НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Теплоенергетичний факультет
Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено:
Завідувач кафедри
_____ Наталія АУШЕВА
«___»_____2022 р.

Дипломна робота

на здобуття ступеня бакалавра
спеціальності 122 «Комп'ютерні науки»
освітня програма «Комп'ютерний моніторинг та геометричне моделювання процесів і систем»

на тему: «Інформаційна-автоматизована система з обробки даних абітурієнтів»

Виконала:
студентка IV курсу, групи ТР-81
Кваша Катерина Олегівна _____

Керівник:
Ст. викладач,
Колумбет Вадим Петрович _____

Рецензент:

_____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2022

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший

спеціальність 122 «Комп’ютерні науки»

освітня програма «Комп’ютерний моніторинг та геометричне моделювання процесів і систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Наталія АУШЕВА
(підпис)
”___”_____2022р.

ЗАВДАННЯ

на дипломну роботу студенту

Кваші Катерині Олегівні
(прізвище, ім’я, по батькові)

1. Тема роботи _Інформаційна-автоматизована система з обробки даних абітурієнтів_____

керівник роботи__Колумбет Вадим Петрович, ст. вик._____
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”25” травня 2022р. № _____

2. Строк подання студентом роботи _10.06.2022_____

3. Вихідні дані до роботи _мова PHP, HTML, CSS, Visual Studio Code_____

4.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)_провести дослідження вимог членів приймальної комісії, на рахунок автоматизації та покращення роботи системи, розробити програмний продукт, що відповідатиме потрібним вимогам, його реалізація_____

5. Перелік ілюстративного матеріалу
25 рисунків, з них 3 схеми системи

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання ”10”___вересня_____2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Затвердження теми роботи	25.05.2022	
2.	Вивчення та аналіз задачі	02-08.05.2022	
3.	Розробка архітектури та загальної структури системи	09-15.05.2022	
4.	Створення інтерфейсу та серверної частини	16-22.05.2022	
5.	Програмна реалізація системи	16-28.05.2022	
6.	Оформлення пояснювальної записки та тестування системи	28.05-02.06.2022	
7.	Захист програмного продукту	23.05-27.05.2022	
8.	Передзахист	06-09.06.2022	
9.	Захист	20-30.06.2022	

Студент _____ _____Кваша К. О._____
(підпис) (прізвище та ініціали,)

Керівник роботи _____ _____Колумбет В. П._____
(підпис) (прізвище та ініціали,)

АНОТАЦІЯ

Даний сервіс призначений для працівників приймальної комісії університету. Він передбачає швидкий пошук та фільтрацію даних, можливість створення типових звітів. Система реалізована в середовищі Microsoft Visual Code 2008. Розділів 4, схем та малюнків - 26, бібліографічних посилань - 18, а загальний обсяг роботи - 53.

THE SUMMARY

This service is designed for employees of the admissions committee of the university. It provides fast search and filtering of data, the ability to create standard reports. The system is implemented in the environment of Microsoft Visual Code 2008. Chapters 4, diagrams and figures - 26, bibliographic references 18, total volume - 53.

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

БД (база даних) — організована колекція структурованої інформації та/або даних, переважно у великих розмірах, яка зазвичай зберігається в електронному вигляді в комп'ютерній системі. Дані структуруються в таблицях.

Веб-сервіс (веб-служба) — це система, до якої можна отримати доступ в інтернет-просторі. Тобто це програма, яка організовує взаємодію між сайтами. Ідентифікувати можна за допомогою URL-рядка.

Back-end — серверна частина веб-сайту, яка зберігає та впорядковує дані, а також забезпечує належну роботу програмного продукту на стороні клієнта. Це частина веб-сайту, яку користувач не може бачити та з взаємодіяти. Тобто це частина програмного забезпечення, яка не контактує безпосередньо з користувачами.

Front-end — частина веб-сайту, з якою безпосередньо взаємодіє користувач, тобто інтерфейс. Її також називають «клієнтською стороною» програми, яка включає все, з чим контактує користувач: кольори та стилі тексту, зображення, графіки та таблиці, кнопки та навігаційне меню.

СКБД (система керування базами даних) — інша відома абревіатура СУБД, система, призначена для керування, доступу та отримання даних, які зберігаються у БД.

ПК – абревіатура для приймальної комісії.

UML – це стандартизована мова моделювання, у складі якої міститься набор діаграм, щоб візуалізувати та задокументувати функції програмних систем, а також для бізнес-моделювання.

URL – своя унікальна адреса веб-сайту, яка визначає місцезнаходження сайту в мережі Інтернет.

ВСТУП

Кожного року наша країна стає на крок ближче до повної діджиталізації. Проте приймальна комісія все ще вимагає багато людських ресурсів, адже процес прийняття, систематизації, сортування та перевірки документів абітурієнтів супроводжується використанням декількох окремих баз даних. Працівникам комісії доводиться вручну співставляти та порівнювати дані з різних джерел для того, щоб отримати необхідні звіти про кількість абітурієнтів, кількість тих, хто подав документи та усі дані вступників.

Також ми маємо врахувати існуючий на території України воєнний конфлікт, що зобов'язує учнів та викладачів проводити навчальний процес та процес вступу дистанційно.

Саме через описані складнощі виникла ідея створити даний продукт, оскільки створена веб-система може допомогти вирішити більшість проблем, з якими стикається приймальна комісія, адже її у подальшому майбутньому можна буде запровадити у нашому університеті. На мою думку, система обліку даних абітурієнтів є актуальною для будь-якого вищого навчального закладу, адже кожного року абітурієнти подають величезну кількість даних, які зберігаються у різних місцях, що є незручними для ефективного управління.

Метою роботи є розробка програмного продукту, що зможе спростити роботу приймальної комісії та автоматизувати процес пошуку та сортування даних, а також створення звітів.

Саме тому перевагами системи, яку було створено, є те, що дані вступників централізовано у єдиному джерелі, доступ до якого можна легко отримати через будь який веб-браузер, пошук інформації максимально зручний та спрощений, додана можливість автоматизованого створення звітів, які можна редагувати та виставляти потрібні поля, до того ж, для зручності користувача усі звіти у подальшому можна легко експортувати в Excel.

Унікальність цієї роботи полягає у тому, що майже усі робота приймальної комісії значно спрощується, та якщо порівнювати цю роботу з подібними аналогами, вони все одно не були в змозі поєднати усі аспекти, які полягають у наступному:

1) робітник приймальної комісії має змогу переглядати загальний список абітурієнтів у зручному форматі та виконувати пошук в ньому по заданим критеріям;

2) данна система дозволятиме працівникам приймальної комісії швидко та зручно формувати звіти з потрібними параметрами.

1 ЗАДАЧА ПОБУДОВИ ІНФОРМАЦІЙНОЇ АВТОМАТИЗОВАНОЇ СИСТЕМИ З ОБРОБКИ ДАНИХ АБІТУРІЄНТІВ

1.1 Актуальність проблеми

За останні декілька років наша країна робить все більше кроків до діджиталізації та спрощення роботи з документами. На даний момент майже усі етапи вступу абітурієнтів до вищого навчального закладу є автоматизовані та відбуваються дистанційно.

Проте усі дані вступників зберігаються у незручній формі та з ними доволі складно працювати. Саме тому, сам процес роботи приймальної комісії дуже ускладняється та є дуже некомфортним. не є оптимальним, а тому є необхідність у створенні певного програмного забезпечення, котре зробить даний процес ефективнішим.

Згідно з вищеописаним, для ефективної організації роботи працівників вищих навчальних закладів, є необхідність у створенні деякого програмного забезпечення та організації доступу до системи з централізованим доступом, де знаходяться усі дані кожного з абітурієнтів, що подали свою заяву на вступ до університету. Така система дозволить робітникам ПК значно зменшити свій об’єм роботи та спростити процеси друкування звітів по потрібним даним та пошук інформації.

Тому тема даної роботи є актуальною для сьогоднішніх потреб працівників приймальної комісії.

Завдання даної дипломної роботи, як уже було описано вище, полягає у розробці автоматизованої системи з обробки даних абітурієнтів у вигляді веб-сайту, який включає в себе інструмент для пошуку та зміни даних. Створений продукт значно спрощує завдання обробки інформації, а також допомагає легко створити потрібний звіт у формі файлу Excel. Користувач, якому адміністратор

надає роль «Робітник» може швидко та легко справитись із описаними завданнями, а також внести будь які зміни до даних, або їх видалити.

1.2 План виконання роботи

Задача полягає у створенні інформаційно-автоматизованої системи з обробки даних абітурієнтів.

Мета роботи: створення програмного продукту, який буде в змозі облегшити роботу приймальної комісії, автоматизувати деякі процеси та спросити процес пошуку та зміни даних абітурієнтів, а також дати можливість робітникам швидко формувати звіти по потрібним даним.

Для досягнення даної мети потрібно розв’язати декілька задач:

1) уважний аналіз існуючої процедури роботи прийомної комісії;
2) систематизація даних абітурієнтів;
3) продумати завдання, які має реалізовувати система;
4) створення use-case діаграми;
5) обрати програмні інструменти для реалізації;
6) створення списку полів для бази даних;
7) власне розробити систему;
8) проектування інтерфейсу програмного продукту;
9) тестування системи;
10) виправлення помилок.

Поставленні задачі я виконувала упродовж всієї переддипломної практики. Для зручності та спрощення розробки програмного забезпечення, а також для правильного розподілення часу протягом усієї дипломної практики було розроблено діаграму Ганта, у якій було ураховано усі задачі та ефективно виділено час на виконання кожної з них.

Діаграма Ґанта — така діаграма, яка використовується для ілюстрації графіку роботи над будь-яким проектом та є одним з засобів планування та управління проектами. Діаграма Ганта представляє всю інформацію візуально за допомогою горизонтальної гістограми. Вона є особливо корисно для проектів, де працює велика кількість членів команди. Тому що вони мають змогу переглядати розклад завдань та прогрес їх виконання, просто подивившись на діаграму. Завчасне планування всіх завдань і, найголовніше, розміщення їх в одному місці – дає можливість командам виконувати проект вчасно та правильно розподіляти ресурси своєї роботи.

Опишемо ключові частини діаграми та як вони функціонують у плані проекту:

1) список завдань: розташований зліва від самої діаграми для опису завдань, що будуть виконуватись у проекті. Може бути організований у групи та підгрупи;

2) часова шкала: проходить горизонтально вгорі діаграми і зображає проміжки часу (дні, тижні, місяці та роки);

3) тривалість завдання: це такий прямокутник, довжина якого відповідає тривалості виконання завдання, показує дати початку та закінчення виконання.

Величезна перевага використання саме цієї діаграми полягає в тому, що вона дає можливість моніторити хід виконання проекту в режимі реального часу, отримуючи розуміння того, як швидко відбувається розробка проекту в порівнянні з його початковим графіком.

Подану діаграму на рисунку 1.1 я розробила у Microsoft Project.

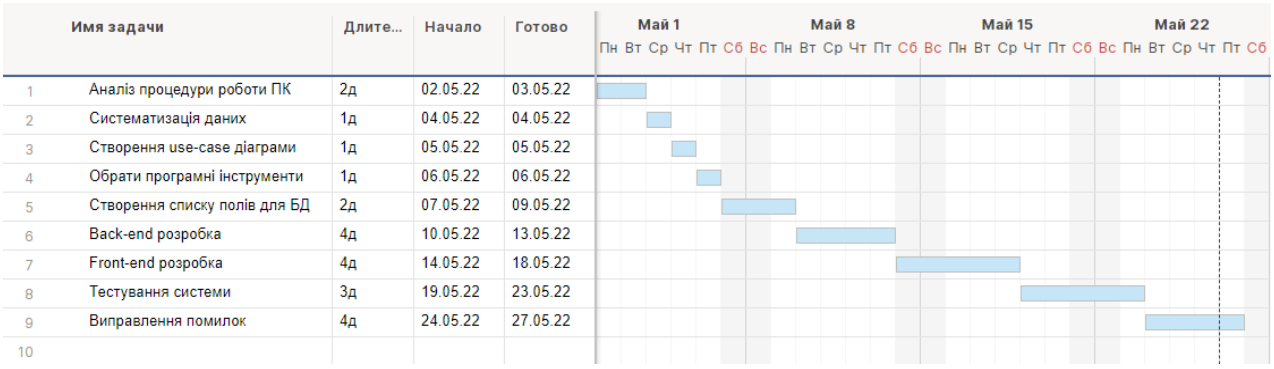

Рисунок 1.1 – Діаграма Ганта

Розроблена діаграма повністю себе виправдала. Завдяки ній, було успішно виконано практику, правильно розраховано час, оцінено об’єм роботи та в повній мірі розроблено програмний продукт.

