МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: 123 «Комп'ютерна інженерія»

Освітня програма: «Комп'ютерна графіка і Web-дизайн»

Група: 4КГ-07

Дипломний проєкт

здобувачки освіти денної форми навчання

КГ.07.07.000.ДП

МАЙСТРЕНКО ДАР'Ї ОЛЕКСАНДРІВНИ

м. Одеса
2024 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: **123 «Комп'ютерна інженерія»**

Освітньо-професійна програма: **«Комп'ютерна графіка і Web-дизайн»**

Група: **4КГ-07**

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проєкту на тему:

Розробка Front-End частини автоматизованої веб-системи обліку руху контингенту здобувачів відділення комп'ютерних систем

Проєктний матеріал складається з пояснювальної записки на **85** сторінках та графічного (презентаційного) матеріалу на **17** аркушах (слайдах).

Дипломник _____ (Майстренко Д. О.)

Керівник _____ (Жадан А. С.)

Консультанти:

з економічного розділу _____ (Іванченков В. С.)

з розділу охорони праці та техніки безпеки _____ (Чорновол Н. І.)

з нормоконтролю _____ (Петрашова В. І.)

старший консультант _____ (Кривченко Ю. В.)

До захисту допущений

Голова циклової комісії _____ (Кривченко Ю. В.)

Завідувач відділення _____ (Скорнякова О. В.)

Захист «18» 06 2024 р. Протокол ЕК № 2

Оцінка ЕК 5 (відмінно) 98 б.

Секретар ЕК _____

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Відділення *комп'ютерних систем* Комісія *КТ та ПІ*

Спеціальність *123 «Комп'ютерна інженерія»*

Освітня програма *«Комп'ютерна графіка і web-дизайн»*

ЗАТВЕРДЖУЮ:

Заст. дир. з НВР *Беркань І. В.*

“ 16 ” 01 *2024 року*

ЗАВДАННЯ

на дипломний проєкт

Здобувачці освіти *Майстренко Дар'ї Олександрівні*

1. Тема проєкту *Розробка Front-End частини автоматизованої веб-системи обліку руху контингенту здобувачів відділення комп'ютерних систем*

Затверджена наказом по коледжу від “ *02* ” *листопада* *2023 р.*, наказ № *244-А2-ОД*

2 Термін здачі закінченого проєкту

3. Вихідні дані до проєкту

1. Передбачити мінімалістичний графічний дизайн користувача (GUI) веб-системи

2. Застосовувати дружній користувальницький досвід (UX) веб-системи

3. Застосовувати Blade-представлення Laravel

4. Використовувати бібліотеку jQuery

4. Зміст розрахунково-пояснювальної записки (перелік питань, які необхідно розробити)

1. Аналіз предметної області. 2. Технології та засоби розробки (проєктування).

3. Проєктування дизайну веб-інтерфейсу. 4. Розробка Front-End частини MVC-застосунку.

5. Тестування створеного веб-застосунку.

6. Економічний розрахунок. 7. Аспекти охорони праці та техніки безпеки

5. Перелік графічного (презентаційного) матеріалу (з точним зазначенням обов'язкових креслень, кількості слайдів)

Презентація Power Point – 17 слайдів

(Дизайн архітектура веб-системи; User Flow; Базова схема Blade-компоненту; Схема AJAX-запитів; Розробка дизайну веб-системи у Figma; Огляд сторінок веб-системи)

6. Консультанти по проєкту, із зазначенням розділів проєкту, що їх стосується

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Основний розділ	*Жадан А. С.*		
Економічний розділ	*Іванченков В. С.*		
Розділ охорони праці	*Чорновол Н. І.*		
Нормоконтроль	*Петрашова В. І.*		
Старший консультант	*Кривченко Ю. В.*		

7. Дата видачі завдання 15 січня 2024р.

Керівник *Жадан А. С.* (підпис)

Завдання прийняв до виконання *Майстренко Д. О.* (підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/р	Назва етапів дипломного проєкту (роботи)	Термін виконання етапів дипломного проєкту (роботи)	Відмітка про виконання
1	*Формування вступу*	*29.04.24*	виконано
2	*Аналіз предметної області*	*10.05.24*	виконано
3	*Підбір технічної літератури*	*19.05.24*	виконано
4	*Вибір технологій та засобів розробки (проєктування)*	*20.05.24*	виконано
5	*Проєктування дизайну веб-інтерфейсу*	*24.05.24*	виконано
6	*Реалізація Front-End частини MVC-застосунку*	*27.05.24*	виконано
7	*Тестування створеного веб-застосунку*	*29.05.24*	виконано
8	*Оформлення пояснювальної записки*	*31.05.24*	виконано
9	*Оформлення графічної (презентаційної) частини*	*01.06.24*	виконано
10	*Економічний розрахунок*	*02.06.24*	виконано
11	*Опис охорони праці та техніки безпеки*	*09.06.24*	виконано
12	*Аналіз результатів проєктування*	*13.06.24*	виконано
13	*Підготовка доповіді для захисту*	*16.06.24*	виконано

Дипломник (підпис)

Керівник (підпис)

Формат	Зона	Поз.	Позначення	Назва	Кіл.	Примітка
				Документація		
			КГ 07. 07 000. 00 ДП	Дипломний проєкт		
А4			КГ 07. 07 000. 00 ДП ПЗ	Пояснювальна записка	1	

Зм.	Арк.	№ докум.	Підпис	Дата	КГ 07. 07 000. 00 ДП ПЗ		
Розроб.		Майстренко Д.О.		10.06	Розробка Front-End частини автоматизованої веб-системи обліку руху контингенту здобувачів відділення комп'ютерних систем	Літ.	Аркуш 4 / Аркушів 85
Перевірив		Жадан А.С.		10.06		Н Д П	
Н. Контр.		Петрашова В.І.		10.06		ВСП "ОТФК ОНТУ" гр.4КГ-07	
Затверд.		Кривченко Ю.В.		10.06			

ЗМІСТ

ВСТУП

Використання інформаційних веб-систем для автоматизації процесів є все більш затребуваним для підприємств, орієнтованих на підвищення продуктивності та зниження витрат. Інформаційні веб-системи включають в себе різноманітні компоненти, які допомагають у зборі, обробці та аналізі даних та управлінні контентом. Автоматизація за допомогою цих систем дозволяє більш ефективно реагувати на зміни у бізнес-середовищі, а також збільшувати конкурентоспроможність та пристосовуватися до різних потреб.

Основною метою цієї роботи є створення веб-системи для автоматизації учбових процесів та забезпечення зручного і ефективного інтерфейсу для управління контингентом здобувачів відділення. Система повинна надавати засоби для організації навчального процесу, спрощуючи надання інформації, щодо студентів, викладачів та необхідних документів. Важливою частиною є розробка дизайну та інтерфейсу користувача, які максимально відповідають потребам користувачів та забезпечують їм зручність та ефективність в роботі.

Для розробки веб-системи використовуються основні технології веб-розробки. HTML, який визначає структуру сторінок, CSS, що відповідає за їхній вигляд та оформлення, а також JavaScript, який забезпечує інтерактивність та динаміку на сторінках. Для спрощення взаємодії з DOM та виконання асинхронних запитів була використана бібліотека jQuery, яка надає прості засоби для роботи з елементами сторінки та AJAX-запитами, що дозволяє швидко та ефективно додавати динамічні функціональні можливості до веб-інтерфейсу.

Результатом роботи є інформаційна веб-система, яка призначена для завідуючого відділення комп’ютерних систем та іншим довіреним особам ОТФК ОНТУ, які відповідають за облік руху контингенту. Ця система забезпечить автоматизацію роботи з документами та даними студентів, що сприятиме покращенню ефективності та точності обліку, оптимізуючи процеси управління.

1 ОСНОВНИЙ РОЗДІЛ

1.1 Аналіз предметної області

1.1.1 Огляд існуючих рішень

На сьогоднішній день у галузі управління даними та документами в навчальних закладах спостерігаються деякі тенденції. Однією з ключових є зростання популярності автоматизованих систем для обліку руху контингенту здобувачів відділення комп'ютерних систем через їхню спроможність спрощувати та оптимізувати учбові процеси навчальних установ.

Перш за все, відзначається збільшення використання веб-систем для цієї мети, оскільки вони забезпечують більшу доступність та зручність управління даними. Це робить процес ведення обліку ефективнішим та зменшує ймовірність помилок. Розглянемо декілька аналогів [1]:

Аналог 1 – OpenEduCat:

На рис. 1.1 зображено перший приклад OpenEduCat. Цей застосунок єявляє собою відкрите програмне забезпечення для управління навчальними закладами, яке включає в себе різні модулі, такі як облік студентів, розклад занять, бібліотеку тощо. Веб застосунки такого типу спрямовані на навчальні заклади різного рівня, від шкіл до вищих навчальних закладів.

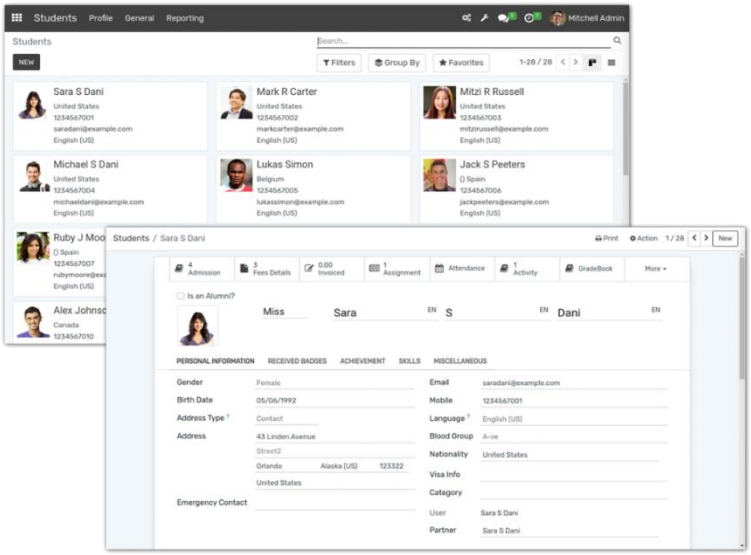


Рисунок 1.1. Інтерфейс застосунку OpenEduCat

Переваги OpenEduCat:

Відкрите програмне забезпечення, що дозволяє змінювати та адаптувати систему під потреби закладу.

Велика спільнота користувачів та розробників, що забезпечує підтримку та постійний розвиток системи.

Недоліки OpenEduCat:

Можливі проблеми зі сумісністю модулів при оновленнях.

Потребує достатньої технічної експертизи для налаштування та підтримки.

Аналог 2 – Fedena:

Fedena є ще одним Достатньо популярним застосунком для управління навчальними закладами. На рис. 1.2 можна розглянути складову інтерфейсу цього веб застосунку. Вона включає в себе модулі для обліку студентів, вчителів, фінансів, бібліотеки, розкладу тощо. Fedena також надає можливість використовувати різні додаткові модулі та розширення.

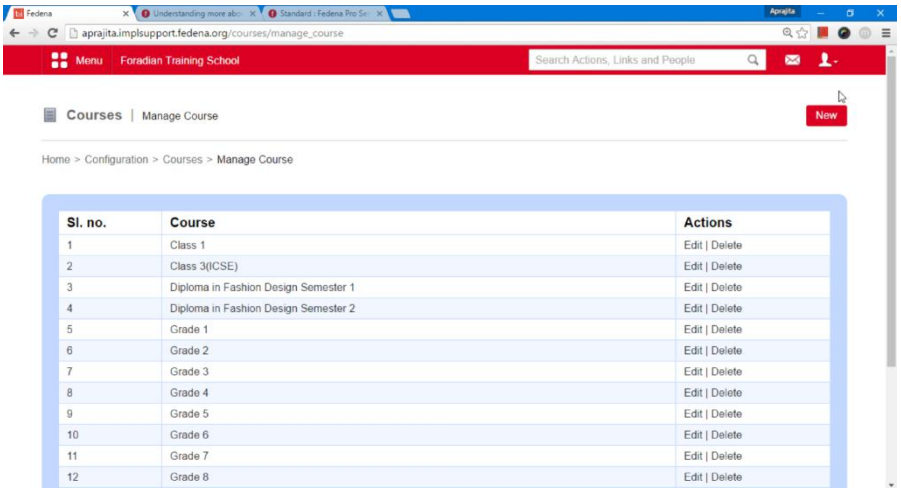


Рисунок 1.2. Інтерфейс застосунку Fedena

Переваги Fedena:

Простий у використанні та налаштуванні.

Має ряд готових модулів та розширень для різних потреб навчальних закладів.

Недоліки Fedena:

Може бути обмежений у функціоналі порівняно з іншими рішеннями.

Можливість обслуговування та підтримки може бути обмеженою залежно від постачальника.

Аналог 3 – SIS (Student Information System):

На рис. 1.3 зображено веб застосунок SIS. Він є одним з найпоширеніших рішень у сфері управління даними про студентів. Це комплексна система, яка включає в себе модулі для реєстрації студентів, обліку присутності, виставлення оцінок, статистики успішності тощо. Вона дозволяє автоматизувати багато аспектів управління студентським контингентом.

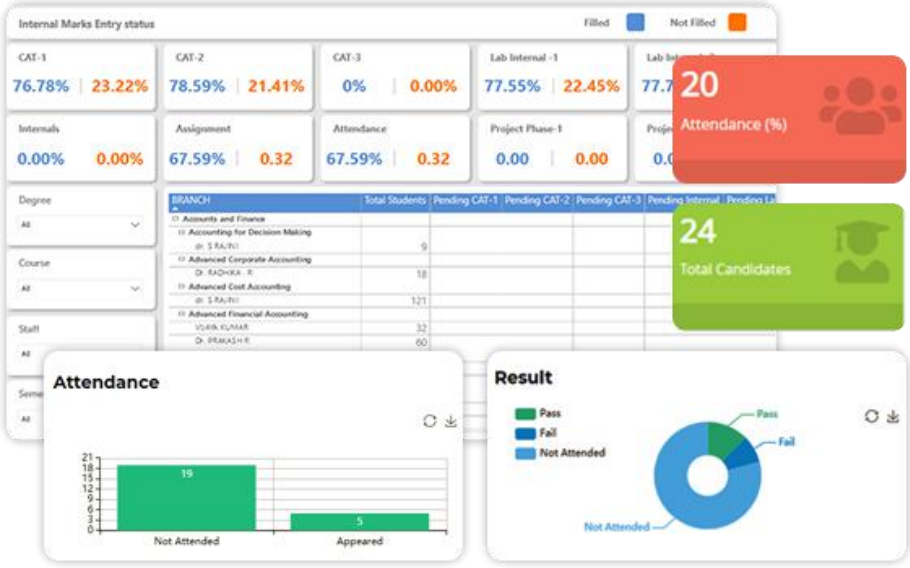


Рисунок 1.3. Приклад функціоналу застосунку SIS

Переваги SIS:

Широкий функціонал, що охоплює багато аспектів управління студентами.

Досвід використання та надійність, оскільки SIS використовується в багатьох навчальних закладах.

Недоліки SIS:

Високі вартості впровадження та підтримки.

Може бути складно налаштовуватися під конкретні потреби закладу.

Для звітування всіх складових обраних веб застосунків та виявлення в них недоліків, запишемо їх функціонал у таблицю 1.1 порівняння характеристик аналогів до веб-системи обліку руху контингенту учбових закладів.

Таблиця 1.1. Порівняльна характеристика аналогів

Характеристика/ Рішення	**SIS**	**OpenEduCat**	**Fedena**
Відкрите програмне забезпечення	Ні	Так	Ні
Функціонал	Широкий, охоплює багато аспектів управління студентами	Різноманітні модулі, включаючи облік студентів, розклад занять тощо	Має модулі для обліку студентів, вчителів, фінансів, бібліотеки тощо
Спільнота користувачів та розробників	Велика	Велика	Можливо не така велика
Вартість	Висока	Залежить від конфігурації та підтримки	Залежить від постачальника
Складність налаштування	Може бути складно	Потребує достатньої технічної експертизи	Простий у використанні та налаштуванні
Підтримка	Доступна	Залежить від спільноти користувачів та розробників	Може бути обмеженою залежно від постачальника

Продовження таблиці 1.1

Характеристика/ Рішення	**SIS**	**OpenEduCat**	**Fedena**
Гнучкість	Обмежена	Велика	Можливо обмежена
Надійність	Висока	Залежить від встановлення та налаштування	Може бути обмеженою залежно від постачальника

1.1.2 Технології та засоби розробки

Під час розробки Front-End частини дипломної роботи були обрані наступні технології: HTML, JavaScript, Blade та jQuery. Вибір цих технологій був обгрунтований їхньою популярністю, потужними можливостями та широким спектром функцій, які вони надають.

HTML був обраний як основна мова розмітки для створення структури веб-сторінок. Він є стандартом у веб-розробці та дозволяє чітко визначати елементи сторінки [2].

JavaScript являє собою мову програмування, яка використовується для створення JavaScript інтерактивних елементів на веб-сторінках. Вона є однією з найпоширеніших мов програмування, що використовується в розробці веб-застосунків. JavaScript використовується для створення різноманітних функцій, таких як взаємодія з користувачем, анімація, валідація даних та динамічна зміна вмісту веб-сторінок без перезавантаження. Ця мова є невід'ємною частиною Front-End розробки та забезпечує веб-застосункам більшу інтерактивність та функціональність. JavaScript використовується як для розробки клієнтської частини застосунків, так і для серверної, завдяки платформам, таким як Node.js. JavaScript був використаний для додавання інтерактивності до веб-сторінок.

Його використання дозволяє реалізувати різноманітні функції, такі як взаємодія з користувачем та динамічне оновлення вмісту сторінок [3].

Також для шаблонів буде використовуватися Blade, який є шаблонним движком, який використовується в Laravel, одному з найпопулярніших фреймворків для розробки веб-застосунків на PHP. Це простий, але потужний механізм створення шаблонів. На відміну від деяких інших механізмів створення шаблонів PHP, Blade не обмежує використання простого коду PHP у шаблонах. Всі шаблони Blade скомпільовані у звичайний PHP-код і зберігаються в кеш-пам'яті, доки їх не буде змінено, що означає, що Blade практично не додає накладних витрат для вашої програми. Файли шаблонів Blade використовують розширення файлу .blade.php і зазвичай зберігаються в каталозі resources/views. Використання Blade спрощує створення та управління представленнями в застосунку [4].

Для бібліотек буде використовуватися jQuery, яка являється швидкою, невеликою та багатофункціональною бібліотекою JavaScript. Її використання значно спрощує низку завдань, таких як обхід HTML-документів, маніпуляції з ними, обробка подій, анімація та робота з AJAX. Завдяки своєму простому у використанні API, jQuery дозволяє ефективно взаємодіяти з елементами веб-сторінки та виконувати різноманітні операції на стороні клієнта. Вона також спрощує роботу з DOM (Document Object Model) та забезпечує сумісність з різними браузерами, що робить її незамінною у розробці веб-застосунків [5].

CSS являє собою мову стилів, що дозволяє описувати зовнішній вигляд веб-сторінок. Він розширює можливості HTML шляхом додавання багатьох корисних функцій, які роблять написання і обслуговування коду більш простим і зручним. Є декілька характеристик CSS:

1. CSS повністю сумісний з усіма версіями HTML.
2. Існує нескінченна кількість фреймворків, створених за допомогою CSS. Деякі з них – Bootstrap, Bourbon і Susy.
3. CSS має більше функцій і можливостей для стилізації веб-сторінок.

4. Знову і знову індустрія обирає CSS як основну мову для створення стилів веб-сторінок.
5. CSS активно підтримується та розвивається консорціумом кількох технологічних компаній.
6. Є можливість використовувати міксіни - це шаблони стилів, які можна використовувати повторно. Вони дозволяють створювати однаковий код для стилів, які використовуються у різних місцях вашого проєкту.
7. CSS дозволяє вам імпортувати інші CSS файли прямо у ваш код, що полегшує використання існуючих бібліотек стилів.

Основна відмінність між CSS та іншими мовами стилів полягає у синтаксисі, а саме те, що CSS використовує плоску модель з явними командами та властивостями [6].

Ці технології були обрані через їхню широку популярність, якість та гнучкість, що сприятиме розробці потужного та ефективного фронтенду для дипломної роботи, що можна побачити на рис. 1.4 з рейтингом мов програмування-2024, взятого з сайту dev.ua.

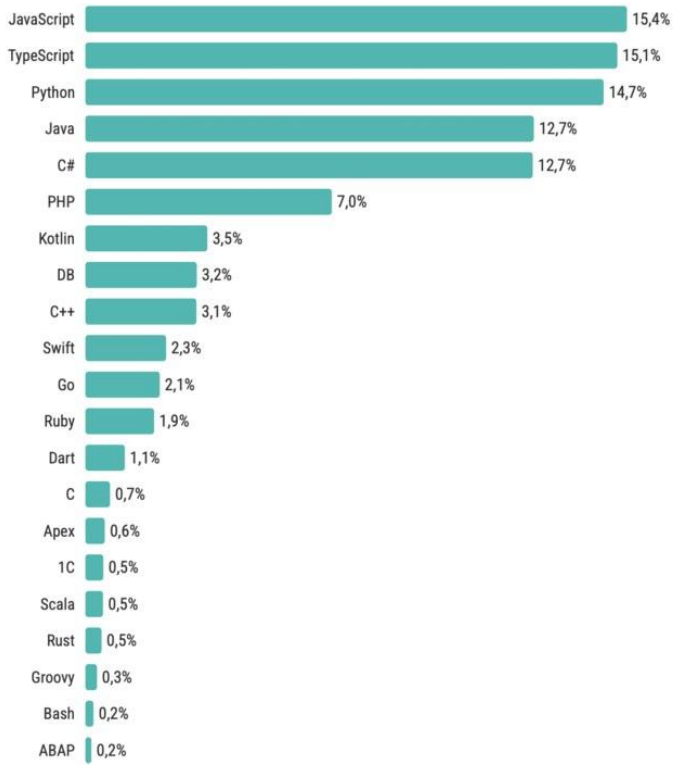


Рисунок 1.4. Рейтинг мов програмування 2024

1.2 Проєктування веб-системи

1.2.1 Технічне завдання на розробку

Розуміння вимог і потреб користувачів є першим і важливим кроком у розробці дизайну, тому після вибору інструменту, треба створити технічне завдання. Тому на початку розробки кожного проєкту, слід зробити правильну постановку завдання: визначити вхідні дані; описати, що планується отримати в результаті виконання проєкту; шляхом дослідження зробити правильний вибір технологій. Дана процедура має важливе значення для успіху застосунку у майбутньому.

Технічне завдання до розробки дизайну веб-системи.

При розробці дизайну автоматизованої веб-системи обліку руху контингенту здобувачів відділення комп'ютерних систем важливо встановити кілька ключових цілей, які забезпечать успішність проєкту. Веб-система повинна мати інтуїтивний та простий інтерфейс користувача, що полегшить навігацію для адміністрації. Це означає, що всі елементи інтерфейсу повинні бути зрозумілими на інтуїтивному рівні і не вимагати додаткових зусиль для освоєння.

При розробці дизайну важливо враховувати UI - користувацький інтерфейс, що має під собою зовнішній вигляд, стиль, кольори, шрифти та інше, та UX - користувацький досвід, тобто наскільки зручно та інтуїтивно зрозуміло користувачеві знаходитися на розробленому проєкті. Сучасний, естетично привабливий дизайн з анімаціями та безліччю різних стилів може мати гарний результат для деякий випадків, але підходить далеко не для кожного проєкту. Такі функціонально-технічні веб-системи мають свої обмеження. У даному випадку я звертаю свою увагу у сторону більш стриманого стилю і дизайну. Використання простих форм і лаконічних деталей сприяє зосередженню користувачів на важливій інформації. Обмежена неяскрава кольорова гама буде найбільш доречною для мого завдання. Кольори мають великий вплив на сприймання інформації, тому приділення переваги природним, нейтральним

відтінкам дозволяє запобігти відволіканню користувача та забезпечує зручне сприйняття інтерфейсу. Використання легко читабельних шрифтів допомагає покращити зрозумілість інформації та знижує втомлюваність при довготривалому використанні системи.

Щодо UX також важливо створити логічну та інтуїтивно зрозумілу структуру інтерфейсу, щоб користувачі могли швидко знаходити необхідну інформацію та функції. Кожен компонент повинен виглядати та функціонувати так, щоб користувачі не стикалися з непередбачуваними змінами або непослідовностями.

Функціональність дизайну повинна підтримувати всі основні можливості системи, включаючи введення, редагування, видалення та перегляд даних про учнів або співробітників. Зручні інструменти для візуалізації даних, такі як таблиці, дозволять адміністраторам легко аналізувати та знаходити необхідну інформацію.

Взаємодія з користувачем повинна бути побудована таким чином, щоб забезпечити зворотний зв'язок при виконанні будь-яких дій. Це включає підтвердження успішного збереження даних або повідомлення про помилки, що допоможе користувачам розуміти, що відбувається на кожному етапі взаємодії з системою.

Швидкість та продуктивність також є критично важливими факторами. Дизайн повинен бути оптимізований для швидкого завантаження і максимальної продуктивності, щоб користувачі не відчували затримок під час роботи з системою.

Також важливо забезпечити безпеку та конфіденційність даних. Дизайн повинен включати елементи, що забезпечують безпечний доступ до даних, захист особистої інформації та конфіденційність користувачів, таких як авторизація адміністрації. Це допоможе запобігти несанкціонованому доступу та забезпечить збереження особистої інформації користувачів.

Технічне завдання до розробки Front-End частини веб-системи.

Необхідно розробити базовий шаблон для аутентифікації. Для його створення потрібно розробити сторінку входу користувача, де будуть поля для введення логіну та пароля. Це необхідно для забезпечення безпеки доступу до системи та ідентифікації користувачів. Також треба вивести повідомлення про успішний або неуспішний вхід.

Для адміністративної панелі треба створити загальний макет з боковим меню, який буде містити основні компоненти для керування даними. Це важливо для зручності навігації та організації робочого простору адміністратора, коли з будь-якої сторінки можна буде перейти на іншу. Також підготувати структуру для відображення основних компонентів: панелі керування, таблиць з даними, форм для редагування і створення записів та забезпечити можливість відображення інформаційних повідомлень та помилок.

Необхідні основні базові компоненти для роботи з даними, такі як таблиці для відображення списків користувачів, груп, студентів та інших сутностей. Для цього необхідно розробити форми для додавання, редагування та видалення записів, а також забезпечити відображення повідомлень про помилки при валідації.

Створення шаблону для пошуку та фільтрації даних важливо для забезпечення зручного доступу до необхідної інформації. Для цього необхідно розробити компонент для введення критеріїв пошуку, таких як ім'я, прізвище, група та інше, і підготувати шаблони для відображення результатів пошуку з можливістю їх вибору.

Створення шаблону для відображення детальної інформації про користувача, групу, наказ, батьків та інші дані, є необхідним для забезпечення доступу до розширеної інформації про об'єкти веб-системи. Для цього необхідно розробити сторінки, які відображають особисті дані та зв'язки з іншими сутностями, і забезпечити можливість відображення даних у зручному форматі.

Необхідно реалізувати AJAX-запити по роутам для виконання пошуку за певними полями у формі та відображення результатів з можливістю їх вибору.

AJAX-запити є асинхронними HTTP-запитами, які використовуються для взаємодії з сервером без перезавантаження сторінки [7].

1.2.2 Проєктування дизайну веб-системи

Перед початком розробки Front-End частини необхідно створити прототипи майбутнього застосунку, у моєму випадку веб-системи. Це дозволить завчасно переглянути можливий результат, усунути можливі недоліки та покращити функціональність. Для проєктування прототипів я використовувала найпопулярніший серед веб дизайнерів онлайн-сервіс для розробки інтерфейсів та прототипування Figma, рис. 1.5. У сервісі передбачено безкоштовний тарифний план для одного користувача, який містить достатню кількість інструментів для розробки макетів.

