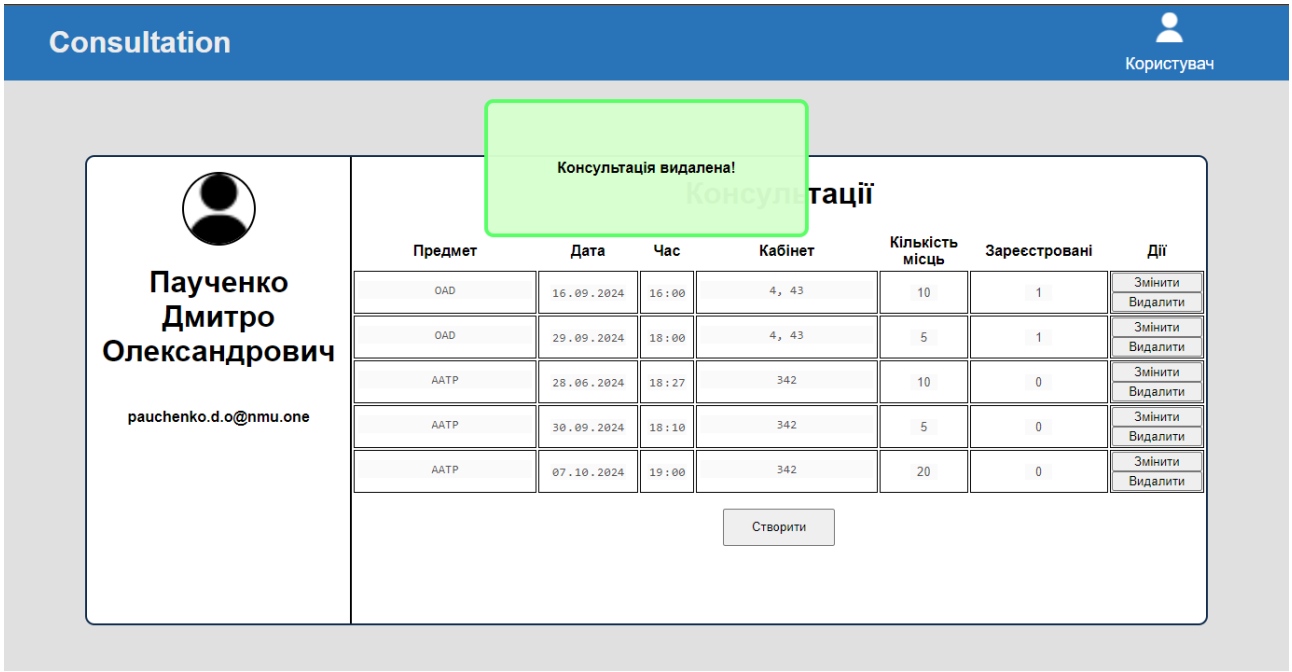

Рис. 2.23. Повідомлення про успішне видалення

При необхідності виправити інформацію про вже існуючу консультацію, викладач повинен натиснути на кнопку «Змінити» (рис 2.24). Після цього він зможе редагувати дані, а також з’являться дві додаткові кнопки. Кнопка «Скасувати» відповідає за скасування введених змін, а кнопка «Зберегти» за береження оновлених даних.