1.3. Призначення та користувачі

Програмний продукт призначений для навчальних закладів вищої освіти. Саме дистанційний режим проведення навчання та вступу повністю висвітлює актуальність даного програмного продукту, завдяки якому вищі навчальні заклади зможуть полегшити роботу приймальної комісії.

Майбутні користувачі системи – працівники ПК нашого університету. Система планується як внутрішня для вищого навчального закладу, доступ до якої надаватиметься тільки адміністратору системи, тобто співробітнику університету, який зможе створювати аккаунти працівникам ПК, тим самим особисті дані абітурієнтів будуть надійно захищені, адже жодні сторонні користувачі не зможуть до них дістатися.

Як вже було зазначено, охорона особистих даних є важливим фактором у даній роботі, саме тому уся інформація, що зберігається та обробляється системою, повинна бути доступна тільки для працівників, а доступом має керувати

адміністратор. Для більшої зручності обов’язковим є забезпечення того, що користувачі з різними ОС будуть мати однаковий доступ до програмного застосунку. Вхід до системи має відбуватись швидко.

Оскільки даний продукт передбачає часте використання у період роботи ПК, потрібно дотримуватись комфорту тривалого користування. Саме тому дизайн системи має бути спроектованим максимально чистим і мінімалістичним, кольори, що будуть використовуватись, потрібно обрати спокійні, не занадто яскраві, а інтерфейс зробити ненав'язливим та зрозумілим.

1.4 Основні задачі

Головна задача програмного продукту, яку він допомагає вирішити, це прискорення, полегшення та покращення роботи приймальної комісії з даними абітурієнтів.

Як вже було визначено вище, пріоритетним завданням системи є збереження даних та подальша обробка даних вступників.

Отож можемо сформувати основні задачі, які виконує система:

- авторизувати користувачів системи;
- можливість редагування та видалення даних в існуючих особових справах вступників;
- зберігання результатів змін у базі даних;
- перегляд даних абітурієнтів;
- пошук по загальному списку даних;
- можливість додання та зміни критеріїв для фільтрації даних;
- виведення фільтрованих даних у вигляді списку;
- можливість створення звіту по заданим параметрам;
- можливість експорту звіту у Excel.

Для оптимальної та зручної фільтрації даних має бути можливість фільтрувати за кількома параметрами одночасно.

Для цього було виділено основні параметри для фільтрації:

- ПІБ;
- дата народження;
- факультет;
- вступ на основі;
- форма навчання;
- громадянин;
- статус заяви;
- курс;
- бал;
- заява електронна;
- номер особової справи;
- чи подано оригінали;
- контракт;
- бюджет;
- тип документа;
- серія документа;
- номер документа;
- дата видачі;
- ким видано;
- відзнака;
- сканкопії;
- час додання заяви до ЄДЕБО;
- потрібно внести зміни в ЗНО;
- контакти;
- електронна пошта;
- пріоритет;

- бере участь в конкурсі на місця держзамовлення;
- пільгові категорії;
- сільський коефіцієнт;
- РНОКПП.

Саме завдяки грамотному виконанню описаних задач, в результаті маємо повноцінну програму, яка вирішує важливі проблеми процесу роботи приймальної комісії і не має ідентичного аналогу у світі.

1.5 Діаграма прецедентів

UML (англ. Unified Modeling Language) — це стандартизована мова моделювання, у складі якої міститься набор діаграм, які спеціально використовуються для допомоги розробникам програмного забезпечення, для того щоб визначити, візуалізувати, сконструювати та задокументувати функції програмних систем, а також для бізнес-моделювання.

Ця мова являє собою набір таких сталих технічних методів, які використовуються при моделюванні великих і складних систем та є дуже важливою частиною розробки об'єктно-орієнтованого програмного забезпечення.

Мова UML є досить візуальною, тобто використовуються у більшості саме графічні позначення для відображення та розуміння процесів програмних проектів. Її використання допомагає групам, що працюють над програмними продуктами, досліджувати потенційні проекти та наочно переглядати компоненти, поведінку й структуру системи чи певного процесу. Вона допомагає передбачити потенційні помилки в поведінці та структурі системи ще на етапі її планування та обговорення.

Як зазначено вище, UML є способом візуального представлення архітектури, проектування та реалізації складних програмних систем. При написанні програми, у програмі з’являються тисячі рядків коду, і інколи буває важко знайти та відстежити взаємозв’язки та ієрархії в програмній системі. Саме для цього

використовують діаграми UML, адже вони поділяють усю систему на компоненти та підкомпоненти. Завдяки такій діаграмі можемо візуально побачити, які функції та можливості мають бути у майбутніх користувачів.

Схеми UML поділяють на дві основні групи, що, в свою чергу, розділяються ще на низку типів та підтипів: діаграми бувають структурні і поведінкові.

Для цього проекту була створена use-case діаграма, яка є підтипом поведінкових схем UML (див. рисунок 1.1).

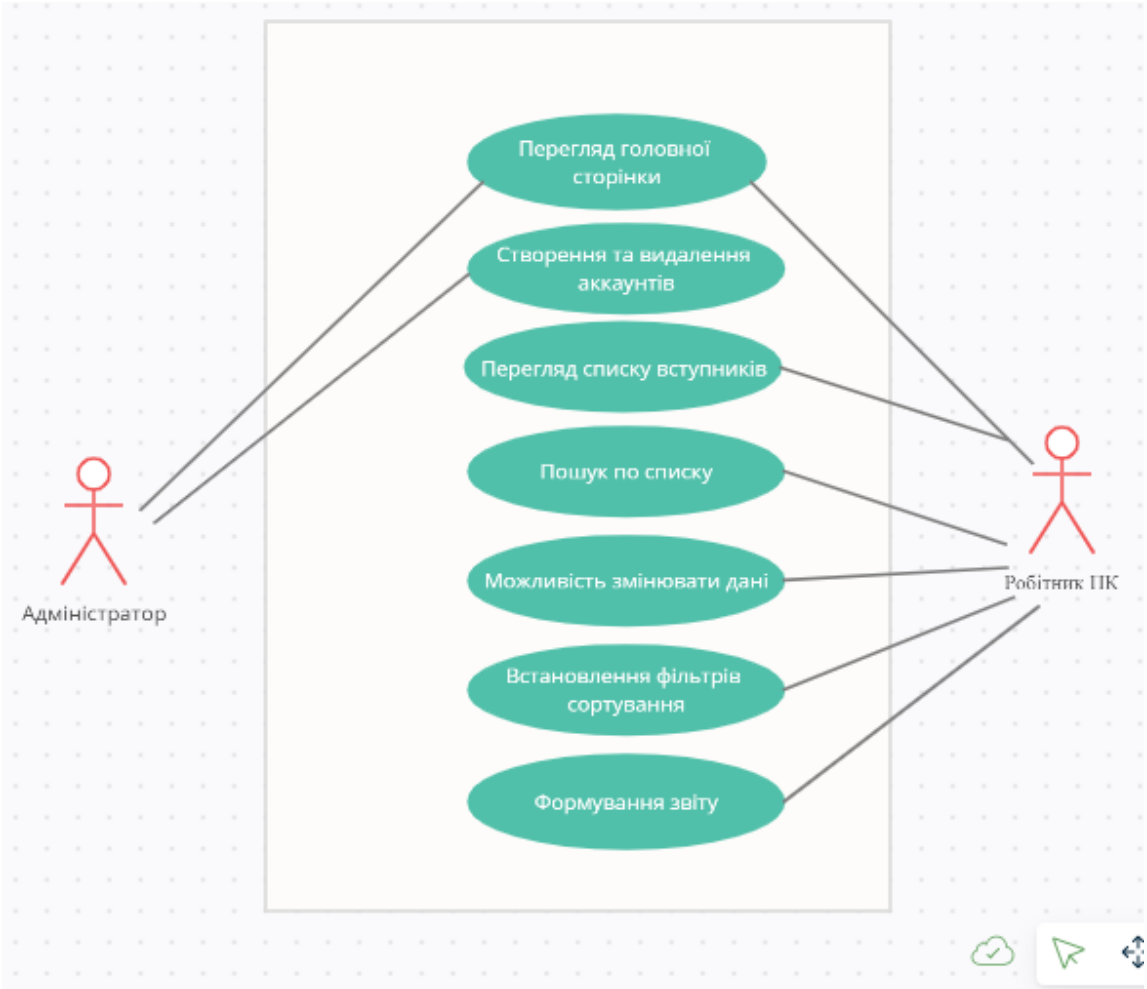

Рисунок 1.2 – Діаграма UML.

Для кращого розуміння діаграми опишемо символи зображені на ній:

1) границі системи — білий прямокутник, який містить назву системи зверху та деякий порядок дій і взаємодій між так званими акторами. Її у деяких випадках також можна назвати сценарієм;

2) актори — користувачі, які власне взаємодіють із створеною системою. Цю роль виконує будь-яка людина, організація або стороння система, що взаємодіє з програмним продуктом. На діаграмі позначається символом людини;

3) прецедент — еліпси, що розташовані горизонтально; представляють собою різні варіанти використання та дії, які може виконувати актор (користувач). Напис на еліпсі обов'язково має бути описом дій системи, але, звичайно, з точки зору актора.

Такий тип діаграм дає змогу візуально побачити огляд акторів, задіяних у системі, різні функції, що необхідні цим акторам, і те, як усі ці функції взаємодіють.

Це чудовий сценарій для початку обговорення та проектування будь-якої системи, оскільки можна легко визначити головних учасників та основні процеси.

1.6 Послідовність дій системи

Після того, як найважливіші функції та можливості системи продумано, далі починаємо прописувати послідовність, у якій будуть оброблятись дії системи.

Отже, черговість дій користувача програмного застосунку така:

1) як завжди, у подібних системах, відбувається авторизація, тобто до вже створеної БД програма відправляє запит, який робить перевірку на існування у системі такого користувача;

2) якщо після перевірки виявляється, що заданого аккаунту не існує, користувач отримує відмову доступу до системи;

3) якщо ж авторизація даних успішна, у працівника приймальної комісії з'являються можливості шукати та передивлятись потрібні дані по заданим

параметрам, також встановлювати фільтри для пошуку, сортувати отриману інформацію, а також завантажувати звіти даних із заданими параметрами;

4) адміністратор системи, в свою чергу може створювати та видаляти аккаунти, тобто надавати доступ до системи користувачам.

1.7 Діаграма станів

У мові UML є ще одна дуже корисна для розробників програм поведінкова діаграма – діаграма діяльності. Такий тип графічного зображення процесів розроблювального продукту використовується для відображення послідовності процесів та дій (як можна зрозуміти із самої назви).

Такі діаграми зображають процес роботи будь-якої системи від її початкової точки та аж до кінцевого етапу з детальним описом усіх процесів та різних варіантів шляхів прийняття рішень, що існують у процесі роботи.

Діаграми діяльності часто використовують для більшої деталізації конкретних ситуацій під час виконання програми, а особливо – коли відбувається паралельна обробка декількох дій під час роботи.

Тому перерахуємо деякі переваги таких діаграм:

1) можна наглядно побачити роботу логіки алгоритму програми, що дозволяє розробникам пришвидшити та покращити деякі процеси, шляхом зменшення дій для обробки;

2) ілюстрація бізнес-процесу між користувачами та системою, що дає можливість візуально побачити наскільки використання створеного програмного застосунку є зручним для користувача;

3) автоматизація та пришвидшення будь-якого процесу, шляхом графічного та наочного прояснення складних випадків використання.

