НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

Навчально-науковий інститут атомної та теплової енергетики

кафедра інженерії програмного забезпечення в енергетиці

ДО ЗАХИСТУ ДОПУЩЕНО
В.о. завідувача кафедри
_____ Олександр КОВАЛЬ
«____»_____2023 р.

Дипломна робота
на здобуття ступеня бакалавра за освітньо-професійною програмою «Інженерія програмного забезпечення інтелектуальних кібер-фізичних систем і веб-технологій» спеціальності 121 Інженерія програмного забезпечення на тему: «Інформаційна система моніторингу успішності студентів»

Виконав:
студент IV курсу, групи _ТВ-91____
Кривда Дмитро Олександрович____ _____
(прізвище, ім'я, по батькові) (підпис)

Керівник:
Бандурка О. І. старший викладач___ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент:
Професор кафедри ЦТЕ, д.т.н., проф. Отрох С.І. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент_____
(підпис)

Київ – 2023

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Навчально-науковий інститут атомної та теплової енергетики
Кафедра інженерії програмного забезпечення в енергетиці
Рівень вищої освіти перший (бакалаврський)
Спеціальність 121 Інженерія програмного забезпечення
Освітньо-професійна програма «Інженерія програмного забезпечення інтелектуальних кібер - фізичних систем і веб-технологій»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ Олександр КОВАЛЬ
«____»_____2023 р.

ЗАВДАННЯ
на дипломну роботу студенту

_____Кривді Дмитру Олександровичу_____
(прізвище, ім'я, по батькові)

1. Тема роботи «Інформаційна система моніторингу успішності студентів»_____
_____
Керівник роботи Бандурка Олена Іванівна, старший викладач_____
(посада, науковий ступінь, вчене звання, прізвище та ініціали)
затверджені наказом по університету від "**29**" травня 2023 року №2039_____
2. Строк подання роботи 12.06.2023р._____
3. Вихідній дані роботи мова програмування мова програмування JavaScript, платформа Node.js, фреймворк Vue JS, мова запитів SQL, мова Python, фреймворк Django._____
4. Зміст (дипломної роботи) пояснювальної записки (перелік завдань, які потрібно розробити) створити систему моніторингу успішності студентів, система повинна відображати дані про успішність студентів у зручному виді, перегляд середнього балу за факультетом, кафедрою, спеціальністю, групою, відстеження успішності на протязі певного проміжку часу, представлення шкали оцінок за п'ятибальною. системою оцінювання.

5. Перелік ілюстративного матеріалу аналіз існуючих аналогів до програми демонстрація інтерфейсу, відображення виконування алгоритмів запитів до бази даних_____

6. Дата видачі завдання "**30**" вересня 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Строки виконання етапів роботи	Примітка
1	Отримання завдання	30.09.2022	Виконано
2	Дослідження предметної області	02.10.2022-16.04.2023	Виконано
3	Розробка програмного продукту	17.04.2023-21.05.23	Виконано
4	Захист програмного продукту	15.05.2023-19.05.2023	Виконано
5	Оформлення дипломної роботи	22.05.2023-04.06.2023	Виконано
6	Передзахист	05.06.2023-11.06.2023	Виконано
7	Захист	19.06.2023-23.06.2023	Виконано

Студент _____ Кривда Д. О.
(підпис) (ім'я, прізвище)

Керівник роботи _____ Бандурка О. І.
(підпис) (ім'я, прізвище)

РЕФЕРАТ

Структура та обсяг дипломної роботи. Робота 70 сторінок, кількість рисунків дорівнює 18, робота містить 2 додатки та 20 бібліографічних посилань.

Метою роботи інформаційної системи моніторингу успішності студентів є надання зручного та ефективного інструменту для відстеження та аналізу навчальних досягнень студентів.

Для досягнення поставленої мети виконано такі завдання:

- проаналізовано аналогічні системи моніторингу успішності;

Розроблено такі інструменти для моніторингу:

- відстеження середнього балу за факультетом, кафедрою, спеціальністю, групами;

- відображення оцінок за п’яти бальною та сто бальною шкалою;

- вивід 3 кращих студентів;

- відстеження оцінок за певний проміжок часу.

Практичне значення одержаних результатів полягає в класифікації та виведенню даних у зручній формі для подальшого аналізу. Всі ці рішення надають можливість максимально точного збору даних та їх виведення для інтерфейсу, де користувач зможе відстежувати результат навчального процесу.

Апробація результатів дипломної роботи

Кривда Д.О., Бандурка О.І., Свинчук О.В. Система автоматизованого планування бізнес-процесів для контингенту кафедри. Матеріали XXIII Всеукраїнської науково-технічної конференції молодих вчених, аспірантів та студентів «Стан, досягнення та перспективи інформаційних систем і технологій – 2023». Одеса, 20-21 квітня 2023 р. С.151-153.

https://ontu.edu.ua/download/konfi/2023/Conference_abstract-IT-21-22-04-23.pdf

Ключові слова: моніторинг, запити, обробка даних, SQL-запити, Django, моделі до БД, кластеризація, класи запитів, стек технологій.

ABSTRACT

Structure and scope of the thesis. The work is 70 pages, the number of figures is 18, the work contains 2 appendices and 20 bibliographic references.

The purpose of the information system for monitoring students' progress is to provide a convenient and effective tool for tracking and analyzing students' educational achievements.

To achieve the set goal, the following tasks were completed:

- similar success monitoring systems were analyzed;

The following monitoring tools have been developed:

- tracking of the average score by faculty, department, specialty, groups;
- display of grades on a five-point and one-hundred-point scale;
- withdrawal of the 3 best students;
- tracking grades over a certain period of time.

The practical significance of the obtained results lies in the classification and output of data in a convenient form for further analysis. The project architecture was developed, as well as the interface for the monitoring program was implemented.

All these solutions provide the possibility of the most accurate data collection and their output for the interface, where the user can monitor the result of the educational process.

Approbation of the results of the thesis

Kryvda D.O., Bandurka O.I., Svynchuk O.V. System of automated planning of business processes for the contingent of the department. Materials of the 23rd All-Ukrainian scientific and technical conference of young scientists, graduate students and students "State, achievements and prospects of information systems and technologies - 2023". Odesa, April 20-21, 2023. P.151-153.

Keywords: monitoring, queries, data processing, SQL queries, Django, database models, clustering, query classes, technology stack.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, ТЕРМІНІВ ТА УМОВНИХ ПОЗНАЧЕНЬ

JSON – це стандартний текстовий формат для відображення структурованих даних на основі синтаксису об’єкту JavaScript.

СУБД – (система управління базами даних) – це комплексне програмне забезпечення, яке є призначене для створення та роботи з базами даних.

БД – це база даних, що використовується для зберігання інформації в залежності від проекту.

HTML (HyperText Markup Language - мова розмітки гіпертексту) - стандартизована мова розмітки документів для перегляду вебсторінок у браузері.