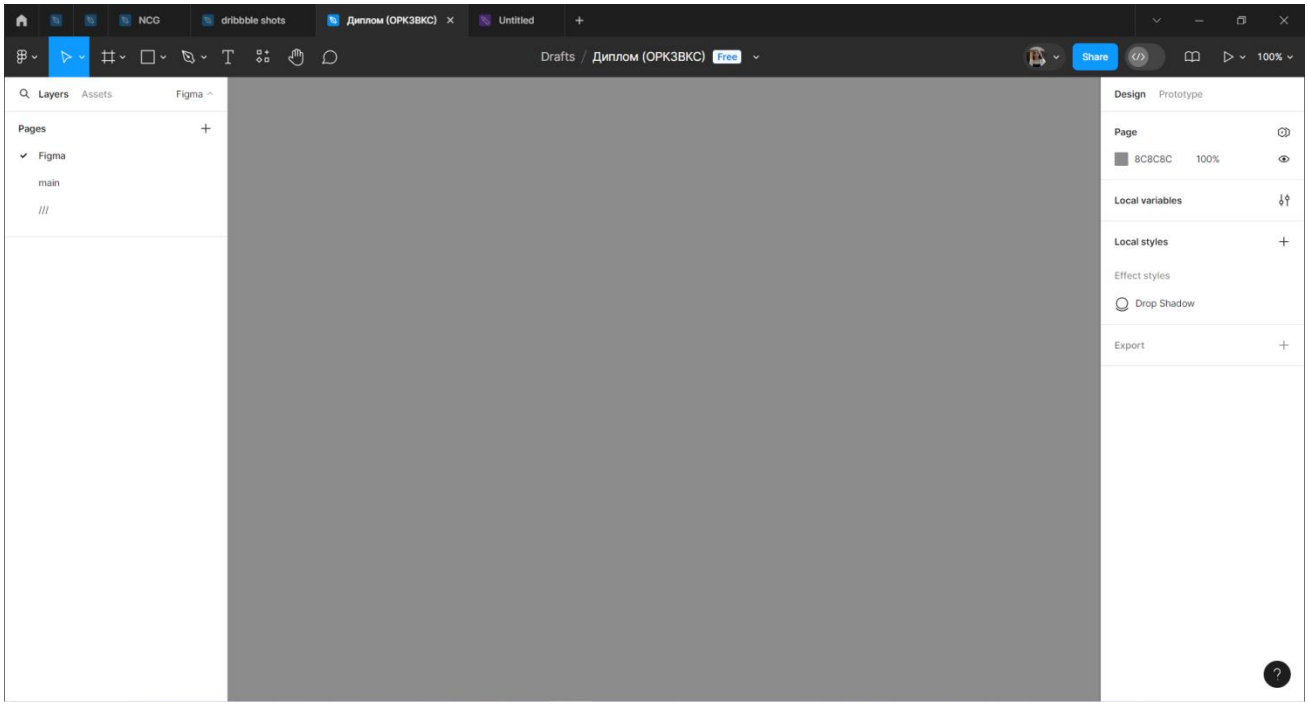


Рисунок 1.5. Інтерфейс онлайн-сервісу Figma

Figma стала незамінним інструментом для багатьох фахівців у сфері веб дизайну, а також для команд, які займаються розробкою веб-інтерфейсів та мобільних застосунків. Ця платформа отримала широку популярність завдяки своїм унікальним можливостям, ось деякі з них:

1. Figma дозволяє користувачам спільно працювати над проєктами в реальному часі, що особливо корисно для команд, що працюють в різних географічних регіонах або віддалено. Будь-які зміни вносяться миттєво, що сприяє ефективній спільній роботі.
2. Figma не вимагає встановлення спеціального програмного забезпечення і працює у веб-браузері, що робить його доступним для користувачів з будь-якого пристрою та операційної системи.
3. Figma надає можливість створювати макети і прототипи в одному інструменті, спрощуючи процес розробки та тестування користувацького досвіду, що є важливим етапом у веб дизайні.
4. Платформа підтримує роботу з растровою графікою, що робить її більш універсальною і корисною для дизайнерів, які працюють зі зображеннями різного типу.
5. Figma дозволяє дизайнерам створювати бібліотеки та компоненти, які можна повторно використовувати в різних проєктах. Це підвищує ефективність роботи та сприяє створенню єдиної стилістики для всіх проєктів компанії або бренда.

Таким чином Figma стала очевидним вибором для створення макетів дизайну для веб-системи.

Далі, коли в нас виявлені основні потреби користувачів та обрано інструмент, можемо перейти до архітектури веб-застосунку. Архітектура веб-застосунку в дизайні являє собою структурну організацію веб-системи, яка визначає його складові елементи, взаємозв'язки між ними та принципи їх взаємодії. Архітектура проєкту, що розробляється, повинна бути, оскільки вона забезпечує зрозумілість для подальшою розробки макетів. Це особливо необхідно при розробці багатосторінкових сайтів, так як вони зазвичай містять в собі багато різних елементів та взаємозв’язків. Через їх кількість є ризик щось забути або переплутати, та архітектура якраз-таки запобігає цьому. На рис. 1.6 зображено архітектуру веб-системи обліку руху контингенту відділення комп’ютерних систем.

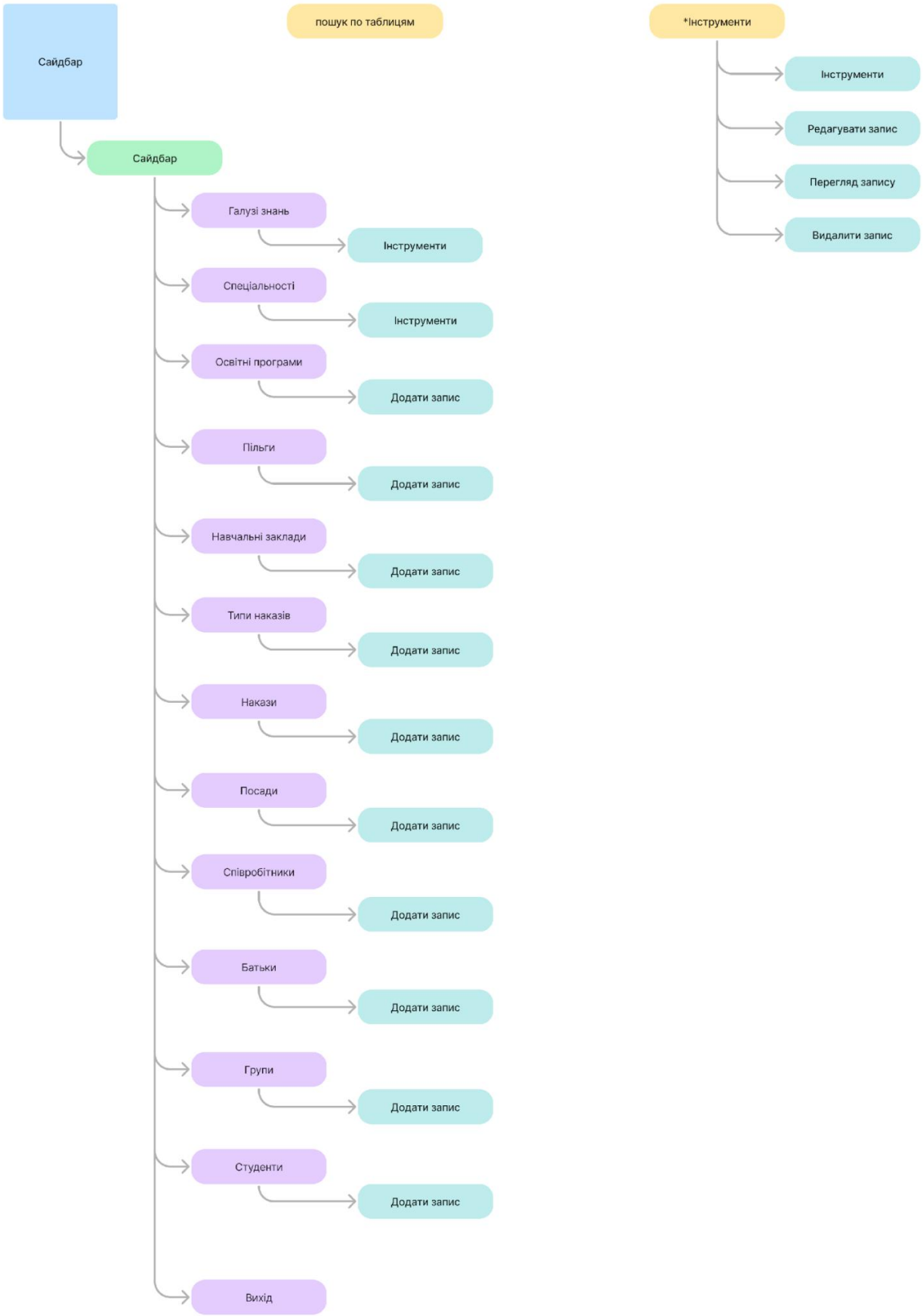


Рисунок 1.6. Архітектура веб-системи обліку руху контингенту

Після створення архітектури можна розробити User Flow, тобто шлях користувача, щоб зрозуміти та впевнитися в тому, що архітектура побудована коректно та користувачам буде зручно виконувати дії. Для розробки шляху, я використаю вбудований інструмент в Figma - FigJam, який створений для розробки схем різного типу. Для прикладу, на рис. 1.7 я розробила шлях користувача від авторизації до успішного додавання даних про нового студента.

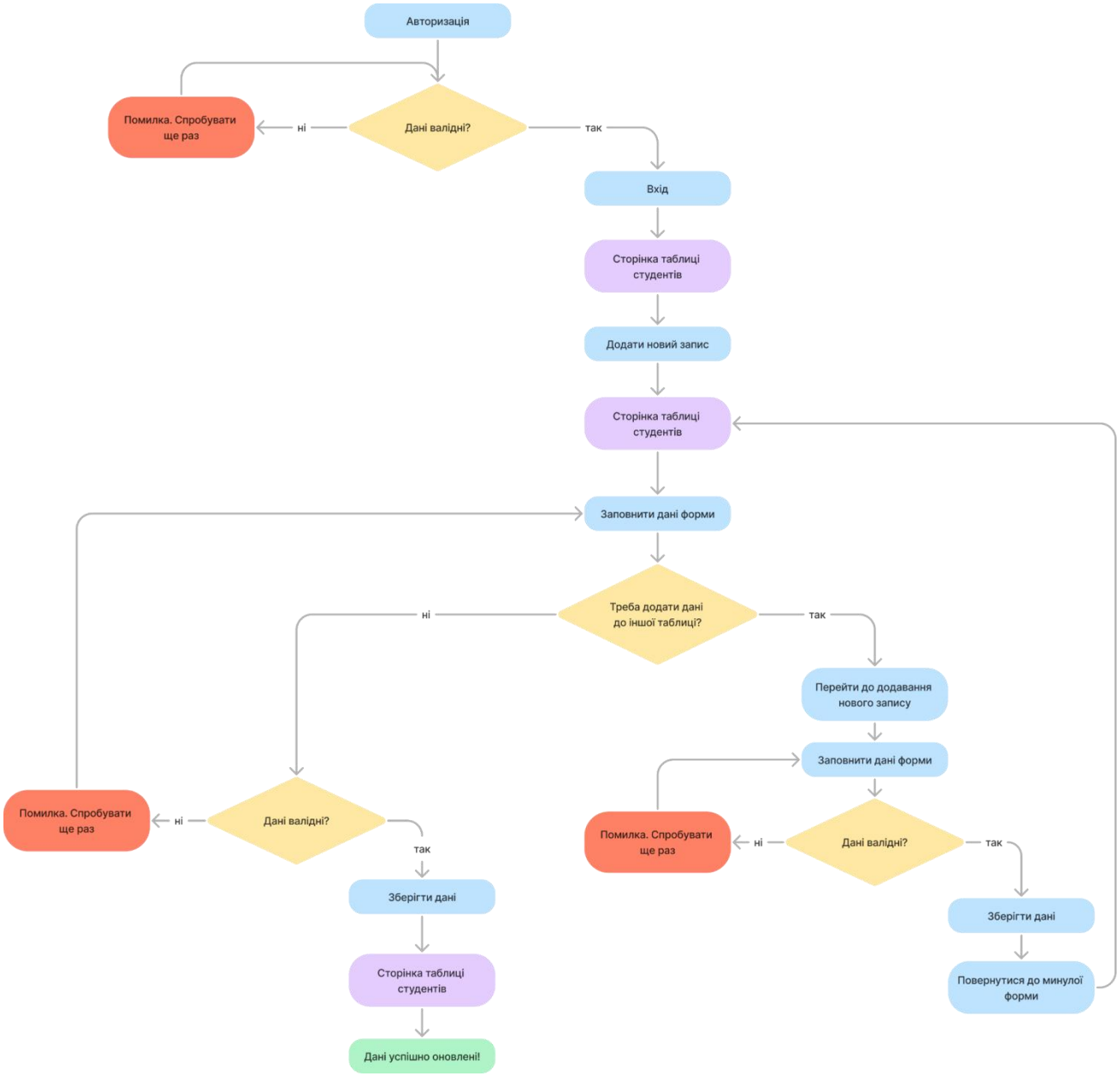


Рисунок 1.7. Шлях користувача від авторизації до успішного додавання даних про нового студента.

Після цих дій, створення архітектури та шляху користувача, можна розпочати розробку дизайн макетів для веб-застосунку. На рис. 1.8 можна побічити розроблені макети для веб-системи у середовищі Figma.

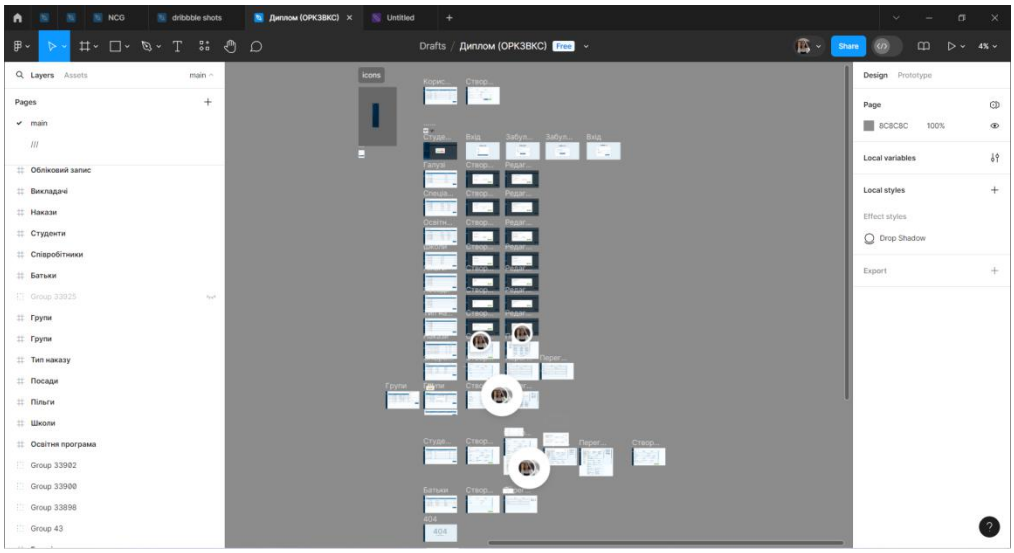


Рисунок 1.8. Розроблені макети для веб-системи у середовищі Figma

Далі розглянемо основні з них трохи ближче.

Створення сторінки авторизації.

Авторизація є невід'ємною частиною будь-якої системи, сервісу чи застосунку. На рис. 1.9 зображено сторінку для авторизації. Вона містить в собі назву навчального закладу та базову форму для авторизації, яка складається з введення своєї електронної пошти та паролю та кнопки входу. Форма також містить можливість відновити пароль у разі того, якщо користувач його забув.

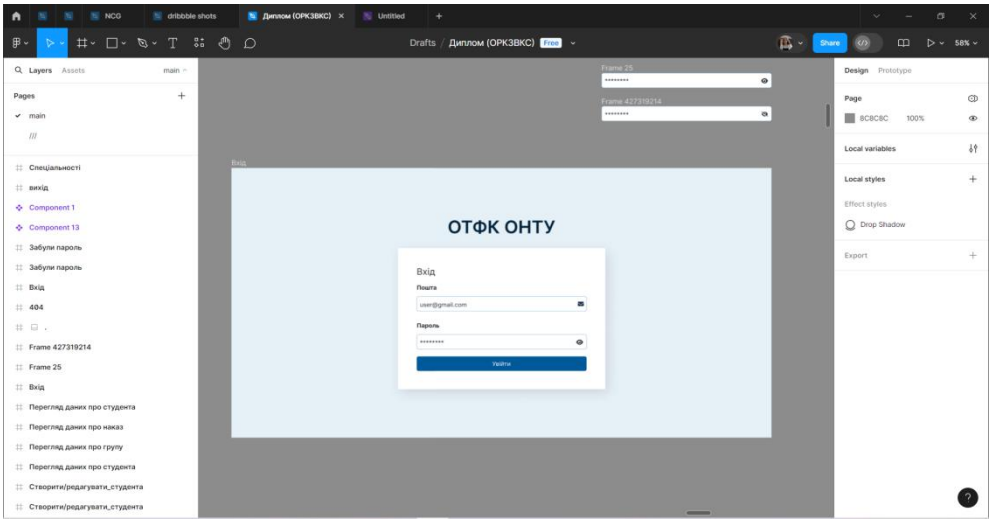


Рисунок 1.9. Сторінка авторизації веб-системи

Наступне, що ми розглянемо буде інтерфейс веб-системи, що розробляється. Розроблені інтерфейси можна розділити на чотири категорії: сторінка таблиці, додавання та редагування нового запису через поп-ап, додавання та редагування нового запису через сторінку, перегляд інформації на окремій сторінці. Перше, що ми розглянемо, буде сторінка таблиці студентів, яка зображена на рис. 1.10.

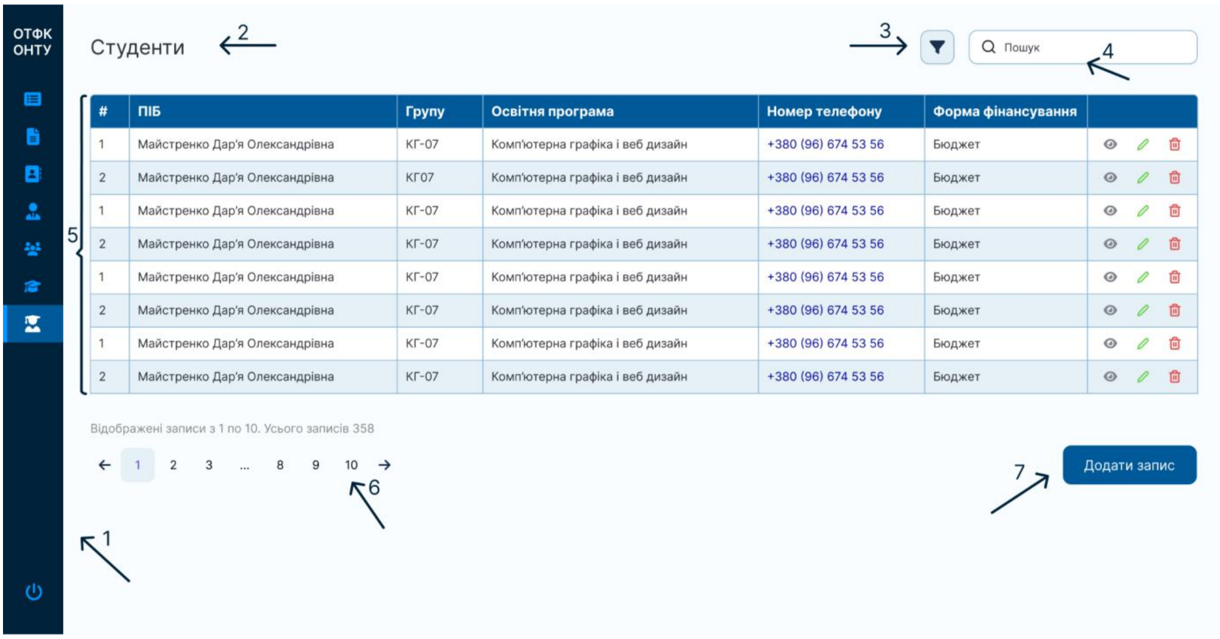


Рисунок 1.10. Основні елементи інтерфейсу системи на прикладі сторінки таблиці студентів

Ми можемо бачити, що в цілому інтерфейс складається з двох частин: бокового меню під номером 1 та основну робочу область, в якій знаходяться інші елементи. Бокове меню містить в собі текстову абревіатуру коледжу, при натиску на яку, відкривається його розширена версія, в якій можна бачити назви всіх табів. Нижче знаходиться список з табів - сторінок веб-системи, які мають фіксоване поле, в якому вони проскролюються. І останнім, найнижчим елементом меню, є кнопка виходу з веб-системи. Під номером 2 на сторінці таблиць завжди буде назва таблиця, так само як фільтрація під номером 3 і пошук по даним таблиці, який знаходиться під номером 4. Під номером 4 знаходиться безпосередньо сама таблиця студентів, яка складається з таких полів, як ID, ПІБ, Група, Номер телефону, Пошта, Форма фінансування та дії, які

можна виконувати з записами. Дії можуть змінюватися в залежності від кожної таблиці. Під таблицею під номером 5 знаходиться дескріпшн, тобто опис до таблиці, а саме, скільки відображено записів у таблиці та скільки їх всього. Це є досить корисним елементом для роботи з таблицями, особливо, коли вона вже буде повністю заповнена адміністратором і в ній буде велика кількість записів. Також на рис. 1.11 зображено пагінацію, яка з'являється, коли записи у таблиці перевищують кількість 10.

Рисунок 1.11. Пагінація до таблиці

Останнім елементом 6 є кнопка додати новий запис, яка при натиску або виведе поп-ап або переведе нас на сторінку додавання запису. Розглянемо перший варіант - додавання нового запису через поп-ап на рис. 1.12 додавання пільги через поп-ап.

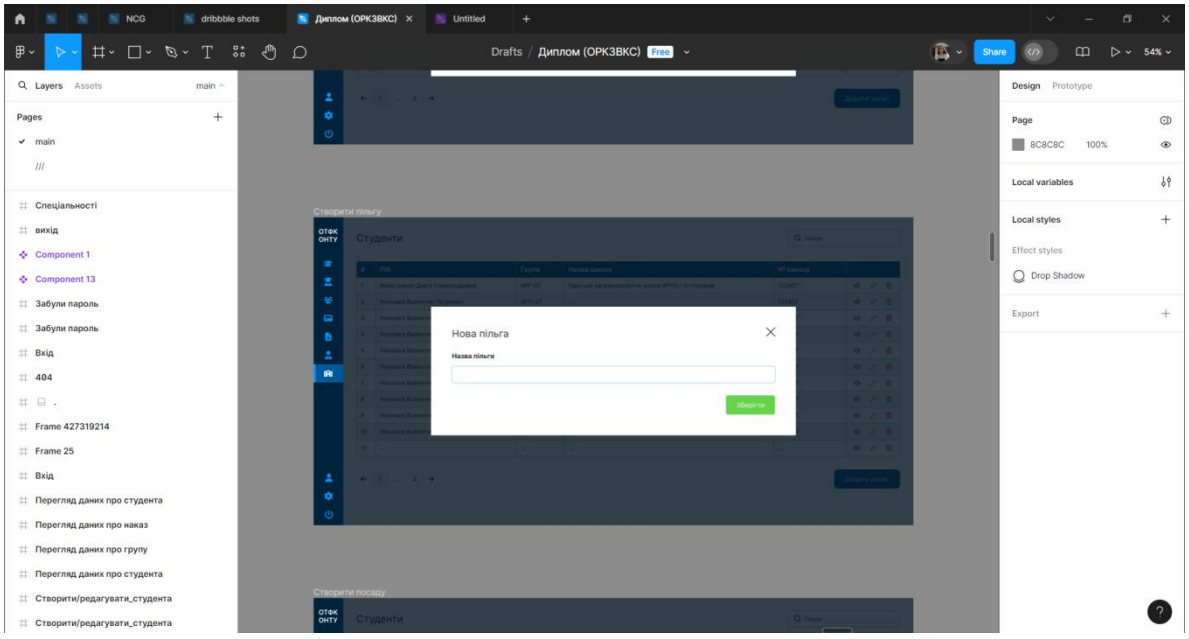


Рисунок 1.12. Додавання пільги через поп-ап

Додавання, а також редагування записів на рис. 1.13 через поп-ап використовується для створення нового запису про галузі знань, спеціальності, освітні програми, школи, пільги, посади та типи наказів. Це зумовлено тим, що

для додавання цих записів потрібна мала кількість інформації, тому більш доцільно використовувати саме поп-апи.

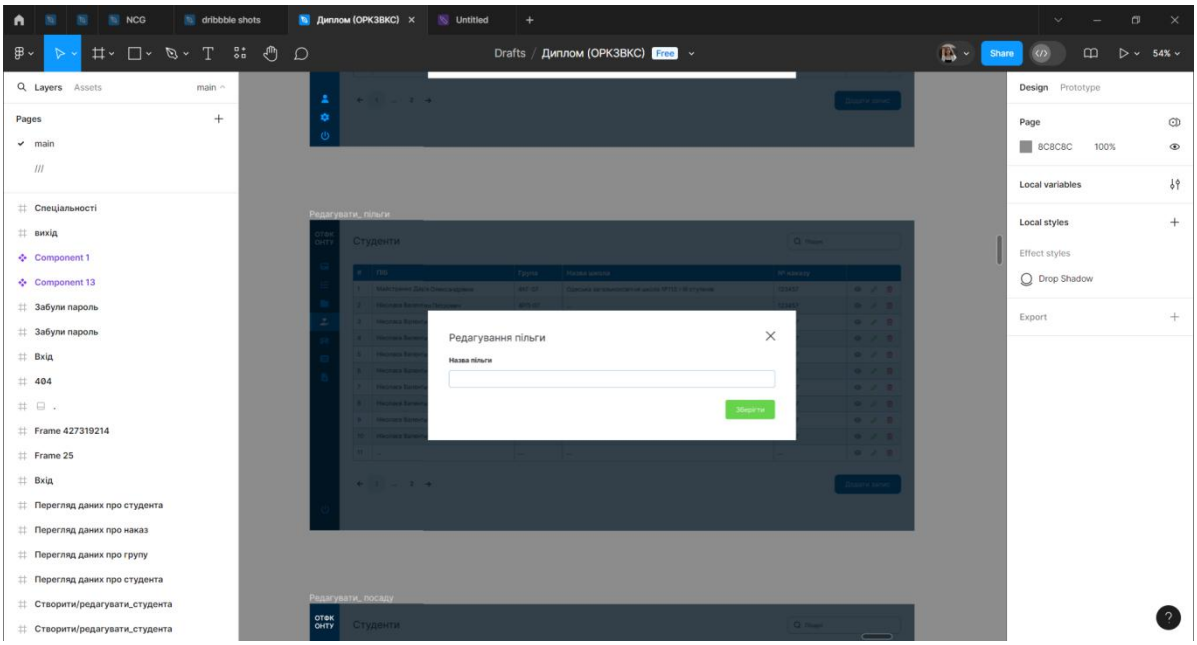


Рисунок 1.13. Редагування пільги через поп-ап

Поп-апи це досить зручно, але є таблиці, записи до яких потребують більшої кількості даних. У цих випадках ми створюємо окрему сторінку додавання або редагування запису. Розглянемо це на прикладі рис. 1.14 додавання нового запису про студента. Дана форма містить в собі поля для вводу ПІБ студента, електронної пошти, номеру телефону, фото студента, дані про вступ, паспорт та інші особисті дані. У таких полях як група, батьки або опікун та пільги, інформація виводиться списком з таблиць цих даних, які були додані раніше. Але важливо було передбачити можливість додавання нових записів відразу при додаванні студента, так як це економить час та мінімізує зусилля.