UML діаграма діяльності містить у собі багато компонентів, а перш ніж починати її складати, обов’язковим є спочатку зрозуміти її склад. Тому опишемо основні з компонентів:

- дія: один крок у процесі діяльності системи, на якому користувач або саме програмне забезпечення виконують описане завдання. Дії позначаються прямокутниками;
- вузол рішення: представлений ромбом та включає в себе один вхід і два або більше виходів. Використовується для демонстрації різних варіантів вирішення завдання;
- потоки керування: з’єднувачі між діями. Позначається стрілками; після завершення кроку потік продовжується стрілкою, що виходить;
- початковий вузол: зображається чорним колом та символізує початок процесу діяльності, тобто є початковим вузлом;
- кінцевий вузол: також зображений чорним колом та означає останній крок у процесі діяльності.

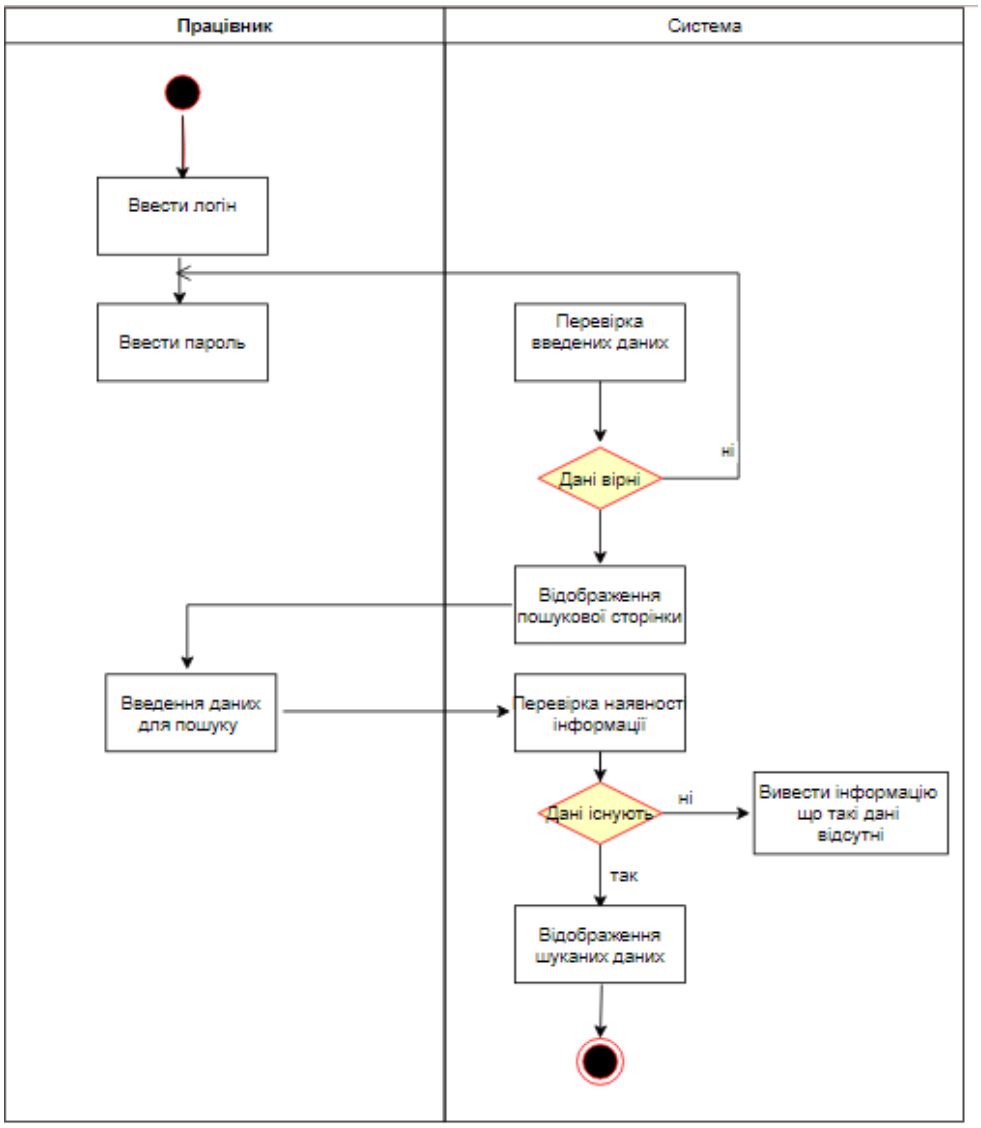

Рисунок 1.3 – Діаграма діяльностей

На рисунку 1.3 зображена діаграма діяльності процесу авторизації та введення даних для пошуку інформації. Оглянемо ситуацію, в якій працівник ПК бажає авторизуватися у системі та здійснити пошук по бажаним параметрам. Дійові особи, що використані у діаграмі – «Працівник» та «Система».

Процес описаної діяльності починається з відкриття працівником ПК в браузері створеного веб-сервісу та його головної сторінки. Першим кроком буде авторизація у системі. Далі програма перевіряє чи існують внесені логін та пароль у нашій БД. Якщо ж такі дані співпали з існуючими у БД (тобто адміністратор

створив такий аккаунт), то працівник без перешкод входить у систему і бачить пошукову стрічку. Якщо ж дані були введені невірно – користувачу відмовлено у доступі до системи.

Працівник може зробити запит на пошук інформації, що його цікавить, по заданим ним параметрам. Якщо такі дані існують серед інформації про абітурієнтів – він отримає табличку з усіма можливими співпадіннями.

Отже, така діаграма дозволила більш детально та візуально показати процеси авторизації та пошуку.

1.8 Діаграма компонентів

У той час як інші діаграми UML описують функціональність системи, діаграма компонентів використовується саме для моделювання тих самих компонентів, на основі яких далі створюються функції програми.

Що таке ж таке ця діаграма компонентів? Її використовують для візуалізації, організації та розуміння зв’язків та залежностей між компонентами системи. Такі діаграми потрібні при моделюванні об’єктно-орієнтованих систем, а також для побудови систем за допомогою прямого та зворотного проектування. Тобто діаграма компонентів, по суті, є діаграмою класів, яка просто зосереджена на компонентах системи. Вона дозволяє розробникам ідентифікувати різні частини системи, аби уся система правильно працювала та виконувала те, що вона повинна робити.

Компонентами можуть виступати база даних або інтерфейс користувача; схема, мікрочіп або пристрій; або бізнес-одиниця, наприклад постачальник, нарахування заробітної плати або доставка.

Діаграма компонентів, на відміну від вищеописаних схем, належить до другого типу діаграм UML – до структурних. Структурні діаграми показують різні

частини заданої системи на різних рівнях її реалізації, а також їх взаємозв'язок між собою.

Розглянемо її переваги та чому при розробці будь-якого програмного продукту її створення є необхідним:

- на ній можна побачити структуру самого майбутнього коду без його написання;
- дає змогу зосередитись саме на зв’язку між компонентами системи, не обдумуючи специфічні деталі;
- на моїй діаграмі компонентів є 6 основних інтерфейсів:
- авторизація — система входу для користувачів системи.
- сторінка з можливістю пошуку по даним — система пошуку по заданим користувачем даним;
- створення звіту — система створення звіту з вибраними параметрами;
- сортування та зміна у даних — дає змогу додавати, видаляти та редагувати дані абітурієнтів із списку.

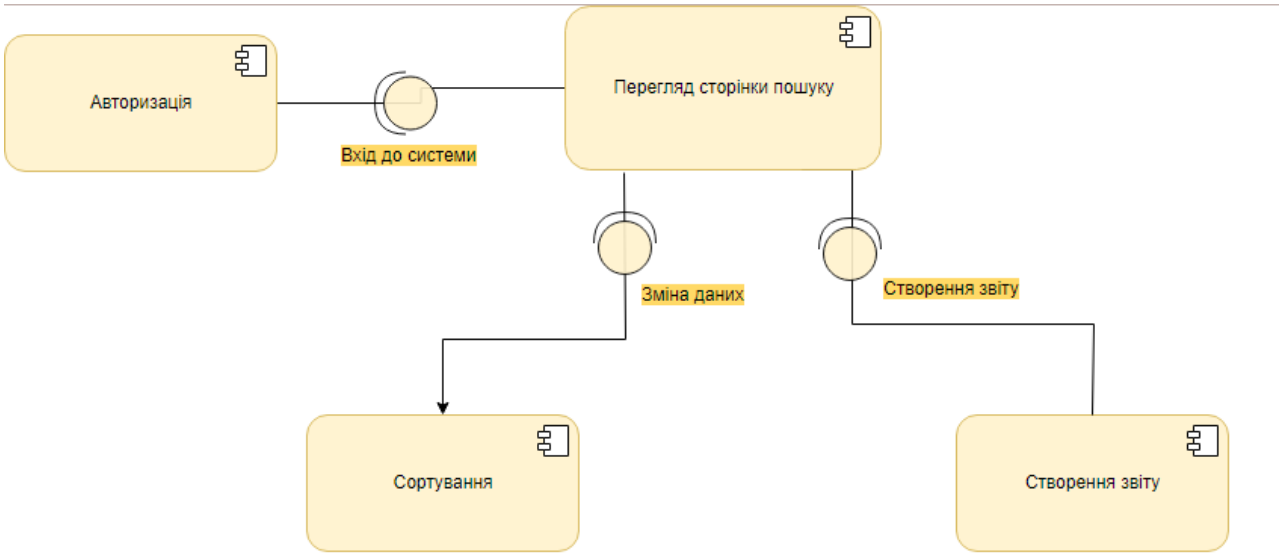

Рисунок 1.4 – Діаграма компонентів

Розроблену діаграму можемо побачити на рисунку 1.4. Її створення відбувалось у веб-сервісі «Diagrams.net», оскільки, на мою думку, він має зручний та доступний інтерфейс.

Отже, створення усіх діаграм допомогло дати розуміння того, які функції потрібні системі, скільки буде ролей у ній та кількість її компонентів.

Тому залишилося дослідити дані, що будуть використовуватись створеним програмним застосунком.

1.9 Дослідження даних

Під час процесу вступу абітурієнтів університет зберігає велику кількість поданих ними даних, серед яких:

- ПІБ;
- дата народження;
- стать;
- номер особової справи;
- пріоритет;
- паспорт;
- ідентифікаційний код;
- назва закінченого учбового закладу;
- чи є відзнака;
- середній бал атестату;
- назви предметів ЗНО;
- середній бал предметів ЗНО;
- факультет;
- спеціальність;
- рівень освіти (бакалавр/ магістр);
- адреса проживання та прописки;

- чи потребує майбутній студент гуртожиток;
- чи є абітурієнт іноземцем;
- чи має пільги;
- адреса електронної пошти – Email;
- домашній та мобільний номер телефону;
- мобільний номер телефону батьків;
- форма навчання (денна/заочна);
- тип конкурсу, за яким отримано рекомендацію.

Усі ці дані зберігаються у Excel-файлі, а розроблена система дозволить організувати швидкий та звучний пошук і сортування інформації.

Аналіз та дослідження демонструє, що створення єдиною системи даних є досить проблематичним. На мою думку, головною проблемою є дуже незручний формат збереження всієї інформації абітурієнтів, так як автоматизувати пошук та швидку зміну або додання даних у Excel-файлі є просто неможливо. До того ж, усі дані кожний навчальний заклад зберігає та оброблює по різному.

Все вищеописане пояснює необхідність та актуальність створення саме локальної системи для обробки даних.

ВИСНОВКИ ДО РОЗДІЛУ

У першому розділі даної записки до дипломної роботи було описано задачу та мету роботи, продумано які користувачі будуть її використовувати. Тобто, описано усі найважливіші етапи декомпозиції проекту. Перерахуємо їх:

1) описана актуальність роботи;

2) створення системи було розбито на етапи, для яких було виділено свої проміжки часу, аби дану роботу можна було максимально ефективно виконати за виділений час та правильно розподілити навантаження – тобто розроблено діаграму Ганта;

3) створено діаграму прецедентів, що дає змогу візуально побачити огляд ролей, задіяних у системі, різні функції, що необхідні користувачам, і те, як усі ці функції взаємодіють;

4) сформовано основні задачі, що має виконувати система, для більшої зручності роботи ПК;

5) описано послідовність дій роботи системи;

6) розроблена діаграма станів, яка зображає процес роботи створеної системи від її початку і до кінця, з детальним описом усіх процесів, що існують у системі;

7) створена діаграма компонентів системи, для її кращого розуміння;

8) та описано дані, які будуть у ній зберігатись.