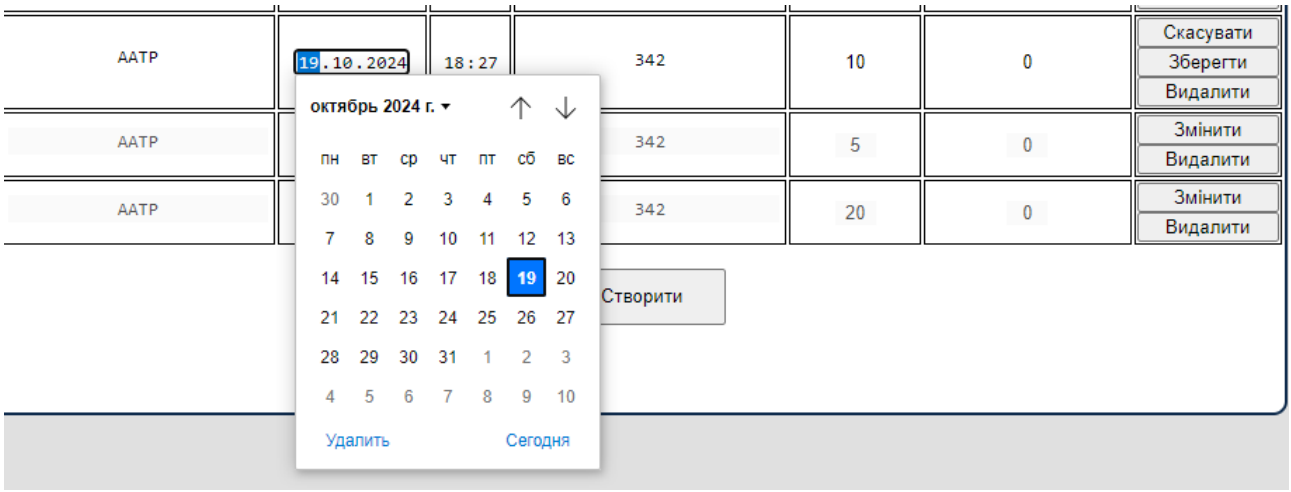

Рис. 2.24. Редагування інформації про консультацію

Після заповнення нової інформації, необхідно натиснути на кнопку «Зберегти» для збереження змін. Якщо інформація була вірно заповнена, то, в результаті, дані оновляться, а викладачю відобразиться повідомлення про успішність запиту(рис 2.25).

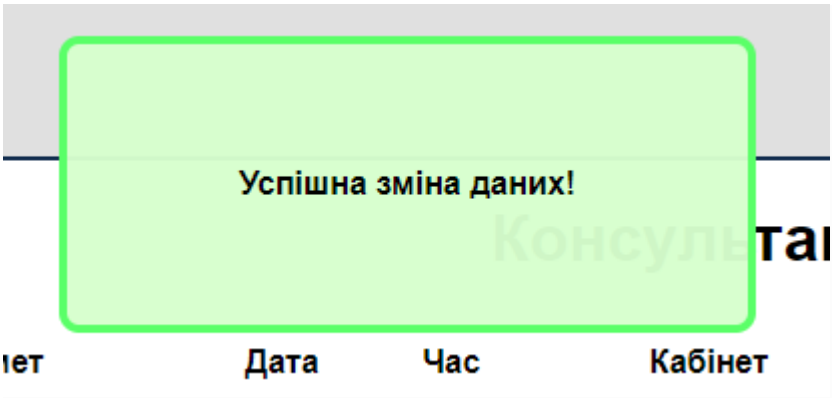

Рис. 2.25. Повідомлення про успішну зміну даних

Якщо необхідно створити нову консультацію, викладач може натиснути на кнопку «Створити», після чого з’явиться нова форма для заповнення (рис. 2.26). В цій формі пропонується заповнити такі поля: назва предмету, дата та час проведення, місце проведення та кількість вільних місць.

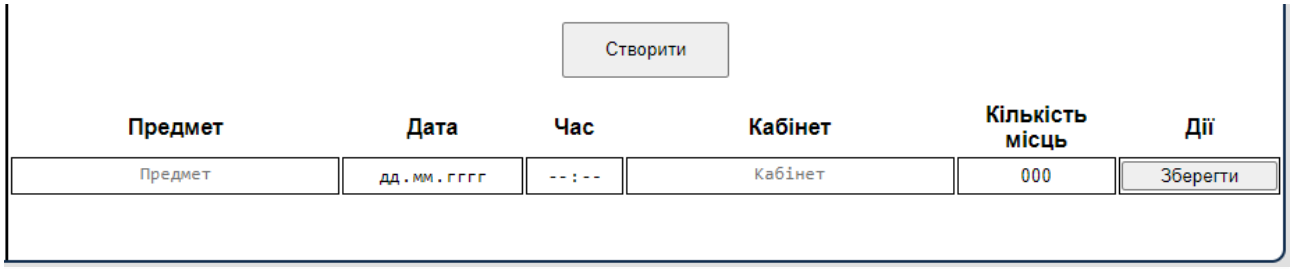

Рис. 2.26. Кнопка і форма для створення консультації

Після натискання на кнопку «Зберегти», нова консультація буде збережена в БД, а на сторінці відобразиться відповідне повідомлення (рис 2.27).