Так як форма є доволі довгою, для кращого сприймання, при розробці дизайну був доданий фіксований блок, у якому і знаходиться вся форма. Завдяки цьому при звичайному вигляді, як на рис. 1.15, форма є зручною та зрозумілою.

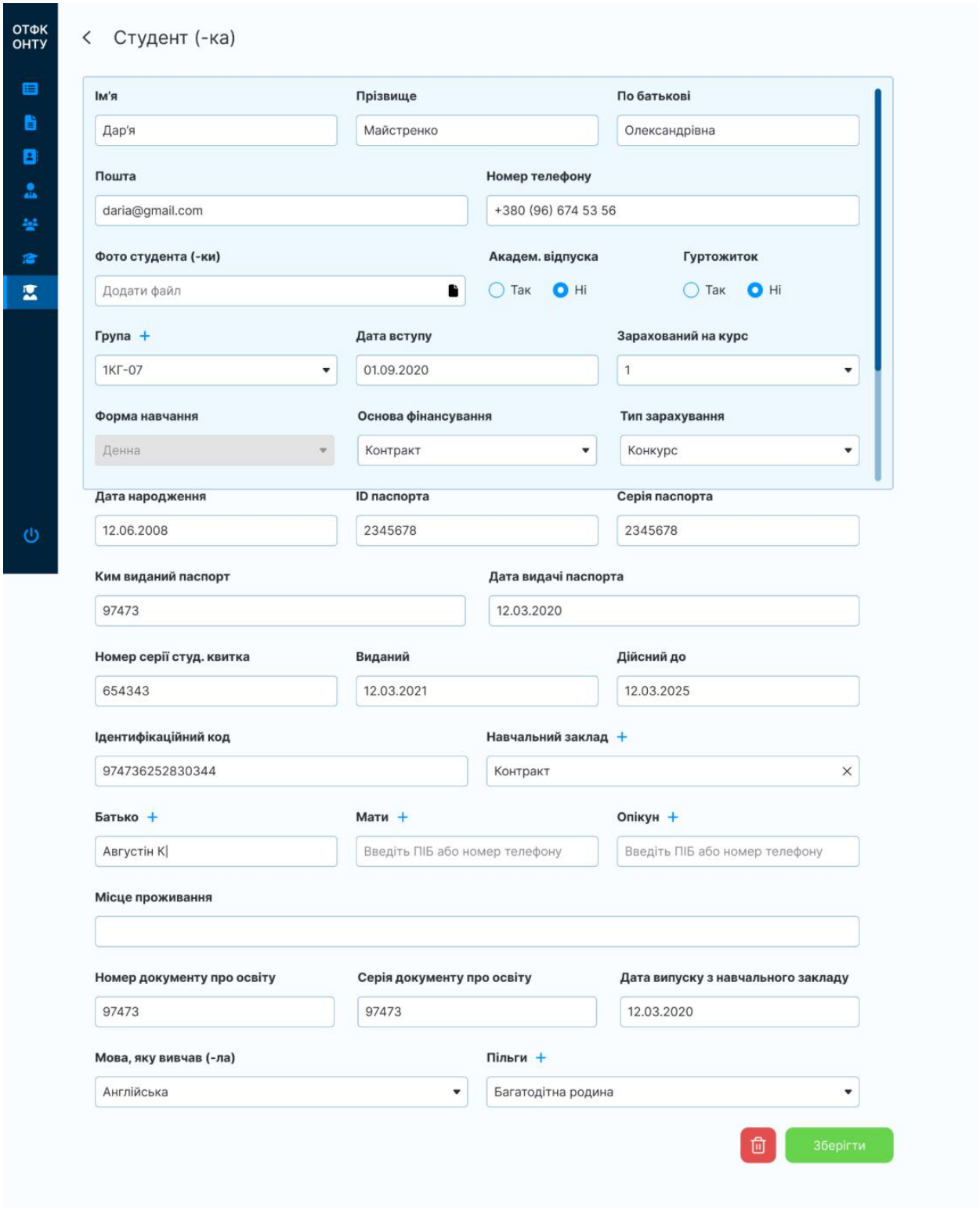


Рисунок 1.14. Повна форма додавання нового запису про студента

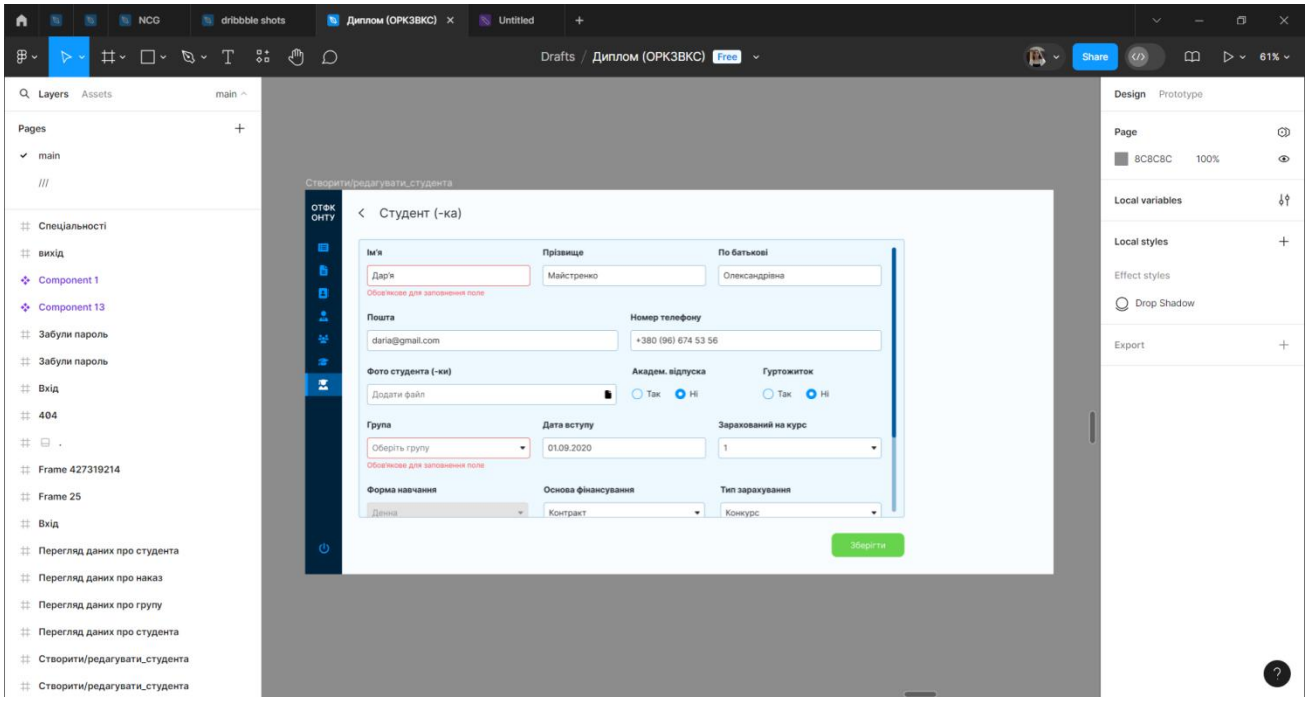


Рисунок 1.15. Звичайна форма додавання нового запису про студента

Редагування студента відбувається на такій самій сторінці, але з вже заповненими до цього даними. На рис. 1.16 зображено сторінку редагування студента.

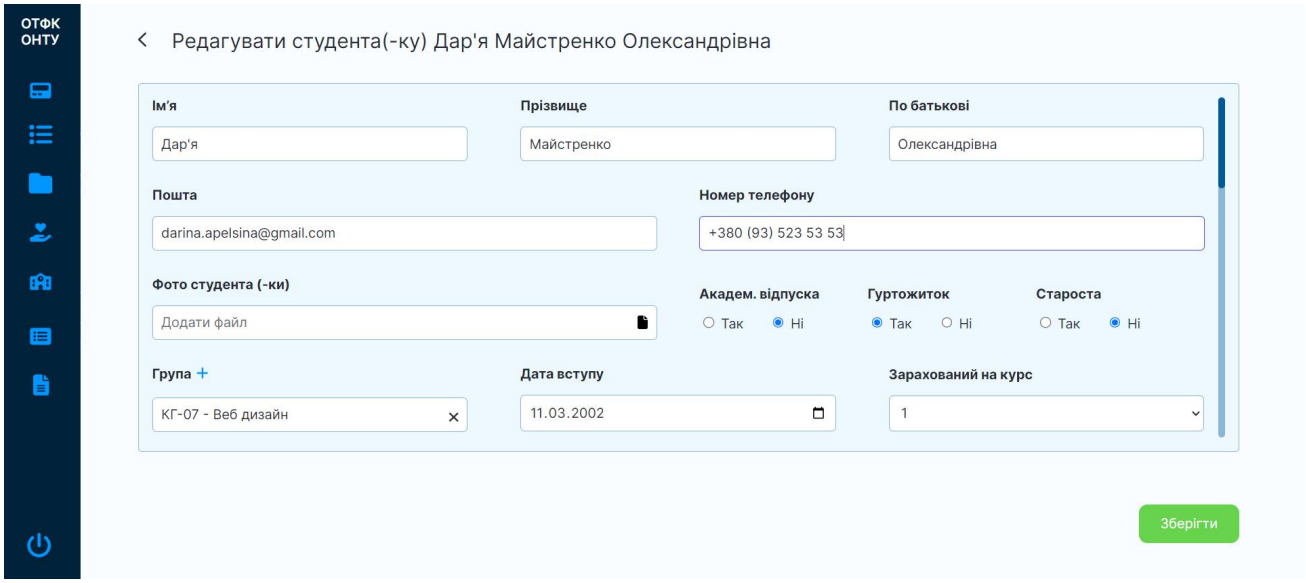


Рисунок 1.16. Сторінка редагування даних про студента

При заповненні кожного поля у формі необхідно валідувати дані, які вводяться. Є вірогідність вводу неправильних або недопустимих значень даних,

для таких випадків необхідно передбачити вигляд та виведення до них помилок, що і зображено на рис. 1.17.

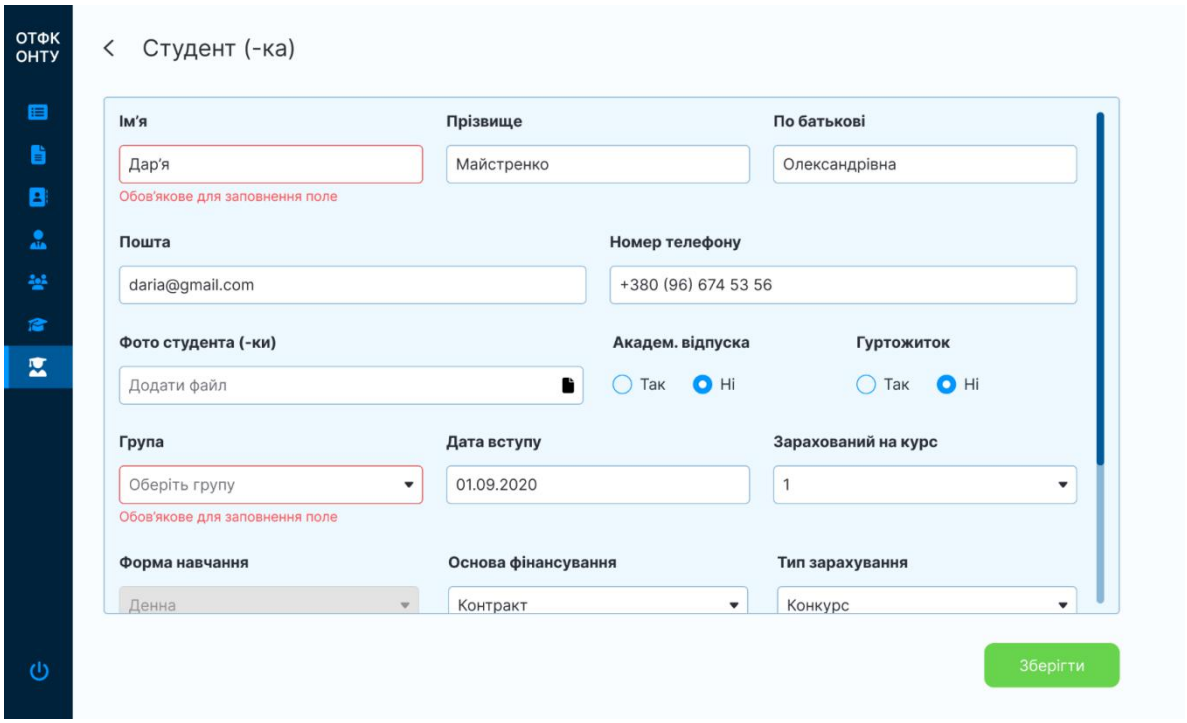


Рисунок 1.17. Виведення помилок при валідації даних

Після редагування даних та збереження, у таблицю виводиться допис у вигляді тегу про успіх редагування, що і зображено на рис. 1.18. Такі самі теги є ще для помилок та попереджень.

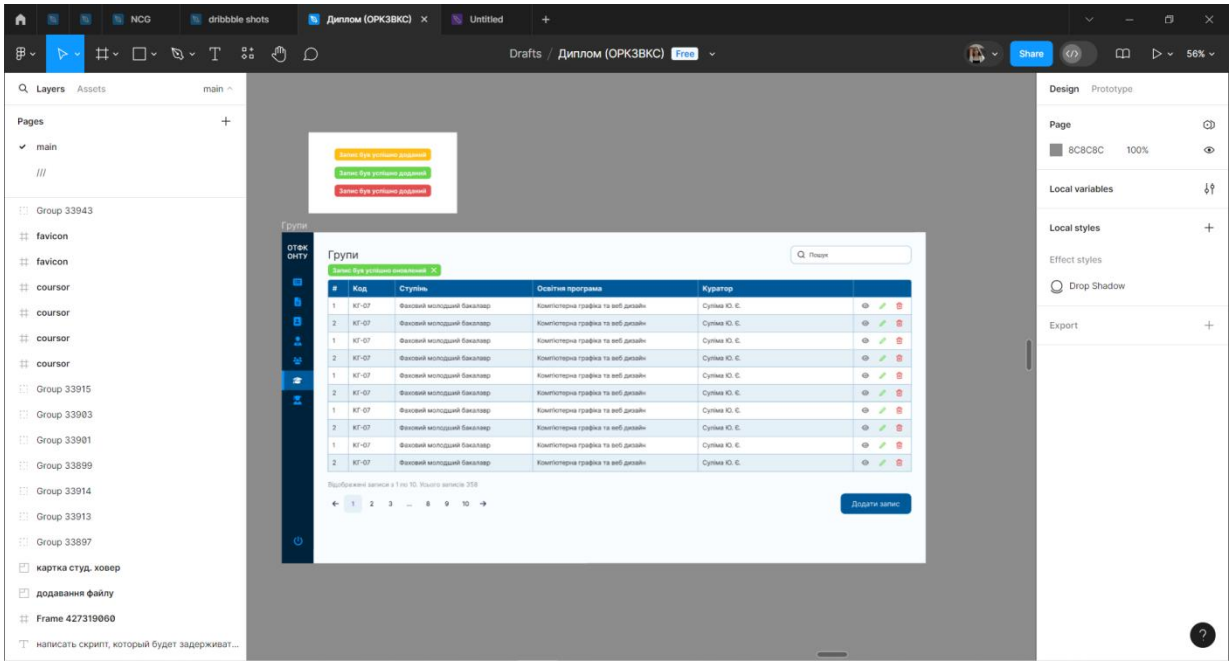


Рисунок 1.18. Тег про успіх редагування даних

Перегляд інформації відбувається на окремій сторінці на рис. 1.19, в якій зібрано усі введені дані про студента, а саме його особисті дані, дані про групу та накази, які коли-небудь відносилися до нього або до його групи. Також передбачено посилання на документи, а саме Особову картку та Індивідуальний план, які будуть генеруватися з даних студента з різних таблиць.

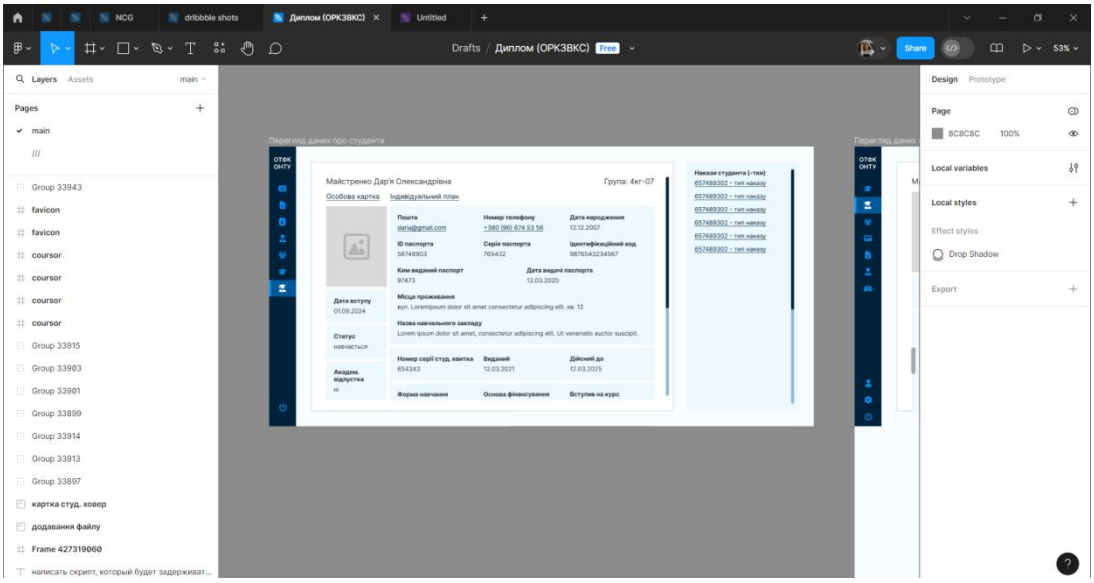


Рисунок 1.19. Перегляд інформації про студента

Для груп перегляд інформації виглядає трохи іншим чином, ніж у перегляду інформації для студента, що показано на рис. 1.20. На сторінці відображено інформацію про групу, накази, які відносилися до певної групи та список студентів з куратором, які є посиланнями на перегляд кожної особи.

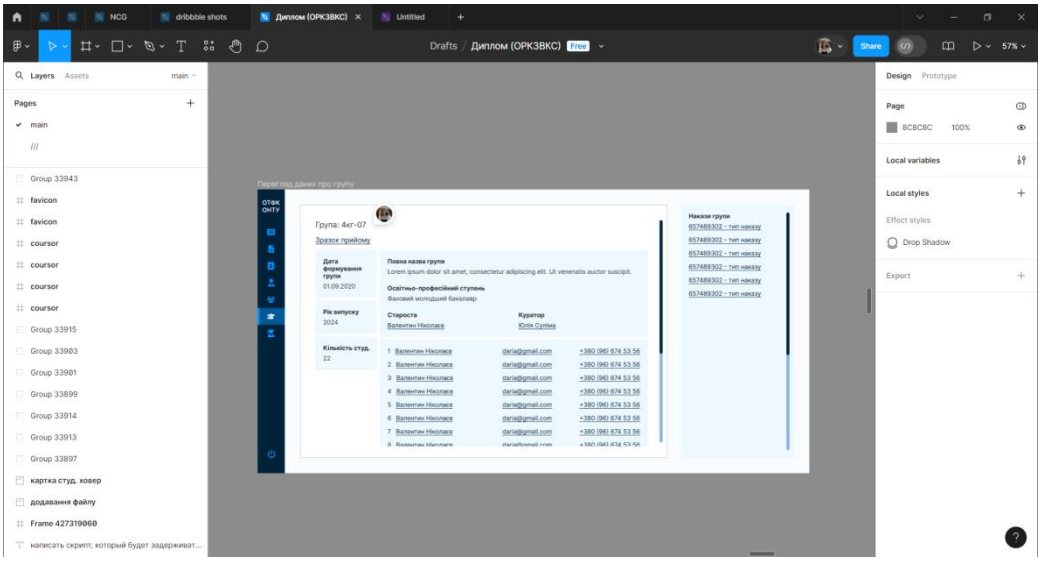


Рисунок 1.20. Сторінка перегляду інформації про групу

Розглянемо також рис. 1.21, на якому зображено поп-ап підтвердження дії при виході з веб-системи.

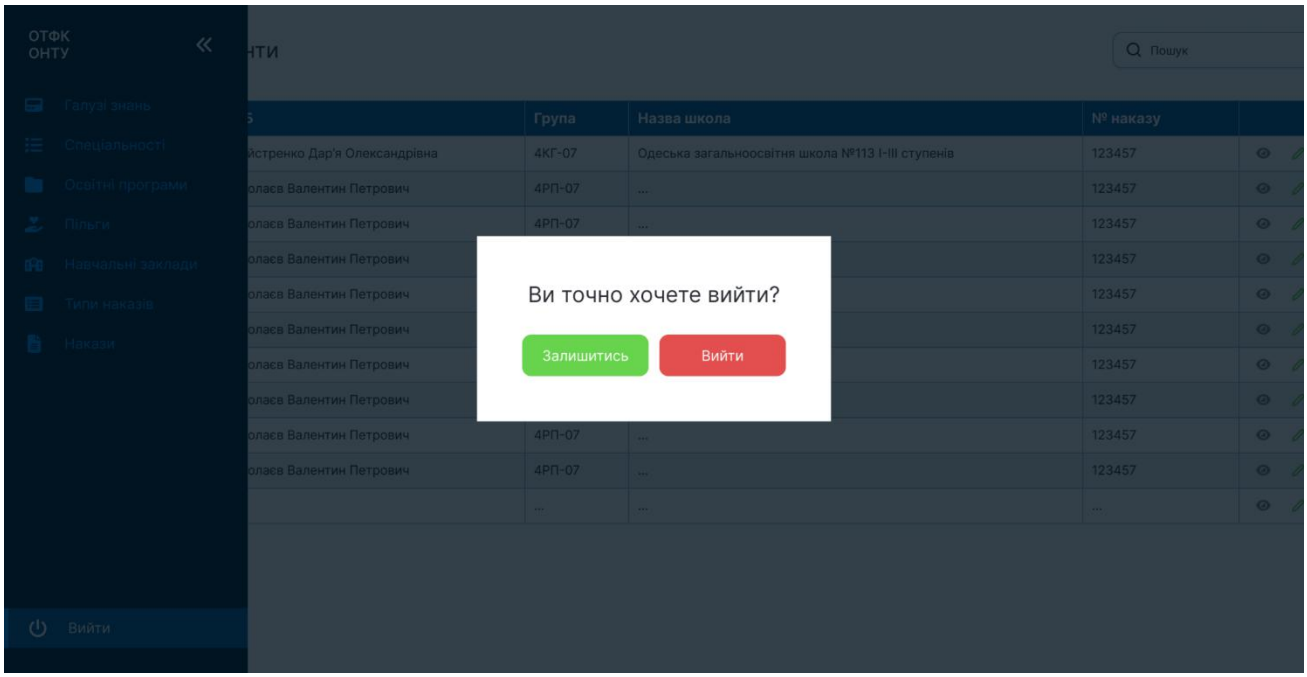


Рисунок 1.21. Поп-ап підтвердження дії при виході з веб-системи

1.3 Реалізація веб-системи

1.3.1 Налаштування базового шаблону

Базовий шаблон, схема якого зображена на рис. 1.22 є основою для побудови всіх сторінок веб-системи. Він забезпечує єдину структуру, стиль для всіх сторінок та використовується для уникнення дублювання однакового коду. Це значно спрощує та оптимізує час розробки Front-End частини [8].

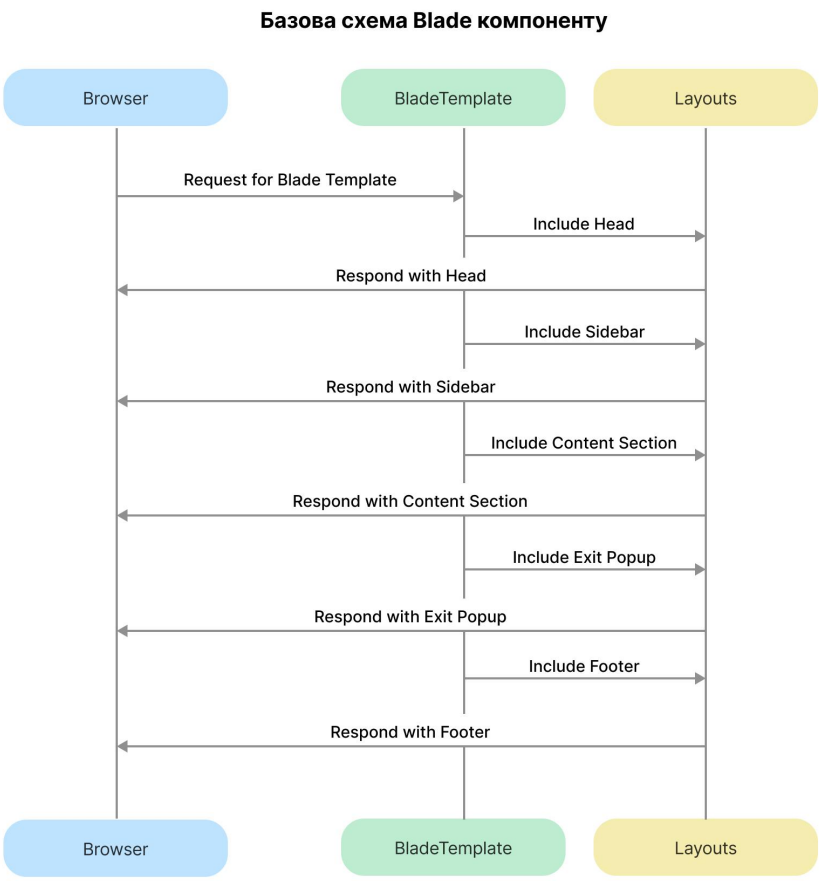


Рисунок 1.22. Схема базового шаблону сторінок

Розглянемо код базового шаблону нижче:

```
<!DOCTYPE html>
<html lang="uk">
@include('layouts._elements._head')
<body class="@yield('body-class', 'home-page')">
<div class="fluid-wrapper">
    <main class="@yield('main-class', 'home-main')">
        @include('layouts._elements.sidebar')
        <section class="@yield('main-section', 'section-main')">
            <div class="container">
                @yield('content')
            </div>
        </section>
    </main>
    @include('layouts._elements.exit-popup')
    <footer class="footer"></footer>
</div>
@include('layouts._elements._footer')
</body>
</html>
```

Цей шаблон починається з оголошення документа HTML5 та встановлення мови сторінки на українську за допомогою тега `<!DOCTYPE html>` і атрибуту `lang="uk"`. Далі, за допомогою директиви

`@include('layouts._elements._head')`, підключається зовнішній файл `_head`, який зазвичай містить метатеги, посилання на стилі та скрипти.

Тег `<body>` визначає клас для тіла документа, який може бути змінений для кожної конкретної сторінки через директиву `@yield`. Якщо клас не заданий, за замовчуванням використовується `home-page`. Далі для налаштовування макету, створюється обгортка для основного контенту за допомогою тега `<div class="fluid-wrapper">`.

Основний блок контенту визначається тегом `<main class="@yield('main-class', 'home-main')">`, який включає бічну панель з файлу `_sidebar` за допомогою директиви `@include('layouts._elements.sidebar')`. Клас основного блоку також можна налаштовувати для кожної сторінки.

Головний розділ сторінки визначається тегом `<section class="@yield('main-section', 'section-main')">`, всередині якого використовується контейнер `<div class="container">` для основного контенту, що буде додаватися через директиву `@yield('content')`.

Шаблон також включає поп-ап для виходу, який підключається за допомогою директиви `@include('layouts._elements.exit-popup')`, та футер сторінки, визначений тегом `<footer class="footer"></footer>`. Футер порожній у шаблоні, але його можна наповнити контентом при необхідності.

Наприкінці шаблону підключається зовнішній файл `_footer`, який зазвичай містить скрипти та інші ресурси, що мають бути завантажені наприкінці сторінки, за допомогою директиви `@include('layouts._elements._footer')`.