Підсумовуючи, усі вищеописані етапи дозволили ефективно розподілити ресурси роботи та швидко запрограмувати систему.

2. АНАЛІЗ АНАЛОГІВ ТА ОПИТУВАННЯ

2.3. Аналіз аналогів

abit-poisk.org.ua – є схожою за функціональністю системою, проте цей сервіс не є локальним. Тут можна знайти дані усіх абітурієнтів з усіх університетів. До того ж, список інформації, що видає сайт, є неповним через захист особистих даних користувачів.

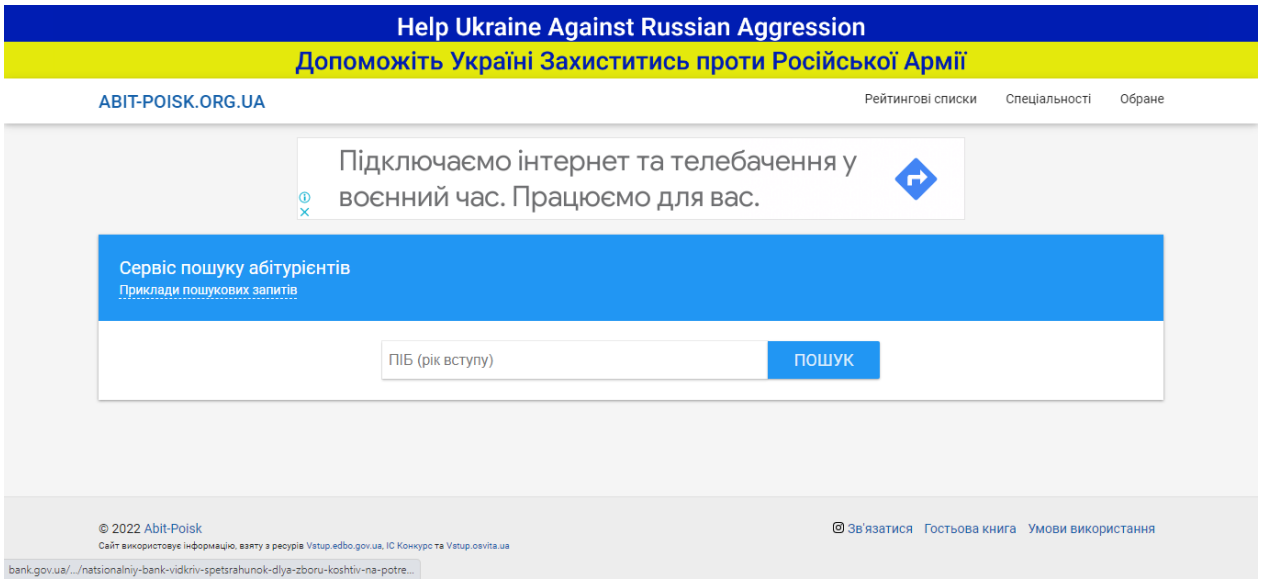

Рисунок 2.1 – Головна сторінка сайту

На рисунку 2.1 та 2.2 продемонстовано головну сторінку веб-сайту та результат пошуку за заданою інформацією. Можна побачити, що тут кнопка реєстрації відсутня. Це означає, що фільтрації користувачів за рівнем доступу на цьому програмному застосунку немає. Пошук по даним працює доволі швидко, проте функціонал фільтрації або сортування тут відсутній.

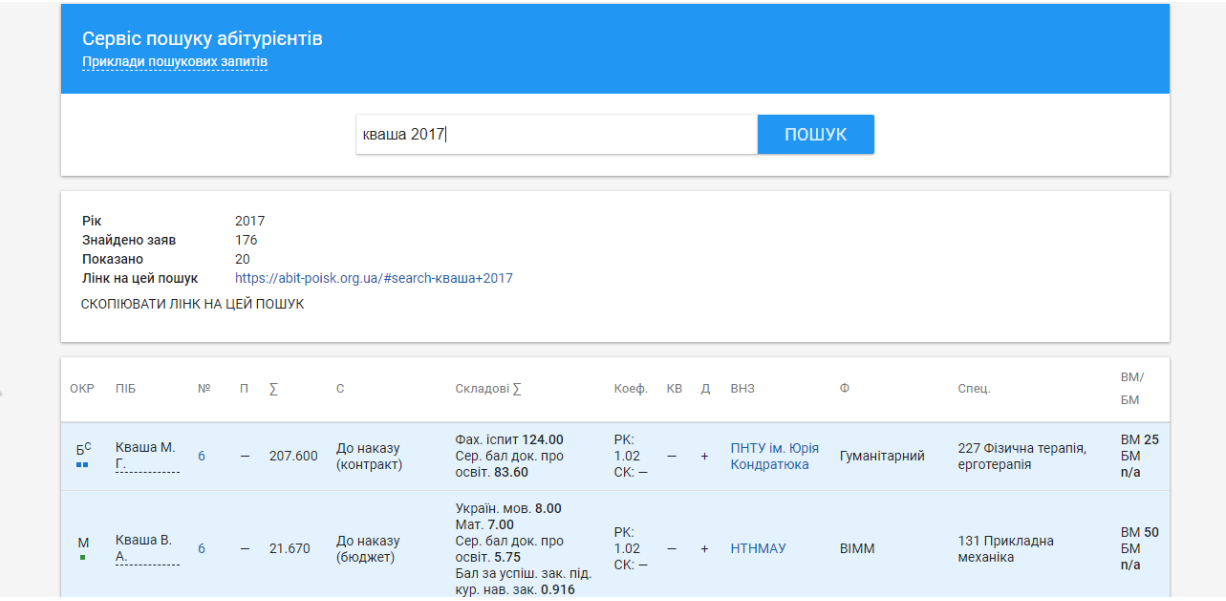

Рисунок 2.2 – Сторінка з результатами пошуку

2.4. Аналіз результатів опитування

Для кращого розуміння того, які функції має мати наша система, а які функції є непотрібними, я провела опитування серед працівників приймальної комісії нашого університету.

Були обговорені наступні питання:

- у який спосіб зараз відбувається зберігання та обробка даних абітурієнтів;
- які недоліки та переваги має цей спосіб;
- які функції ви хотіли б бачити у нашому програмному застосунку;
- найчастіші проблеми, з якими стикаються працівники при створенні звітів та при пошуку потрібної інформації.

Після результатів обговорення стало зрозуміло, що функції створення звітів та пошуку потребують автоматизації та покращення.

3. ЗАСОБИ РОЗРОБКИ

Розробка складалась з двох частин:

1) розробка Back-end (програмно-апаратна частина сервісу)
2) розробка Front-end (інтерфейс)

3.1 Back-end розробка

Для цієї розробки було використано мову PHP та SQL, середовище розробки Visual Code.

3.2 Мова PHP

Для створення даної системи було обрано популярну мову програмування PHP, що раніше мало значення Personal Home Page Tools — перекладається як засоби для особистої домашньої сторінки, проте зараз є рекурсивним акронімом Hypertext Preprocessor — гіпертекстовий препроцесор.

Ця мова програмування існує вже досить давно, адже її перша версія була випущена ще 27 років тому — у далекому 1995 році, на даний момент мова PHP пережила вже вісім версій, останнє доповнення було випущено у січні 2022 року, проте її сьома версія залишається найбільш широко розповсюдженою серед розробників.

Рисунок 3.1 – Логотип мови PHP

PHP є скриптовою мовою програмування та має досить багато переваг, через які я вибрала саме її, до того ж, вона є основною мовою програмування для багатьох веб серверів протягом останніх 15 років.

Отож, переваги PHP, які я виділила особисто для себе:

- кросплатформенність — ця мова не залежна від платформи використання, тобто користувачу не потрібно мати певну операційну систему, щоб нею користуватися, з PHP можна працювати і на Mac, і на Windows та навіть на Linux. Це є досить зручно, бо мова стає доступною для великої кількості клієнтів;
- open source: PHP — програмне забезпечення з відкритим вихідним кодом. Оригінальний код надається всім, хто хоче розробити його, тому розробник може вибрати будь який з фреймворків для роботи. Відкритий код допомагає веб-розробці стати швидшою через надзвичайно швидке надання інструментів та інших функцій розробнику;
- також значною перевагою є простота вивчення PHP, адже її синтаксис дуже простий у використанні та вивченні навіть для початківців;
- простота інтеграції з іншою мовою програмування та базою даних (реляційною та нереляційною), адже це дозволяє заощадити багато зусиль та часу розробників.

Після того, як користувач переходить за посиланням (створює запит), він створює запит до сервера. Найпоширеніший випадок - коли викликають HTML сторінку — тобто текстовий файл. Браузер, який використовує користувач, робить

запит на сервер, далі отримує сам код сторінки, а також додатково завантажує файли, що будуть відображати дизайну та коректної роботи веб-сторінки. Після цього користувач може побачити результат у своєму вікні.

3.3 Використання баз даних в середовищі розробки

Як відомо, великий обсяг даний потребує управління. А оскільки у завдання мого проекту входить зберігання даних абітурієнтів, то, очевидно, використання бази даних є необхідним. Що ж таке база даних? Це така електронно створена система для управління інформацією, вона мусить ефективно обробляти обсяги даних будь якого розміру, без помилок та протиріч.

У цій системі було вирішено застосовувати мову структурованих запитів, а тобто SQL, яка може виконувати такі завдання:

- створення, зміна нової або існуючої структури бази даних (БД);
- налаштування та модифікування параметрів безпеки системи;
- налаштування повноважень певних користувачів для доступу до інформації;
- та звичайно, отримання самої інформації з БД.

SQL підтримують такі системи управління базами даних: Microsoft Acces, Oracle, Sybase. Синтаксис цієї мови досить гнучкий та простий до вивчення та розуміння.

3.4 Система управління базами даних

СУБД є абревіатурою для систем, управляють базами даних, вони підтримують зберігання та обробку даних, які на них знаходяться.

Найпопулярнішими операційним системами управління базами даних є: Oracle, Informix, Microsoft SQL Server або MySQL.

Після дослідження різних систем управління БД було прийнято рішення, що у цій роботі проекті буде використовуватись СУБД під назвою MySQL, адже вона є перевіреною часом та доволі простою у використанні, і, найголовніше, у багатьох проектах часто поєднується з PHP.

Що ж таке СУБД MySQL? Ця система була розроблена шведами у 1993 році, являє собою програмне забезпечення для управління інформацією з відкритим вихідним кодом.

Цікавий факт, ця система стала найпопулярнішою у світі та вважається найстабільнішим рішенням в сфері СУБД. До того ж, MySQL є доволі гнучкою: її можна використовувати на більш ніж 20 платформах, включаючи Linux, Windows і Mac OS. За її допомогою створюються динамічні сайти.

Тому було виділено основні переваги MySQL:

- використовується на різних платформах;
- оцінка запитів проводиться у короткий відрізок часу;
- є доволі простою для вивчення та використання навіть початківцям.

Незважаючи на широкий список переваг, ця система має і свої недоліки, такі як обмежена функціональність, якщо порівнювати з більшими системами управління БД. Проте у нашому проекті це не є принциповим, тому цей недолік не є суттєвим.

Отже, враховуючи усі переваги та недоліки було обрано для використання саме СУБД MySQL.

3.5 Front-end розробка

Під терміном frontend у загальному понятті розуміють інтерфейс програмного продукту, тобто те, що бачить користувач. Інтерфейс дозволяє йому

отримувати інформацію із програмного застосунку, взаємодіяти з ним та керувати програмним продуктом. Для прикладу, якщо ми говоримо про веб-сайт, то frontend це сторінки. Користувачі переглядають їх, а також взаємодіють з ними – наприклад, клікають лінки, проскролюють, натискають кнопки, здійснюють вибір зі списків, тощо. Отримані від користувача команди передаються для виконання на сервер (backend) або обробляються і виконуються безпосередньо на комп'ютері користувача.