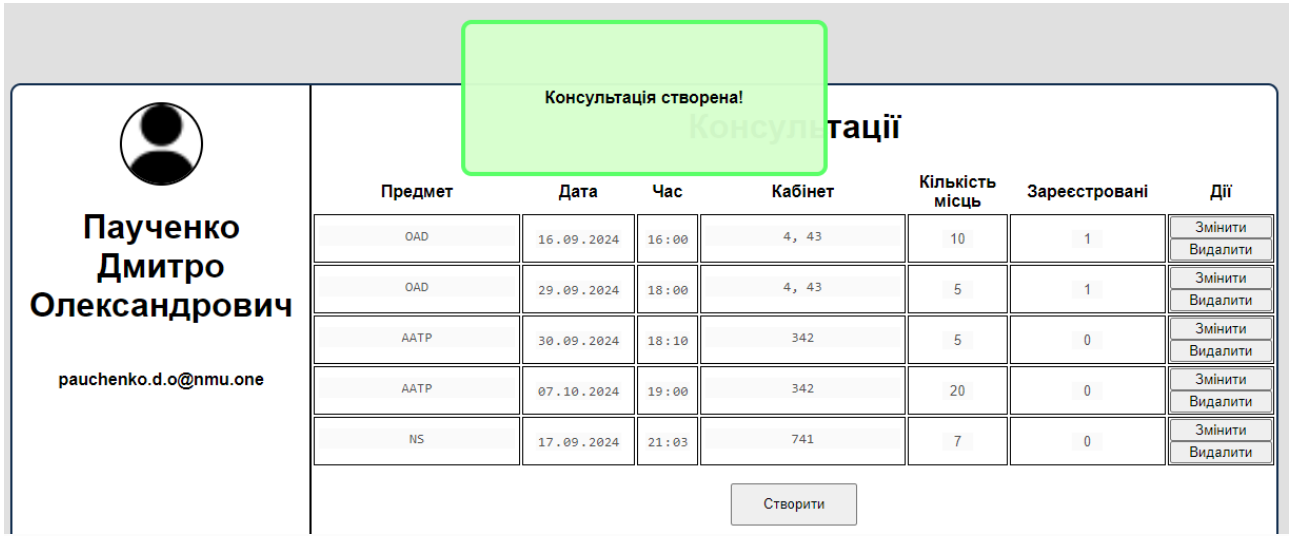

Рис. 2.27. Повідомлення про успішне створення нової консультації

Якщо адміністратор вебдодатку бажає авторизуватися, йому потрібно ввести у відповідні поля дані йому пошту та пароль. При успішній авторизації та переході на головну сторінку, адміністратору стають доступні додаткові функції пошуку (рис 2.28). За допомогою них, він може обрати бажаний проміжок часу, за який виведуться всі консультації у урахуванням запиту. Адміністратор може

переглянути кількість консультацій за створений період та їх відвідуваність студентами.