CSS – (Cascading Style Sheets «каскадні таблиці стилів») - формальна мова декорування та опису зовнішнього вигляду документа (веб-сторінки), написаного з використанням мови розмітки, найчастіше це HTML.

JavaScript – це кросплатформна об’єктно-орієнтована мова програмування, яка зазвичай використовується для створення інтерактивних веб сторінок.

Vue.js – JavaScript-фреймворк з відкритим вихідним кодом для створення інтерфейсів користувача. Легко інтегрується у проекти з використанням інших JavaScript-бібліотек.

Python – високорівнева мова програмування загального призначення з динамічною строгою типізацією та автоматичним управлінням пам'яттю, орієнтована на підвищення продуктивності розробника, читання коду та його якості, а також на забезпечення переносимості написаних на ньому програм.

Django (Джанго) – високорівневий відкритий Python-фреймворк (програмний каркас) для розробки вебсистем.

ВСТУП

У сучасному світі інформаційні технології відіграють важливу роль у багатьох сферах життя, включаючи освіту. Системи моніторингу успішності студентів є невід'ємною частиною сучасного освітнього процесу, які дозволяють університетам та іншим навчальним закладам стежити за успішністю та активністю студентів, оцінювати ефективність викладання та надавати студентам зворотний зв'язок про їхні навчальні досягнення.

Для автоматизації процесу моніторингу та підвищення ефективності управління навчальним процесом розробка інформаційної системи моніторингу успішності студентів є актуальним завданням. У дипломному проекті буде розглянуто створення такої системи з використанням фреймворку Python Django та бази даних Microsoft SQL Server. Даний звіт описує технологічний стек, використаний у процесі розробки системи, включаючи Python Django, models у Python Django та Microsoft SQL Server.

Передбачається створення функціоналу, який здійснюватиме моніторинг статистичних даних успішності: загальний рейтинг студентів, середній бал кожного студента, середній бал кафедри, середній бал груп, спеціальностей, напрямків, які є на кафедрі, а також графіки середнього балу за певний проміжок часу.

1 ЗАДАЧА ІНФОРМАЦІЙНОЇ СИСТЕМИ МОНІТОРИНГУ СТУДЕНТІВ

Останнім часом, у зв’язку з повною інформатизацією суспільства, освітній процес не став виключенням серед інших сфер життя, що зазнали певних інноваційних змін. Системи моніторингу успішності студентів є невід'ємною частиною сучасного освітнього процесу, які дозволяють університетам та іншим навчальним закладам стежити за успішністю та активністю студентів, оцінювати ефективність викладання та надавати студентам зворотний зв'язок про їхні навчальні досягнення.

Мета інформаційної системи моніторингу успішності студентів – забезпечити ефективне управління навчальним процесом, підвищити якість освіти та допомогти студентам у їхній навчальній діяльності.

Для досягнення цієї мети інформаційна система має вирішувати такі завдання:

- Автоматизувати процес збору, зберігання та обробки інформації про студентів, їх успішність та активність у навчальному процесі;
- Забезпечити моніторинг успішності та активності студентів, а також надання студентам зворотного зв'язку про їхні навчальні досягнення;
- Оцінювати ефективність роботи викладачів та навчальних програм, надаючи аналітичні дані для прийняття управлінських рішень;
- Підвищити рівень інформаційної безпеки та захистити персональні дані студентів;
- Забезпечити зручний та ефективний інтерфейс для користувачів системи.

Вирішення цих завдань дозволить суттєво підвищити ефективність управління навчальним процесом та забезпечити якість освіти на вищому рівні.

2 АКТУАЛЬНІСТЬ ІНФОРМАЦІЙНОЇ СИСТЕМИ МОНІТОРИНГУ УСПІШНОСТІ СТУДЕНТІВ

В даний час домінуючою тенденцією розвитку суспільства є його інформатизація, що висуває особливі вимоги до сучасної освіти. Варто також відзначити зростаючу роль мас-медіа в суспільному житті, та й найбільш активного впровадження медіа технологій (мультимедіа технологій) у всі сфери життєдіяльності. І це, природно, має своє відображення і в освітній сфері, що закріплено в концепції довгострокового розвитку до 2020 р., суть якої полягає у створенні механізму ефективного і динамічного функціонування та розвитку освіти, що забезпечує вирішення зовнішніх завдань, що стоять перед ним, у відповідність до його логіки внутрішнього розвитку за умов сучасного інформаційного суспільства.

Активне впровадження інформаційно-комунікаційних технологій в освіту відбувається досить нерівномірно, що у свою чергу проявляється до певної міри в недостатній увазі до адміністративної частини процесу. І при цьому слід зазначити, що останнім часом ефективність навчання знизилася як з об'єктивних, так і суб'єктивних причин. Тому необхідні нові методичні розробки, технології, методи, прийоми, форми навчання, за яких можна повніше відповідати тим вимогам, які відображені в державних освітніх стандартах середньої, середньої спеціальної, вищої освіти, які пред'являє суспільство випускнику освітньої установи.

Система контролю та оцінки роботи учнів є найбільш консервативною та усталеною протягом багатьох років, незважаючи на всілякі інновації в цій галузі в системі освіти та реформи в цій системі в нашій країні та за кордоном. Існуюча система вже не може повною мірою задовольняти сучасні вимоги як до навчального процесу, де відбувається спільна діяльність педагога та студента, так і в результаті до якості підготовки випускників. Якщо при цьому враховувати характерну для теперішнього часу слабку мотивацію навчальної діяльності, то

можна вважати, що роль контролю та оцінки знань та умінь різко зростає. У рамках розвитку цього напряму було зроблено певні кроки. Як приклад можна навести використання електронних журналів, проектів «мережева школа». Але варто зазначити, що ці проекти не замінили собою паперові версії: так, в освітній установі одночасно вчитель веде і паперовий, і електронний журнали, що створює, з одного боку, додаткове навантаження на вчителя, а з іншого боку, не є повноцінним вирішенням проблеми. Варто зазначити, що журнал обліку відвідуваності та успішності вирішує лише частину завдань, які стоять перед педагогом. Але крім обліку успішності та відвідуваності учнів запроваджено та застосовуються й інші критерії.

У рамках вирішення проблеми успішності навчання, що є перспективним напрямом освітнього процесу, необхідно враховувати низку критеріїв для ухвалення правильного рішення. У рамках такого підходу можна говорити про запровадження в освітній процес поняття моніторингу успішності студентів.

Моніторинг успішності – це процес систематичного чи безперервного критеріального збору інформації про параметри багатокомпонентного об'єкта чи діяльності визначення тенденцій формування компетентності. На наш погляд, вирішенням проблеми моніторингу успішності може бути створення програмно-апаратного комплексу з інтегрованою системою автоматичного збирання даних. База даних програмно-апаратного комплексу повинна містити основну та необхідну інформацію, достатню для проведення аналізу даних, що надходять. Сформулюємо методичну задачу: необхідно підвищити ефективність проведення моніторингу успішності та мінімізувати вплив суб'єктивної складової у рамках підвищення якості навчання. Варто звернути особливу увагу на мінімізацію суб'єктивної складової у навчальному процесі.