Для front-end розробки було використано такі інструменти:

HTML — це мова розмітки гіпертексту, яка вказує браузеру, як саме демонструвати текст і зображення на веб-сторінці. За допомогою тегів розробник позначає різні елементів, визначає розташування абзаців, заголовків, додає зображення і відео.

Рисунок 3.2 – Логотип мови HTML

CSS — мова стилів , що складається з каскадної таблиці стилів, додає до створеного веб-сайту такі стилі як: шрифти, кольори, макети, фонові зображення, поля; і тим самим робить сайт більш естетичним.

Рисунок 3.3 – Логотип мови CSS

Таким чином, HTML дозволяє побудувати структуру сайту, а CSS її оживляє.

Мова тегів HTML повідомляє браузерам, яка частина веб-сторінки є заголовком (header), яка нижньою частиною (footer), де знаходяться абзаци, де розміщуються зображення, фото та відео тощо.

Суть роботи така: браузер користувача бере написаний програмістом HTML-вміст сторінки і перекладає його на те, що користувач бачить на екрані свого пристрою.HTML також є загальновизнаним стандартом для того, щоб пошукові системи могли «знайти» веб-сайти на основі пошукових слів, які користувач вводить в рядок пошуку. Саме тому HTML є скелетом практично будь-якого веб-сайту.

Документи CSS використовуються для визначення стилю веб-сторінки, а потім зв’язуються з документом HTML, який, як вже описано вище, містить вміст і структуру сторінки. Тобто файл CSS містить правила та значення, які веб-браузер потім застосовує, щоб правильно відображати вміст сторінки. Стиль можна встановити і безпосередньо в документі HTML, але це не рекомендується. Документи цієї мови можна створювати в будь-якому текстовому редакторі, навіть у Блокноті на Windows.

Переваги CSS:

- швидкий час завантаження: чим більше коду на сайті, тим він важче і довше завантажується. Використовуючи CSS ви пишете код лише один раз, замість того, щоб додавати цей самий код у багатьох місцях у документі HTML, щоб веб-сайт був легшим і швидшим;
- простий у використанні: усі стилі сторінки знаходяться в одному місці, тому користувач може потім повторно використовувати їх на багатьох інших системах. Якщо виникне потреба щось змінити пізніше – наприклад, шрифт, який використовується в усіх абзацах сайту, – у файлі CSS це можна змінити один раз і зміна автоматично застосується на всіх сторінках;
- більше варіантів стилю сторінки: CSS надає багато розширених функцій, дозволяючи контролювати більше аспектів зовнішнього вигляду веб-сайту, ніж у мовах розмітки.;
- більш сучасний: багато функцій у HTML уже застаріли, і, ймовірно, ця тенденція збережеться і надалі. При використанні CSS для стилізації, якщо якісь функції будуть видалені з HTML, це ніяк не вплине на файл користувачів.

3.6 Програма Visual Code

Visual Studio Code — це сервіс для розробки додатків. Він є кроссплатфоменним, адже підтримує розробку на macOS, Linux і Windows, тому розробник можете почати роботу незалежно від платформи, якою він користується. Visua Studio Code позиціонує себе як надзвичайно швидкий та легкий сервіс із простим інтерфейсом. У даному редакторі, як і у більшості додатків такого типу, вбудовані відладник коду, засоби навігації та рефакторингу коду, автодоповнення і контекстні підказки.

Рисунок 3.4 — Логотип програми Visual Code

Великою переваго та плюсом даного сервісу є його підтримка надзвичайно великої кількості мов програмування, серед них : Python, PHP (), C ++, C #, JavaScript, TypeScript, jade, R, Objective-C, XML, PowerShell, Batch, JSON, HTML, CSS, Haxe.

Переваги Visual Code:

- додаток є безкоштовним, що автоматично робить його доступним для всіх бажаючих;
- сервіс легко налаштовується під користувача. Можна встановити різні комбінації клавіш, які буде зручно використовувати саме розробнику, та бажані кольорові схеми;
- величезна кількість доповнень у бібліотеці і готових рішень;
- мультифункціональність (як вже було зазначено вище, редактором може користуватися дійсно будь-який розробник, адже сервіс підтримує практично усі мови програмування, які використовуються у сучасному світі для створення програм та додатків);
- швидкість роботи та інтуїтивний інтерфейс.

JavaScript – мова програмування, що була створена, аби «оживити веб-сторінки».

Програми написані цією мовою називаються скриптами. Вони можуть бути вписані прямо в HTML веб-сторінки та автоматично запускатися під час завантаження веб-сторінки. Тобто, JavaScript це мова скриптів, яка дозволяє

створювати вміст сторінки, що буде динамічно оновлюваний, дає можливість керувати мультимедіа та анімувати зображення.

Рисунок 3.5 – Логотип мови JS

4. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Даний веб-застосунок побудований на основі шаблону MVC (модель-вигляд-контролер). Це шаблон по якому будується майбутня програма. Його метою є відокремлення логіки етапів програмування від того, що бачить користувач – тобто його інтерфейсу. Ця мета досягається шляхом описаного цим шаблоном способу побудови структури програми,. Як результат ми отримаємо набагато легшу масштабованість і реалізацію програми. І до того ж, систему, побудовану на основі цього шаблону, дуже легко тестувати та знаходити помилки.

Простими словами, це такий спосіб організації написання коду, що передбачає виділення блоків, відповідальних для вирішення різних завдань. Усі три блоки мають свої задачі, перший (можель) – містить дані програми, другий (вигляд) – її зовнішній вигляд, а третій контролює процес виконання самої програми.

Розглянемо компоненти MVC кожен окремо:

- Модель — центральний компонент схеми. Він безпосередньо керує даними, логікою та правилами програми, незалежно від інтерфейсу користувача.
- Вигляд – це те, з чим взаємодіє користувач, іншими словами інтерфейс програми. Тобто код компонента «view» визначає зовнішній вигляд системи та способи її використання. Він містить HTML-розмітку і невеликі вставки PHP-коду для форматування та відображення даних. Цей компонент можна поділити на загальний або основний шаблон, де міститься розмітку, що буде однаковою для всіх сторінок (наприклад, «footer» і «header») та деякі маленькі компоненти шаблону, які використовують відображення форм для введення даних.
- Контролер – відповідає за зв'язок між компонентами «model» та «view». Його код визначає, як реагує сайт на дії користувача. По суті, це мозок MVC-програми.

Загальний варіант взаємодії елементів продемонстровано на рисунку 4.1.

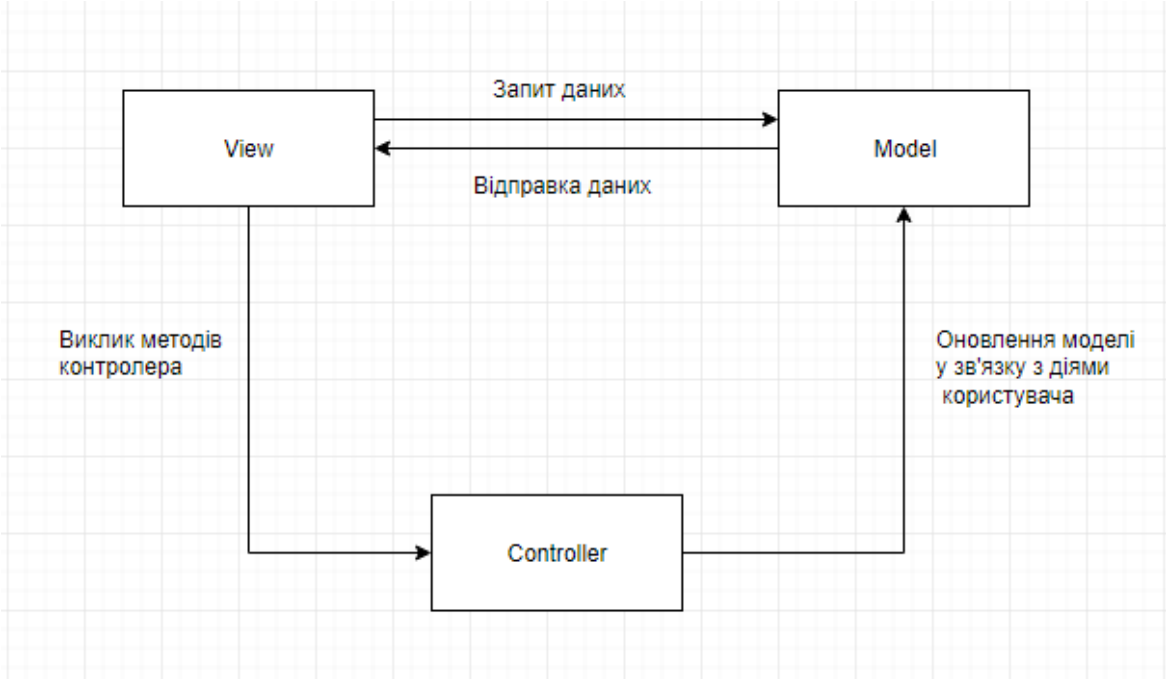

Рисунок 4.1 — Схема роботи моделі MVC

Для мого проекту, враховуючи використання моделі MVC, було створено наступну структуру файлів та папок, які зображені на рисунку 4.2.

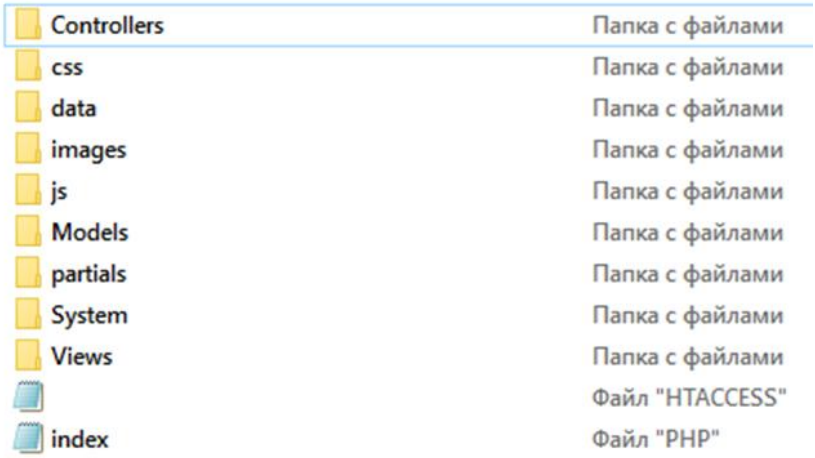

Рисунок 4.2 – Структура папок

Пошук та сортування даних абітурієнтів є основними функціями розроблюваної системи. У даному програмному застосунку основний контролер той, у якому описуються функції взаємодії з БД: тобто пошук інформації, фільтрація, видалення/редагування в таблицях з даними абітурієнтів,.

Також у системі наявні дві основні сторінки (два «вигляди» за моделлю «модель-вигляд-контролер»), з якими взаємодія користувача буде найчастішою. Перший – це відображення таблиці з усіма записами про абітурієнтів з можливістю сортування за обраними критеріями (для працівника ПК). Другий – сторінка з можливістю задання параметрів для формування звіту.