JavaScript бібліотеки потрібні для спрощення написання коду та додавання функціональності до веб-сторінок. Тому нижче розглянемо, як саме підключалися скрипти [9]:

```
<script src="{{ asset('assets/js/min/jquery-3.6.0.min.js') }}"></script>
<script src="{{ asset('assets/js/min/inputmask.min.js') }}"></script>
<script src="{{ asset('assets/js/main.js') }}"></script>
<script src="{{ asset('assets/js/mask.js') }}"></script>
<script src="{{ asset('assets/js/ajax-search.js') }}"></script>
<script type="text/javascript">
    $(document).ready(function () {
```

```
            $('.popup-btn-stay').on('click', function () {
                let modal = $(this).closest('[data-modal="exit-popup"]');
                modal.modalClose();
            });
        });
        $('.confirmation').on('click', function () {
            return confirm('Are you sure?');
        });
    </script>
    @yield('js')
```

Спочатку підключаються необхідні JavaScript файли за допомогою директиви {{ asset('...') }}, яка генерує URL для зазначених файлів. Це дозволяє легко керувати бібліотеками та власними скриптами. Я підключила:

- jquery-3.6.0.min.js – бібліотека jQuery, яка спрощує роботу з DOM та надає потужні засоби для створення динамічних веб-сторінок;
- inputmask.min.js – плагін для jQuery, який дозволяє створювати маски для вводу даних у форми;
- main.js, mask.js, ajax-search.js – власні JavaScript файли, які містять користувацький код для додаткової функціональності веб-застосунку.

Далі йде блок JavaScript коду, який виконується після завантаження документа. Код всередині $(document).ready(function () { ... }); виконується після того, як весь документ завантажиться та буде готовим до взаємодії.

На елементи з класом .popup-btn-stay навішується обробник події кліку, який знаходить найближчий елемент з атрибутом data-modal="exit-popup" та закриває його за допомогою методу modalClose().

На елементи з класом .confirmation навішується обробник події кліку, який виводить підтверджуюче повідомлення користувачу з питанням "Are you sure?". Якщо користувач підтверджує, то повертається true і дія продовжується, інакше повертається false.

Для забезпечення коректної роботи веб-системи, додавання стилів та метаданих до коду, пропишемо наступний код у тезі <head>:

```
    <head>
        <meta charset="utf-8" />
        <meta
            name="viewport"
            content="width=device-width, initial-scale=1, maximum-scale=1"
        />
        <!-- Add the slick-theme.css if you want default styling -->
```

```
    <link  rel="stylesheet"  href="{{  asset('assets/css/min/slick.min.css')  }}"
referrerpolicy="no-referrer"/>
    <link rel="stylesheet" href="{{ asset('assets/css/main.css') }}" />
    <!--favicons-->
    <link           rel="icon"           type="image/png"           sizes="32x32"
href="{{ asset('/assets/images/favicon.png') }}" />
    @yield('css')
    <title>@yield('title', 'Page')</title>
    <meta name="description" content="@yield('description', 'Page')" />
</head>
```

Спочатку встановлюються метадані: кодування символів на UTF-8 і налаштування viewport для забезпечення коректного відображення сторінки на різних пристроях. Далі підключаються CSS файли для налаштування зовнішнього вигляду сторінки. Файл - main.css містить основні стилі сторінки. Також додається фавікон, який відображається у вкладці браузера поруч із назвою сторінки.

Для підключення додаткових CSS файлів у дочірніх шаблонах використовується директива @yield('css').

Далі встановлюється заголовок та опис сторінки. Використання директив @yield дозволяє змінювати ці значення у дочірніх шаблонах, задаючи унікальні заголовки та описи для кожної сторінки.

Цей код забезпечує налаштування основних параметрів веб-сторінки, підключення необхідних стилів та метаданих для покращення відображення та роботи сайту.

Для реалізації функціоналу поп-апу виходу з веб-системи використовується наступний код. Розглянемо цей код детальніше:

```
<div class="modal modal--sm js-modal" data-modal="exit-popup">
    <div class="modal-overlay js-close-modal"></div>
    <div class="popup-content">
        <h3 class="popup-title">Ви точно хочете вийти?</h3>

        <div class="popup-btn-wrapper">
            <form action="{{ route('admin.logout') }}" method="post">
                @csrf
                @method('post')
                <button type="submit" class="popup-btn-exit">Вийти</button>
            </form>
            <div class="popup-btn-stay">Залишитись</div>
        </div>
    </div>
</div>
```

Він створює модальне вікно, яке спливає при натисканні на кнопку виходу. Модальне вікно має клас modal і містить невеликий розмір модального вікна modal--sm. Додатково, використовується клас js-modal для інтерактивності.

Внутрішня частина модального вікна складається з двох основних елементів: затемненої області фону для закриття вікна при кліку за його межами, та основного вмісту поп-апу. Заголовок поп-апу <h3 class="popup-title">Ви точно хочете вийти?</h3> - повідомляє користувача про намір вийти із системи. Потім йде блок кнопок, який містить форму для підтвердження виходу з системи. Форма відправляє POST-запит до маршруту admin.logout, використовуючи захист CSRF. Кнопка "Вийти" відправляє форму, викликаючи вихід з системи, а кнопка "Залишитись" закриває модальне вікно, дозволяючи користувачу залишитися в системі.

Цей код забезпечує зручний інтерфейс для підтвердження виходу з веб-системи, надаючи користувачу можливість вибору та захищаючи від випадкового виходу.

1.3.2 Створення компонентів

Розглянемо компонент components.table-page.header, який додає заголовок на сторінки таблиць з полем для пошуку. Він приймає заголовок сторінки, який потрібно відображати в хедері сторінки. Код компоненти наведу нижче.

```
<div class="section-main-heading">
    <h2 class="section-title">{{ $title }}</h2>
    <div class="search-input-wrapper">
        <form method="get">
            <input       type="text"       name="search"       placeholder="Пошук"
value="{{ request('search') }}" class="search-input"/>
            <button  type="submit"  class="search-icon"  style="background:  none;
border: none; padding: 0; cursor: pointer;">
                <svg  width="20"  height="20"  viewBox="0  0  20  20"  fill="none"
xmlns="http://www.w3.org/2000/svg">
                    <path
                        d="M16.5   16.5L13.1167   13.1167M14.9444   8.72222C14.9444
12.1587   12.1587   14.9444   8.72222   14.9444C5.28578   14.9444   2.5   12.1587   2.5
8.72222C2.5 5.28578 5.28578 2.5 8.72222 2.5C12.1587 2.5 14.9444 5.28578 14.9444
8.72222Z"
                        stroke="#2A2D34"      stroke-width="1.66667"      stroke-
linecap="round" stroke-linejoin="round"/>
                </svg>
            </button>
        </form>
```

```
    </div>
</div>
```

Разом з цією компонентою створена компонента, код якої наведено нижче. Вона відповідає за відображення помилок, статусів на сторінках таблиць і це є компонента components.table-page.message.

```
@props(['type'])
@php
    $bgColor = match($type) {
        'success' => '#68D34D',
        'error' => '#E34F4F',
        'warning' => '#FFBE15',
        default => '#FFBE15',
    };
@endphp
<div class="alert-message" style="background-color: {{ $bgColor }}; padding: 5px 10px 5px 10px; padding-right: 30px; border-radius: 7px; margin-bottom: 20px; display: inline-block; position: relative;">
    {{ $slot }}
    <span class="close-button" style="cursor: pointer; position: absolute; top: 5px; right: 10px;">✖</span>
</div>
@section('js')
    <script>
        $(document).ready(function () {
            const alertMessages = $('.alert-message');
            alertMessages.each(function () {
                const message = $(this);
                setTimeout(function () {
                    message.fadeOut();
                }, 10000);
            });
            $('.close-button').on('click', function () {
                $(this).parent().fadeOut();
            });
        });
    </script>
@stop
```

В коді задані можливі кольори в залежності від того, що саме відображено на поточний момент. Для помилок заданий червоний колір, для попереджень - жовтий, а для успішних відповідей - зелений.

Також я написала js обробник, який відповідає за закриття цього повідомлення через 10 секунд після його появи або при кліку на хрестик в кінці повідомлення.

Наступна компонента яку необхідно розглянути - components.pagination.pagination, яка додає на сторінку таблиць пагінацію, яка надає можливість переключатися між сторінками, та передивлятися усі записи.

Він приймає як параметр колекцію об’єктів які вміють пагінуватися, в нашому випадку це може бути студент. Розглянемо її код.

```
<div class="pagination-wrapper">
    @if ($paginator->previousPageUrl())
        <a href="{{ $paginator->previousPageUrl() }}" class="pagination-arr">
            <svg  width="34"  height="32"  viewBox="0  0  34  32"  fill="none"
xmlns="http://www.w3.org/2000/svg">
                <path
                    d="M10.293  15.2937C9.90234  15.6843  9.90234  16.3187  10.293
16.7093L15.293  21.7093C15.6836  22.0999  16.318  22.0999  16.7086  21.7093C17.0992
21.3187  17.0992  20.6843  16.7086  20.2937L13.4117  16.9999H22.9992C23.5523  16.9999
23.9992  16.553  23.9992  15.9999C23.9992  15.4468  23.5523  14.9999  22.9992
14.9999H13.4148L16.7055  11.7062C17.0961  11.3155  17.0961  10.6812  16.7055
10.2905C16.3148  9.8999  15.6805  9.8999  15.2898  10.2905L10.2898  15.2905L10.293
15.2937Z"
                    fill="#1E3050"/>
            </svg>
        </a>
    @endif
    @if($paginator->total() > $paginator->perPage())
        <ul class="paginstion-list">
            @for ($i = 1; $i <= $paginator->lastPage(); $i++)
                @if ($i <= 1 || $i > $paginator->lastPage() - 1 || ($i >=
$paginator->currentPage() - 1 && $i <= $paginator->currentPage() + 1))
                    <li class="paginstion-item @if ($paginator->currentPage() ===
$i) current @endif">
                       <a    href="{{    $paginator->appends(request()->query())-
>onEachSide(10)->url($i) }}">{{ $i }}</a>
                    </li>
                    @php
                        $dots = false;
                    @endphp
                @else
                    @if (!$dots)
                        <li class="paginstion-item">
                            <span>...</span>
                        </li>
                        @php
                            $dots = true;
                        @endphp
                    @endif
                @endif
            @endfor
        </ul>
    @endif
    @if ($paginator->nextPageUrl())
        <a href="{{ $paginator->nextPageUrl() }}" class="pagination-arr">
            <svg  width="34"  height="32"  viewBox="0  0  34  32"  fill="none"
xmlns="http://www.w3.org/2000/svg">
                <path
                    d="M23.7054  15.2937C24.0961  15.6843  24.0961  16.3187  23.7054
16.7093L18.7054  21.7093C18.3148  22.0999  17.6804  22.0999  17.2898  21.7093C16.8992
21.3187  16.8992  20.6843  17.2898  20.2937L20.5867  16.9999H10.9992C10.4461  16.9999
9.99919  16.553  9.99919  15.9999C9.99919  15.4468  10.4461  14.9999  10.9992
14.9999H20.5836L17.2929  11.7062C16.9023  11.3155  16.9023  10.6812  17.2929
10.2905C17.6836  9.8999  18.3179  9.8999  18.7086  10.2905L23.7086  15.2905L23.7054
15.2937Z"
                    fill="#1E3050"/>
```

```
                </svg>
            </a>
        @endif
    </div>
```

Тут розраховується всі елементи та їх значення, які мають відображатись. Якщо у пагінатора є URL для попередньої сторінки ($paginator->previousPageUrl()), відображається стрілка для перемикання на попередню сторінку. Якщо загальна кількість записів більше ніж записів на одній сторінці ($paginator->total() > $paginator->perPage()), відображається список сторінок. За допомогою циклу for перебираються всі сторінки від 1 до останньої ($paginator->lastPage()). Якщо номер сторінки є в діапазоні від поточної сторінки мінус один до поточної сторінки плюс один, або якщо це перша чи остання сторінка, сторінка відображається. Якщо сторінка знаходиться поза вказаним діапазоном, додається "..." для позначення пропущених сторінок.

Якщо у пагінатора є URL для наступної сторінки ($paginator->nextPageUrl()), відображається стрілка для перемикання на наступну сторінку.

Цей підхід дозволяє користувачам легко переміщатися між сторінками, переглядати всі доступні записи та отримувати візуальний індикатор поточної сторінки та можливих сторінок для перемикання. А сама компонента дає можливість швидко інтегрувати їх на будь які сторінки з пагінацією.

Компонента components.pagination.pagination-info відображає діапазон відображаємих наразі елементів та загальну кількість елементів.

Компонента components.form-page.margin-wrapper використовується для додавання обгортки на поля вводу для сторінки форм.

Компоненти components.form-page.search-field-many - компонента для пошуку з мультіселектом та components.form-page.search-field-only - компонента для пошуку одного запису використовується на сторінках створення та редагування записів та забезпечується швидке створення та налаштування цих полів. Нижче приведу код компоненти для мультипошуку.

```
@props([
    'name' => 'name',
    'id' => '',
    'fieldName' => 'Пошукове поле',
```

```
    'placeholderSearchField' => 'Почніть вводити, для пошуку',
    'class' => '',
    'oldList' => [],
    'createObjectRoute' => '',
])
<div class="student-form-field search-many {{ $class }}  @error($name) error
@enderror" id="{{ $id }}">
    <label for="{{ $name }}" class="student-form-label">{{ $fieldName }}
        @if($createObjectRoute)
            <a href="{{ $createObjectRoute }}" target="_blank" style="vertical-
align: middle;">
                <svg width="19" height="19" viewBox="0 0 19 19" fill="none"
xmlns="http://www.w3.org/2000/svg">
                    <path d="M4 9.56055L14.8787 9.56055" stroke="#0096FF" stroke-
width="2" stroke-linecap="round"/>
                    <path d="M9.56055 4L9.56055 14.8787" stroke="#0096FF" stroke-
width="2" stroke-linecap="round"/>
                </svg>
            </a>
        @endif
    </label>
    <input
        type="text"
        id="{{ $name }}"
        name="{{ $name }}_search"
        placeholder="{{ $placeholderSearchField }}"
        class="student-form-input search-field"
        autocomplete="off"
    />
    <div class="student-form-search-result">
        <ul class="student-form-search-result-list"></ul>
    </div>
    <ul class="student-form-selected-options show">
        @if(!empty($oldList))
            @foreach($oldList as $item)
                <li      class="student-form-selected-option"      data-selected-
id="{{ $item['value'] }}" data-name="{{ $name }}">
                    <span>{{              $item['text']              }}</span><img
src="/assets/images/icons/cross.svg" alt="cross icon"/>
                </li>
            @endforeach
        @endif
    </ul>
    @error($name)
        <span class="student-form-error-text">{{ $message }}</span>
    @enderror
</div>
```

Єдина різниця з компонентою для пошуку одного поля це наявність таких параметрів як oldList, в той час як для одиничного пошуку замість списку приймається айді старого запису, та його ім’я, який необхідні для заповнення попередньо обраних значень.

1.3.3 Створення Blade-представлень

Розглянемо типові прикладу побудови представлень на прикладі студента. Для початку розглянемо представлення студента та визначимо які компоненти в ньому використовуються. Код представлення наведу нижче.

```
@php use App\Models\Enums\FinancingForms;use App\Models\Student; @endphp
@extends('layouts.page')
@section('title', 'Список студентів')
@section('content')
@component('components.table-page.header', ['title' => 'Студенти'])@endcomponent
    @if(session('success'))
        @component('components.table-page.message', ['type' => 'success'])
            {{ session('success') }}
        @endcomponent
    @endif
    @if(session('error'))
        @component('components.table-page.message', ['type' => 'error'])
            {{ session('error') }}
        @endcomponent
    @endif
    @if($errors->any())
        @component('components.table-page.message', ['type' => 'warning'])
            @foreach ($errors->all() as $error)
                {{ $error }}<br>
            @endforeach
        @endcomponent
    @endif
    <table class="user-table">
        <thead class="user-table-head">
        <tr>
            <th>#</th>
            <th>ПІБ</th>
            <th>Група</th>
            <th>Освітня програма</th>
            <th>Номер телефону</th>
            <th>Форма фінансування</th>
            <th>Дії</th>
        </tr>
        </thead>
        <tbody class="user-table-body">
        @php
            /**
             * @var Student[] $students
             */
        @endphp
        @foreach($students as $student)
            <tr class="user-table-row">
                <td>{{ $student->id }}</td>
                <td>{{   $student->firstname   }}   {{   $student->lastname   }}
{{ $student->surname }}</td>
                <td><a     href="{{     route('admin.groups.show',     $student-
>group) }}">{{ $student->group->code }}</td>
                <td>
                    <a href="{{ route('admin.educational-programs.edit', $student-
>educational_program_id) }}">{{ $student->getEducationProgram()?->name }}</a>
                </td>
```

```
                <td><a     href="tel:{{     $student->phone     }}">{{     $student-
>phone }}</a></td>
                <td>{{     FinancingForms::getLabel(FinancingForms::from($student-
>financing_form)) }}</td>
                <td>
                    <div class="user-table-icons">
                        <a href="{{ route('admin.students.show', [$student])}}">
                           <!-- svg прибрана для економії місця -->
                        </a>
                        <a href="{{ route('admin.students.edit', [$student])}}">
                            <!-- svg прибрана для економії місця -->
                        </a>
                        <form                                              method="post"
action="{{ route('admin.students.destroy', [$student]) }}">
                            @method('delete')
                            @csrf
                            <button type="submit" class="confirmation"
                                    style="background: none; border: none; padding:
0; cursor: pointer;"><!-- svg прибрана для економії місця --></button>
                        </form>
                    </div>
                </td>
            </tr>
        @endforeach
        </tbody>
    </table>
    @component('components.pagination.pagination-info',       ['paginator'       =>
$students])@endcomponent
    <div class="section-main-bottom">
        @component('components.pagination.pagination',       ['paginator'       =>
$students])@endcomponent
        <a  href="{{  route('admin.students.create')  }}"  class="add-btn">Додати
запис</a>
    </div>
@stop
```

З наведеного вище коду видно, що представлення успадковується від базового шаблону веб-застосунку, та встановлюю title для конкретної сторінки. В секції content описується весь контент сторінки, сайдбар буде підключено автоматично, так як він підключений у базовому шаблоні. В представлені використовуються компоненти, які розглядались у попередньому розділі, а саме компоненти для пагінації, який додасть на сторінку пагінацію для студентів. Також використовується компонента components.table-page.header, яка додає поле пошуку та заголовок на сторінку, створення представлень стає дуже швидким за допомогою компонент та дозволяє ефективно будувати сторінки.

Розглянемо скорочену версію представлення для створення студентів, через його велику розміри. Розглянемо базову структуру та приклади компонент, які в ньому використовуються, код наведено нижче.

```
@extends('layouts.page')
```

```
    @section('title', 'Список студентів')
    @section('body-class', 'student-page')
    @section('main-class', 'student-main')
    @section('main-section', 'student-section')
    @section('content')
        @component('components.form-page.header', [
           'title' => 'Додати студента(-ку)',
           'route' => route('admin.students.list')
       ])@endcomponent
        <form  action="{{  route('admin.students.store')  }}"  class="student-form"
method="post" enctype="multipart/form-data">
            @method('POST')
            @csrf
        <div class="student-form-wrapper">
            <div class="student-form-inner">
                    @component('components.form-page.margin-wrapper',  ['class'  =>
'third', 'name' => 'firstname'])
                    <label for="firstname" class="student-form-label">Ім’я</label>
                    <input
                        type="text"
                        id="firstname"
                        name="firstname"
                        placeholder="Ім’я"
                        class="student-form-input name_mask"
                        autocomplete="off"
                        value="{{ old('firstname') }}"
                    />
                @endcomponent
                @component('components.form-page.search-field-only', [
                    'name' => 'group_id',
                    'fieldName' => 'Група',
                    'placeholderSearchField' => 'Введіть код або назву групи',
                    'class' => '',
                    'createObjectRoute' => route('admin.groups.create'),
                ])@endcomponent
            </div>
        </div>

        <div class="student-form-btn-wrapper">

            <button type="submit" class="student-form-btn submit">
                Зберігти
            </button>
        </div>
    </form>
@stop

@section('js')
    <script type="text/javascript">
        $(document).ready(function () {
            $('#group_id').on('input', function () {
                performAjaxRequest(
                    'group_id',
                    '{{ route('admin.search-groups') }}',
                    'GET',
                    function (resultList, item) {
resultList.append('<li   class="student-form-search-result-item"   data-id="'   +
item.id + '">' + item.code + ' - ' + item.name + '</li>');
                    },
                );
            });
```

```
        });
    </script>
@stop
```

Вище приведено структуру представлення для створення студентів та два типових компоненти, які в ньому використовуються, а саме компонент, який огортає будь які поля, для виведення помилок при валідації даних. Та компонента для пошуку одного запису, які ми розглянули вище, для налаштування пошуку використовую Ajax запити, схема яких відображена на рис. 1.23, до певного роуту системи. На прикладі вище видно що під час вводу в поле викликається функція performAjaxRequest, яку ми розглянемо в наступному підрозділі, та яка створює запит до роуту. Вона приймає такі значення як ID поля, роут, метод запиту, та анонімну функцію яка має заповнити вікно пошуку даними, які є.

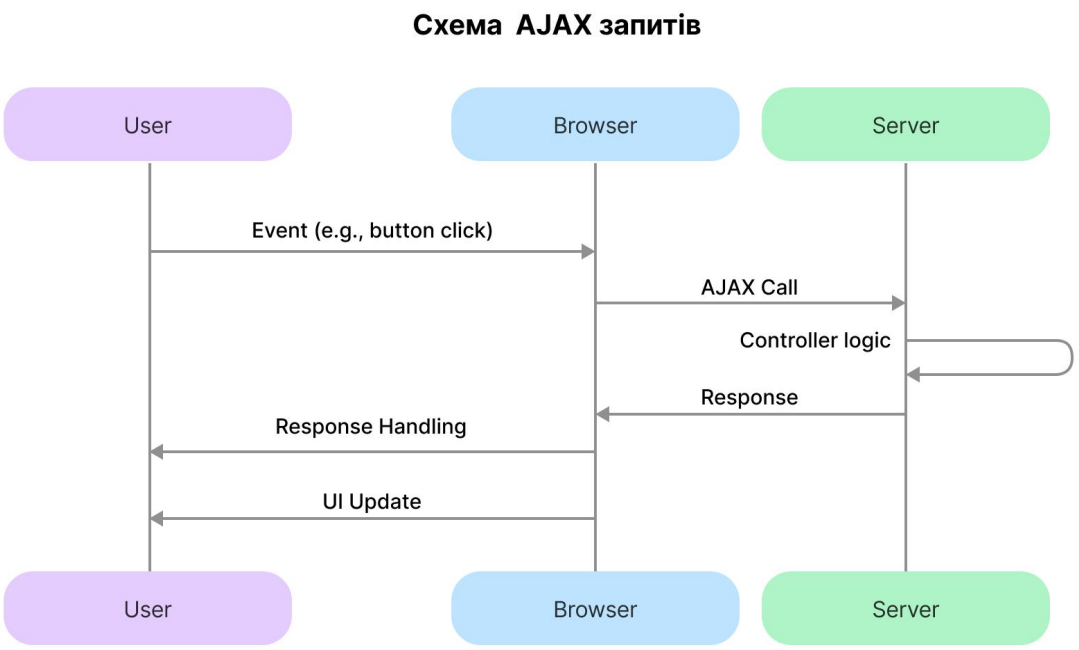


Рисунок 1.23. Схема AJAX-запиту до серверу

1.3.4 Налаштування AJAX-запитів та масок для полів

Розглянемо функцію performAjaxRequest, який відповідає за налаштування ajax запиті, а саме за запити для пошукових полів для компонентів з мільтівибором да одиночного пошуку для полів форм [10]. На вхід запит

приймає такі дані як fieldId, куди передається ID поля, для якого треба додати результати пошуку, url приймає роут до якого надсилається запит, method приймає метод запиту, далі передається анонімна функція яка опрацьовує результати пошуку. Для мультипошуку додані такі параметри як resultItemClass, selectedIds, які приймають масив обраних записів, та додає клас до елементів списку з відповідями, що дозволяє налаштувати пошук з мультивибором. Також метод приймає на вхід мінімальну довжину пошукового значення, при якому дозволено здійснювати запит на сервер, для отримання даних. Розглянемо його код.

```
function performAjaxRequest(fieldId, url, method, callback, resultItemClass = '', selectedIds = [], triggerLength = 3) {

let $field = $('#' + fieldId);
let searchValue = $field.val();
let $parentForm = $field.closest('.student-form-field');

if (searchValue.length >= triggerLength) {
$.ajax({
      url: url,
method: method,
      data: {
            search: searchValue,
            selected_ids: selectedIds
      },
      success: function (response) {
           var resultList = $parentForm.find('.student-form-search-result-list');
               resultList.empty();

               if (response.length > 0) {
                   $.each(response, function (index, item) {
                       callback(resultList, item);
                   });
               } else {
                   resultList.append('<li  class="student-form-search-result-item ' + resultItemClass + '" disabled>Нічого не знайдено</li>');
               }

               $parentForm.find('.student-form-search-result').addClass('show');
           },
           error: function (xhr, status, error) {
               console.error(error);
           }
       });
   } else {
       $parentForm.find('.student-form-search-result').removeClass('show');
   }
}
```

При дозволеній довжині пошукового значення, здійснюється запит до серверу за допомогою $.ajax(“тіло запиту та обробка”). В ньому вказується url до якого має відправити запит, після чого вказано метод, параметр data використовується для передачі параметрів пошуку, а саме search - для передачі значення для пошуку та selectedIds, яке містить в собі список ідентифікаторів, які буде потрібно виключити з пошуку.

Після вказується дії, які необхідно виконати при наступних відповідях запиту:

success - виконає код прописаний у блоці success, виведе та відобразить список з результатами пошуку, під полем для якого здійснювався пошук.

error - виконує код прописаний у блоці error та виведе у консоль браузера те, що сталася помилка.

В разі коли довжина значення для пошуку менша ніж вказана - запит приховує список результатів для поля.

Також розглянемо маски для полів електронної пошти, та поля номеру телефону, код яких наведено нижче.

```
$(document).ready(function () {
    // phone mask
    let phones = $('.phone_mask');
    if (phones.length) {
        phones.each(function () {
            Inputmask({
                "mask": "+380 (99) 999 99 99",
              }).mask($(this).get(0));
        });
    }
    // email mask
    let emails = $('.email_mask');
    if (emails.length) {
        emails.each(function () {
            Inputmask({
                "alias": "email",
            }).mask($(this).get(0));
        });
    }
});
```

Код виконується після повного завантаження документа, використовуючи функцію $(document).ready(function () { ... });. Це забезпечує, що всі елементи готові до маніпуляцій. Скрипт перебирає яким полям необхідна кожна з масок та

якщо такі елементи є - створює для них екземпляри класу Inputmask, із заданими параметрами та застосовує маску на поточному полі за допомогою mask($(this).get(0), де get(0) повертає сам елемент з jQuery-об’єкта.

1.4 Мануальне тестування

1.4.1 Тестування сторінки входу

Розглянемо сторінку входу до веб-системи.

Сторінка входу до веб-системи є ключовим елементом безпеки, яка захищає від небажаного доступу та забезпечує аутентифікацію користувачів. Для того, щоб увійти до веб-системи, необхідно ввести свою електронну пошту та пароль.

На рис. 1.24 зображено сторінку входу до веб-системи.