Система підтримки прийняття рішень (СППР) створюється для прийняття складних рішень з опорою на багатофакторну інформацію, отриману, як правило, за допомогою інтеграції баз даних та управлінських систем.

На мій погляд, вирішення сформульованої задачі можливе за допомогою часткової автоматизації процесу збирання даних, що можна зробити за допомогою адаптації технологій обліку в освітньому процесі. Як один із методів рішення може виступити впровадження системи контролю та управління доступом, що успішно застосовується в різних сферах життєдіяльності. Ця система була розроблена спочатку як механізм захисту від несанкціонованого фізичного доступу до об'єкта, що охороняється. Таким чином, можна зробити висновок, що система може бути впроваджена в освітній процес при внесенні низки змін.

Варто відзначити, що природні обмеження розробленого програмного забезпечення не можуть бути повною мірою адаптовані для вирішення сформульованої задачі, таким чином, можна зробити висновок, що реалізація можлива лише використовуючи апаратну частину системи контролю управління доступом (СКУД). В рамках реалізації програмної частини комплексу можливе використання існуючого програмного забезпечення (наприклад, MS SQl Server), а також розробка за допомогою середовища об'єктно-орієнтованого програмування спеціалізованого програмного забезпечення (наприклад, Python та його фреймворк Django); розробка web-базованої програми. Варто зазначити, що повноцінна та ефективна реалізація можлива у рамках обраного стека комп'ютерних технологій.

Реалізація за допомогою середовища об'єктно-орієнтованого програмування має ряд переваг: написання програми можливе з урахуванням усіх вимог, можливість реалізації доступу різного рівня, можливість реалізації клієнтської та серверної частини, віддалене розміщення бази даних.

Найбільш перспективним, на наш погляд, способом є створення web-дозування, що базується. Такий спосіб створення має низку переваг: розробка з урахуванням усіх вимог; можливість реалізації доступу різного рівня; віддалене розміщення бази даних; відсутність необхідності інсталяції доступ здійснюється за допомогою браузера користувача з будь-якого пристрою, що має вихід в

Інтернет. Реалізація web-базованого додатка дозволить підвищити інформативну складову, оскільки користувач може отримати необхідну інформацію незалежно від свого розташування. При цьому можна забезпечити різнорівневий доступ користувачів до необхідної інформації.

2.1 АНАЛІЗ ІСНУЮЧИХ СИСТЕМ

На сьогоднішній день існує дуже багато різних програм, веб-додатків, які дозволяють отримати інформацію стосовно успішності студентів. Зазвичай дана інформація використовуються у навчальних закладах відповідними викладачами та експерту для моніторингу та аналізу даних про навчальний процес студентів. А саме аналіз відбувається на отриманих результатах, які отримуються впродовж навчального процесу. Основи які закладені до цих програм це, зручний та простий користувацький інтерфейс. Також, в залежності від сфери моніторингу такі сервіси надають різній функціонал для більш точної обробки та формування даних.

2.2 Продукт "LMS"

Після визначення актуальності даної системи, наступний крок був розгляд аналогів, аналіз їх функціональності, стилю оформлення, архітектурна складова системи, взаємодія між програмою та користувачем, визначення рівня доступу до програмі.

LMS (Learning Management System) - це програмне забезпечення, розроблене для організації, управління та поширення навчальних матеріалів у віртуальному навчальному середовищі. Основна мета використання LMS полягає у поліпшенні процесу навчання і сприянні успішності студентів.

LMS має низку своїх сильних сторін та явних переваг у сфері організації роботи зі студентами.

Основні плюси використання LMS:

1. Централізоване зберігання та доступ до матеріалів: LMS дозволяє зберігати всі навчальні матеріали в одному місці, що спрощує їх доступність для студентів. Викладачі можуть завантажувати лекції, підручники, відео та інші ресурси, які студенти можуть переглядати в будь-який зручний для них час.

2. Інтерактивне взаємодія: LMS надає інструменти для взаємодії між студентами та викладачами. Це можуть бути форуми обговорень, системи миттєвих повідомлень, електронна пошта тощо. Це створює можливість для активного обміну ідеями, питаннями та спільної роботи над завданнями.

3. Оцінювання та надання зворотного зв'язку: LMS дозволяє викладачам створювати онлайн-тести, завдання та інші форми оцінювання. Вони можуть бути автоматично перевірені, що дозволяє швидко надати результати студентам. Також LMS забезпечує можливість надання зворотного зв'язку та індивідуальної підтримки для кожного студента.

4. Моніторинг прогресу: LMS дозволяє викладачам та адміністрації відстежувати прогрес та успішність студентів. Вони можуть бачити, які завдання були виконані, які матеріали були переглянуті, результати тестів та іншу інформацію. Це допомагає виявляти слабкі місця студентів та надавати додаткову підтримку.

5. Флексібільність та доступність: LMS забезпечує гнучкість у навчанні, оскільки студенти можуть отримувати доступ до матеріалів та завдань з будь-якого місця та в будь-який час. Він також сприяє дистанційному навчанню, що особливо актуально в сучасному світі.

Особливості програми LMS можуть варіюватися залежно від конкретної платформи, але вони зазвичай включають структуроване представлення курсів, можливість завантаження та організацію навчальних матеріалів, інструменти для комунікації та співпраці, оцінювання та моніторинг прогресу, аналітику та звітність, а також можливості настройки для відповідності потребам конкретного навчального закладу.

Як вже було сказано, LMS має багаторівневу роботу с програмою (рис. 2.1), багатий та зручний функціонал, ось кілька яскравих елементів інтерфейсу програми, які слугували основую при розробці власного інтерфейсу.