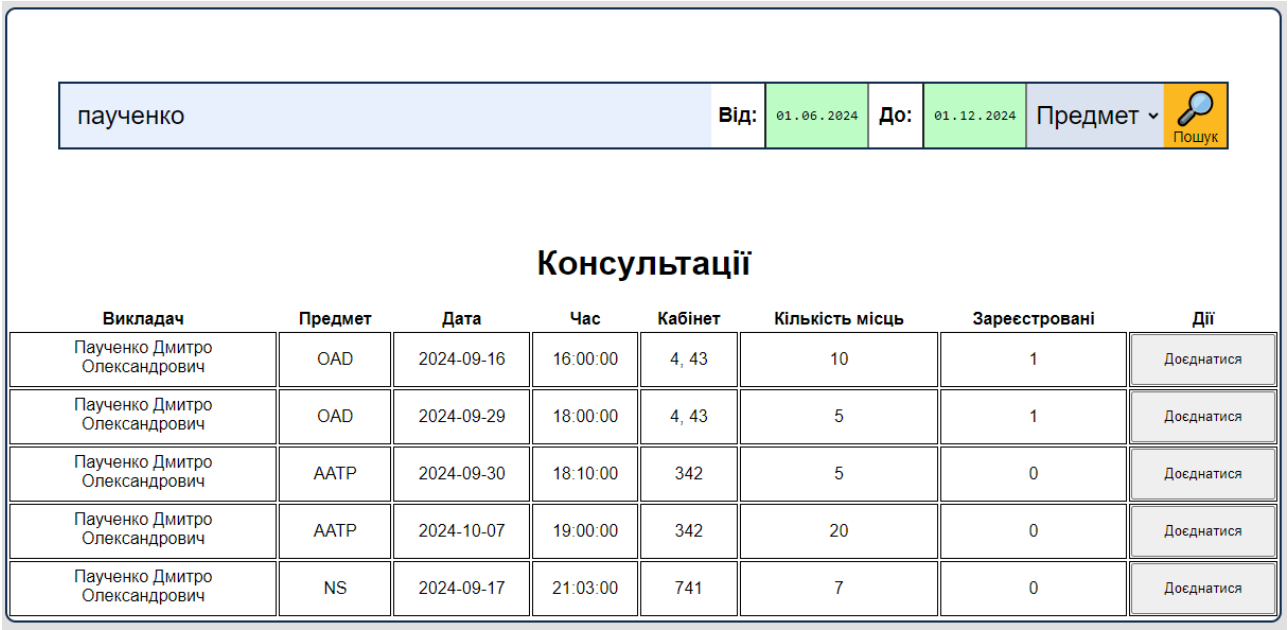

Рис. 2.28. Пошук консультацій за встановлений проміжок часу

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості розробки програмного забезпечення

Вихідні дані:

1. Передбачуване число операторів програми: 2266;
2. Коефіцієнт складності програми: 1,4;
3. Коефіцієнт корекції програми в ході її розробки: 0,09
4. Годинна заробітна плата розробника рівня Middle: 629,53 грн/год. Згідно зі статистикою наданою сайтом DOU за посиланням https://jobs.dou.ua/salaries/?period=2023-12&position=Middle%20SE. Було визначено що медіана заробітної плати для розробника рівня Middle становить 2500 доларів за місяць. При 160 робочих годин на місяць, ми можемо визначити погодинну плату, яка становить 15,625 дол/год. Звернувшись до сайту Національного банку України за посиланням https://bank.gov.ua/ua/markets/exchangerate-chart?cn%5B%5D=USD, знаходимо поточний курс долара, а саме 40,29 на момент написання диплому. Маючи всі дані можна розрахувати годинну заробітну плату, яка дорівнює 629,53 грн/год.
5. Коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі: 1,25;
6. Коефіцієнт кваліфікації програміста, залежний від стажу роботи: 1;
7. Вартість машино-години ЕОМ: 3,38 грн/год. Для розробки вебдодатку до кваліфікаційної роботи було використано ноутбук, який споживає електроенергію та використовує постійне підключення до інтернету. Щомісячна плата за інтернет від провайдера Dnepro.net становить 180 грн/місяць. Ноутбук при роботі споживає 90Вт. На розробку вебдодатку було витрачено близько 120 годин впродовж двух місяців. Згідно тарифного плану, який можна знайти на сайті Yasno за посиланням https://yasno.com.ua/b2c-tariffs , кВт/год коштує 4,32 грн з урахуванням ПДВ. Маючи всі дані ми можемо порахувати, що під час

роботи ноутбук за годину споживав 90 Вт і по тарифу це виходить 0,38 грн/год. Вартість інтернету на годину становить 3 грн/год, яку ми отримали поділивши місячну плату за інтернет на кількість годин проведених за роботою в одному місяці. Загальна вартість машино-години вийшла 3,38 грн/год.

Трудомісткість розробки програмного забезпечення можна розрахувати за допомогою формули:

$$t = t_0 + t_u + t_а + t_n + t_{отл} + t_д \,, \tag{3.1}$$

де t_o - витрати праці на підготовку й опис поставленої задачі (приймається 50);

t_u - витрати праці на дослідження алгоритму рішення задачі;

t_a - витрати праці на розробку блок-схеми алгоритму;

t_n - витрати праці на програмування по готовій блок-схемі;

$t_{отл}$ - витрати праці на налагодження програми на ЕОМ;

$t_д$ - витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у програмному забезпеченні, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q \cdot C \cdot (1 + p) \,, \tag{3.2}$$

де q – передбачуване число операторів (2266);

C – коефіцієнт складності програми (1,4);

p – коефіцієнт корекції програми в ході її розробки (0,09).