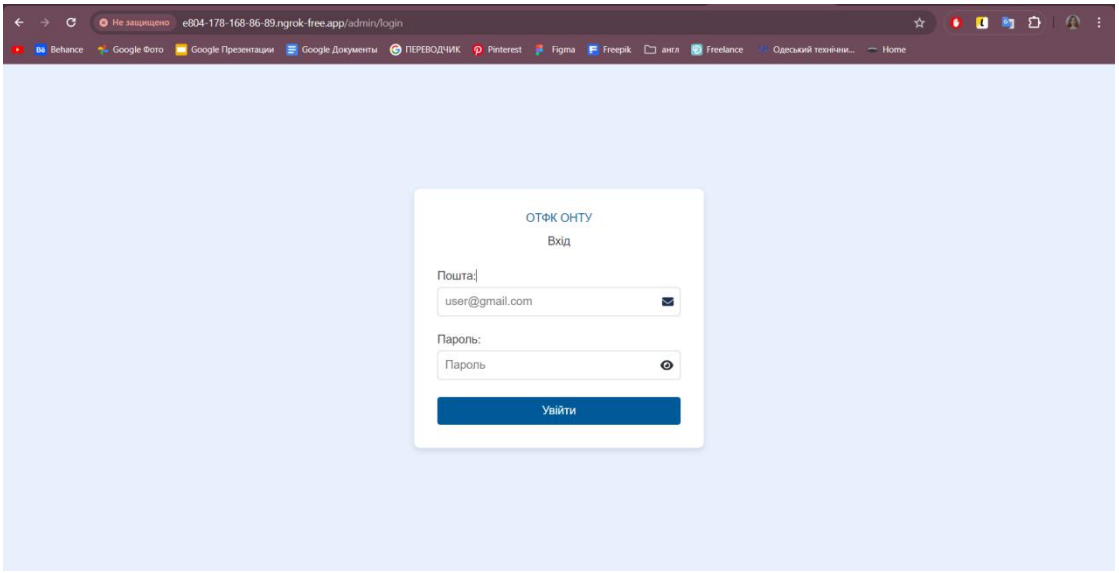


Рисунок 1.24. Сторінка входу до веб-системи

1.4.2 Тестування сторінки таблиці студентів

Сторінка таблиці студентів є ключовим компонентом системи обліку контингенту здобувачів. Вона надає можливість переглядати, редагувати та видаляти інформацію про студентів, а також додавати нові записи про студентів.

На сторінці представлена таблиця, яка включає в себе такі колонки як унікальний ідентифікатор запису, ПІБ студентів, номер групи, до якої належить студент, освітню програму, яку вивчає студент, номер телефону студента та тип

фінансування навчання (бюджет або контракт). Хоча при додаванні кожного студента збирається набагато більше інформації, в таблицю виводиться лише основна, яка може часто бути необхідна для адміністраторів веб-системи.

Також таблиця має такі функціональні можливість переглядати детальну інформацію про студента за допомогою іконки "око", можливість редагувати дані студента за допомогою іконки "олівець" та можливість видаляти запис про студента за допомогою іконки "смітник".

Додатково, на сторінці є можливість фільтрації та пошуку даних. Фільтрація дає можливість фільтрувати записи студентів по таким критеріям: ті, хто в академічній відпустці, ті, хто потребує гуртожиток або просто всі записи. Поле пошуку розташоване у верхньому правому куті, що дозволяє швидко знаходити необхідну інформацію. Також є кнопка для додавання нових записів, яка розташована у нижньому правому куті.

Сторінка таблиці студентів має зручну пагінацію, яка є зручною для перегляду великого обсягу даних, розділених на сторінки. Це забезпечує зручне користування таблицями. На рис. 1.25 зображено сторінку таблиці студентів.

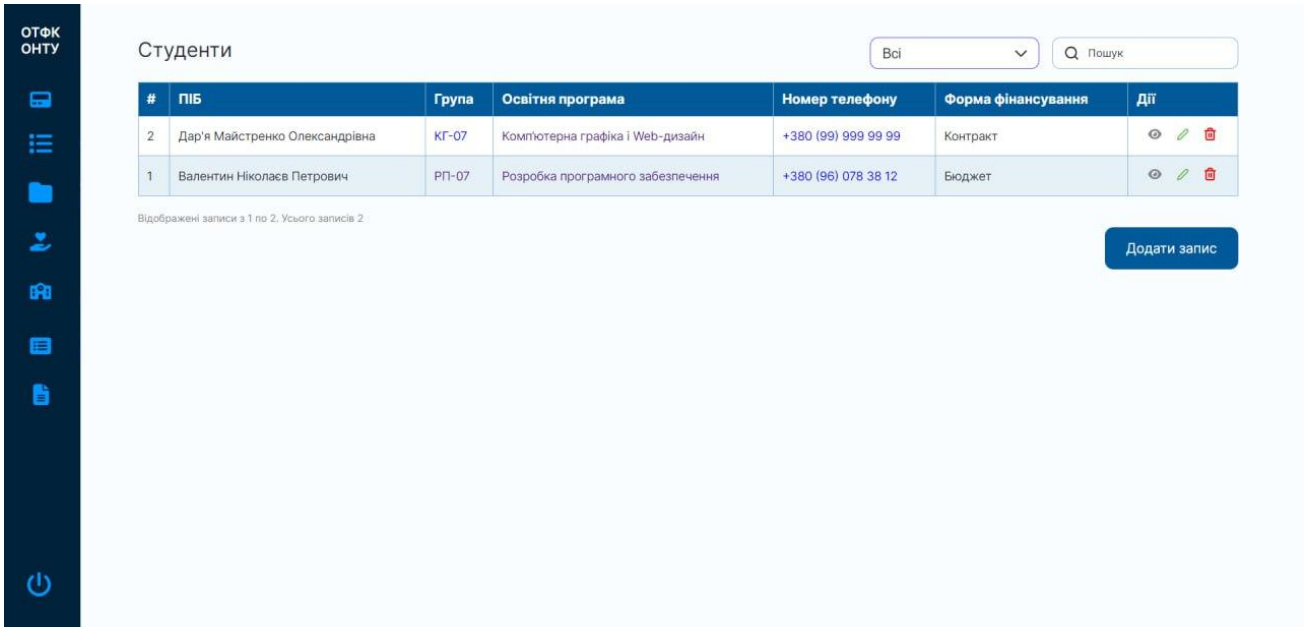


Рисунок 1.25. Сторінка таблиці студентів

1.4.3 Тестування бокової панелі

Бокова панель є однією з основних частин інтерфейсу веб-системи, яка відображається на всіх екранах. Завдяки панелі можна мати швидкий доступ до основних розділів та функцій системи, що значно полегшує навігацію для користувачів.

Бокова панель має два вигляди. Згорнутий вигляд є за замовчуванням, в якому відображаються лише іконки розділів, що забезпечує компактність і економію місця на екрані. Другий, тобто розгорнутий вигляд, з’являється при натисканні на "ОТФК ОНТУ". В ньому відображаються текстові назви кожної іконки, що полегшує орієнтацію серед розділів.

Також на панелі є кнопка "Вийти" для виходу з системи. Бокова панель забезпечує користувачам легкий доступ до необхідної інформації та функцій, роблячи роботу з веб-системою більш зручною та ефективною. На рис. 1.26 зображено два вигляди бокової панелі.

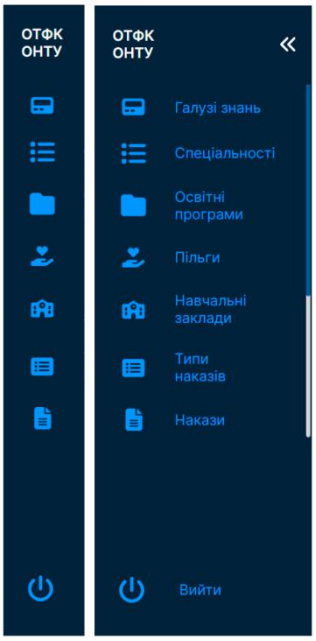


Рисунок 1.26. Згорнутий та розгорнутий вигляди бічної панелі

1.4.4 Тестування сторінки додавання запису про студента

Для додавання нового запису необхідно натиснути на синю кнопку під таблицею, яка переведе нас на сторінку додавання нового запису. Як і було передбачено на етапі розробки дизайну, форма на сторінці додавання має фіксований блок зі скролом, в якому знаходять всі поля, які треба заповнити для додавання запису про нового студента. Також біля деяких назв полів є блакитні плюси, при натиску на які можна перейти на форму додавання нового запису обраного поля у новому вікні, якщо у базі даних не буде необхідного. На рис. 1.27 зображено сторінку додавання нового запису про студента.

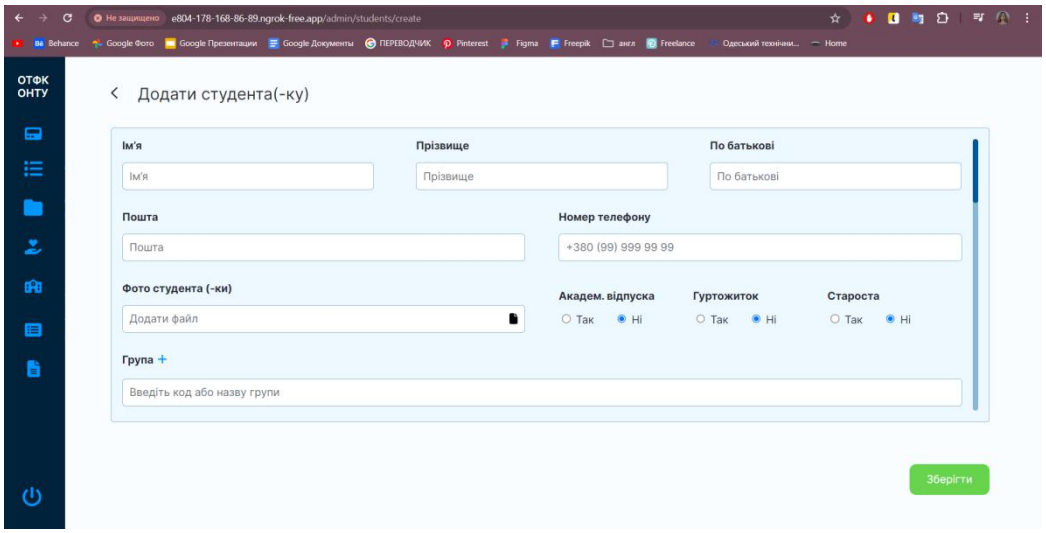


Рисунок 1.27. Сторінка додавання нового запису про студента

1.4.5 Тестування сторінки редагування запису про студента

Якщо у системі вже є готовий запис про студента, але треба оновити якісь дані, це можна зробити на сторінці редагування запису, на яку можна потрапити натиснувши зелений олівець у стовпці дії на сторінці таблиці студентів. На рис. 1.28 зображено сторінку редагування даних про студента.

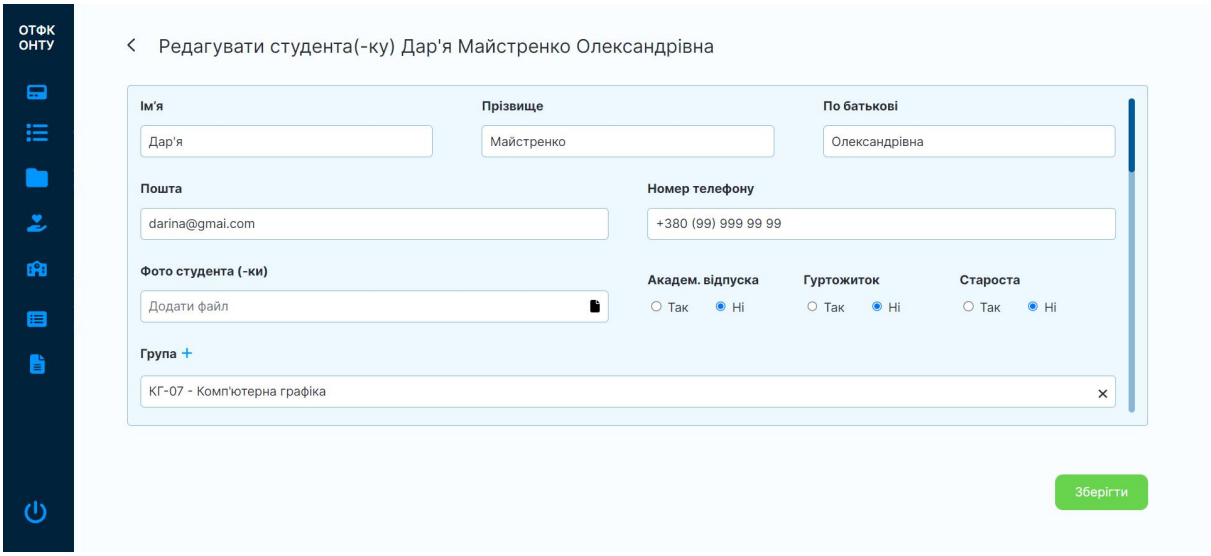


Рисунок 1.28. Сторінка редагування даних про студента

1.4.6 Функціональний огляд сторінки перегляду запису про студента

Сторінка перегляду інформації про студента дозволяє користувачам отримати детальну інформацію про кожного студента. Ця сторінка відображає всю необхідну інформацію про студента у зручному форматі.

Інформація про студента розділена на логічні блоки. Зверху виведено ПІБ та групу студента. При наведенні коло ПІБ з’являються кнопки дії, а саме можливість редагування та видалення запису. У лівому стовпці нижче є фотографія студента та інформація про дату вступу, чи треба гуртожиток та чи знаходиться студент у академічній відпустці. Праворуч знаходить три інформаційні блоки:

- особисті та контактні дані: електронна пошта, номер телефону, дата народження, ID паспорта, ким він виданий, дата видачі паспорта, ідентифікаційний код та місце проживання;
- освітні дані: освітня програма, форма навчання, на який курс вступив студент та тип зарахування;
- дані щодо минулого місця навчання: назва та адреса навчального закладу, номер документу про освіту, дата випуску зі школи та мова, яку вивчав студент.

Ще праворуч знаходиться поле у якому виведено всі накази які коли-небудь відносилися до студента. Сторінка містить в собі дані, при кліці на які є перехід до інших пов’язаних розділів.

На рис. 1.29 зображено сторінку перегляду даних про студента.

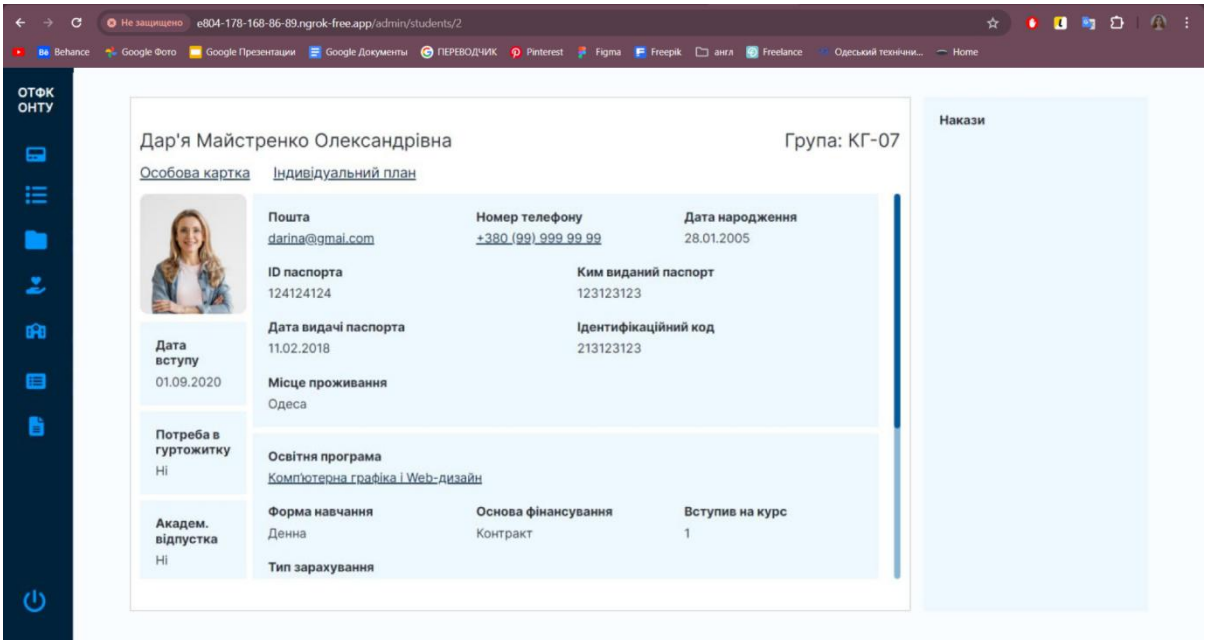


Рисунок 1.29. Сторінка перегляду даних про студент

2 ЕКОНОМІЧНИЙ РОЗДІЛ

В дипломному проєкті створена веб-система для Front-End частини автоматизованої веб-системи обліку руху контингенту здобувачів відділення комп'ютерних систем. Цей сайт не є комерційним і розроблений на замовлення. Він буде використовуватись завідувачем відділення та довіреними особами для забезпечення ефективного та зручного управління обліком здобувачів.

При оцінці ефективності створюваного сайту слід виходити з того, що залежно від результату, що досягається, можуть бути визначені наступні види ефективності сайту: економічна, функціональна та соціальна ефективність. В дипломному проєкті веб-система, яка розроблялася, не є комерційною, доходу від неї не буде, тому розраховуємо лише витрати на її розробку та функціональну і соціальну ефективності веб-системи.

Загальні витрати ($В_з$) на створення сайту розраховуються за формулою:

$$В_з = В_р + В_в + В_е,$$

Де $В_р$ – витрати на розробку сайту; $В_в$ – витрати на впровадження сайту; $В_е$ – витрати на експлуатацію сайту;

Витрати на розробку сайту ($В_р$) є одноразовими та складаються з вартості наступних видів робіт зі створення сайту :

6. Аналіз та розробка дизайну сайту: розробка фірмового стилю, основних та сторінок перегляду, редагування інформації веб-системи, меню
7. Підготовка сторінок-шаблонів
8. Програмна розробка сайту: створення Front-End частини веб-системи

Для визначення витрат на розробку сайту ($В_р$) розраховуємо оплату праці виконавців, безпосередньо притягнених до її виконання. Над реалізацією проєкту Web-системи працювали Front-End та Back-End розробники.

Для визначення трудомісткості розробки сайту ($В_р$) складемо план-графік по розробці веб-системи і тривалості виконання робіт. Розподіл робіт по етапах і видах виконавців наведено в таблиці 2.1.

Таблиця 2.1. План-графік по розробці веб-системи

№	Назва етапу	Час виконання (годин)	Посада виконавця
1	Аналіз та розробка дизайну веб-системи	40 годин	Web дизайнер
2	Розробка Front-End частини веб-системи	70 годин	Front-End розробник
ВСЬОГО:		110 годин	

У таблиці 2.2 описані етапи розробки та витрати на заробітну плату.

Таблиця 2.2. Витрати на заробітну плату

№	Персонал	Етапи розробки	Кількість робочих годин (або днів)	Погодинна ставка (або денна ставка), грн.	Заробітна плата, грн.
1	Веб дизайнер	Аналіз предметної області.	8 годин	120грн./год.	960 грн.
2	Веб дизайнер	Аналіз конкурентів та цільової аудиторії.	10 годин	120грн./год.	1 200 грн.
3	Веб дизайнер	Створення дизайну	22 години	120грн./год.	2 640 грн.
4	Front-End розробник	Розробка Front-End частини	70 годин	120грн./год.	8 400 грн.
				ВСЬОГО:	$В_{зп}$ = 13 200 грн.

Візьмемо до уваги, що розробка Front-End частини автоматизованої веб-системи обліку руху контингенту здобувачів відділення комп'ютерних систем є

лише частиною розробки для цілої веб-системи і є комплексною роботою. Тому для розрахунку повної суми витрат на заробітну плату я додам суму заробітної плати Back-End розробник. В цю суму входить ініціалізація проєкту веб-системи (5 годин), створення структури та зв'язків баз даних (14 годин), написання CRUD системи для веб-системи (90 годин) та генерація документів (20 годин). Розраховуємо суму з погодинною оплатою у 140грн./год. Таким чином необхідно буде витратити ще 18 060 грн. Тепер можемо розрахувати вартість заробітної плати на повну розробку веб-системи:

$$В_{зп} = 13\ 200 + 18\ 060 = 31\ 260 \text{ грн.}$$

До витрат також треба віднести єдиний соціальний внесок, який становить 22% від заробітної плати і обчислюється за такою формулою:

$$В_{єсв} = В_{зп} \text{ x } 0{,}22 = 31\ 260 \text{ x } 0{,}22 = 6\ 877 \text{ грн.}$$

Розрахуємо загальні витрати ($В_р$) на розробку веб-сайту, в які треба додати суму витрат на заробітну плату праці персоналу ($В_{зп}$) та єдиного соціального внеску ($В_{єсв}$):

$$В_р = В_{зп} + В_{єсв} = 31\ 260 + 6\ 877 = 38\ 137 \text{ грн.}$$

Витрати на впровадження сайту ($В_в$) складаються з двох складових :

1. За даними сайту imena.ua, вартість на реєстрацію доменного імені .com зазвичай становить 756 грн. (Вв1);
2. Витрати на реєстрацію в пошукових системах (Вв2), наприклад Google через Google Search Console є безкоштовною.

$$В_в = В_{в1} + В_{в2} = 756 + 0 = 756 \text{ грн.}$$

Витрати на експлуатацію сайту ($В_е$) включають вартість робіт з підтримки сайту в робочому стані і вартість послуг по продовженню доменного імені на 1 рік. У нашому випадку вираховуємо вартість хостингу на 1 рік. У цю суму також входить вартість налаштування параметрів серверу хостингу, забезпечення щомісячного захисту веб-системи та створення резервних копій системи. В таблиці 2.3 визначаються постійні витрати як сума витрат на впровадження та експлуатацію сайту протягом року.

Таблиця 2.3. Постійні витрати

№	Стаття витрат	Вартість за рік, грн.
1	Хостинг на 1 рік	8 707 грн.
	Всього:	$В_е$ = 8 707 грн.

Вартість послуг хостингу на 1 рік розраховувалося на основі рис. 2.1 даних з сайту хмарної платформи DigitalOcean, де вартість хостингу на місяць становить 18 доларів – 725,58 грн., на рік – 216 доларів – 8 707 грн.

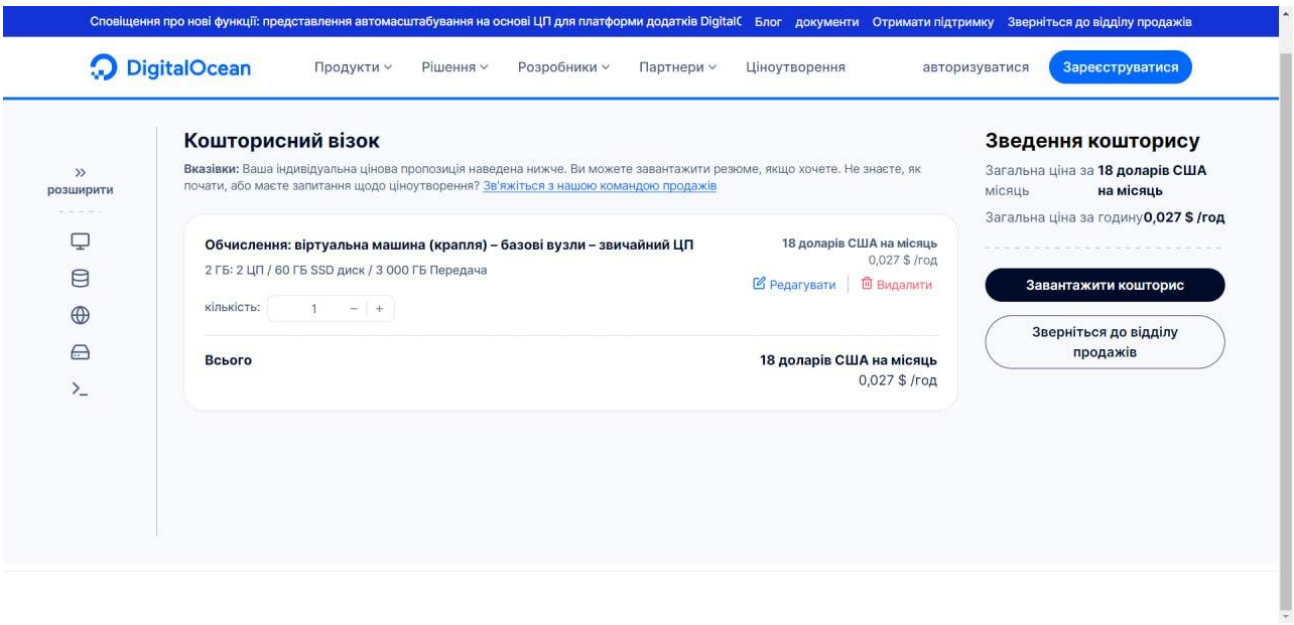


Рисунок 2.1 Вартість хостингу з даних сайту хмарної платформи DigitalOcean

Загальні витрати (Вз) на розробку, впровадження та експлуатацію веб-сайту розраховуються за наступною формулою:

$$В_з = В_р + (В_в + Ве) = 31\ 260 + (756 + 8\ 707) = 40\ 723 \text{ грн.}$$

Функціональна ефективність

Функціональна ефективність автоматизованої веб-системи обліку руху контингенту здобувачів відділення комп'ютерних систем полягає в забезпеченні захищеного зберігання усієї інформації в одному місці. Веб-система надає можливість зберігати, оновлювати та отримувати детальну інформацію про студентів, групи, батьків, спеціальності, освітні програми, накази, типи наказів, співробітників, посади, навчальні заклади та пільги у будь-який зручний для

користувача час. Це гарантує, що завідувач відділення та довірені особи завжди мають доступ до актуальних даних.

Система оптимізує технологічні процеси. Завідувач відділення може додати нового студента, вводячи його особисті дані в систему. Інші дані, такі як група, галузь знань або навчальний заклад, автоматично підтягнуться з попередньо заповнених таблиць. При необхідності, є можливість одразу додати нові суміжні дані. Це значно скорочує час і зусилля, необхідні для введення або перегляду та пошуку даних вручну кожен раз. Всі дані зберігаються в централізованій базі, що дозволяє уникнути дублювання інформації та помилок, пов'язаних з ручним введенням. Автоматизоване управління та обробка даних дозволяє швидко генерувати звіти та необхідні документи.

Використання автоматизованої системи забезпечує значне підвищення продуктивності праці, оскільки багато повторюваних операцій виконуються автоматично. Система дозволяє легко відстежувати зміни та оновлення, що сприяє більш ефективному управлінню контингентом здобувачів. Отже, функціональна ефективність автоматизованої веб-системи обліку руху контингенту здобувачів полягає в забезпеченні високого рівня упорядкування даних, їхньої точності та доступності, а також у значному спрощенні та прискоренні процесів керування та обробки інформації.

Соціальна ефективність залежить від типу об'єкту сайтобудування установи для якої створюється сайт. У моїй дипломній роботі це відображається так: система спрощує завідувачу відділення та довіреним особам ефективно керування даними про студентів, групи, спеціальності та інші аспекти освітнього процесу, що сприяє кращій організації та управлінню. Це допомагає розширити можливості комунікації між студентами, їх батьками та адміністрацією, що підвищує загальний рівень задоволеності учасників освітнього процесу.

Крім того, система сприяє формуванню позитивного іміджу відділення комп'ютерних систем як сучасного інноваційного освітнього закладу. Завдяки автоматизації багатьох процесів, пов'язаних з обліком та управлінням

контингентом здобувачів, система демонструє відкритість до новітніх технологій та підвищену ефективність роботи, що є привабливим фактором для потенційних студентів та їхніх батьків. Це також допомагає зміцнити довіру до адміністрації та її здатності забезпечити високий рівень організації навчального процесу.

Важливим фактором є те, що доступ до даних у системі мають лише завідувач відділення та довірені особи, що забезпечує конфіденційність та безпеку інформації. Таким чином, соціальна ефективність веб-системи проявляється у покращенні комунікації, підвищенні рівня задоволеності учасників освітнього процесу та зміцненні позитивного іміджу освітнього закладу.