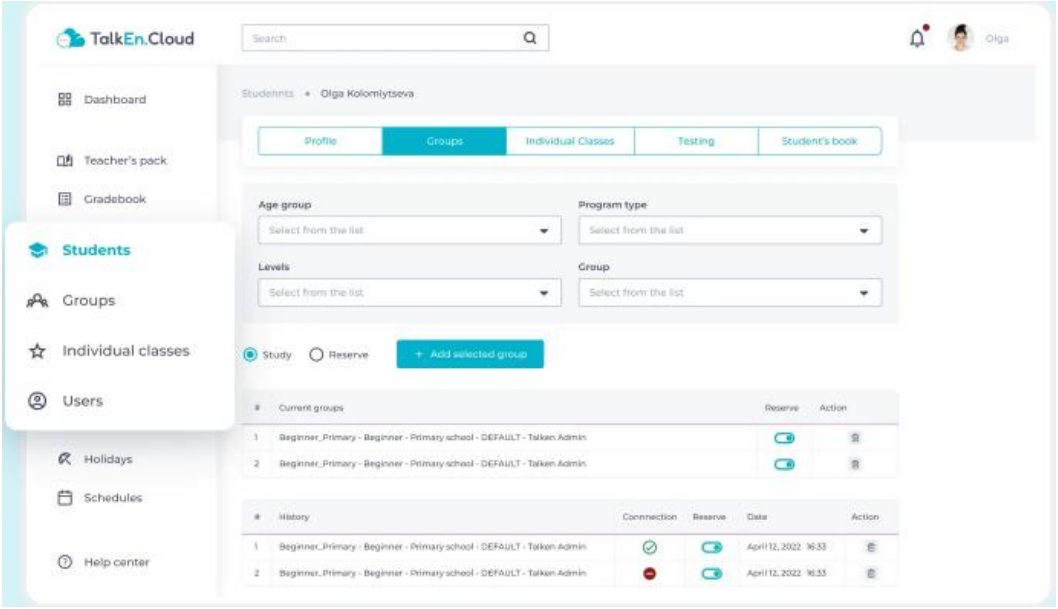


Рисунок 2.1 – Панель адміністратора в застосунку LMS

Панель адміністратора в програмі LMS надає широкий набір функціональних можливостей для управління системою та контролю над навчальними процесами. Ось детальний опис можливостей панелі адміністратора в програмі LMS:

- керування користувачами: адміністратор може керувати користувачами системи, включаючи студентів, викладачів та інших адміністраторів. він може додавати нових користувачів, редагувати їхні дані, встановлювати ролі та права доступу, блокувати акаунти або видаляти користувачів;
- управління курсами: адміністратор має можливість створювати, редагувати та видаляти курси. він також може налаштовувати параметри курсів, встановлювати дати початку та закінчення, призначати викладачів, керувати доступом студентів та налаштовувати категорії курсів;
- забезпечення безпеки: адміністратор може здійснювати заходи для забезпечення безпеки системи lms. це включає налаштування прав доступу до

системи, захист даних, резервне копіювання, моніторинг активності користувачів та захист від несанкціонованого доступу;

• моніторинг та аналітика: адміністратор може отримувати доступ до звітів, статистики та аналітичних даних щодо активності студентів, викладачів та курсів. він може відстежувати прогрес студентів, оцінки, активність в системі та інші важливі показники;

• налаштування системи: адміністратор може налаштовувати різні параметри системи lms, такі як мови, шаблони, логотипи, навігацію та інтерфейс користувач.

Наступний скріншот це особистий кабінет викладача в програма LMS, (рис. 2.2).

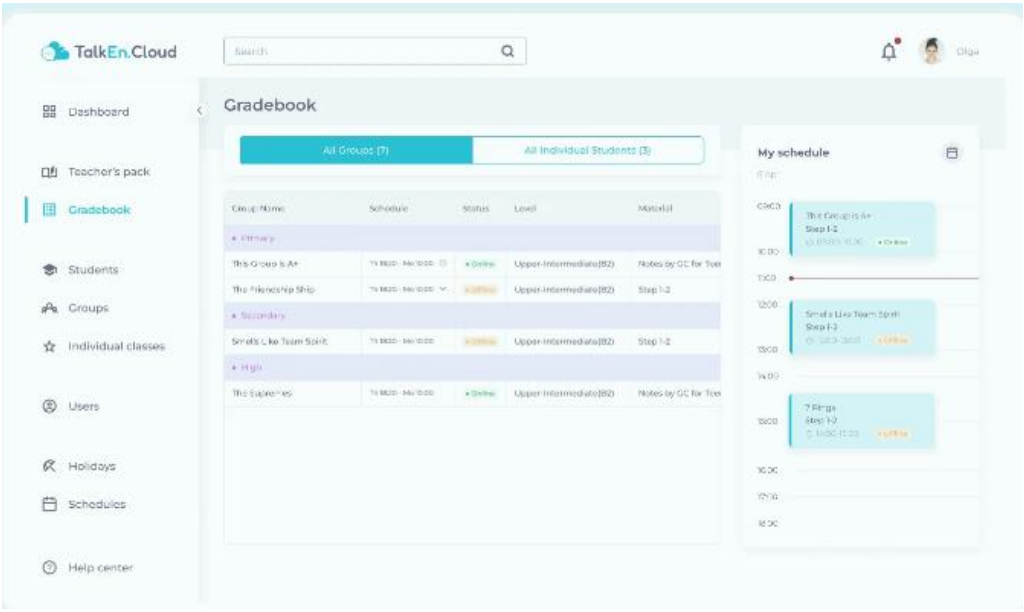


Рисунок 2.2 – Особистий кабінет викладача в застосунку LMS

Особистий кабінет викладача в програмі LMS надає викладачам ряд функціональних можливостей для ефективного управління навчальним процесом. Ось детальний опис можливостей особистого кабінету викладача:

• організація курсів: викладач може створювати та налаштовувати курси в особистому кабінеті. це включає встановлення назви курсу, опису, додавання

навчальних матеріалів, завдань та ресурсів, а також налаштування параметрів доступу для студентів;

- завантаження навчальних матеріалів: викладач може завантажувати лекції, презентації, додаткові матеріали, відео або аудіо файлів для подальшого використання студентами. вони будуть доступні студентам відповідно до розкладу курсу або потреби;
- створення завдань та тестів: викладач може створювати завдання, вікторини та онлайн-тести для оцінювання навчальних досягнень студентів. вони можуть містити питання різного типу, такі як багато вибірні, з заповненням пропусків, короткі відповіді тощо. викладач також може налаштовувати критерії оцінювання та автоматичне перевіряння відповідей;
- комунікація зі студентами: в особистому кабінеті викладач може спілкуватися зі студентами за допомогою форумів, системи миттєвих повідомлень або електронної пошти. це надає можливість вирішувати питання студентів, надавати індивідуальну підтримку та проводити обговорення;
- оцінювання та зворотний зв'язок: викладач може переглядати та оцінювати роботи студента.

Наступний скріншот це особистий кабінет учня в програмі LMS.

Кабінет студента (рис. 2.3), у програмі LMS надає широкий спектр функціональних можливостей для доступу до навчальних ресурсів та управління особистим навчальним процесом.