Після підстановки значень в формулу (3.2) ми отримаємо такий результат:

$$Q = 2266 \cdot 1{,}4 \cdot (1 + 0{,}09) = 3457{,}916$$

Витрати праці на вивчення опису задачі t_u визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{\text{Q}\cdot\text{B}}{(75..85)\cdot\text{k}}, \quad (3.3)$$

де B – коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі (1,25);

k – коефіцієнт кваліфікації програміста, обумовлений стажем роботи з даної спеціальності (1);

Після підстановки значень в формулу (3.3) ми отримаємо такий результат:

$$t_u = \frac{3457{,}916\cdot 1{,}25}{77\cdot 1} = 56{,}14 \text{ людино-годин}$$

Витрати праці на розробку алгоритму рішення задачі визначаються за формулою:

$$t_{\text{a}} = \frac{Q}{(20..25)\cdot\text{k}}, \quad (3.4)$$

де Q – умовне число операторів програми (3457,916);

k – коефіцієнт кваліфікації програміста (1).

Після підстановки значень в формулу (3.4) ми отримаємо такий результат:

$$t_{\text{a}} = \frac{3457{,}916}{22\cdot 1} = 157{,}18 \text{ людино-годин}$$

Витрати на складання програми по готовій блок-схемі визначається за формулою:

$$t_n = \frac{Q}{(20..25)\cdot\text{k}}, \quad (3.5)$$

Після підстановки значень в формулу (3.5) ми отримаємо такий результат:

$$t_n = \frac{3457,916}{24 \cdot 1} = 144,08 \text{ людино-годин}$$

Витрати праці на налагодження програми на ЕОМ:

– за умови автономного налагодження одного завдання:

$$t_{\text{отл}} = \frac{Q}{(4..5) \cdot \text{k}}, \quad (3.6)$$

Після підстановки значень в формулу (3.6) ми отримаємо такий результат:

$$t_{\text{отл}} = \frac{3457,916}{4,4 \cdot 1} = 785,89 \text{ людино-годин}$$

– за умови комплексного налагодження завдання:

$$t_{\text{отл}}^{k} = 1,5 \cdot t_{\text{отл}}, \quad (3.7)$$

Після підстановки значень в формулу (3.7) ми отримаємо такий результат:

$$t_{\text{отл}}^{k} = 1,5 \cdot 785,89 = 1178,84 \text{ людино-годин}$$

Витрати праці на підготовку документації:

$$t_{\text{д}} = t_{\text{др}} + t_{\text{до}}, \quad (3.8)$$

де $t_{\text{др}}$ трудомісткість підготовки матеріалів і рукопису;

$$t_{\text{др}} = \frac{Q}{(15..20) \cdot \text{k}}, \quad (3.9)$$

Після підстановки значень в формулу (3.9) ми отримаємо такий результат:

$$t_{\text{др}} = \frac{3457{,}916}{17 \cdot 1} = 203{,}41 \text{ людино-годин}$$

де $t_{\text{до}}$ - трудомісткість редагування, печатки й оформлення документації:

$$t_{\text{до}} = 0{,}75 \cdot t_{\text{др}} , \qquad (3.10)$$

Після підстановки значень в формулу (3.10) ми отримаємо такий результат:

$$t_{\text{до}} = 0{,}75 \cdot 203{,}41 = 152{,}56 \text{ людино-годин}$$

В результаті за формулою (3.8) отримаємо такий результат:

$$t_{\text{д}} = 203{,}41 + 152{,}56 \text{ людино-годин}$$

Повертаючись до формули (3.1), отримаємо повну оцінку трудомісткості розробки програмного забезпечення:

$$t = 50 + 56{,}14 + 157{,}18 + 144{,}08 + 785{,}89 + 152{,}56 = 1345{,}85$$

За результатами розрахунків, загальна трудомісткість розробки даного програмного засобу складає 1688,8 людино-годин.

3.2. Розрахунок витрат на створення програмного забезпечення

Витрати на створення ПЗ $К_{ПО}$ включають витрати на заробітну плату виконавця програми $З_{ЗП}$ і витрат машинного часу, необхідного на налагодження програми на ЕОМ:

$$K_{ПО} = З_{ЗП} + З_{МВ}\ , \quad (3.11)$$

$З_{ЗП}$ – заробітна плата виконавців, яка визначається за формулою:

$$З_{ЗП} = t \cdot C_{ПР}\ , \quad (3.12)$$

де t - загальна трудомісткість, людино-годин;

$C_{ПР}$ – середня годинна заробітна плата програміста, грн/година.