3 РОЗДІЛ ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

Охорона праці є важливим аспектом будь-якої професійної діяльності, адже вона спрямована на забезпечення здоров'я та безпеки працівників. Оскільки в моїй роботі над дипломом при розробці веб-системи більшість часу я провела за комп’ютером, то це безпосередньо стосується мене, як користувача комп’ютером на персональному робочому місці.

Охорона праці включає в себе комплекс заходів, правил і норм, які спрямовані на запобігання виробничих травм, професійних захворювань та аварій на робочих місцях. Основним завданням охорони праці є створення таких умов, які б дозволили мінімізувати ризики для життя та здоров'я працівників, а також сприяти підвищенню продуктивності праці за рахунок зменшення кількості нещасних випадків і втрат робочого часу. .

3.1 Негативні фактори під час роботи за комп’ютером

Робота за комп'ютером може має негативні чинників та може призвести до проблем, які впливають на здоров'я та самопочуття людини. Однією з головних проблем є нарушення зору, яке може виникнути через тривале зосередження на екрані монітора. Це може спричинити синдром комп'ютерного зору, який зазвичай проявляється у вигляді сухості та подразнення очей, головного болю, зниження гостроти зору та інших факторів. Також, неправильне освітлення робочого місця може посилити ці симптоми, підвищуючи дискомфорт і втому очей.

Основною проблемою є болі у спині, шиї та плечах, які виникають через тривале сидіння за комп’ютером у неправильному положенні. Неправильна висота стільця, недостатня підтримка для спини або неправильне розташування монітора можуть призвести до розвитку хронічних болей у спині, шиї та плечах. Погано відрегульована висота крісла, відсутність опори для спини та неправильне розташування монітора можуть призвести до тривалого дискомфорту в спині, шиї та плечах. Важливо зазначити, що тривале використання комп’ютера може призвести до синдрому зап’ястного каналу,

який виникає через надмірне навантаження на зап’ястя під час друкування або використання миші. Багато офісних працівників ведуть спосіб життя, який передбачає тривале сидіння та відсутність фізичних вправ, що може призвести до різних захворювань, таких як надмірна вага, проблеми з серцем та інші проблеми зі здоров’ям.

Для запобігання цих негативних чинників слід дотримуватися ергономічних рекомендацій щодо облаштування робочого місця, регулярно робити перерви для відпочинку очей, змінювати положення сидячи, а також займатися фізичною активністю, щоб підтримувати загальний фізичний стан.

3.2 Гігієнічні вимоги до виробництва

Відповідно дипломному проєктуванню робоче місце знаходиться вдома, розглянемо як правильно його облаштувати згідно правил охорони праці. Для забезпечення здорових і безпечних умов праці необхідно дотримуватися ряду гігієнічних вимог до виробничого середовища. До основних аспектів належать приміщення, мікроклімат, освітлення, рівень шуму, робоче місце та електробезпека. Ці вимоги спрямовані на збереження здоров'я працівника та підвищення продуктивності праці. .

3.2.1 Приміщення

Робоче місце вдома повинно бути облаштоване у спеціально відведеній кімнаті або в окремій частині приміщення. Приміщення має бути достатньо просторим, щоб вміщати необхідні меблі та обладнання, при цьому залишаючи місце для вільного пересування. Площу приміщень, в яких розташовують персональні комп’ютери, визначають згідно з чинними нормативними документами. Відповідно до них з розрахунку на одне робоче місце, обладнане ПК, встановлено такі норми: площа – не менше 6,0 кв. м; об’єм – не менше 20,0 куб. м.

3.2.2 Освітлення робочих місць

Робочі місця, згідно санітарним нормам, слід розташовувати відносно світлових прорізів так, щоб природне світло падало переважно з лівого боку. Робоче місце необхідно розміщувати таким чином, щоб уникнути попадання прямого світла в очі, натомість вікна повинні забезпечувати рівномірне розсіювання світла по всій робочій поверхні.

Штучне освітлення приміщення має здійснюватися системою загального рівномірного освітлення. У приміщеннях при переважній роботі з документами допускається використання системи комбінованого освітлення, тобто встановлення світильників місцевого освітлення додатково до загального. Згідно норм охорони праці, система загального освітлення має бути у вигляді суцільних або переривчатих ліній світильників, що розташовані збоку від робочих місць паралельно лінії зору працівників..

3.2.3 Робоче місце

Правильна організація робочого місця є ключовою для збереження здоров'я працівника. На рис. 3.1 зображені основні нормативні розміри для робочого місця.

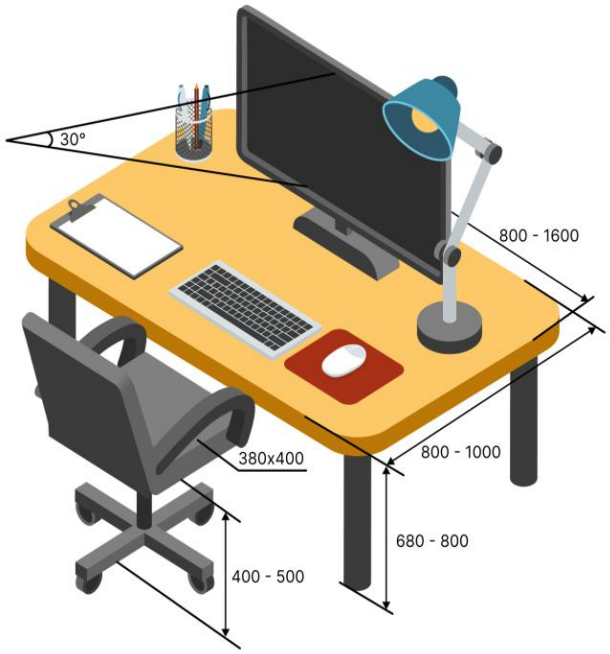


Рисунок 3.1. Основні нормативні розміри для робочого місця

Робочий стіл має відповідати ергономічним стандартам, що забезпечують оптимальне розташування обладнання (монітора, клавіатури) та документів. Висота стільниці повинна регулюватися в межах 680-800 мм, а її ширина та глибина мають дозволяти виконання всіх необхідних завдань у межах досяжності (рекомендовані розміри: ширина - 600-1400 мм, глибина - 800-1000 мм). Робочий стіл повинен мати достатньо простору для ніг: висота простору для ніг не менше 600 мм, ширина - не менше 500 мм, глибина на рівні колін - не менше 450 мм, на рівні витягнутої ноги - не менше 650 мм.

Робочий стілець повинен бути підйомно-поворотним, з можливістю регулювання висоти та кута нахилу сидіння та спинки. Сидіння має бути рівним, з закругленим переднім краєм. Регулювання кожного параметра має здійснюватися незалежно, легко та надійно фіксуватися. Висота сидіння повинна регулюватися в межах 400-500 мм, ширина і глибина сидіння мають бути не менше 400 мм. Кут нахилу сидіння має регулюватися від 15 вперед до 5 назад. Висота спинки стільця повинна бути 300±20 мм, ширина - не менше 380 мм. Відстань від спинки до переднього краю сидіння має регулюватися в межах 260-400 мм. Для зменшення статичної напруги м'язів рук слід використовувати стаціонарні або знімні підлокітники довжиною не менше 250 мм, шириною 50-70 мм, з регулюванням по висоті над сидінням у межах 230-260 мм, а відстань між підлокітниками повинна бути в межах 350-500 мм. Поверхня сидіння та спинки стільця повинна бути напівм'якою, з нековзким, повітронепроникним покриттям, яке легко чиститься і не електризується.

Монітор комп'ютера повинен бути розташований на рівні очей або трохи нижче, щоб уникнути напруження шиї та очей. Відстань від очей до екрану повинна бути не меншою за 50-70 см. Клавіатура та миша повинні бути розташовані так, щоб руки знаходилися під прямим кутом або трохи більше при роботі.

3.2.4 Мікроклімат

Мікроклімат в приміщенні відіграє важливу роль у забезпеченні комфортних умов праці.

У приміщенні на робочому місці мають забезпечуватись оптимальні значення параметрів мікроклімату: температура повітря повинна становити 22–25°С, відносна вологість повітря повинна бути в межах 40-60%. Занадто сухе повітря може спричинити подразнення слизових оболонок, швидкість руху повітря — не більше 0,1 м/с.

3.2.5 Шум

Рівень шуму на робочому місці повинен бути мінімальним, щоб не відволікати від роботи і не спричиняти стрес. Для зниження шуму можна використовувати шумоізоляційні матеріали, встановити вікна з подвійним склопакетом або користуватися навушниками з шумозаглушенням.

Допустимий еквівалентний рівень шуму для робочого місця оператора складає 65 дБА.

3.2.6 Електробезпека

По завершенню роботи персональний комп'ютер і периферійні пристрої необхідно відключати від електромережі. У разі виникнення аварійної ситуації слід негайно вимкнути комп'ютер і периферійні пристрої від електромережі.

Персональні комп'ютери та периферійні пристрої повинні підключатися до електромережі лише за допомогою справних штекерних з'єднань і розеток заводського виготовлення. Штекерні з'єднання та електричні розетки, крім контактів фазного та нульового робочого проводів, повинні мати спеціальні контакти для підключення нульового захисного проводу. Їх конструкція має забезпечувати підключення нейтрального захисного проводу раніше, ніж фазного і нульового робочих проводів.

Послідовність відключення повинна бути зворотною. Необхідно унеможливити з'єднання контактів фазних проводів з контактами нейтрального захисного проводу. Підключення комп'ютерів і периферійних пристроїв до

звичайної двопровідної мережі електропостачання, включаючи використання перехідних пристроїв, є неприпустимим.

3.3 Пожежна безпека

При робочому місці з комп’ютером удома, існує певна пожежна небезпека, яка може виникнути, якщо ви неправильно використовуєте електричне обладнання. Найпоширенішими причинами пожежі є занадто багато електрики, погані розетки чи шнури, а також недостатньо повітря для комп’ютера через розміщення його в тісному місці або можливість накопичення пилу. Щоб уникнути цих небезпек, потрібно часто оглядати вилки та дроти в розетках, не використовувати обірвані або зношені шнури та не використовувати одну розетку для великої кількості обладнання одночасно.

Щоб уникнути пожежі, необхідно пам’ятати про правила обережного використання комп’ютера. Перед початком роботи слід переконатися, що комп’ютер отримує достатньо повітря, не стоїть його надто близько до стін або речей, які можуть блокувати потік повітря. Тримати свій комп’ютер у чистоті від пилу важливо, оскільки це зменшує ймовірність його перегрівання та займання. При пожежі краще використовувати вуглекислотний вогнегасник типу ВВК, оскільки він може загасити електричну пожежу, не пошкодивши електроніку.

3.4 Висновки з охорони праці

При забезпеченні безпеки праці та пожежної безпеки під час роботи за комп’ютером вдома, дотримання ергономічних вимог відіграє основну роль у створенні відповідного робочого місця. При цьому враховується забезпечення комфортних умов мікроклімату та низький рівень шуму. Такі проблеми зі здоров’ям, як синдром комп’ютерного зору або біль у спині, можна запобігти, якщо правильно розмістити обладнання та робити регулярні перерви для виконання вправ. Крім того, слід дотримуватися правил електробезпеки — правильно підключати штепсельні з’єднання та не забувати від’єднувати комп’ютер від мережі після роботи. З точки зору запобігання пожежі, необхідно

тримати комп'ютер в чистоті від пилу і мати коло робочого місця вогнегасник типу ВВК. Усі ці дії спрямовані на те, щоб зробити робоче місце безпечним і комфортним без шкоди для роботи за комп’ютером вдома.

ВИСНОВКИ

В дипломному проєкті створено Back-End частину автоматизованої веб-системи обліку руху контингенту здобувачів відділення комп'ютерних систем.

Для керування доступами була розроблена система авторизації за електронною поштою та паролем та відповідно для виходу з системи. Всі роути, що відповідають за безпосереднє керування обліком студентів проходять попередню перевірку на авторизацію адміністратора у системі, задля обмеження доступів неавторизованим користувачам. Під цю переврку також було доданий роут, який відповідає за отримання файлів з локального сховища, що гарантує що неавторизований користувач не може отримати доступ до фотографій студентів та інших файлів, які будуть там зберігатись, що забезпечує деяку безпеку інформації.

При створенні Back-End частини веб-системи для відділення комп’ютерних частин, було задіяно Docker для забезпечення портативності та масштабованості застосунку; в якості серверної мови програмування - PHP 8.2, а саме був використаний фреймворк Laravel 10; як веб-сервер - Nginx, який швидко опрацьовує запити та передає їх далі на обробку, а для зберігання інформації був обраний MySQL 8.0.

В пояснювальній записці розглянуті всі питання, які передбачені технічним завданням дипломного проєктування. Проведено аналіз предметної області, детально описано огляд існуючих аналогів та технології, які використовувалися при створені проєкту. Описане технічне завдання та показано основні етапи розробки Front-End частини веб-застосунку. У частині тестування показано роботу тестування основної ефективності роутів через Postman. Проведено розрахунок економічної створення та впровадження веб-системи, розглянуті питання охорони праці та наведений перелік використаних джерел.

ПЕРЕЛІК ВИКОРИСТАНИХ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

1. 12 Найкращих Систем І Програмного Забезпечення Для Управління Школами. squeezegrowth. [Веб-сайт] URL: https://squeezegrowth.com/uk/best-school-management-software-systems/.
2. HTML і CSS: що це, кому та для чого потрібно. goit. [Веб-сайт] URL: https://goit.global/ua/articles/html-i-css-shcho-tse-komu-ta-dlia-choho-potribno/.
3. https://goit.global/ua/articles/shcho-take-javascript-i-dlia-choho-vin-potriben/. goit. [Веб-сайт] URL: https://goit.global/ua/articles/shcho-take-javascript-i-dlia-choho-vin-potriben/.
4. Blade Templates. laravel. [Веб-сайт] URL: https://laravel.com/docs/11.x/blade#supercharging-blade-with-livewire.
5. jQuery API. jQuery. [Веб-сайт] URL: https://api.jquery.com/.
6. CSS with superpowers. sass. [Веб-сайт] URL: https://sass-lang.com/.
7. К. В. Двірничук, Д. О. Вацек. Веб-програмування та веб-дизайн: Навчальний посібник. – «Чернівецький національний університет імені Юрія Федьковича», 2022.
8. Веб-шаблони як компоненти. learn.microsoft. URL: https://learn.microsoft.com/uk-ua/power-pages/configure/web-templates-as-components.
9. В. В. Босько, Л. В. Константинова, К. М. Марченко, О. С. Улічев. Web-програмування. Частина 1 (Frontend): Навчальний посібник. – «Кропивницький центральноукраїнський національний технічний університет», 2022.
10. Laravel Ajax Request Example. medium. URL: https://raviyatechnical.medium.com/laravel-ajax-request-example-10e4aa681cd.

ДОДАТОК А. Програмний код основної логіки веб-застосунку

```
// Ajax-search.js

function performAjaxRequest(fieldId, url, method, callback, resultItemClass = '',
selectedIds = [], triggerLength = 3) {
    let $field = $('#' + fieldId);
    let searchValue = $field.val();
    let $parentForm = $field.closest('.student-form-field');

    if (searchValue.length >= triggerLength) {
        $.ajax({
            url: url,
            method: method,
            data: {
                search: searchValue,
                selected_ids: selectedIds
            },
            success: function (response) {
                var resultList = $parentForm.find('.student-form-search-result-list');
                resultList.empty();

                if (response.length > 0) {
                    $.each(response, function (index, item) {
                        callback(resultList, item);
                    });
                } else {
                    resultList.append('<li class="student-form-search-result-item ' +
resultItemClass + '" disabled>Нічого не знайдено</li>');
                }

                $parentForm.find('.student-form-search-result').addClass('show');
            },
            error: function (xhr, status, error) {
                console.error(error);
            }
        });
    } else {
        $parentForm.find('.student-form-search-result').removeClass('show');
    }
}


// Main.js

/******/ (() => { // webpackBootstrap
    var __webpack_exports__ = {};
    /*!************************!*\
      !*** ./src/js/main.js ***!
      \************************/
// import $, { Callbacks } from 'jquery';

    function fixHeight() {
        var vh = window.innerHeight;
        document.documentElement.style.setProperty('--vh', "".concat(vh, "px"));
    }
    $(document).ready(function () {
```

```
fixHeight();
window.addEventListener('resize', function () {
    fixHeight();
});
$.fn.modalOpen = function () {
    $('.js-modal').modalCloseAll();
    $('body').addClass('is-hidden');
    $(this).fadeIn(1);
    $(this).addClass('is-open');
};
$.fn.modalClose = function () {
    $(this).fadeOut(1);
    $(this).removeClass('is-open');
    $('body').removeClass('is-hidden');
    return this;
};
$.fn.modalCloseAll = function () {
    $('.js-modal').modalClose();
    return this;
};
$(document).on('click', '.js-close-modal', function () {
    $('.js-modal').modalCloseAll();
});
$(document).on('click', '.js-modal-link', function (e) {
    var target = $(e.currentTarget).attr('data-target');
    var $modal = $(".js-modal[data-modal=\"".concat(target, "\"]"));
    if ($modal.length) {
        $modal.modalOpen();
    }
});

if ($('.side').length > 0) {
    $(document).on('click', '.side-heading-name', function () {
        $('.side-overlay').removeClass('hidden');
        $('html').addClass('is-hidden');
    });
    $(document).on('click', '.side-heading-arr', function () {
        $('.side-overlay').addClass('hidden');
        $('html').removeClass('is-hidden');
    });
}

// ============== form add file
if (document.querySelector('.student-form')) {
    var inputFile = document.getElementById('fileInput');
    var selectedFileName = document.getElementById('selectedFileName');
    var selectedFile = document.getElementById('selectedFile');
    var fileInputLabel = document.getElementById('fileInputLabel');
    if (inputFile) {
        inputFile.addEventListener('change', function () {
            fileInputLabel.style.display = 'none';
            selectedFileName.innerHTML = inputFile.files[0].name;
            selectedFile.classList.add('show');
        });
        $('#removeSelectedFile').on('click', function () {
            fileInputLabel.style.display = 'flex';
            selectedFile.classList.remove('show');
            inputFile.value = '';
        });
    }
}
$(document).ready(function () {
```

```
            $('.student-form-field').on('click', '.student-form-search-result-item',
function () {
                var currentField = $(this).closest('.student-form-field');
                currentField.find('.student-form-search-result-item').removeClass('is-
selected');
                $(this).addClass('is-selected');
                var selectedText = $(this).text().trim();
                var selectedId = $(this).data('id');

                if ($(this).hasClass('multiselect')) {
                    var selectedName = $(this).data('name');
                    currentField.find('.search-field').val('');
                    var selectedOptionsList = currentField.find('.student-form-selected-
options');
                    selectedOptionsList.addClass('show')
                    var newOption = $('<li class="student-form-selected-option" data-
selected-id="' + selectedId + '" data-name="'
                        + selectedName +'">' +
                        '<span>' + selectedText + '</span><img
src="/assets/images/icons/cross.svg" alt="cross icon"/>' +
                        '</li>');
                    selectedOptionsList.append(newOption);
                    newOption.find('img').on('click', function() {
                        $(this).parent().remove();
                    });
                } else {
                    currentField.find('.search-field').val(selectedText);
                    currentField.find('.value-field').val(selectedId);
                    currentField.find('.search-field').prop('readonly', true);

                    var clearButton = $('<span class="clear-button">&#10006;</span>');
                    currentField.find('.search-field').after(clearButton);
                    clearButton.on('click', function () {
                        currentField.find('.search-field').val('');
                        currentField.find('.value-field').val('');
                        clearButton.remove();
                        currentField.find('.search-field').prop('readonly', false); //
Corrected line
                    });
                }

                currentField.find('.student-form-search-result-list').empty();
                currentField.find('.student-form-search-result').removeClass('show');
            });
        });

        //  видалення елемента при кликові на img
        $('.student-form-selected-option img').click(function () {
            $(this).parent().remove();
        });
    });
    /******/ })()
;
//# sourceMappingURL=main.js.map


Mask.js
$(document).ready(function () {
    let currentUrl = window.location.href.replace(/^https?:\/\//, '');
    let bestMatchLink = null;
    let bestMatchLength = 0;
    $('.side-list-link').each(function () {
```

```
        let href = $(this).attr('href').replace(/^https?:\/\//, '');
        let commonPrefixLength = 0;
        while (commonPrefixLength < href.length && commonPrefixLength <
currentUrl.length && href[commonPrefixLength] === currentUrl[commonPrefixLength]) {
            commonPrefixLength++;
        }
        if (commonPrefixLength > bestMatchLength) {
            bestMatchLink = $(this);
            bestMatchLength = commonPrefixLength;
        }
    });
    if (bestMatchLink !== null) {
        bestMatchLink.addClass('active');
    }
});

// masks
$(document).ready(function () {
    // phone mask
    let phones = $('.phone_mask');
    if (phones.length) {
        phones.each(function () {
            Inputmask({
                "mask": "+380 (99) 999 99 99",
            }).mask($(this).get(0));
        });
    }

    // email mask
    let emails = $('.email_mask');
    if (emails.length) {
        emails.each(function () {
            Inputmask({
                "alias": "email",
            }).mask($(this).get(0));
        });
    }
```

Page.blade.php

```
<!DOCTYPE html>
<html lang="uk">
@include('layouts._elements._head')

<body class="@yield('body-class', 'home-page')">
<div class="fluid-wrapper">
    <main class="@yield('main-class', 'home-main')">
        @include('layouts._elements.sidebar')
        <section class="@yield('main-section', 'section-main')">
            <div class="container">
                @yield('content')
            </div>
        </section>
    </main>

    @include('layouts._elements.exit-popup')
    <footer class="footer"></footer>
</div>
@include('layouts._elements._footer')
</body>
```

```
</html>

// Sidebar.blade.php

<aside class="side">
    <div class="side-overlay hidden">
        <div class="side-content">
            <div class="side-content-top">
                <div class="side-heading">
                    <h1 class="side-heading-name">
                        ОТФК <br />
                        ОНТУ
                    </h1>
                    <svg
                        width="17"
                        height="16"
                        viewBox="0 0 17 16"
                        fill="none"
                        xmlns="http://www.w3.org/2000/svg"
                        class="side-heading-arr"
                    >
                        <path
                            d="*svg"
                            fill="white"
                        />
                    </svg>
                </div>

                <ul class="side-list">
                    <li class="side-list-item">
                        <a href="{{ route('admin.branch-of-knowledge.list') }}"
class="side-list-link">
                            <svg width="40" height="25" viewBox="0 0 20 20" fill="none"
xmlns="http://www.w3.org/2000/svg">
                                <path d="*svg"/>
                            </svg>
                            <span>Галузі знань</span>
                        </a>
                    </li>

                    <li class="side-list-item">
                        <a href="{{ route('admin.specialties.list') }}" class="side-
list-link">
                            <svg width="40" height="25" viewBox="0 0 20 18" fill="none"
xmlns="http://www.w3.org/2000/svg">
                                <path d="*svg" fill="#0096FF"/>
                            </svg>
                            <span>Спеціальності</span>
                        </a>
                    </li>

                    <li class="side-list-item">
                        <a href="{{ route('admin.educational-programs.list') }}"
class="side-list-link @if(request()->routeIs('educational-programs.list')) active
@endif">
                            <svg width="40" height="25" viewBox="0 0 20 18" fill="none"
xmlns="http://www.w3.org/2000/svg">
                                <path d="*svg" fill="#0096FF"/>
                            </svg>
                            <span>Освітні програми</span>
                        </a>
                    </li>
```

```
                    <li class="side-list-item">
                        <a href="{{ route('admin.benefits.list') }}" class="side-list-
link @if(request()->routeIs('benefits.list')) active @endif">
                            <svg width="40" height="25" viewBox="0 0 20 20" fill="none"
xmlns="http://www.w3.org/2000/svg">
                                <path d="*svg" fill="#0096FF"/>
                            </svg>
                            <span>Пільги</span>
                        </a>
                    </li>

                    <li class="side-list-item">
                        <a href="{{ route('admin.education-institutes.list') }}"
class="side-list-link @if(request()->routeIs('education-institutes.list')) active
@endif">
                            <svg width="40" height="25" viewBox="0 0 20 20" fill="none"
xmlns="http://www.w3.org/2000/svg">
                                <path d="*svg" fill="#0096FF"/>
                            </svg>
                            <span>Навчальні заклади</span>
                        </a>
                    </li>

                    <li class="side-list-item">
                        <a href="{{ route('admin.decree-types.list') }}" class="side-
list-link @if(request()->routeIs('decree-types.list')) active @endif">
                            <svg width="40" height="25" viewBox="0 0 20 20" fill="none"
xmlns="http://www.w3.org/2000/svg">
                                <path d="*svg" fill="#0096FF"/>
                            </svg>
                            <span>Типи наказів</span>
                        </a>
                    </li>

                    <li class="side-list-item">
                        <a href="{{ route('admin.decrees.list') }}" class="side-list-
link">
                            <svg width="40" height="25" viewBox="0 0 20 20" fill="none"
xmlns="http://www.w3.org/2000/svg">
                                <path d="*svg" fill="#0096FF"/>
                            </svg>
                            <span>Накази</span>
                        </a>
                    </li>

                    <li class="side-list-item">
                        <a href="{{ route('admin.positions.list') }}" class="side-list-
link @if(request()->routeIs('positions.list')) active @endif">
                            <svg width="40" height="25" viewBox="0 0 20 20" fill="none"
xmlns="http://www.w3.org/2000/svg">
                                <path d="*svg" fill="#0096FF"/>
                            </svg>
                            <span>Посади</span>
                        </a>
                    </li>

                    <li class="side-list-item">
                        <a href="{{ route('admin.employees.list') }}" class="side-list-
link @if(request()->routeIs('employees.list')) active @endif">
                            <svg width="40" height="25" viewBox="0 0 20 20" fill="none"
xmlns="http://www.w3.org/2000/svg">
```

```
                                <path d="*svg" fill="#0096FF"/>
                            </svg>
                            <span>Співробітники</span>
                        </a>
                    </li>

                    <li class="side-list-item">
                        <a href="{{ route('admin.student-parents.list') }}" class="side-
list-link @if(request()->routeIs('student-parents.list')) active @endif">
                            <svg width="40" height="25" viewBox="0 0 20 20" fill="none"
xmlns="http://www.w3.org/2000/svg">
                                <path d="*svg" fill="#0096FF"/>
                            </svg>
                            <span>Батьки</span>
                        </a>
                    </li>

                    <li class="side-list-item">
                        <a href="{{ route('admin.groups.list') }}" class="side-list-link
@if(request()->routeIs('groups.list')) active @endif">
                            <svg width="40" height="25" viewBox="0 0 20 20" fill="none"
xmlns="http://www.w3.org/2000/svg">
                                <path d="*svg" fill="#0096FF"/>
                            </svg>
                            <span>Групи</span>
                        </a>
                    </li>

                    <li class="side-list-item">
                        <a href="{{ route('admin.students.list') }}" class="side-list-
link">
                            <svg width="40" height="25" viewBox="0 0 20 20" fill="none"
xmlns="http://www.w3.org/2000/svg">
                                <path d="*svg" fill="#0096FF"/>
                            </svg>
                            <span>Студенти</span>
                        </a>
                    </li>
                </ul>
            </div>

            <div class="side-content-bottom">
                <ul class="side-list">
                    {{--<li class="side-list-item">
                        <button type="button" class="side-list-btn">
                            <svg
                                width="23"
                                height="26"
                                viewBox="0 0 23 26"
                                fill="none"
                                xmlns="http://www.w3.org/2000/svg"
                            >
                                <path
                                    d="*svg"
                                    fill="#0096FF"
                                />
                            </svg>
                            <span>Обліковий запис</span>
                        </button>
                    </li>--}}

                    {{--<li class="side-list-item">
```

```
                        <button type="button" class="side-list-btn">
                            <svg
                                width="25"
                                height="26"
                                viewBox="0 0 25 26"
                                fill="none"
                                xmlns="http://www.w3.org/2000/svg"
                            >
                                <path
                                    d="*svg"
                                    fill="#0096FF"
                                />
                            </svg>

                            <span>Налаштування</span>
                        </button>
                    </li>--}}

                    <li class="side-list-item">
                        <button
                            type="button"
                            class="side-list-btn js-modal-link"
                            data-target="exit-popup"
                        >
                            <svg width="30" height="30" viewBox="0 0 26 28" fill="none"
xmlns="http://www.w3.org/2000/svg">
                                <path d="*svg" fill="#0096FF"/>
                            </svg>
                            <span>Вийти</span>
                        </button>
                    </li>
                </ul>
            </div>
        </div>
    </div>
</aside>
Exit-popup/blade.php

<div class="modal modal--sm js-modal" data-modal="exit-popup">
    <div class="modal-overlay js-close-modal"></div>
    <div class="popup-content">
        <h3 class="popup-title">Ви точно хочете вийти?</h3>

        <div class="popup-btn-wrapper">
            <form action="{{ route('admin.logout') }}" method="post">
                @csrf
                @method('post')
                <button type="submit" class="popup-btn-exit">Вийти</button>
            </form>
            <div class="popup-btn-stay">Залишитись</div>
        </div>
    </div>
</div>

// Pagination.blade.php

<div class="pagination-wrapper">
    @if ($paginator->previousPageUrl())
        <a href="{{ $paginator->previousPageUrl() }}" class="pagination-arr">
            <svg width="34" height="32" viewBox="0 0 34 32" fill="none"
xmlns="http://www.w3.org/2000/svg">
                <path
```

```
                    d="M10.293 15.2937C9.90234 15.6843 9.90234 16.3187 10.293
16.7093L15.293 21.7093C15.6836 22.0999 16.318 22.0999 16.7086 21.7093C17.0992 21.3187
17.0992 20.6843 16.7086 20.2937L13.4117 16.9999H22.9992C23.5523 16.9999 23.9992 16.553
23.9992 15.9999C23.9992 15.4468 23.5523 14.9999 22.9992 14.9999H13.4148L16.7055
11.7062C17.0961 11.3155 17.0961 10.6812 16.7055 10.2905C16.3148 9.8999 15.6805 9.8999
15.2898 10.2905L10.2898 15.2905L10.293 15.2937Z"
                    fill="#1E3050"/>
            </svg>
        </a>
    @endif

    @if($paginator->total() > $paginator->perPage())
        <ul class="paginstion-list">
            @for ($i = 1; $i <= $paginator->lastPage(); $i++)
                @if ($i <= 1 || $i > $paginator->lastPage() - 1 || ($i >= $paginator-
>currentPage() - 1 && $i <= $paginator->currentPage() + 1))
                    <li class="paginstion-item @if ($paginator->currentPage() === $i)
current @endif">
                        <a href="{{ $paginator->appends(request()->query())-
>onEachSide(10)->url($i) }}">{{ $i }}</a>
                    </li>
                    @php
                        $dots = false;
                    @endphp
                @else
                    @if (!$dots)
                        <li class="paginstion-item">
                            <span>...</span>
                        </li>
                        @php
                            $dots = true;
                        @endphp
                    @endif
                @endif
            @endfor
        </ul>
    @endif

    @if ($paginator->nextPageUrl())
        <a href="{{ $paginator->nextPageUrl() }}" class="pagination-arr">
            <svg width="34" height="32" viewBox="0 0 34 32" fill="none"
xmlns="http://www.w3.org/2000/svg">
                <path
                    d="M23.7054 15.2937C24.0961 15.6843 24.0961 16.3187 23.7054
16.7093L18.7054 21.7093C18.3148 22.0999 17.6804 22.0999 17.2898 21.7093C16.8992 21.3187
16.8992 20.6843 17.2898 20.2937L20.5867 16.9999H10.9992C10.4461 16.9999 9.99919 16.553
9.99919 15.9999C9.99919 15.4468 10.4461 14.9999 10.9992 14.9999H20.5836L17.2929
11.7062C16.9023 11.3155 16.9023 10.6812 17.2929 10.2905C17.6836 9.8999 18.3179 9.8999
18.7086 10.2905L23.7086 15.2905L23.7054 15.2937Z"
                    fill="#1E3050"/>
            </svg>
        </a>
    @endif
</div>

// Message.blade.php

@props(['type'])
@php
    $bgColor = match($type) {
        'success' => '#68D34D',
        'error' => '#E34F4F',
```

```
        'warning' => '#FFBE15',
        default => '#FFBE15',
    };
@endphp
<div class="alert-message" style="background-color: {{ $bgColor }}; padding: 5px 10px
5px 10px; padding-right: 30px; border-radius: 7px; margin-bottom: 20px; display: inline-
block; position: relative;">
    {{ $slot }}
    <span class="close-button" style="cursor: pointer; position: absolute; top: 5px;
right: 10px;">×</span>
</div>
@section('js')
    <script>
        $(document).ready(function () {
            const alertMessages = $('.alert-message');
            alertMessages.each(function () {
                const message = $(this);
                setTimeout(function () {
                    message.fadeOut();
                }, 10000);
            });

            $('.close-button').on('click', function () {
                $(this).parent().fadeOut();
            });
        });
    </script>
@stop
```