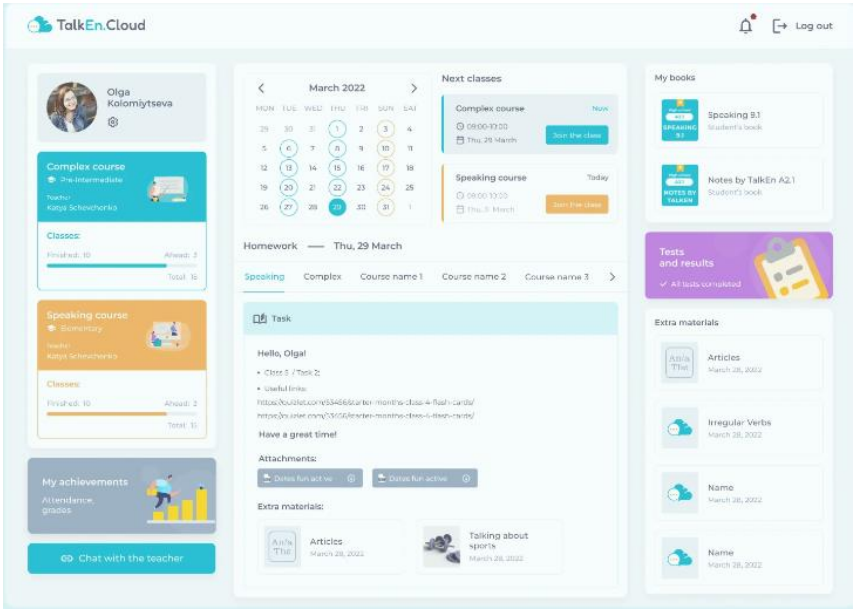


Рисунок 2.3 – Особистий кабінет учня в застосунку LMS

Ось детальний опис можливостей кабінету студента в програмі LMS:

- перегляд навчальних матеріалів: студент може отримати доступ до навчальних матеріалів, які викладачі завантажили в систему. це можуть бути лекції, підручники, відеоуроки, презентації та інші матеріали, які студент може переглядати в будь-який зручний для нього час;
- виконання завдань та тестів: студент може виконувати завдання, вікторини та тести, які викладачі створили в системі lms. вони можуть містити питання різного типу, такі як багатовибірні, з заповненням пропусків, короткі відповіді тощо. після виконання завдань студент може надіслати їх для оцінювання викладачем;
- комунікація з викладачем та співстудентами: кабінет студента надає можливість спілкуватися з викладачами та іншими студентами через форуми обговорень, систему миттєвих повідомлень або електронну пошту. це дозволяє студентам задавати питання, обговорювати матеріали курсу та спілкуватися з колегами;
- оцінювання та отримання зворотного зв'язку: студент може переглядати свої оцінки за завдання та тести, які виконав. викладачі можуть надати зворотний зв'язок щодо виконання завдань та оцінок, що дозволяє студентам зрозуміти свій прогрес та вдосконалити свої навички;
- моніторинг прогресу: студент може відстежувати свій прогрес у навчанні, переглядаючи свої оцінки, виконані завдання, прогрес по курсу та іншу статистику. це допомагає студентам оцінювати свої знання та розуміння теми;
- розклад та дедлайни: в кабінеті студента може бути розклад занять, який відображає дати та час проведення лекцій, семінарів або інших занять. також можуть бути вказані дедлайни для виконання завдань та відправки робіт;
- доступ до додаткових ресурсів: студент може мати доступ до додаткових ресурсів, таких як рекомендована література, посилання на веб-ресурси, додаткові матеріали для самостійного вивчення теми;

- налаштування профілю: студент може налаштовувати свій профіль, включаючи особисту інформацію, фотографію, налаштування повідомлень та інші параметри відображення.

Хоча система управління навчанням (LMS) має багато переваг, вона також має кілька потенційних недоліків.

Ось деякі з них:

1. Складність використання: Для деяких викладачів та студентів використання LMS може бути складним і вимагати часу на ознайомлення з інтерфейсом та функціями системи. Це може викликати певні труднощі та вимагати додаткової підтримки технічних спеціалістів.

2. Залежність від технології: Використання LMS передбачає наявність комп'ютера або іншого пристрою з доступом до Інтернету. Це означає, що студенти та викладачі, які не мають доступу до цих технологій, можуть бути виключені з можливості повноцінного використання системи.

3. Відсутність особистого контакту: Використання LMS може призвести до зменшення особистого контакту між викладачами та студентами. Відсутність фізичної присутності може ускладнити вирішення складних запитань або обговорення навчального матеріалу.

4. Великий обсяг інформації: LMS може містити велику кількість навчальних матеріалів, завдань та ресурсів. Це може призвести до перенасичення інформацією та ускладнити студентам орієнтуватися та знаходити необхідну інформацію.

5. Відсутність безпосереднього нагляду: LMS може не забезпечувати безпосереднього нагляду за студентами під час виконання завдань або тестів. Це може створювати можливість для шахрайства або підробки результатів.

2.3 Застосунок "CANVAS"

Canvas - це популярна програма управління навчанням (LMS), яка надає широкий спектр інструментів для навчання та спілкування в освітніх закладах. Вона має багато можливостей, спрямованих на полегшення навчання, співпраці та оцінювання. Ось детальний опис програми Canvas:

1. Мета використання: Основна мета Canvas полягає в покращенні процесу навчання і наданні зручної та ефективної платформи для взаємодії між викладачами та студентами. Вона сприяє організації навчального матеріалу, виконанню завдань, комунікації, співпраці та оцінюванню.

2. Головні плюси використання:

- зручний інтерфейс: canvas пропонує зрозумілий і інтуїтивний інтерфейс користувача, що полегшує навігацію та використання системи;
- широкий спектр функцій: canvas надає багато інструментів, таких як завдання, календар, форуми обговорень, система оцінювання, відеоконференції, миттєві повідомлення та багато іншого;
- гнучкість: canvas дозволяє викладачам налаштовувати курси згідно зі своїми потребами, включаючи структуру курсу, розклад занять, завдання та оцінки;
- мобільність: canvas має мобільні додатки для ios та android, що дозволяють студентам та викладачам отримувати доступ до курсів та матеріалів навіть зі своїх мобільних пристроїв;
- інтеграція з іншими системами: canvas може інтегруватися з іншими програмами та сервісами, такими як системи відеоконференцій, електронні ресурси бібліотек та інші, що забезпечує зручну і одночасно повноту використання системи;

особливості програми:

- модульність курсів: викладачі можуть створювати курси з розділами, підрозділами та модулями для організації навчального матеріалу;

- комунікація та співпраця: canvas надає можливості для обговорень у форумах, чатів, спільної роботи над завданнями та проектами;
- оцінювання та звітність: викладачі можуть встановлювати завдання, тести, опитування та інші форми оцінювання, а також відстежувати прогрес студентів та генерувати звіти;
- інтерактивність: canvas дозволяє використовувати різноманітні мультимедійні матеріали, такі як відео, аудіо, зображення та інтерактивні елементи, для залучення студентів у навчальний процес;
- canvas є потужною та гнучкою програмою управління навчанням, яка забезпечує зручність та ефективність навчання та спілкування у освітніх закладах;
- платформа canvas має багато переваг, які зробили її популярною серед освітніх закладів;

ось деякі головні плюси використання платформи canvas:

- зручний інтерфейс: canvas має інтуїтивно зрозумілий і легкий у використанні інтерфейс, що полегшує навігацію та забезпечує зручність використання для викладачів, студентів та адміністраторів;
- широкий спектр функцій: canvas пропонує багато інструментів та можливостей, що дозволяють викладачам створювати та керувати курсами, завданнями, оцінками, обговореннями, співпрацювати зі студентами, проводити віртуальні заняття, спілкуватися тощо;
- гнучкість та налаштовування: canvas надає можливість викладачам налаштовувати курси згідно зі своїми потребами, включаючи структуру курсу, навчальні матеріали, завдання, терміни подачі, критерії оцінювання тощо;
- мобільність: canvas має мобільні додатки для ios та android, що дозволяють студентам та викладачам доступатися до курсів, матеріалів, завдань, сповіщень тощо зі своїх мобільних пристроїв;

• зручна комунікація: canvas надає засоби для ефективної комунікації між викладачами та студентами, включаючи відправлення повідомлень, форуми обговорення, коментарі до завдань тощо;

• відстеження прогресу: canvas дозволяє студентам та викладачам відстежувати прогрес у навчанні, переглядати оцінки, коментарі до завдань, відстежувати виконання завдань та прогрес по курсу;

• інтеграція з іншими системами: canvas може інтегруватися з іншими платформами та сервісами, такими як відеоконференції, електронні ресурси бібліотеки, системи електронних підручників тощо;

• підтримка та спільнота користувачів: canvas має активну спільноту користувачів, де можна знайти підтримку, поради та ресурси для вдосконалення використання платформи.

Ці плюси роблять платформу Canvas потужним інструментом для організації навчання, співпраці та оцінювання, що сприяє поліпшенню якості освіти та ефективності навчального процесу.

Далі буде представлено певний функціонал застосунку Canvas.

Панель адміністратора (рис. 2.4), в програмі Canvas LMS надає доступ до різноманітних інструментів та налаштувань для керування системою та навчальними процесами.

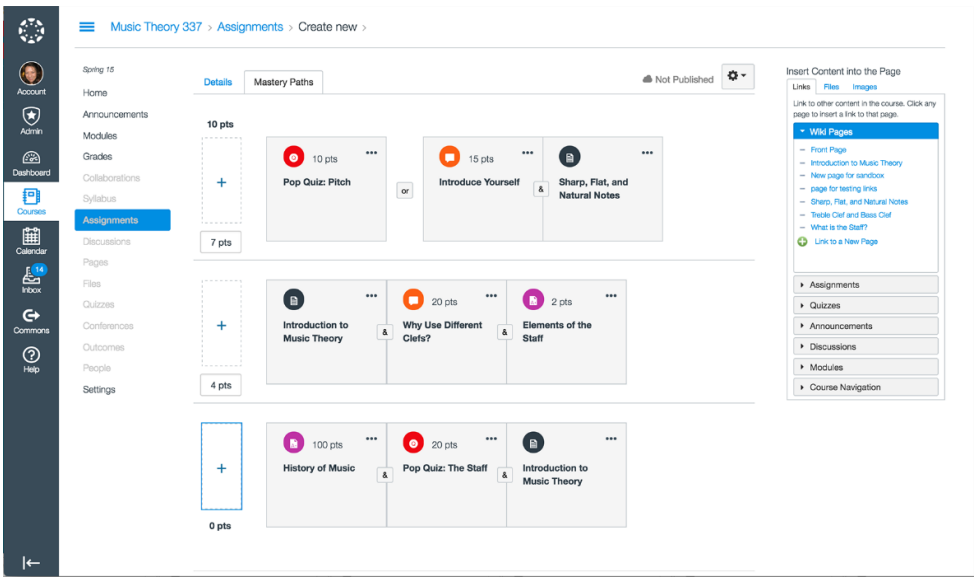


Рисунок 2.4 - Панель адміністратора в застосунку Canvas

Ось детальний опис можливостей панелі адміністратора у програмі Canvas:

1. Керування користувачами:

- створення та управління обліковими записами користувачів: адміністратор може створювати нові облікові записи для викладачів, студентів та інших користувачів системи, а також керувати даними профілю та правами доступу;
- управління ролями та дозволами: адміністратор може призначати ролі користувачам і надавати їм відповідні дозволи та обмеження в системі;
- імпорт та експорт користувачів: адміністратор може імпортувати масові дані про користувачів зовнішніх джерел або експортувати дані користувачів для зручного управління масовими діями;

налаштування курсів та програм:

- створення та керування курсами: адміністратор може створювати курси, встановлювати їх параметри, включаючи назву, опис, терміни, доступність та інші налаштування;
- реєстрація на курси: адміністратор може керувати процесом реєстрації студентів на курси, додавати або видаляти їх з курсів та керувати обмеженнями реєстрації;
- налаштування навчальних програм: адміністратор може створювати та керувати навчальними програмами, встановлювати послідовність курсів, вимоги та інші налаштування;

моніторинг та аналітика:

- статистика та звіти: адміністратор може отримувати статистику та звіти про активність користувачів, виконання завдань, оцінки та інші показники, що допомагають в оцінці ефективності навчання та виявленні проблемних областей;

• система трекінгу та безпеки: адміністратор може відстежувати доступ та активність користувачів, встановлювати правила безпеки, забороняти певні дії та моніторити потенційні загрози безпеці.

2. Налаштування системи:

• інтеграція з зовнішніми сервісами: адміністратор може налаштовувати інтеграцію з іншими системами, такими як системи електронного навчання, електронні бібліотеки, системи відеоконференцій тощо;

• налаштування безпеки та приватності: адміністратор може встановлювати правила безпеки, шифрування, політику паролів та інші заходи для забезпечення безпеки та приватності даних в системі;

• налаштування сповіщень та комунікації: адміністратор може налаштовувати сповіщення, повідомлення та комунікаційні засоби, щоб забезпечити ефективну комунікацію між користувачами системи.

Ці можливості дозволяють адміністраторам ефективно керувати системою Canvas LMS, налаштовувати її під потреби освітнього закладу та забезпечувати ефективність та безпеку навчального процесу.

Наступний функціонал це сторінка викладача.

Сторінка викладача (рис. 2.5), у програмі Canvas LMS надає викладачам широкий спектр інструментів та можливостей для керування та організації навчального процесу.