При створенні моєї системи перш за все була розроблена база даних, у яку будуть вигружатися дані з Excel-файлу. Розробка велася мовою PHP. На рисунку 4.3 додаючи або комірки з масиву з іменами стовпців таблиці Excel, ми вказуємо, які дані необхідно завантажити з цієї таблиці (властивість $cells класу Config, міститься масив з іменами стовпців таблиці Excel, які підлягають вивантаженню до БД. При цьому, імена стовпців використовуються як ключі, а як значення - заголовки кожного стовпця. Тому, відповідно, заголовки стовпців однойменні з полями нашої створеної таблиці).

```
class Config {

  public $cells = array(
 'A'=>'id',
 'B'=>'entrant',
 'C'=>'faculty',
 'E'=>'form_of_study',
 'F'=>'citizen',
 'G'=>'status',
 'H'=>'course',
 'I'=>'mark',

 );
}
```

Рисунок 4.3 – Фрагмент коду програми

А на рисунку 4.4 можна побачити власне приклад стовпців з даними у файлі Excel.

id	entrant	faculty	form_of_study	citizen	status	course	mark
9520096	Головач Євген Тарасович 17.12.1999	Факульте	Денна	Так	Скасовано вступником	1 Курс	174.750
9520097	Бублик Олена Юріївна 20.07.2004	Видавнич	Денна	Так	Допущено	1 Курс	142.800
9520108	Конопльов Дмитро Борисович 15.03.1999	Механіко	Денна	Так	Рекомендовано (бюджет)	1 Курс	187.565
9520133	Чернушевич Мирослава Андріївна 13.01.2004	Факульте	Денна	Так	До наказу	1 Курс	162.200
9520134	Кас`яненко Георгій Дмитрович 25.06.2004	Фізико-те	Денна	Так	Відмова	1 Курс	114.700
9520144	Прокопець Дмитро Олександрович 10.04.2004	Факульте	Денна	Так	Скасовано (зарах. на бюджет)	1 Курс	181.600
9520169	Конопльов Дмитро Борисович 15.03.1999	Механіко	Денна	Так	До наказу	1 Курс	187.565
9520176	Рудика Богдан Геннадійович 12.06.2004	Теплоене	Денна	Так	Скасовано (зарах. на бюджет)	1 Курс	189.200
9520189	Кас`яненко Георгій Дмитрович 25.06.2004	Факульте	Денна	Так	Відмова	1 Курс	114.700
9520192	Багінський Любомир Ігорович 29.04.2004	Факульте	Денна	Так	Рекомендовано (бюджет)	1 Курс	191.150
9520202	Маковоз Ангеліна Сергіївна 12.01.2004	Факульте	Денна	Так	Скасовано (втрата пріор.)	1 Курс	122.100
9520218	Чернушевич Мирослава Андріївна 13.01.2004	Інститут п	Денна	Так	До наказу	1 Курс	162.200
9520245	Кас`яненко Георгій Дмитрович 25.06.2004	Факульте	Денна	Так	Відмова	1 Курс	114.700
9520264	Чернушевич Мирослава Андріївна 13.01.2004	Теплоене	Денна	Так	Допущено	1 Курс	162.200
9520297	Кас`яненко Георгій Дмитрович 25.06.2004	Факульте	Денна	Так	Відмова	1 Курс	114.700
9520304	Чернушевич Мирослава Андріївна 13.01.2004	Факульте	Денна	Так	Рекомендовано (контракт)	1 Курс	162.200
9520307	Сень Ангеліна Олександрівна 14.12.2003	Факульте	Денна	Так	Скасовано (зарах. на бюджет)	1 Курс	182.300
9520313	Філонюк Євгеній Костянтинович 14.11.2003	Теплоене	Денна	Так	Скасовано (зарах. на бюджет)	1 Курс	176.500
9520325	Сеїт-Джеліль Ільмі Абібуллайович 24.11.2000	Фізико-ма	Денна	Так	До наказу	1 Курс	128.250
9520343	Нагорна Єлизавета Вікторівна 24.07.2004	Факульте	Денна	Так	Скасовано (зарах. на бюджет)	1 Курс	189.800
9520368	Чернушевич Мирослава Андріївна 13.01.2004	Факульте	Денна	Так	Допущено	1 Курс	162.200
9520376	Луценко Корнєй Романович 24.09.2003	Теплоене	Денна	Так	Скасовано (зарах. на бюджет)	1 Курс	170.180

Рисунок 4.4 – Список даних абітурієнтів у Excel

За завантаження файлу з даними на сервер відповідає метод export(), класу Controller. На рисунку 4.5 зображено його опис.

```
public function export() {
    if(!empty($_FILES['xls']['tmp_name'])) {
        $file = $this->uploadFile($_FILES);

        if($this->xlsToMysql($file)) {
            header('Location:index.php');
        }
    }
}
```

Рисунок 4.5 – Метод export()

Після успішного завантаження файлу на сервер викликаємо для виконання метод xlsToMysql($file), який власне і перенесе дані з таблиці Excel в базу даних

MySQL. В цьому методі саме цикл foreach(), надає нам змогу пройтись по об'єкту і отримати доступ до кожного рядка таблиці окремо (див рисунок 4.6).

```
$arr = array();
foreach ($rowIterator as $row) {

  if($row->getRowIndex() != 1) {

    $cellIterator = $row->getCellIterator();
    foreach ($cellIterator as $cell) {
      $cellPath = $cell->getColumn();

        $arr[$row->getRowIndex()][$this->config->cells[$cellPath]] = $cell->getCalculatedValue();
      }
      }


  }

}
```

Рисунок 4.6 – Цикл foreach()

В результаті, отримаємо масив даних, у кожній комірці якого міститься масив з даними по окремому рядку таблиці Excel. Відповідно, ці дані передаємо методу моделі insertExcel(), який сформує SQL запит для вставки даних у таблицю бази даних MySql.

Основне підключення бази даних до веб-сторінки відбувалось таким чином – папка «Models», файл «Model» (рисунок 4.7).

```
<?php

namespace Models;
class Model {
    public $pdo;

    public function __construct()
    {
        $dbOptions = ['host' => 'localhost', 'dbname' => 'id18929014_test', 'user' => 'id18929014_test_user', 'password' => '~yr94RD^$w2hOz->'];

        $this->pdo = new \PDO(
            'mysql:host=' . $dbOptions['host'] . ';dbname=' . $dbOptions['dbname'],
            $dbOptions['user'],
            $dbOptions['password']
        );
        $this->pdo->exec('SET NAMES UTF8');
    }
}
```

Рисунок 4.7 – Підключення БД

Опишемо реалізацію контролерів. Контролер отримує введені дані користувача при авторизації, обробляє їх і відправляє назад результат обробки. На рисунку 4.8 зображено один із контролерів.

```php
namespace Controllers;

use Models\Order;
use Models\User;
class MainController
{
    public function admin($view)
    {
        if (isset($_SESSION['user']) && $_SESSION['user']['name'] == 'admin') {
            $orderObj = new Order();
            $orders = $orderObj->getOrders();
            $orderTypes = Order::TYPE_ORDER;
            include "Views/{$view}.php";
        } else {
            header('Refresh:2; url=/');
            echo 'Ви не маєте доступа до сторінки адміністратора.
            Вас буде перенаправлено на головну сторінку через 2с';
        }

    }
```

Рисунок 4.8 – Вигляд одного з контролерів

Даний контролер перевіряє чи може користувач зайти на сторінку адміністратора.

І останній етап програмування нашого веб-сайту – створення графічного інтерфейсу, тобто того, що буде бачити користувач. За це відповідає компонент «View» моделі «MVC». У даному сайті створені такі сторінки:

- головна;
- авторизація;
- сторінка пошуку;
- сторінка вибору параметрів для формування звіту.

Для кожної зі сторінок є свій вид з папки «Views». Деякі сторінки можуть використовувати модель або моделі з папки «Models». На рисунку 4.9 можемо

побачити як відбувається побудова інтерфейсу входу у свій аккаунт, тобто авторизації, із застосуванням контролерів.

```
<body>
    <div class="wrap">
        <!-- header -->
        <?php include_once 'partials/header.php' ?>
        <!-- content -->
        <div class="form-block">
            <form action="/login" method="post">
                <h3>Вхід</h3>
                <p>Електронна пошта</p>
                <input type="email" name="email" placeholder="Логін" required>
                <p>Пароль</p>
                <input type="password" name="password" placeholder="Пароль" required>
                <button type="submit">Вхід</button>
            </form>
        </div>
        <!-- footer -->
        <?php include_once 'partials/footer.php' ?>
    </div>
</body>
</html>
```

Рисунок 4.9 – Інтерфейс користувача

За допомогою вище описаної схеми будувався та розроблявся увесь проект, саме використання моделі MVC дозволило правильно організувати процес розробки системи, і відповідно завдяки цьому зменшились витрати часу та збільшилась продуктивність роботи.

5. РОБОТА КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ

Перша дія для початку роботи із створеним веб-додатком – зайти на відповідну сторінку в браузері. На Рисунку 5.1 показаний інтерфейс головної сторінки.

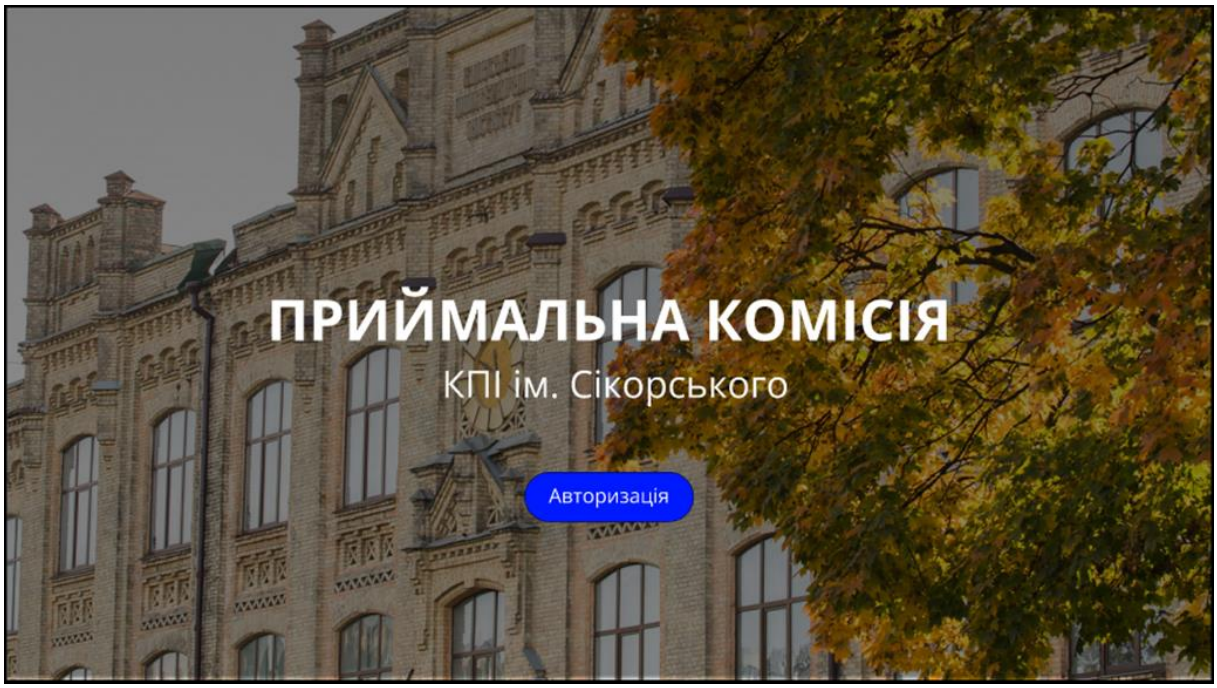

Рисунок 5.1 – Головна сторінка

Можна звернути увагу, що дизайн сторінки виповнений у мінімалістичному стилі, тут зображено лише найголовніше. Для того щоб мати змогу переглядати особисті дані абітурієнтів, потрібно авторизуватися. Для цього можна перейти за посиланням «Авторизація», що знаходиться у центрі сторінки.