ДОДАТОК Б. Слайди мультимедійної презентації

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

КГ.07.07.000.ДП

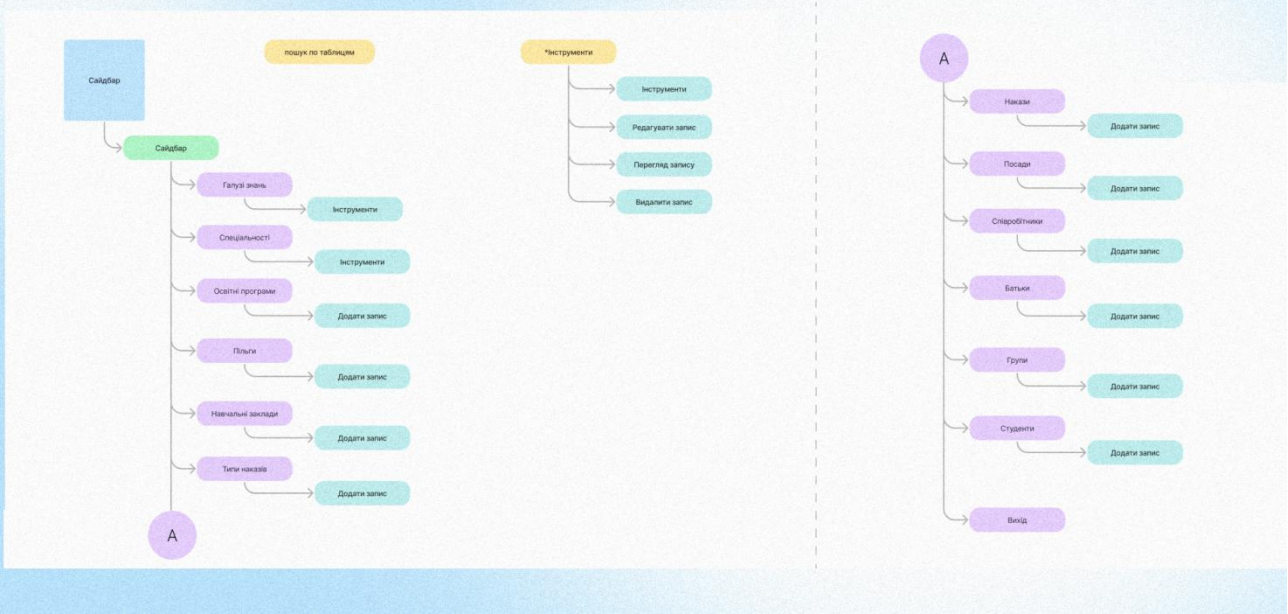
ДИПЛОМНИЙ ПРОЄКТ

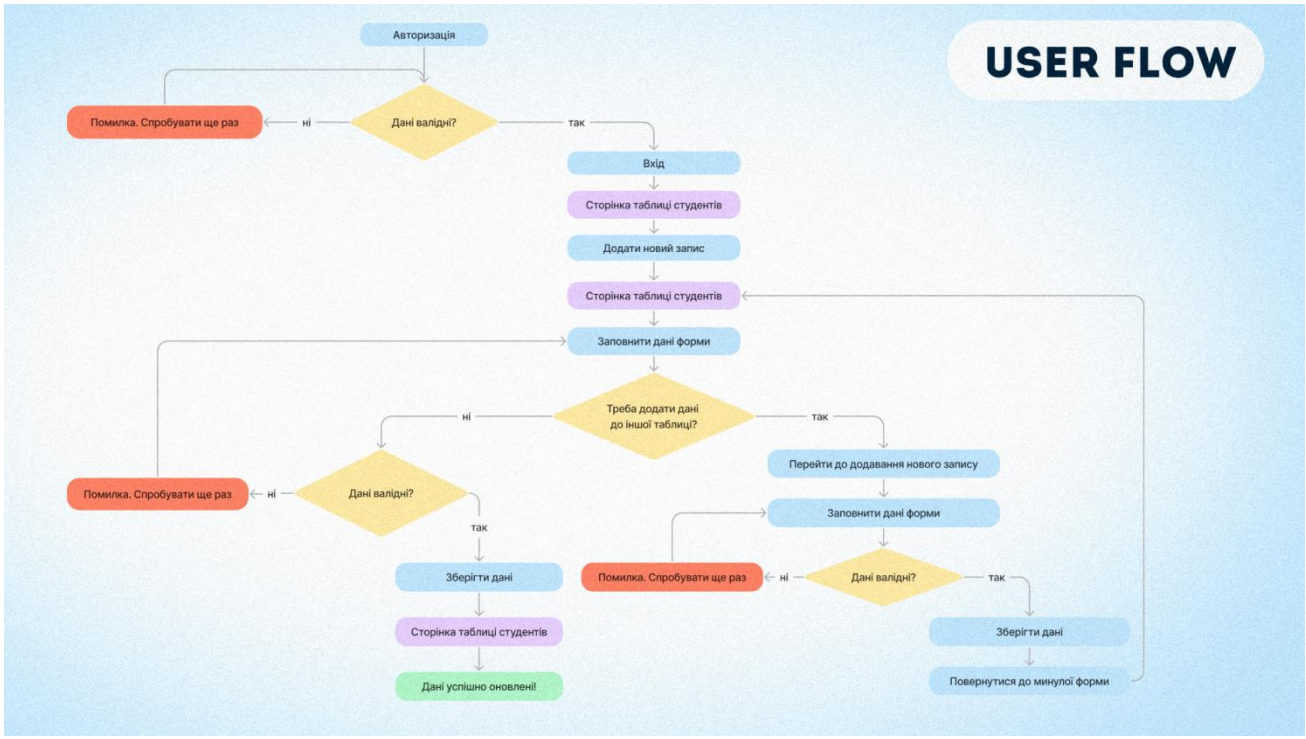
На тему:

Розробка Front-End частини автоматизованої веб-системи обліку руху контингенту здобувачів відділення комп'ютерних систем

Майстренко Дар'ї Олександрівни
Спеціальність: 123 «Комп'ютерна інженерія»
Освітньо-професійна програма: «Комп'ютерна графіка і Web-дизайн»

ДИЗАЙН АРХІТЕКТУРА ВЕБ-СИСТЕМИ




USER FLOW
Авторизація
Дані валідні?
ні
Помилка. Спробувати ще раз
так
Вхід
Сторінка таблиці студентів
Додати новий запис
Сторінка таблиці студентів
Заповнити дані форми
Треба додати дані до іншої таблиці?
ні
так
Дані валідні?
ні
Помилка. Спробувати ще раз
так
Зберігти дані
Сторінка таблиці студентів
Дані успішно оновлені!
Перейти до додавання нового запису
Заповнити дані форми
Дані валідні?
ні
Помилка. Спробувати ще раз
так
Зберігти дані
Повернутися до минулої форми

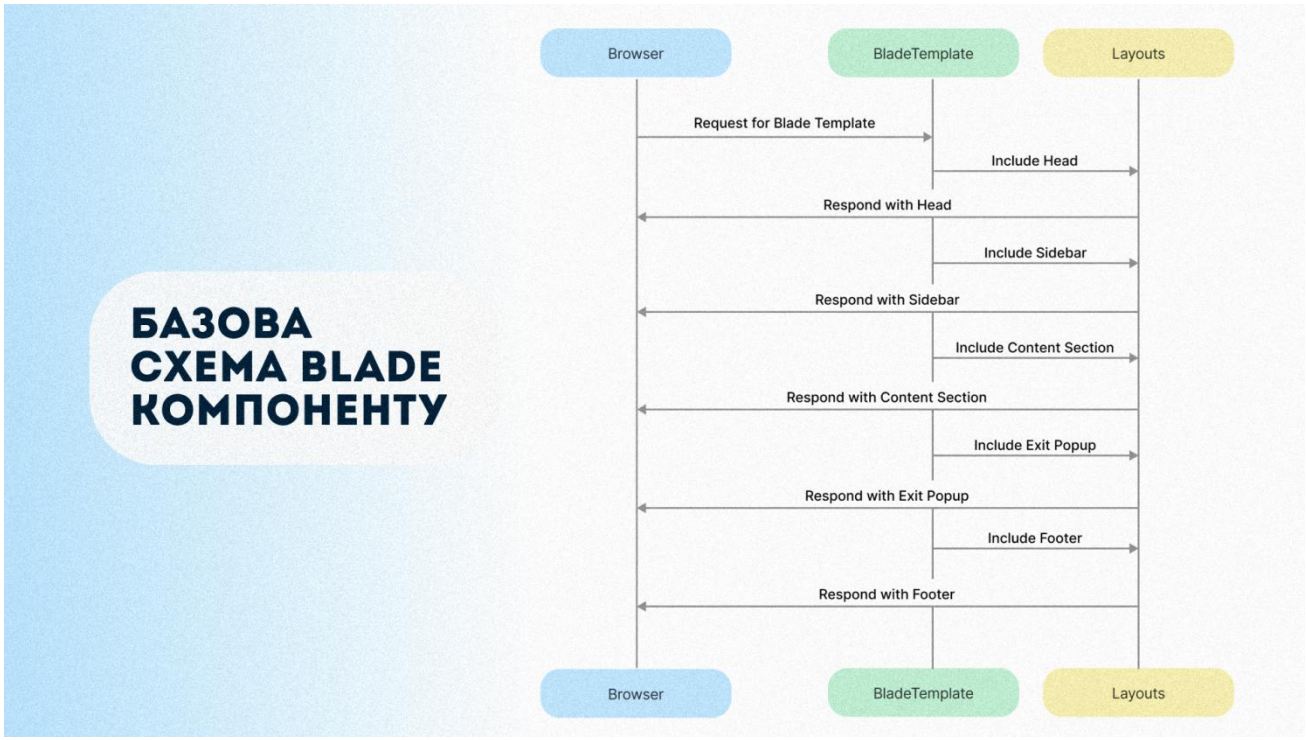
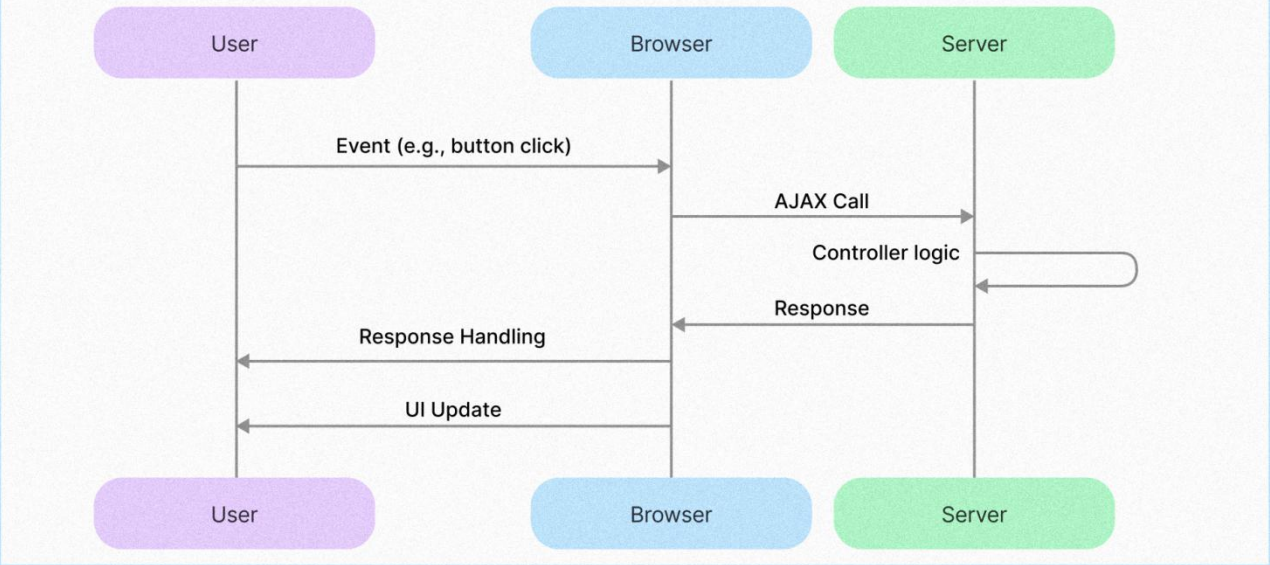
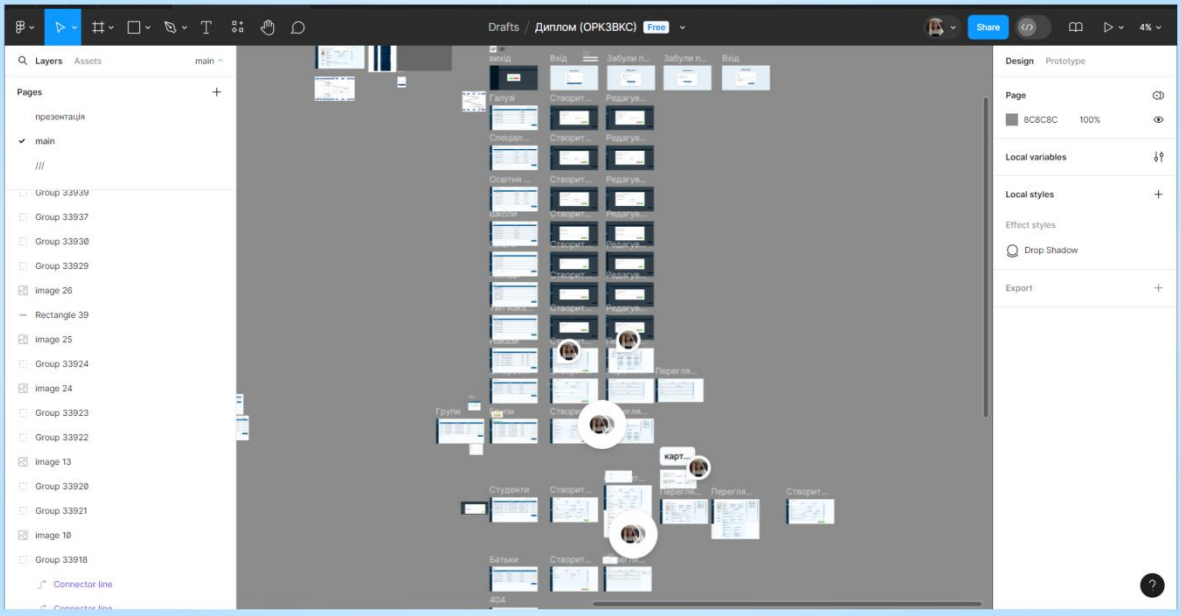

БАЗОВА СХЕМА BLADE КОМПОНЕНТУ
Browser
BladeTemplate
Layouts
Request for Blade Template
Include Head
Respond with Head
Include Sidebar
Respond with Sidebar
Include Content Section
Respond with Content Section
Include Exit Popup
Respond with Exit Popup
Include Footer
Respond with Footer
Browser
BladeTemplate
Layouts