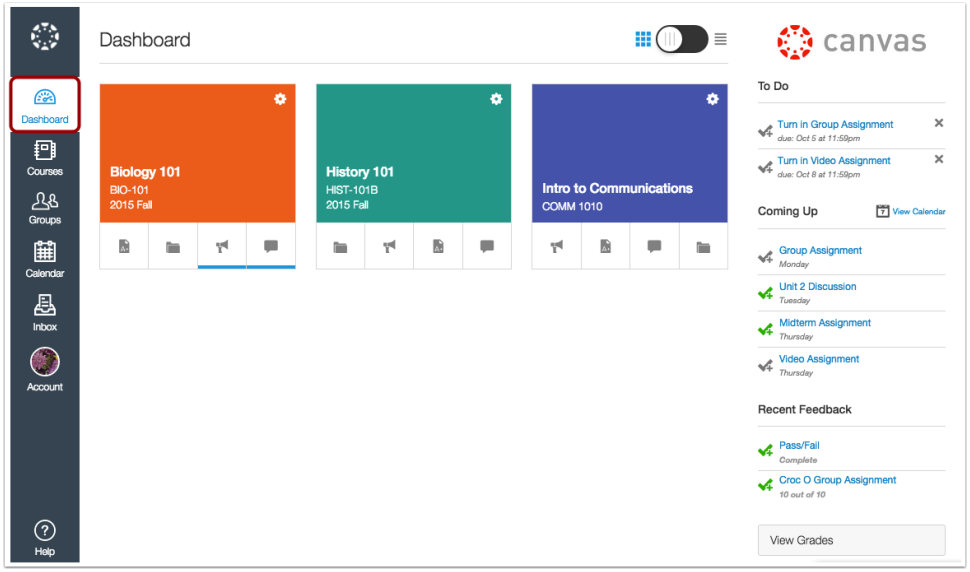


Рисунок 2.5 – Сторінка викладача в застосунку Canvas

Ось детальний опис можливостей сторінки викладача у програмі Canvas LMS:

1. Створення та управління курсами:

- створення курсів: викладач може створювати нові курси, встановлювати назву, опис, терміни, доступність та інші параметри курсу;
- керування курсами: викладач може налаштовувати структуру курсу, створювати модулі, завдання, оголошення, обговорення та інші розділи курсу;
- редагування матеріалів курсу: викладач може додавати навчальні матеріали, відео, презентації, файли, зовнішні посилання та інші ресурси до курсу;
- встановлення завдань та оцінювання: викладач може створювати завдання, встановлювати терміни подачі, встановлювати критерії оцінювання, оцінювати та коментувати роботи студентів;

комунікація та співпраця:

- обговорення та форуми: викладач може створювати обговорення та форуми, де студенти можуть спілкуватися, ділитися думками, задавати питання та обговорювати матеріали курсу;
- повідомлення та сповіщення: викладач може надсилати повідомлення студентам, сповіщати їх про оновлення, зміни в курсі та нагадування про важливі дати та завдання;
- групова робота: викладач може створювати групи студентів для спільної роботи над проектами, завданнями, дискусіями тощо.

2. Оцінювання та звітність:

- оцінювання завдань: викладач може оцінювати завдання студентів, встановлювати критерії оцінювання, надавати коментарі та оцінки;
- графіки та звіти: викладач може переглядати графіки успішності студентів, аналізувати прогрес, отримувати звіти про виконання завдань та іншу статистику.

3. Керування Студентами:

• реєстрація на курс: викладач може керувати процесом реєстрації студентів на курс, відкривати та закривати доступ до курсу, додавати або видаляти студентів;

• персоналізовані налаштування: викладач може налаштовувати відображення курсу для студентів, встановлювати правила та обмеження доступу до матеріалів.

4. Інтеграція та розширення:

• інтеграція з зовнішніми сервісами: викладач може інтегрувати зовнішні сервіси, такі як системи відеоконференцій, електронні ресурси бібліотеки, системи електронних підручників тощо;

• використання додатків: викладач може використовувати додатки, розширення та плагіни для розширення функціональності та налаштування системи під власні потреби.

Ці можливості роблять сторінку викладача у програмі Canvas LMS потужним інструментом для організації навчального процесу, співпраці зі студентами, оцінювання та моніторингу прогресу студентів та забезпечення ефективного навчання.

Наступний функціонал це сторінка студента (рис. 2.6).

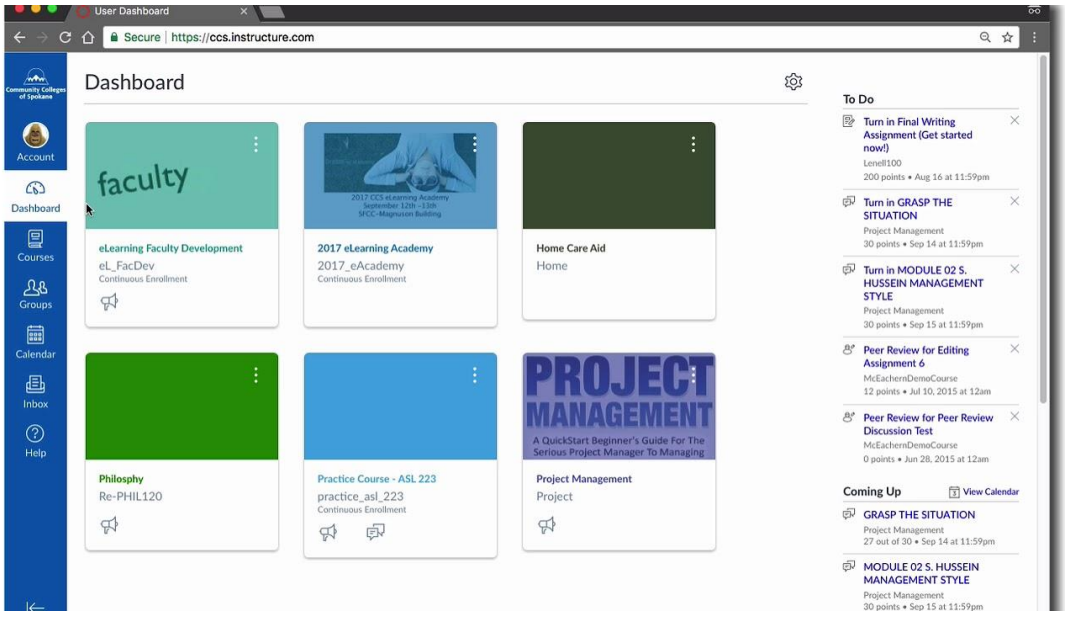


Рисунок 2.6 - Сторінка студента в застосунку Canvas

Сторінка студента у програмі Canvas LMS надає студентам доступ до всіх необхідних інструментів та ресурсів, що допомагають їм успішно навчатися.

Ось детальний опис можливостей сторінки студента у програмі Canvas LMS:

1. Перегляд та доступ до курсів:

- список курсів: Студент може бачити список всіх курсів, до яких він зареєстрований;
- матеріали курсу: Студент може переглядати навчальні матеріали, презентації, відео, завдання та інші ресурси, які надаються в рамках курсу;
- розклад занять: Студент може переглядати розклад занять, дати важливих подій та дедлайнів.

2. Завдання та оцінювання:

- перегляд завдань: Студент може бачити список завдань, які потрібно виконати, та їх терміни подачі;
- завантаження завдань: Студент може завантажувати виконані завдання та передавати їх для оцінювання;
- оцінки: Студент може переглядати свої оцінки за завдання та курс в цілому.