Середня плата за одну годинну роботи програміста становить 629,53 грн/год. Після підстановки значень в формулу (3.12) ми отримаємо такий результат:

$$З_{ЗП} = 1345{,}85 \cdot 629{,}53 = 847252{,}95 \text{ грн.}$$

Вартість машинного часу, необхідного для налагодження програми на ЕОМ, визначається за формулою:

$$З_{МВ} = t_{отл} \cdot C_{мч}\ , \quad (3.13)$$

де $t_{отл}$– трудомісткість налагодження програми на ЕОМ, год;

$C_{мч}$ – вартість машино-години ЕОМ, грн/год (3,38).

Після підстановки значень в формулу (3.13) ми отримаємо такий результат:

$$З_{МВ} = 785{,}89 \cdot 3{,}38 = 2656{,}31 \cdot \text{грн.}$$

Звідси за формулою (3.11) розраховуємо витрати на створення програмного забезпечення:

$$K_{ПО} = 847252{,}95 + 2656{,}31 = 849909.26 \text{ грн.}$$

Очікуваний період створення ПЗ:

$$T = \frac{\mathrm{t}}{B_k \cdot \mathrm{F}_p}, \tag{3.14}$$

де B_k – число виконавців (1);

F_p – місячний фонд робочого часу (при 40 годинному робочому тижні F_p = 176 годин).

Після підстановки значень в формулу (3.14) ми отримаємо такий результат:

$$T = \frac{1345{,}85}{1 \cdot 176} = 7{,}6 \text{ міс.}$$

Висновок: для розробки вебдодатку «Consultations» за підрахунками знадобиться 1345,85 людино-години. З урахуванням курсу долара та середньої заробітної плати програміста, на момент написання роботи, для розробки програмного забезпечення знадобиться 849909,26 грн. Приблизний час, який знадобиться для розробки складає 7,6 місяців при роботі одного програміста рівня Middle, зі стажем роботи 2 роки та тривалістю робочого часу не більше 40 годин на тиждень.

ВИСНОВКИ

Під час виконання даної роботи було розроблено вебдодаток для запису на оффлайн консультації з використанням HTML, CSS, PHP, JavaScript. Окрім додатку, було створено базу даних з використанням MySQL.

Створений додаток дозволяє студентам та викладачам НТУ «Дніпровська політехніка» вести зручний графік, швидко узгоджувати дату та час консультацій, завчасно планувати свій робочий день.

Для виконання поставлених завдань та досягнення мети у вебдодатку були реалізовані такі можливості, як: створення, редагування та видалення консультацій, запис на створені консультації, пошук зручних консультацій за предметом, викладачем, часом та датою, заповнення та перегляд контактних даних у разі виникнення непередбачуваних подій.

При написані додатку було дотримано всіх інформаційних, технічних та функціональних вимог зазначених у першому розділі роботи.

В результаті, розроблена платформа дає змогу автоматизувати процеси запису на оффлайн консультації, покращити комунікацію, підвищити ефективність роботи викладачів та надати можливість студентам зручно та швидко знаходити консультації за їх потребами.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Simpson K. You Don't Know JS Yet: Get Started. – 2020. 143p.
2. Eric Elliott. Programming JavaScript Applications: Robust Web Architecture with Node, HTML5, and Modern JS Libraries. – O'Reilly Media, 2014. 254 p.
3. Zandstra M. PHP Objects, Patterns, and Practice. – Apress, 2016. 576 p.
4. Haverbeke M. Eloquent JavaScript Eloquent JavaScript 3rd edition – San Francisco, No Starch Press, 2018. 472 p.
5. Martin R. C. Clean code: a handbook of agile software craftsmanship. – Pearson Education, 2009. 464 p.
6. Robbins J. N. Learning web design: A beginner's guide to HTML, CSS, JavaScript, and web graphics. 4th ed. – O'Reilly, Beijing. 2012. 603 p.
7. Harris A. HTML5 for dummies quick reference. – John Wiley & Sons, 2011. 1104 p.
8. Frain B. Responsive Web Design with HTML5 and CSS: Build future-proof responsive websites using the latest HTML5 and CSS techniques. – Packt Publishing Ltd, 2022.
9. Robbins J. N. Learning Web Design: A Beginner's Guide to (X) HTML, StyleSheets, and Web Graphics. – " O'Reilly Media, Inc.", 2007.
10. Welling L., Thomson L. PHP and MySQL Web Development (4th Edition) (Developer's Library). – Sams, 2007. 950 p.
11. Навчіться кодувати за допомогою Visual Studio Code. [Електронний ресурс] – URL: https://code.visualstudio.com/learn (дата звернення 25.05.2024).
12. Сайт для вивчення вебдизайну. Flux Academy - Learn How To Become a Web Designer. [Електронний ресурс] – URL: https://www.flux-academy.com/ (дата звернення 25.05.2024).
13. Сайт для вивчення HTML, CSS, JS, SQL. W3Schools Online Web Tutorials. [Електронний ресурс] – URL: https://www.w3schools.com/ (дата звернення 23.05.2024).