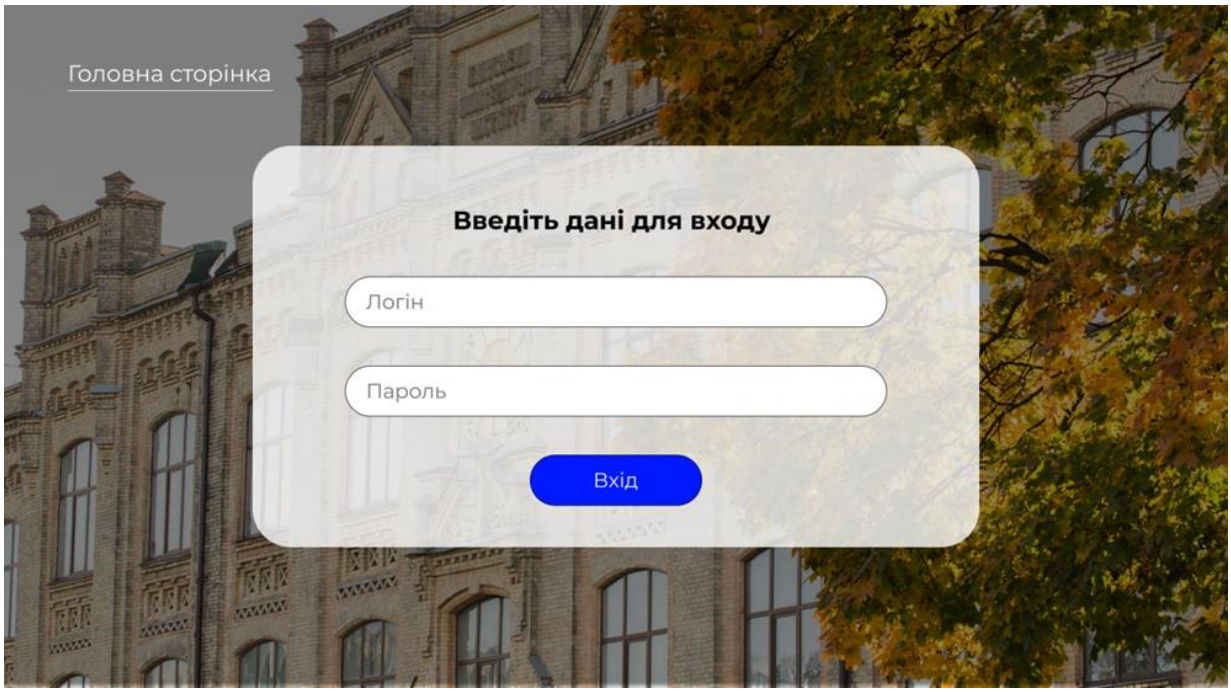

Рисунок 5.2 – Вікно авторизації

Далі користувач входить в систему. При спробі зайти в аккаунт з незаповненими обов’язковими полями з’являтиметься попередження, а також при неправильному вводі даних також буде з’являтися повідомлення про помилку. Авторизація можлива тільки тоді, коли введені дані відповідають очікуваному формату.

Після успішного введення даних робітник приймальної комісії може здійснити пошук по даним абітурієнтів за різними параметрами (прізвище, ім’я, факультет, спеціальність, форма навчання, контракт чи бюджет, бакалавр чи магістр, конкурсний бал (див. рисунок 5.3).

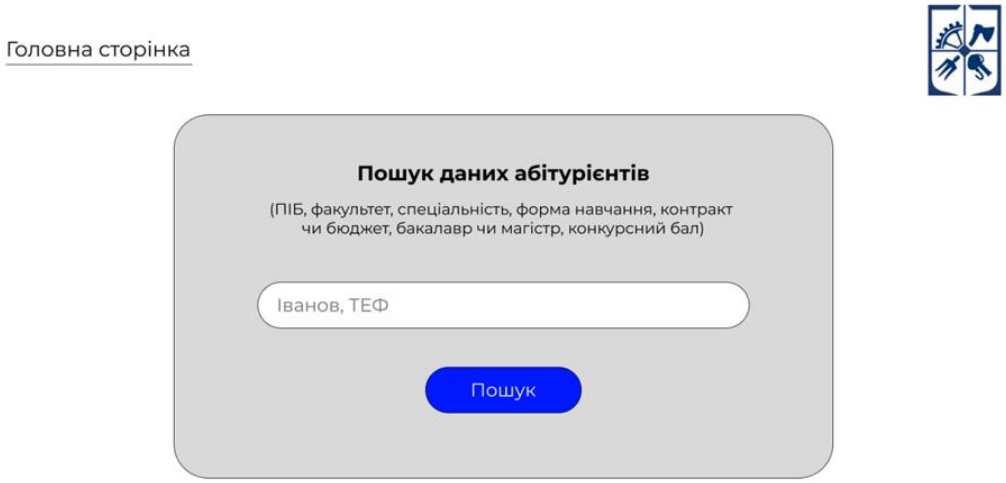

Рисунок 5.3 – Вікно пошуку

Далі на рисунку 5.4 користувач може побачити результати пошуку.

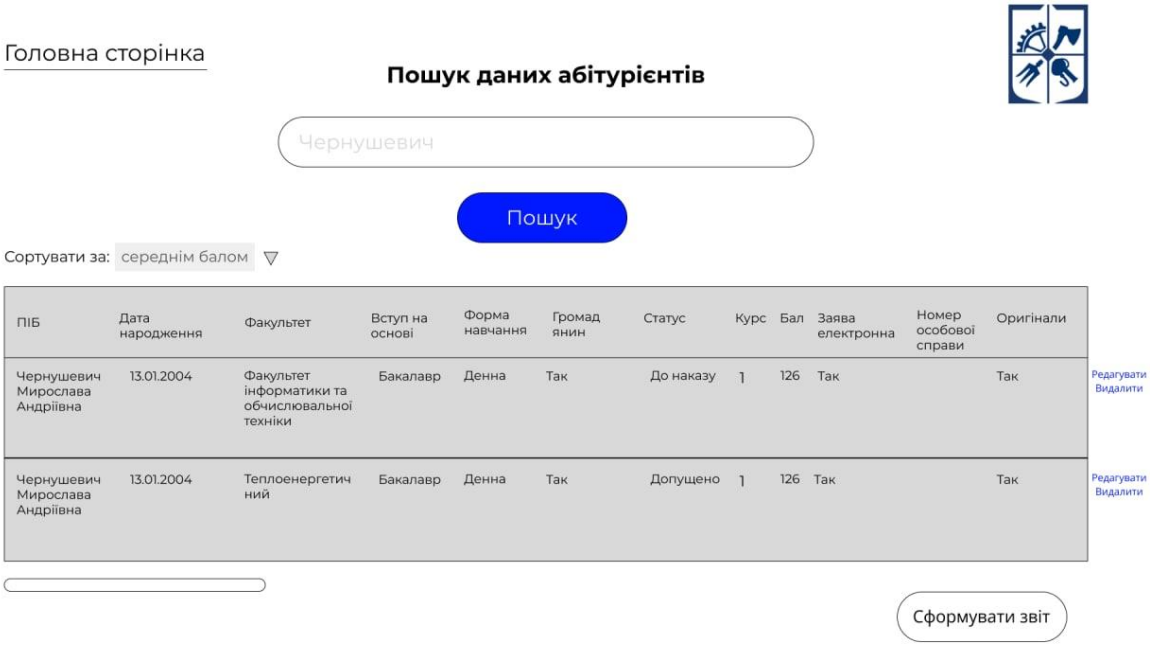

Рисунок 5.4 – Результат пошуку

Усі дані абітурієнтів можна легко змінювати натиснувши кнопку «Редагувати», а також дані можна видалити натиснувши «Видалити», що значно спрощує роботу ПК, адже працювати з документом Excel є набагато менш зручно та швидко.

При вводі прізвища «Чернушевич» було знайдено два співпадіння. Усі дані можна побачити, проскролюючи сторінку вправо.

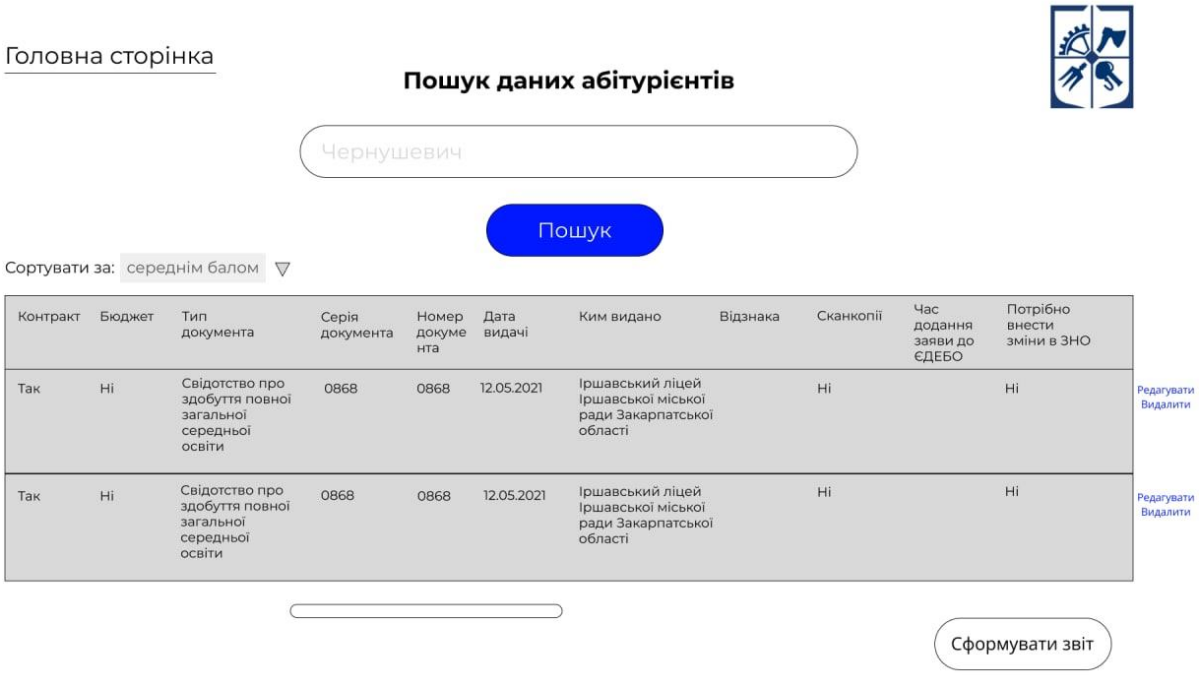

Рисунок 5.5 – Результати пошуку по прізвищу

На сторінці відображення результатів пошуку є мінімальна кількість налаштувань та кнопок, аби не перегружати візуально, адже у самій таблиці вже і так наявна велика кількість стовпців та даних. Усі найважливіші функції виділено, а тому працівнику не буде важко розібратися із системою та він швидкр зможе знайти усі важливі налаштування.

Також натиснувши кнопку «Сформувати звіт» користувач може створити звіт по вибраним даним та вибрати параметри, які будуть там відображатись. На рисунку 4.6 зображено описаний інтерфейс.

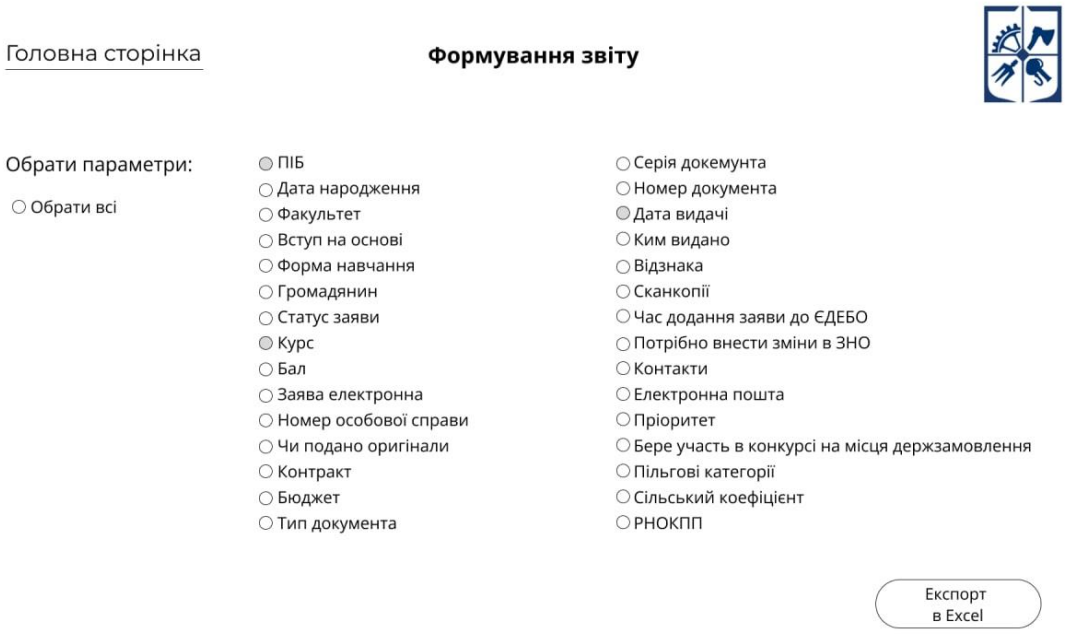

Рисунок 5.6 – Сторінка формування звіту.