СХЕМА AJAX ЗАПИТІВ



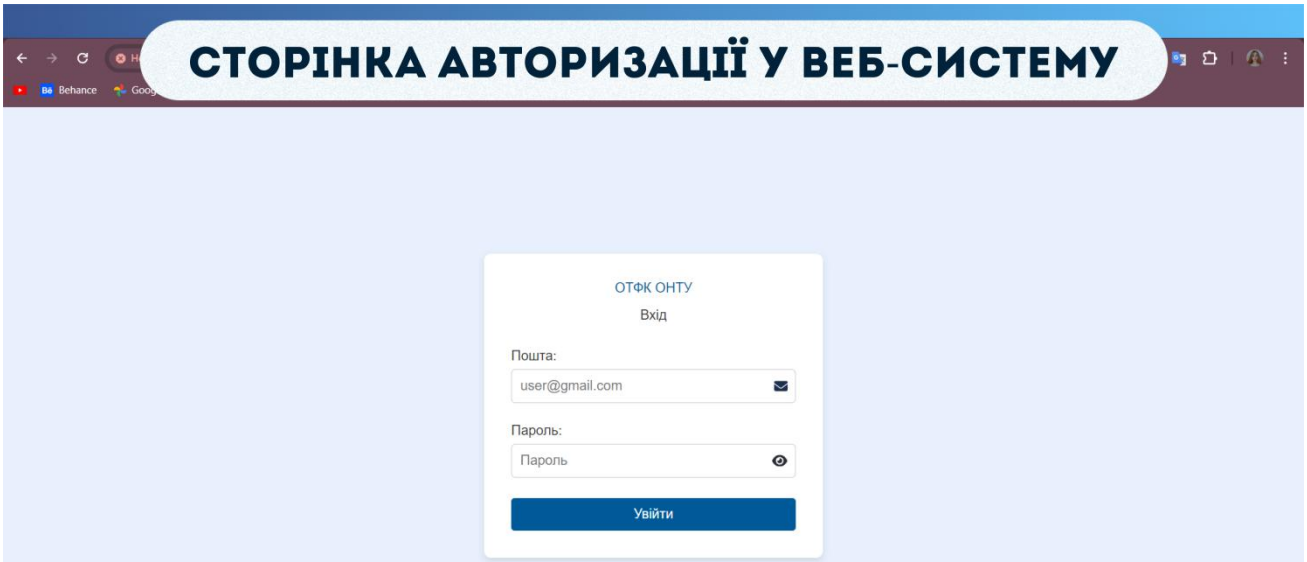
РОЗРОБКА ДИЗАЙНУ ВЕБ-СИСТЕМИ У FIGMA

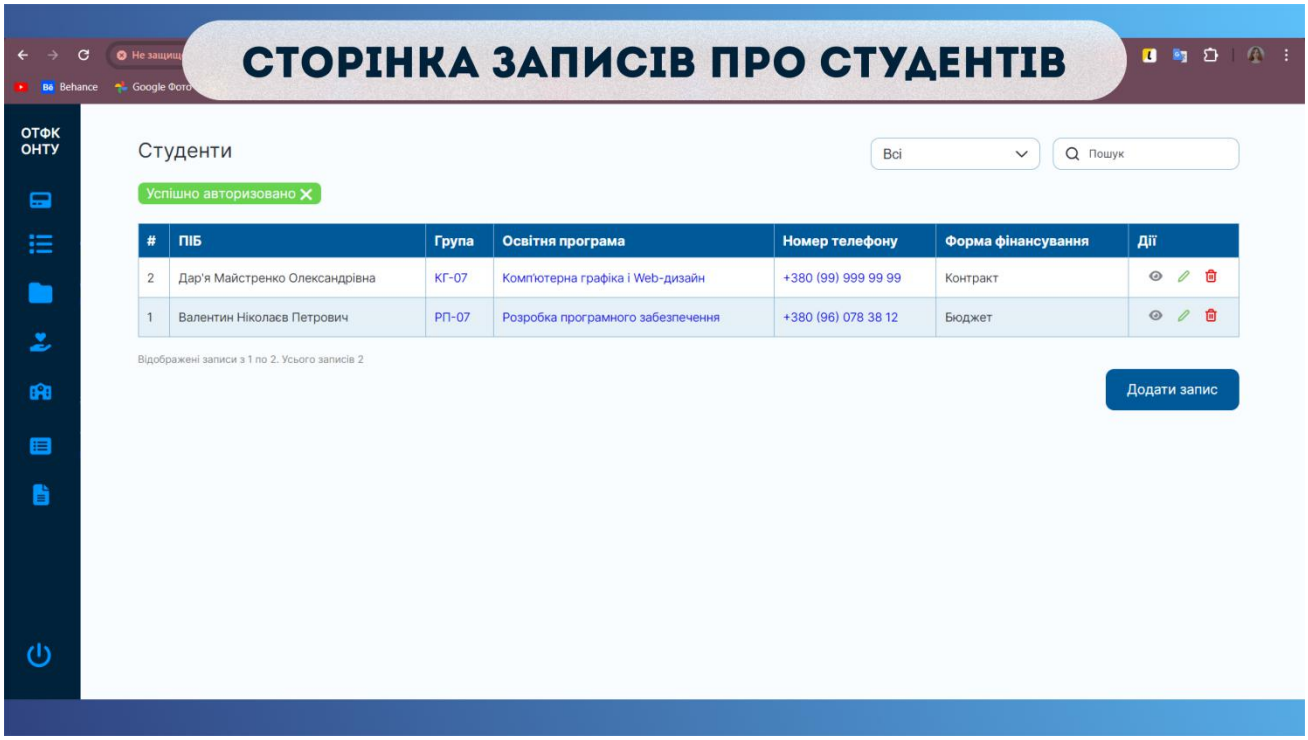


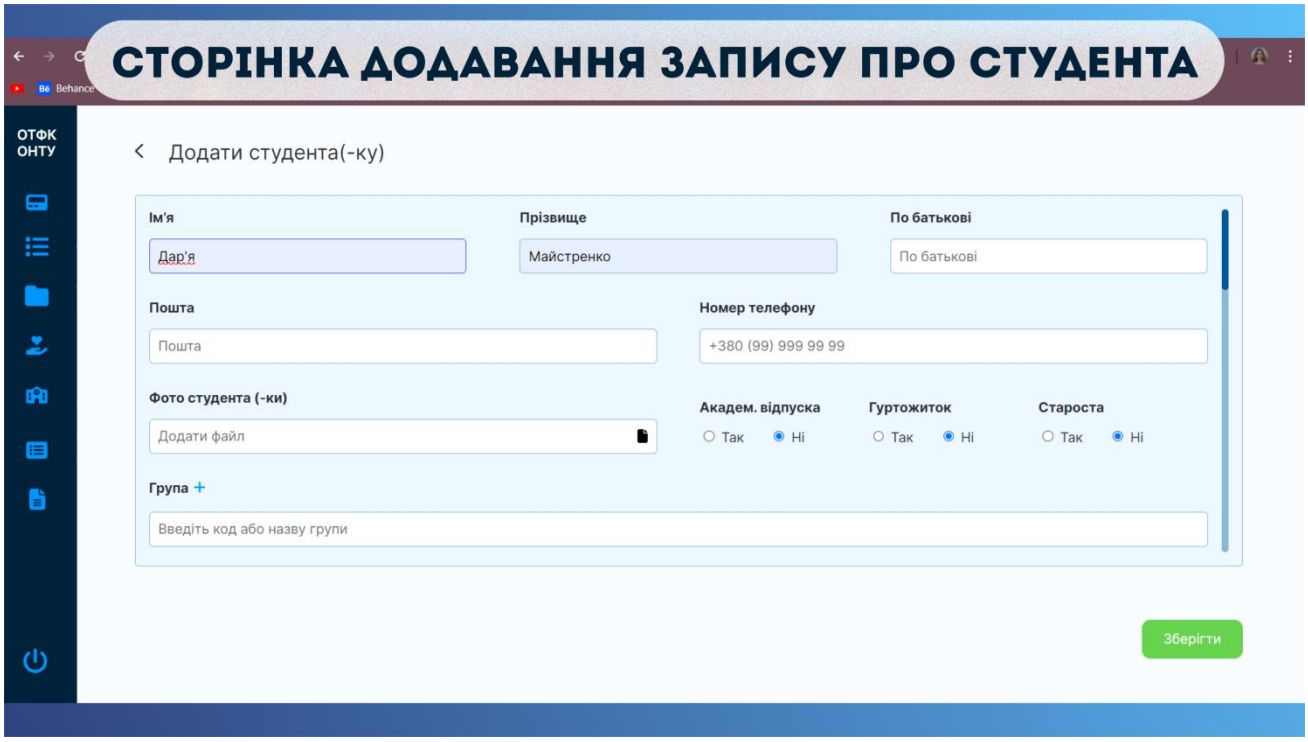
УНІВЕРСАЛЬНИЙ AJAX ЗАПИТ НА ПОШУК

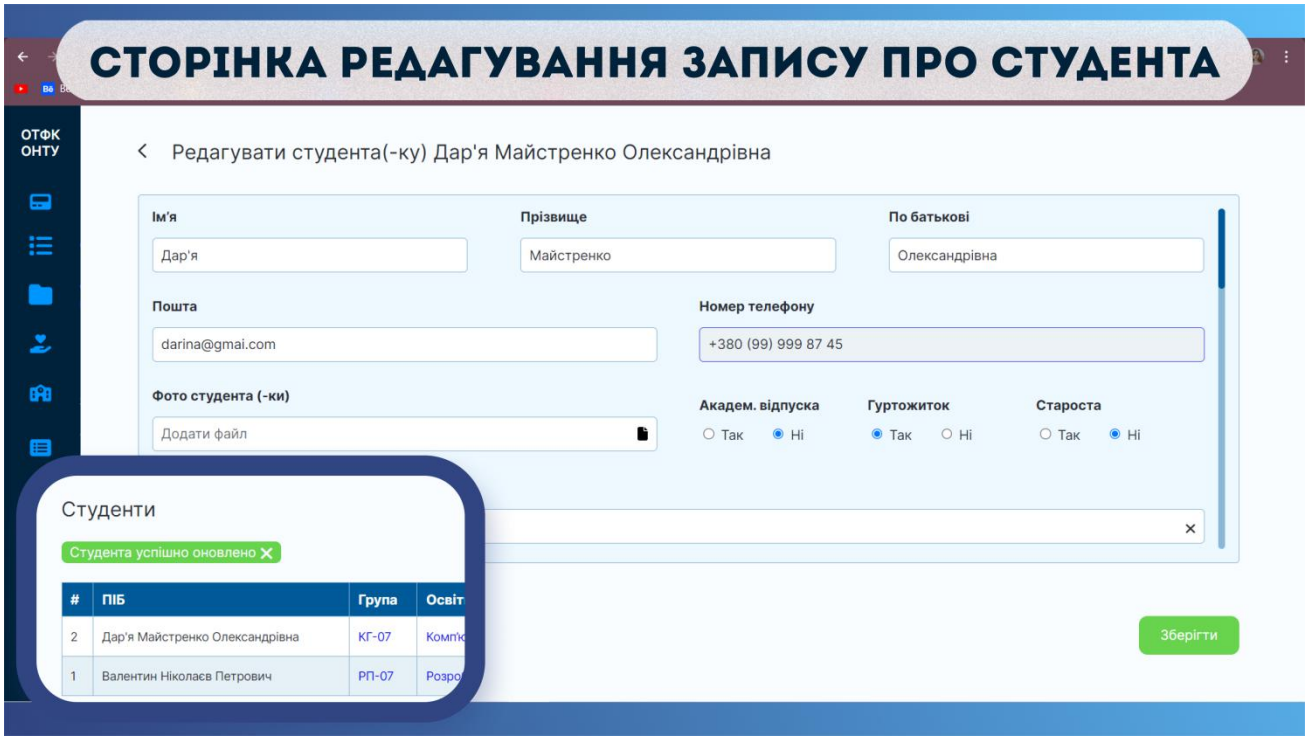
```
function performAjaxRequest(fieldId, url, method, callback, resultItemClass = '', selectedIds = [], triggerLength = 3) {
let $field = $('#' + fieldId);
let searchValue = $field.val();
let $parentForm = $field.closest('.student-form-field');

if (searchValue.length >= triggerLength) {
$.ajax({
       url: url,
method: method,
       data: {
            search: searchValue,
            selected_ids: selectedIds
       },
       success: function (response) {
            var resultList = $parentForm.find('.student-form-search-result-list');
                resultList.empty();

                if (response.length > 0) {
                    $.each(response, function (index, item) {
                        callback(resultList, item);
                    });
                } else {
                    resultList.append('<li class="student-form-search-result-item ' + resultItemClass + '" disabled>Нічого не знайдено</li>');
                }

                $parentForm.find('.student-form-search-result').addClass('show');
            },
            error: function (xhr, status, error) {
                console.error(error);
            }
        });
    } else {
        $parentForm.find('.student-form-search-result').removeClass('show');
    }
}
```

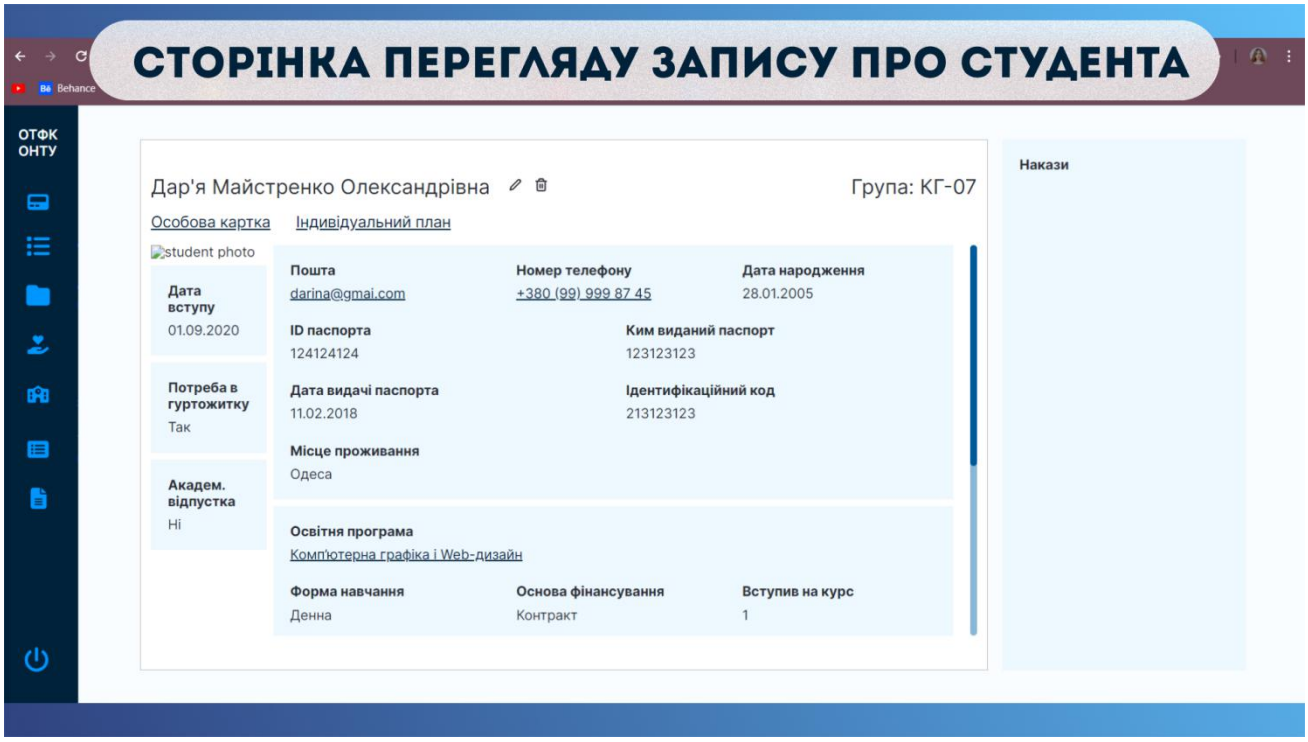
СТОРІНКА АВТОРИЗАЦІЇ У ВЕБ-СИСТЕМУ

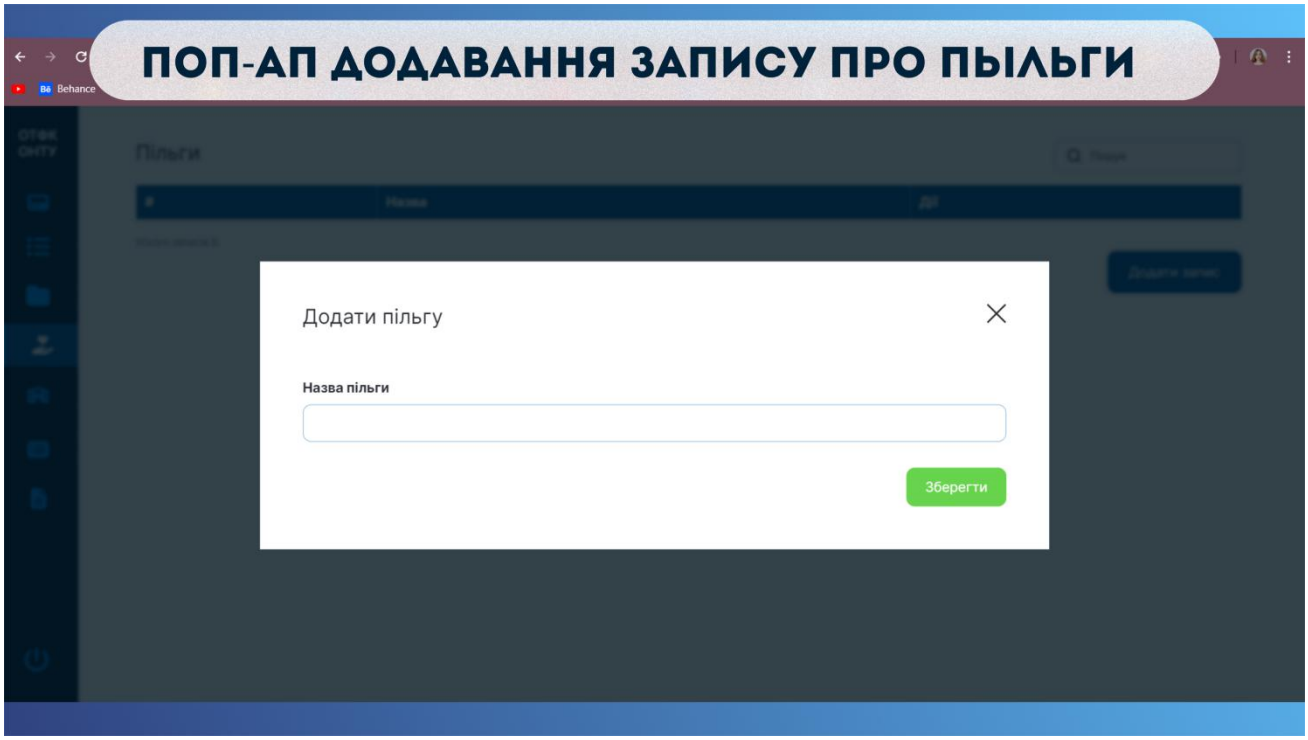


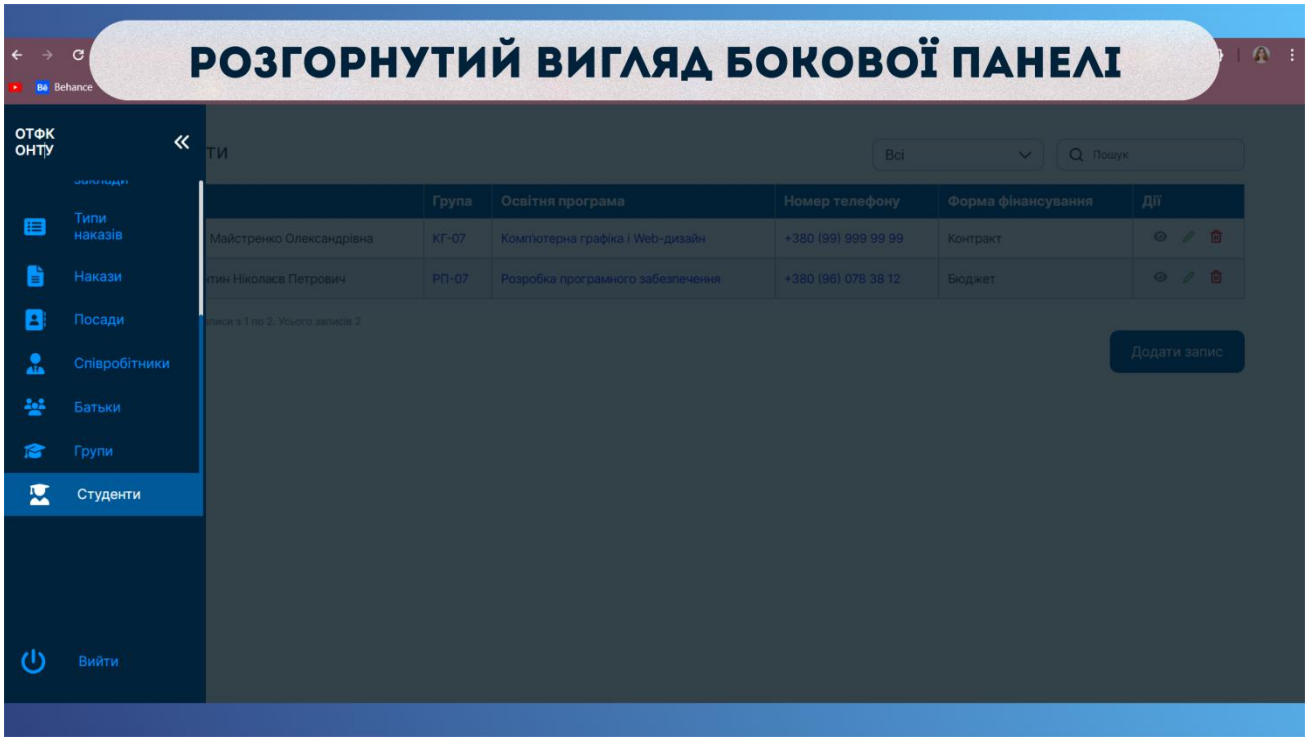

СТОРІНКА ЗАПИСІВ ПРО СТУДЕНТІВ
ОТФК ОНТУ
Студенти
Всі
Пошук
Успішно авторизовано
ПІБ Група Освітня програма Номер телефону Форма фінансування Дії
2 Дар'я Майстренко Олександрівна КГ-07 Комп'ютерна графіка і Web-дизайн +380 (99) 999 99 99 Контракт
1 Валентин Ніколаєв Петрович РП-07 Розробка програмного забезпечення +380 (96) 078 38 12 Бюджет
Відображені записи з 1 по 2. Усього записів 2
Додати запис


СТОРІНКА ДОДАВАННЯ ЗАПИСУ ПРО СТУДЕНТА
ОТФК ОНТУ
Додати студента(-ку)
Ім'я
Дар'я
Прізвище
Майстренко
По батькові
По батькові
Пошта
Пошта
Номер телефону
+380 (99) 999 99 99
Фото студента (-ки)
Додати файл
Академ. відпуска
Так Ні
Гуртожиток
Так Ні
Староста
Так Ні
Група +
Введіть код або назву групи
Зберігти


СТОРІНКА РЕДАГУВАННЯ ЗАПИСУ ПРО СТУДЕНТА
ОТФК ОНТУ
Редагувати студента(-ку) Дар'я Майстренко Олександрівна
Ім'я
Дар'я
Прізвище
Майстренко
По батькові
Олександрівна
Пошта
darina@gmai.com
Номер телефону
+380 (99) 999 87 45
Фото студента (-ки)
Додати файл
Академ. відпуска
Так
Ні
Гуртожиток
Так
Ні
Староста
Так
Ні
Студенти
Студента успішно оновлено
ПІБ Група Освіт
2 Дар'я Майстренко Олександрівна КГ-07
1 Валентин Ніколаєв Петрович РП-07
Зберігти


СТОРІНКА ПЕРЕГЛЯДУ ЗАПИСУ ПРО СТУДЕНТА
ОТФК ОНТУ
Дар'я Майстренко Олександрівна
Група: КГ-07
Особова картка
Індивідуальний план
student photo
Дата вступу
01.09.2020
Потреба в гуртожитку
Так
Академ. відпустка
Ні
Пошта
darina@gmai.com
Номер телефону
+380 (99) 999 87 45
Дата народження
28.01.2005
ID паспорта
124124124
Ким виданий паспорт
123123123
Дата видачі паспорта
11.02.2018
Ідентифікаційний код
213123123
Місце проживання
Одеса
Освітня програма
Комп'ютерна графіка і Web-дизайн
Форма навчання
Денна
Основа фінансування
Контракт
Вступив на курс
1
Накази


ПОП-АП ДОДАВАННЯ ЗАПИСУ ПРО ПИЛЬГИ
Behance
Пільги
Додати пільгу
Назва пільги
Зберегти


РОЗГОРНУТИЙ ВИГЛЯД БОКОВОЇ ПАНЕЛІ
Behance
ОТФК ОНТУ
Типи наказів
Накази
Посади
Співробітники
Батьки
Групи
Студенти
Вийти
Всі
Пошук
Група
Освітня програма
Номер телефону
Форма фінансування
Дії
Майстренко Олександрівна
КГ-07
Комп'ютерна графіка і Web-дизайн
+380 (99) 999 99 99
Контракт
РП-07
Розробка програмного забезпечення
+380 (96) 078 38 12
Бюджет
Додати запис

В РОБОТІ БУЛО ПРОВЕДЕНО ЕКОНОМІЧНИЙ РОЗРАХУНОК:

Загальні витрати (Вз) на створення веб-системи розраховуються за формулою:

Вз = Вр+ Вв+ Ве

Вз = Вр+ (Вв+ Ве) = 40 723 грн.

Де **Вр** – витрати на розробку сайту;
Вв – витрати на впровадження сайту;
Ве – витрати на експлуатацію сайту;

Автоматизована веб-система обліку руху контингенту здобувачів не буде приносити дохід, оскільки це не комерційний проєкт. Однак вона принесе велику користь у контексті роботи відділення комп'ютерних систем:

1. Централізоване та захищене зберігання даних
2. Ефективне управління інформацією
3. Оптимізація технологічних процесів
4. Покращення комунікації
5. Підвищення продуктивності
6. Формування позитивного іміджу

В РОЗДІЛІ ОХОРОНИ ПРАЦІ БУЛИ РОЗГЛЯНУТІ:

Негативні фактори:

- **Проблеми із зором:** сухість та подразнення очей, головний біль, зниження гостроти зору).
- **Болі у спині, шиї та плечах:** Неправильне положення, недостатня підтримка для спини, неправильне розташування монітора.
- **Синдром зап'ястного каналу:** Надмірне навантаження на зап’ястя під час друкування або використання миші.
- **Інші проблеми зі здоров'ям:** Надмірна вага, проблеми з серцем через тривале сидіння.

Гігієнічні вимоги:

- **Приміщення:** Площа не менше 6 кв.м на робоче місце).
- **Освітлення:** Природне світло з лівого боку, рівномірне розсіювання штучного світла.
- **Робоче місце:** Ергономічний стіл і стілець, правильне розташування монітора та клавіатури.
- **Мікроклімат:** Температура 22-25°С, вологість 40-60%, швидкість руху повітря не більше 0,1 м/с.
- **Шум:** Мінімальний рівень шуму (до 65 дБА), використання шумоізоляційних матеріалів.
- **Електробезпека:** Відключення обладнання після роботи, використання справних штекерних з'єднань.

Пожежна безпека:

- **Причини пожежі:** Перевантаження електромережі, погані розетки, недостатнє повітря для комп'ютера, накопичення пилу.
- **Профілактика:** Огляд вилок та проводів, використання якісного обладнання, забезпечення вентиляції комп'ютера, регулярне очищення від пилу.
- **Гасіння пожежі:** Використання вуглекислотного вогнегасника типу ВВК для електричних пожеж.

ДЯКУЮ ЗА УВАГУ!

Чи є якісь питання у шановної комісій?

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

ВІДГУК

керівника на дипломний проект здобувача (здобувачки) освіти

відділення комп'ютерних систем

Майстренко Дар'ї Олександрівни

(прізвище, ім'я та по батькові)

Спеціальність: _____

Освітня програма: *123 «Комп'ютерна інженерія»*

«Комп'ютерна графіка і Web-дизайн»

Тема дипломного проекту: *Розробка Front-End частини автоматизованої веб-системи обліку руху контингенту здобувачів відділення комп'ютерних систем*

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ

а) обсяг і якість виконання проекту (графічного матеріалу і розрахунково-пояснювальної записки) *Дипломний проект виконано відповідно технічному завданню. Пояснювальна записка до дипломного проекту містить 85 сторінок. У пояснювальній записці описано етапи розробки Front-End частини автоматизованої веб-системи обліку руху контингенту здобувачів відділення комп'ютерних систем засобами HTML/CSS/JS/PHP. Графічна частина складається з окремих слайдів, оформлених у вигляді презентації, передбачених технічним завданням. Якість виконання пояснювальної записки та слайдів відмінна.*

б) самостійність роботи над проектом: *Протягом виконання дипломного проекту здобувачка освіти Майстренко Дар'я поступово та послідовно виконувала всі етапи, проявляла ініціативу в створенні загальної концепції та реалізації роботи. Всі роботи здобувачка освіти виконувала самостійно, з оглядом на рекомендації керівника.*

в) теоретична підготовка випускника (випускниці): *Здобувачка освіти Майстренко Дар'я під час роботи над дипломним проектом вивчила достатньо багато літературних та інтернет-джерел за даною тематикою. Вважаю, що теоретична підготовка дипломниці достатня і він готовий до захисту проекту.*

г) вміння розв'язувати виробничі та конструкторські питання *Під час виконання дипломного проекту здобувачка освіти Майстренко Дар'я показала вміння організовано працювати над поставленим завданням, застосовувати знання у галузі програмування та математики, розробляти, встановлювати та налаштовувати спеціалізоване програмне забезпечення, оформлювати слайди та складати презентації, користуючись сучасними комп'ютерними програмними засобами, такими як MS VS Code, jQuery, Microsoft PowerPoint, Microsoft Visio та ін.*

Оцінка розрахункової частини *Відмінно*

Оцінка графічної частини *Відмінно*

Загальна оцінка *Відмінно*

Прізвище, ім'я, по батькові керівника дипломного проекту *Жадан Артур Сергійович*

Місце роботи і посада керівника дипломного проекту *ВСП «Одеський технічний фаховий коледж ОНТУ», викладач спецдисциплін циклової комісії комп'ютерної техніки та програмної інженерії*

Підпис

«10» червня 2024 р.

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

РЕЦЕНЗІЯ

на дипломний проект здобувача (здобувачки) освіти

відділення комп'ютерних систем

Майстренко Дар'ї Олександрівни

(прізвище, ім`я та по батькові)

Спеціальність *123 «Комп'ютерна інженерія»*

Освітня програма *«Комп'ютерна графіка і Web-дизайн»*

Керівник дипломного проекту (роботи) *Жадан Артур Сергійович*

(прізвище, ім`я та по батькові)

Тема дипломного проекту (роботи) *Розробка Front-End частини автоматизованої веб-системи обліку руху контингенту здобувачів відділення комп'ютерних систем*

Обсяг розрахунково-пояснювальної записки *85* сторінок

Обсяг графічної (презентаційної) частини *17* аркушів (слайдів)

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ (РОБОТИ)

а) заключення про ступінь відповідності виконаного дипломного проекту завданню

Представлений на рецензію дипломний проект відповідає затвердженій темі та виконаний відповідно технічному завданню. Дипломний проект присвячений проблемі обліку руху контингенту здобувачів на відділені та складається з пояснювальної записки, додатку з програмним кодом та мультимедійної презентації, що містить приклади роботи програми.

б) характеристика виконання кожного розділу дипломного проекту

Пояснювальна записка складається з основного розділу (аналізу предметної області, проектування застосунку, реалізації застосунку, тестування застосунку), економічного розділу, розділу охорони праці та додатків. Перелічені розділи поетапно охоплюють розробку, виконані докладно та обґрунтовано. Розділ охорони праці містить загальну інформацію та вимоги до техніки безпеки оператора КТ. Економічний розділ проекту містить розрахунок витрат на НДР та реалізацію проекту.

в) оцінка якості виконання пояснювальної записки та графічної частини дипломного проекту

Графічна частина складається з 17 слайдів мультимедійної презентації, виконаної у програмному продукті MS PowerPoint, які містять ілюстративні схеми, скріншоти роботи програмного застосунку, передбачені технічним завданням. Пояснювальна записка виконана акуратно та у відповідності до норм. Якість виконання графічної частини проекту та пояснювальної записки відмінна, розробку виконано у повному обсязі.

г) перелік позитивних якостей дипломного проекту *Реалізовано Front-End частину автоматизованої веб-системи, що дозволяє зручно взаємодіяти з веб-системою завдяки асинхронним запитам.*

Стилізація графічного інтерфейсу відповідає тематиці обраної предметної області.

Розроблена веб-система готова до впровадження.

д) основні недоліки дипломного проекту

Front-End частина автоматизованої веб-системи оброблює лише основні запити асинхронно.

Незважаючи на тему дипломної роботи, для цілісності сприйняття матеріалу пояснювальної записки рекомендовано додати розділ про Back-End частину.

Оцінка розрахункової частини *Відмінно*

Оцінка графічної частини *Відмінно*

Загальна оцінка *Відмінно*

Прізвище, ім'я, по батькові рецензента *Василіу Євген Вікторович*

Місце роботи і посада рецензента *Державний університет інтелектуальних технологій і зв'язку, д.т.н., проф. кафедри КБ та ТЗІ*

Підпис

«14» *червня* 2024 р.

UNICHECK
by Turnitin

Ім'я користувача:
Катерина Григоріївна Краснокутська

ID перевірки:
1016329110

Дата перевірки:
06.06.2024 19:46:46 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
06.06.2024 19:49:06 EEST

ID користувача:
100011688

Назва документа: **4КГ-07_Майстренко_Д**

Кількість сторінок: **51** Кількість слів: **7950** Кількість символів: **60318** Розмір файлу: **4.40 MB** ID файлу: **1016128576**

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

3.96%
Схожість

Найбільша схожість: 0.92% з Інтернет-джерелом (https://swissbeauty.in/blogs/news/why-is-it-important-to-have-cruelty.

3.96% Джерела з Інтернету | **575** ..... Сторінка 53

Не знайдено джерел з Бібліотеки

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Підозріле форматування | **12 сторінок**

ДОЗВІЛ
НА РОЗМІЩЕННЯ
ВИПУСКНОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
(ДИПЛОМНОГО ПРОЕКТУ)
В ЕЛЕКТРОННОМУ РЕПОЗИТАРІЇ ВСП «ОТФК ОНТУ»

Ми, що нижче підписалися,

Майстренко Дар'я Олександрівна,
здобувачка освіти гр. 4КГ-07, та

Жадан Артур Сергійович,
керівник дипломного проекту,

не заперечуємо щодо розміщення електронного варіанту пояснювальної записки до дипломного проекту фахового молодшого бакалавра на тему:

«Розробка Front-End частини автоматизованої веб-системи обліку руху контингенту здобувачів відділення комп'ютерних систем» (авторка роботи – Майстренко Д.О., керівник роботи – Жадан А.С.)

виконаного у ВСП «Одеський технічний фаховий коледж Одеського національного технологічного університету» в 2024 році, у повному обсязі в електронному репозитарії ВСП «ОТФК ОНТУ» для вільного доступу через мережу Інтернет.

Несемо відповідальність за ідентичність електронного та друкованого варіантів випускної кваліфікаційної роботи і даємо згоду на обробку персональних даних.

Виконавець _____ / Майстренко Д.О. /

Керівник _____ / Жадан А.С. /

«10» червня 2024 р.