14. Документація з HTML, CSS, JS для розробників. MDN Web Docs – Mozilla. [Електронний ресурс] – URL: https://developer.mozilla.org/. (дата звернення 23.05.2024).

15. Документація з PHP. [Електронний ресурс] – URL: https://www.php.net/docs.php. (дата звернення 17.05.2024).

16. Документація та приклади використання PHP. [Електронний ресурс] – URL: https://www.geeksforgeeks.org/php-tutorial/. (дата звернення 17.05.2024).

17. Документація з Regex. Regular Expression Language - Quick Reference. [Електронний ресурс] – URL: https://learn.microsoft.com/en-us/dotnet/standard/base-types/regular-expression-language-quick-reference. (дата звернення 03.05.2024).

18. MySQL Документація для розробників. [Електронний ресурс] – URL: https://dev.mysql.com/doc/ (дата звернення 10.05.2024).

19. Вивчення та туторіали з MySQL. [Електронний ресурс] – URL: https://www.techfry.com/mysql-tutorial (дата звернення 10.05.2024).

20. Що таке MySQL і як він працює. [Електронний ресурс] – URL: https://www.hostinger.com/tutorials/what-is-mysql (дата звернення 25.05.2024).

ДОДАТОК А

КОД ПРОГРАМИ

index.html:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-16"/>
    <meta http-equiv="X-UA-Compatible" content="IE=edge"/>
    <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
    <link rel="stylesheet" href="style.css"/>
    <title>Consultation</title>
  </head>
  <body>
    <div class="header">
      <a href=""><h1>Consultation</h1></a>
    </div>
    <div class="main-container" id="main-container">
      <div class="login-container">
        <div class="login-button" id="log-but" onclick="changePageToLogin()">Вхід</div>
        <div class="registration-button" id="reg-but"
onclick="changePageToRegistration()">Реєстрація</div>
        <div class="login" id="log-page">
          <form action="loginFormHandling.php" method="POST" id="log-form">
            <input type="email" class="text-input" placeholder="Пошта" name="email">
            <input type="password" class="text-input" placeholder="Пароль" name="password">
            <button class="confirm-button" type="button" id="log-button">
              <span>Вхід</span>
            </button>
          </form>
        </div>
        <div class="registration" id="reg-page">
          <form action="registrationFormHandling.php" method="POST" id="reg-form">
            <input type="email" class="text-input" placeholder="Пошта" name="email">
            <input type="text" class="text-input" placeholder="Прізвище" name="surname">
            <input type="text" class="text-input" placeholder="Ім'я" name="name">
            <input type="text" class="text-input" placeholder="По батькові" name="fathers_name">
            <select name="occupation" id="occupation-select" onchange="changeOccupation(this)">
              <option value="student">Студент</option>
              <option value="teacher">Викладач</option>
            </select>
```

```
                    <input type="text" class="text-input" id="group-input-id" placeholder="Група" name="group">
                    <input type="text" class="text-input" placeholder="Факультет" name="faculty">
                    <input type="password" class="text-input" placeholder="Пароль" name="password">
                    <input type="password" class="text-input" placeholder="Повторіть пароль"
name="password_repeat">
                    <button class="confirm-button" type="button" id="reg-button">
                      <span>Реєстрація</span>
                    </button>
                  </form>
                </div>
              </div>
            </div>
            <script src="/diplom/main.js"></script>
            <script src="/diplom/utilities/errorHandling.js"></script>
          </body>
        </html>
```

registrationFormHandling.php:

```
        <?php session_start();

        if($_SERVER["REQUEST_METHOD"] === "POST")
        {
          try {
            require_once("utilities/connectToDB.php");
            require_once("utilities/queryDB.php");
          } catch (Exception $e) {
            header("Location: index.html?message=no_db_connection");
            die();
          }
          if(!isset($_POST["email"], $_POST["name"], $_POST["surname"], $_POST["fathers_name"],
$_POST["occupation"], $_POST["faculty"], $_POST["password"], $_POST["password_repeat"]))
          {
            header("Location: index.html?message=fields_empty");
            die();
          }
          if(!filter_var($_POST["email"], FILTER_VALIDATE_EMAIL))
          {
            header("Location: index.html?message=not_valid_email");
            die();
          }
          $onlyLettersPattern = '/^[A-zA-яіІїЇґҐєЄ]+$/u';
```

```
        if(preg_match($onlyLettersPattern, $_POST["name"]) == 0 || preg_match($onlyLettersPattern,
$_POST["surname"]) == 0 || preg_match($onlyLettersPattern, $_POST["fathers_name"]) == 0 )
        {
           header("Location: index.html?message=only_letters_in_name");
           die();
        }
        if(preg_match($onlyLettersPattern, $_POST["faculty"]) == 0 )
        {
           header("Location: index.html?message=only_letters_in_faculty");
           die();
        }
        $nmuEmailPattern = "/^([A-Za-z]+(\.[A-Za-z]+)+)@nmu\.one$/";
        if(preg_match($nmuEmailPattern, $_POST["email"]) == 0)
        {
           header("Location: index.html?message=wrong_email");
           die();
        }
        if($_POST["password"] !== $_POST["password_repeat"])
        {
           header("Location: index.html?message=diffrent_passwords");
           die();
        }
        if(isEmailAlreadyExist($connection, $_POST["email"]))
        {
           header("Location: index.html?message=email_already_exist");
           die();
        }

        $hashedPassword = password_hash($_POST["password"], PASSWORD_BCRYPT, ['cost' => 12]);

        if($_POST["occupation"] == "student")
        {
           if(insertNewStudent($connection, $_POST["email"], $hashedPassword, $_POST["name"],
$_POST["surname"], $_POST["fathers_name"], $_POST["group"], $_POST["faculty"]) > 0)
           {
              header("Location: index.html?message=successfull_registration");
              die();
           }
           else
           {
              header("Location: index.html?message=something_went_wrong");
              die();
```

```
            }

        }
        else if($_POST["occupation"] == "teacher")
        {
            if(insertNewTeacher($connection, $_POST["email"], $hashedPassword, $_POST["name"],
$_POST["surname"], $_POST["fathers_name"], $_POST["faculty"]) > 0)
            {
                header("Location: index.html?message=successfull_registration");
                die();
            }
            else
            {
                header("Location: index.html?message=something_went_wrong");
                die();
            }
        }
        else
        {
            header("Location: index.html?message=occupation_unknown");
            die();
        }
    }
    else
    {
        header("Location: index.html?message=no_post_method");
        die();
    }

    ?>
```

Решта коду додається окремо.

ДОДАТОК Б

ВІДГУК

керівника економічного розділу

на кваліфікаційну роботу бакалавра на тему:

«Розробка вебдодатку для запису на оффлайн консультації з використанням технологій HTML, CSS, JavaScript, PHP»

студента групи 122-20-3 Даниша Олега Дмитровича

Керівник економічного розділу

доц. каф. ПЕП та ПУ, к.е.н **Л.В. Касьяненко**

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файлу	Опис
Пояснювальні документи	
Даниш О.Д. кваліфікаційна робота.docx	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Даниш О.Д. кваліфікаційна робота.pdf	Пояснювальна записка до кваліфікаційної роботи у форматі PDF
Програма	
Даниш О.Д. програма.zip	Архів. Містить коди програми і скомпільовану програму
Презентація	
Даниш О.Д. презентація.pptx	Презентація кваліфікаційної роботи