Працівник ПК має змогу обрати саме ті дані для звіту, які йому потрібні. А сам звіт з вибраними даними користувач може легко експортувати в Excel, натиснувши «Експорт в Excel» у нижньому правому куті екрану.

5.3. Ролі у системі, які можуть бути надані користувачу

У створеній системі існує дві ролі для користувачів:

1) адміністратор;
2) робітник ПК.

Адміністратор має свій створений аккаунт та має можливість додавати та видаляти аккаунти для робітників ПК. Дані про абітурієнтів, їх кількість та їхні ролі зберігаються у базі даних сайту та доступні тільки членам приймальної комісії. Неавторизований користувач не матиме змоги увійти у систему та переглянути дані. Тому опишемо кожну роль більш детально.

5.4. Роль «Адміністратор»

Тільки користувач, який має доступ до системи, тобто створений аккаунт адміністратором, може здійснювати якійсь дії у даному програмному застосунку.

Адміністратор відповідає за надання доступу до ресурсу працівникам приймальної комісії. Він може додавати та видаляти аккаунти у системі.

5.5. Роль «Робітник ПК»

Член приймальної комісії має право переглядати усі особові справи усіх абітурієнтів, виконувати пошук по одному чи декільком параметрам, вносити зміни у дані або їх видаляти. Також робітник має змогу швидко та просто сформувати звіт з подальшим експортом в Excel та вибрати дані, які будуть там відображені.

ВИСНОВКИ

Створена автоматизована система з обробки даних абітурієнтів є результатом даної дипломної роботи. Протягом дипломної практики було спроектовано і реалізовано веб-сайт, який допоможе робітникам приймальної комісії нашого університету та зможе автоматизувати деякі її процеси, такі як створення звітів по заданим параметрам.

В результаті дослідження вимог самих робітників ПК та аналізу предметної області було визначено основі вимоги та дії, які має виконувати система. На основі отриманої інформації були обрані технології для реалізації та створення веб-додатку. Отож, спроектована система дозволить користувачам швидко та просто здійснювати пошук, сортувати по обраним фільтрам та передивлятись дані по заданих параметрах, а також робити звіти з обраними критеріями.

Для захисту даних була реалізована авторизація та виконано розподілення доступу для авторизованих і не авторизованих користувачів, адже у створеному програмному застосунку зберігаються особисті дані абітурієнтів. Саме тому було створена роль адміністратора веб-додатку, який має можливість надавати доступ або видаляти користувачів.

Для системи був створений зрозумілий, мінімалістичний і легкий у користуванні інтерфейс. Проектування виконувалось в середовищі розробки Visual Code з використанням мови PHP, HTML, CSS.

Для даного веб-додатку можуть бути актуальні наступні доробки:

- створення сховища для електронних копій документів абітурієнтів;
- додання можливості студенту бачити себе у рейтингу;
- створення паттернів параметрів для звіту, які зможе редагувати користувач.

Таким чином, досягнута мета, поставлена в даній дипломній роботі, а саме: створення автоматизованої системи з обробки даних абітурієнтів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Лаура Томпсон, Люк Веллінг Розробка WEB-додатків на PHP та MySQL: навч. посібник Вид. 2-ге, переробл. і допов., Лос-Анджелес, 2003. 672 с.
2. Бублик В. В. Об’єктно-орієнтоване програмування: навч. посібник. Київ, 2015. 624 с.
3. Пол Бейнон-Девіс Системи бази даних: навч. посібник. Лондон: Palgrave Macmillan, 2003. 273 с.
4. Сибельшатц А., Сударшан С. Концепції системи баз даних: навч. посібник Вид 7-е. Нью-Йорк: McGrawHill, 2011, 1344 с.
5. Herrera F. Multilabel classification: problem analysis, metrics and techniques – Springer International Publishing Switzerland, 2016. P. 76.
6. Guttag J. V. Introduction to Computation and Programming Using Python: With Application to Understanding Data. — MIT Press. ISBN, 2006. P. 123.
7. PHP: веб-сайт. URL: https://www.php.net/manual/ru/language.basic-syntax.php (дата звернення: 07.05.2022).
8. Перевозчикова О. В. Інформаційні системи і структура даних: навч. Посібник. Київ, 2007. 288 с.
9. Ерік Фрімен, Елізабет Россман, Бейт Бернс, Кеті Сієра Head First. Патерни проєктування: навч. посібник / пер. з англійської Ганни Якубовської. Київ: Фабула, 2020. 672 с.
10. Синтаксис CSS: приклади використання: веб-сайт. URL: https://hi-news.pp.ua/kompyuteri/13302-sintaksis-css-prikladi-vikoristannya.html (дата звернення: 02.05.2022).
11. Класифікація БД: варіанти, моделі даних і основні характеристики: веб-сайт. URL: https://hi-news.pp.ua/kompyuteri/16677-klasifkacya-bd-varanti-model-danih-osnovn-harakteristiki.html (дата звернення: 10.05.2022).
12. Береза А. М. Основи створення інформаційних систем: навч. посібник Вид 2-ге, перероблене і доп., Київ: КНЕУ, 2001. 214 с.

13. Синтаксис мови HTML: веб-сайт. URL: https://stud.com.ua/97588/informatika/sintaksis_movi_html (дата звернення: 03.05.2022).

14. Надія Балик, Віктор Мандзяк MySQL: лабораторний практикум. Полтава, Навчальна книга – Богдан, 2008. 88 с.

15. Програма Model-View-Controller MVC: веб-сайт. URL: https://hi-news.pp.ua/kompyuteri/14628-programa-model-view-controller-mvc-scho-ce-osoblivost-opis.html (дата звернення: 10.05.2022).

16. Борис Погріщук, Ю. Паночишин Комп'ютерна техніка та інформаційні технології: навч. посібник. Київ, 2012. 263 с.

17. Роман Мельник Програмування веб-застосувань (фронт-енд та бек-енд): навч. посібник. Львів: Львівська політехніка, 2018. 248 с.

18. Веб-розробка: веб-сайт. URL: https://dou.ua/lenta/articles/web-development-status-2020/ (дата звернення: 12.05.2022).

ДОДАТОК А

Інформаційна-автоматизована система з обробки даних абітурієнтів

Текст програми

УКР.НТУУ“КПІ ім. Ігоря Сікорського”ТЕФ_АПЕПС_ТР81333_22б12-1

Аркушів 5

Київ – 2022

Приклад програмного коду PHP експорту даних з файлу Excel до бази даних системи (export.php)

```
public function export() {
    if(!empty($_FILES['xls']['tmp_name'])) {
      $file = $this->uploadFile($_FILES);

      if($this->xlsToMysql($file)) {
        header('Location:index.php');
      }
    }
  }

public function xlsToMysql ($file) {
    $this->model = $this->getModel();
    $this->xls = $this->getPhpExcel($file);

    $this->xls->setActiveSheetIndex(0);
    $sheet = $this->xls->getActiveSheet();

    $rowIterator = $sheet->getRowIterator();
    $arr = array();
    foreach ($rowIterator as $row) {

      if($row->getRowIndex() != 1) {

        $cellIterator = $row->getCellIterator();
```

```
            foreach ($cellIterator as $cell) {
              $cellPath = $cell->getColumn();
              if(isset($this->config->cells[$cellPath])) {

                if($cellPath == 'U') {
                  if($cell->getCalculatedValue() == '00.00.0000' || $cell->getCalculatedValue() == '') {
                    $t = '0000-00-00';
                  }
                  else {
                  $t = date( 'Y-m-d', \PHPExcel_Shared_Date::ExcelToPHP( $cell->getCalculatedValue() ) );
                  }
                  $arr[$row->getRowIndex()][$this->config->cells[$cellPath]] =$t;
                  continue;
                }

                if($cellPath == 'F') {
                  if($cell->getCalculatedValue() == '00.00.0000' || $cell->getCalculatedValue() == '') {
                    $t = '0000-00-00';
                  }

        else {
                  $t = date( 'Y-m-d', \PHPExcel_Shared_Date::ExcelToPHP( $cell->getCalculatedValue() ) );
                  }
                  $arr[$row->getRowIndex()][$this->config->cells[$cellPath]] =$t;
                  continue;
```

```
                }
                $arr[$row->getRowIndex()][$this->config->cells[$cellPath]] = $cell->getCalculatedValue();
              }
              }

          }
        }

        $res = $this->model->inserExcel($arr);
        if(!$res) {
          exit();
        }
        return TRUE;
      }
    public function insertExcel($arr) {

        $fields = '';

        foreach($arr[2] as $key => $cell) {
          $fields .= '`'.$key.'`',';
        }
        $fields = trim($fields,',');

        $str = '';
        // INSERT INTO `main` (``,``,``..) VALUES ("","",),(),(),();
        foreach($arr as $item) {
          $str .= "(";
          foreach($item as $cell) {
```

```
      $str .= "'".$this->db->real_escape_string($cell)."',";
    }
    $str = trim($str,",");
    $str .= "),";
  }
  $str = trim($str,",");
  $query = "INSERT INTO `main` (".$fields.") VALUES ".$str;

  $result = $this->db->query($query);
  if($result) {
    return TRUE;
  }
 }
public function insertExcel($arr) {
  $fields = '';

  foreach($arr[2] as $key => $cell) {
    $fields .= '`'.$key.'`'.',';
  }
  $fields = trim($fields,',');

  $str = '';
  // INSERT INTO `main` (``,``,``..) VALUES ("","",),(),(),();
  foreach($arr as $item) {
    $str .= "(";
    foreach($item as $cell) {
      $str .= "'".$this->db->real_escape_string($cell)."',";
    }
    $str = trim($str,",");
```

```
      $str .= "),";
    }
    $str = trim($str,",");
    $query = "INSERT INTO `main` (".$fields.") VALUES ".$str;
    $result = $this->db->query($query);
    if($result) {
      return TRUE;
    }
  }
```

Приклад програмного коду PHP входу у систему (login.php)

```
<?php
namespace Controllers;

use Models\Order;
use Models\User;
class MainController
{
    public function index($view = '')
    {
        if ($view) {
            if (($view == 'login' || $view == 'reg') && !empty($_SESSION['user'])) {
                header('Location: /');
            } else {
                include "Views/{$view}.php";
            }
        }
    }
```

```
    public function admin($view)
    {
        if (isset($_SESSION['user']) && $_SESSION['user']['name'] == 'admin') {
            $orderObj = new Order();
            $orders = $orderObj->getOrders();
            $orderTypes = Order::TYPE_ORDER;
            include "Views/{$view}.php";
        } else {
            header('Refresh:2; url=/');
            echo 'Ви не маєте доступа до аккаунту адміністратору. ';
        }
    }

    public function login()
    {
        $user = new User();
        $loginUserArr = $user->login($_POST);

        if ($loginUserArr['isAdmin'] == true) {
            $_SESSION['user']['id'] = 0;
            $_SESSION['user']['name'] = 'admin';
            header('Location: /admin');
        }
        $loginUser = $loginUserArr['user'];
        if (empty($loginUser)) {
            header('Refresh:2; url=/login');
            echo 'Не знайдено користувача, спробуйте ще. Вас буде перенаправлено через 2с';
        } else {
```

```
            $_SESSION['user']['id']   = $loginUser['id'];
            $_SESSION['user']['name'] = $loginUser['name'];
            header('Location: /account');
        }
    }

    public function registration()
    {
        $user = new User();
        $statusRegister = $user->register($_POST);

        if ($statusRegister) {
            header('Location: /login');
        } else {
            header('Refresh:2; url=/registration');
            echo 'Дані введені невірно, спробуйте ще. Вас буде перенаправлено через 2с';
        }
    }

    public function logout()
    {
        $user = new User();
        $user->logout();
        header('Location: /');
    }

    public function account($view)
    {
```

```
            if (isset($_SESSION['user']) && $_SESSION['user']['name'] != 'admin') {
                $orderObj = new Order();
                $orderTypes = Order::TYPE_ORDER;
                $userId = $_SESSION['user']['id'];
                $userOrders = $orderObj->getOrdersbyUserId($userId);
                include "Views/{$view}.php";
            } else {
                header('Refresh:1; url=/login');
                echo 'Авторизуйтесь для доступу в аккаунт. Вас буде перенаправлено на головну сторінку через 1с';
            }
        }
```