3. Комунікація та співпраця:

- обговорення та форуми: Студент може приєднуватися до обговорень, задавати питання, обмінюватися думками з викладачами та іншими студентами;
- повідомлення та сповіщення: Студент отримує повідомлення та сповіщення про важливі оновлення, зміни в курсі та нагадування про дедлайни.

4. Групова робота:

- робота в групах: Студент може співпрацювати з іншими студентами в межах проектів, завдань та обговорень;
- спільні файли та документи: Студент може спільно працювати з іншими студентами над документами, презентаціями та іншими матеріалами.

5. Особисті налаштування:

- профіль: Студент може налаштовувати свій профіль, додавати фотографію та інформацію про себе;
- налаштування повідомлень: Студент може налаштовувати спосіб отримання повідомлень та сповіщень.

Ці можливості сторінки студента у програмі Canvas LMS допомагають студентам ефективно організовувати свій навчальний процес, взаємодіяти з викладачами та іншими студентами, отримувати необхідну інформацію, виконувати завдання та відстежувати свій прогрес у навчанні.

Як і будь-яка програма, Canvas LMS має свої мінуси які були враховані під час своєї розробки.

- складність для початківців: Для користувачів, які вперше зустрічаються з програмою Canvas LMS, інтерфейс та функціонал можуть здаватися складними для освоєння. Потрібен час та навчання, щоб повністю оволодіти всіма можливостями платформи;
- велика кількість налаштувань: Canvas LMS надає багато налаштувань та опцій, що можуть бути корисними для досвідчених користувачів, але можуть збивати з толку початківців або вимагати багато часу для правильного налаштування системи;
- обмежена можливість настройки інтерфейсу: В деяких випадках, користувачі можуть відчувати обмежені можливості настройки інтерфейсу за своїми потребами та вимогами. Деякі користувачі можуть прагнути до більшої гнучкості та кастомізації, яку не завжди можна знайти у програмі Canvas LMS;
- нестабільність та технічні проблеми: Як будь-яка програмна система, Canvas LMS може бути схильна до технічних проблем, помилок або непередбачених ситуацій. Це може призвести до недоступності системи, втрати даних або інших незручностей для користувачів;
- вартість: Використання програми Canvas LMS може бути пов’язано з витратами, особливо для великих установ або організацій. Ліцензійні витрати,

підтримка та інші витрати можуть становити фінансове виклик для деяких користувачів.

Важливо враховувати, що деякі з перерахованих мінусів можуть бути відносної природи і залежати від індивідуальних потреб та очікувань користувачів. Кожна організація або особа повинна ретельно оцінити переваги та недоліки перед використанням програми Canvas LMS або будь-якої іншої системи управління навчанням.

Висновки по розділу

Порівнявши програми Canvas LMS і LMS (Learning Management System), з моєю програмою можна зробити деякі висновки, були визначені основні розбіжності між програмами, було визначено які переваги та недоліки має моя програма в порівнянні з двома попередніми.

Обидві програми є потужними системами управління навчанням, що надають широкий спектр можливостей для організації навчального процесу. Вони дозволяють створювати та керувати курсами, надавати навчальні матеріали, спілкуватися з викладачами та іншими студентами, виконувати завдання та відстежувати прогрес у навчанні.

Canvas LMS відзначається своєю простотою використання та інтуїтивним інтерфейсом, що полегшує користування для студентів та викладачів. Вона надає гнучкі можливості налаштування та кастомізації, а також велику кількість інтегрованих інструментів і додатків. Крім того, вона підтримує багатомовність, що дозволяє використовувати її в різних мовних середовищах.

LMS (Learning Management System) є більш загальним терміном і може включати різні програми та платформи, такі як Canvas LMS, Moodle, Blackboard і багато інших. Вибір конкретної системи LMS залежить від потреб і вимог користувача.

Кожна програма має свої переваги та недоліки. Наприклад, Canvas LMS відзначається своїм інтуїтивним інтерфейсом та простотою використання, а LMS може надати більше гнучкості та кастомізації в залежності від потреб користувача.

Загальний висновок полягає в тому, що обидві програми є потужними інструментами для управління навчанням і мають свої особливості. Вибір між ними залежить від конкретних потреб, вимог та пріоритетів користувача.

3 ПРОЕКТУВАННЯ БАЗИ ДАНИХ

Проектування бази даних є важливим етапом при розробці будь-якого інформаційного додатку. Це дозволяє ретельно проробити структуру даних, збереження та обробку даних, що відображають реальний світовий стан речей. Коректно спроектована база даних забезпечує ефективну та швидку роботу додатку, полегшує процес розробки та зменшує кількість помилок. В результаті, користувач отримує надійний та зручний продукт, що відповідає його потребам та очікуванням.

3.1 Стек технологій для розробки БД

Мова SQL (Structured Query Language) є стандартною мовою управління базами даних. Вона дозволяє взаємодіяти з базою даних, створювати та змінювати таблиці, вставляти, оновлювати та видаляти дані з бази даних. SQL дозволяє користувачам виконувати складні запити до бази даних, що дозволяє швидко та ефективно отримувати потрібну інформацію [3].

Django – це високорівневий веб-фреймворк, який дозволяє легко та швидко створювати веб-додатки на мові Python. Django має вбудовану підтримку баз даних, що дозволяє використовувати SQL Server та інші реляційні бази даних для зберігання даних. Django надає безліч готових інструментів для роботи з базами даних, таких як ORM (Object-Relational Mapping), який дозволяє легко взаємодіяти з базою даних без написання складних запитів SQL.

Django також забезпечує підтримку для міграцій баз даних, що дозволяє автоматично оновлювати базу даних при зміні моделей. Це дозволяє розробникам зосередитися на функціональності додатку, не хвилюючись про внутрішню структуру бази даних.

У проектуванні бази даних з використанням SQL Server та Django можна використовувати ORM Django для створення моделей бази даних та міграцій для оновлення структури бази даних. SQL Server може бути використаний для зберігання та обробки даних, а також для виконання складних запитів до бази даних. Застосування цих інструментів дозволяє створити ефективну та масштабовану базу даних для веб-додатків.

Побудова концептуальної моделі бази даних є важливим етапом проектування інформаційної системи і представлена на рисунку 4.1.

3.2 Розробка концептуальної та логічної моделі БД

Концептуальна модель є одним із ключових інструментів проектування інформаційних систем та баз даних. Вона описує сутності, атрибути та зв’язки між ними на високому рівні абстракції, відображаючи суть предметної галузі та її логічну структуру.

Концептуальна модель (рис. 3.1), дозволяє встановити загальне розуміння предметної області між замовником та розробниками, а також є основою для проектування логічної та фізичної моделей даних. Вона допомагає розробникам подати інформацію у структурованому вигляді та проаналізувати можливі проблеми та невідповідності в даних до початку фактичної реалізації інформаційної системи.

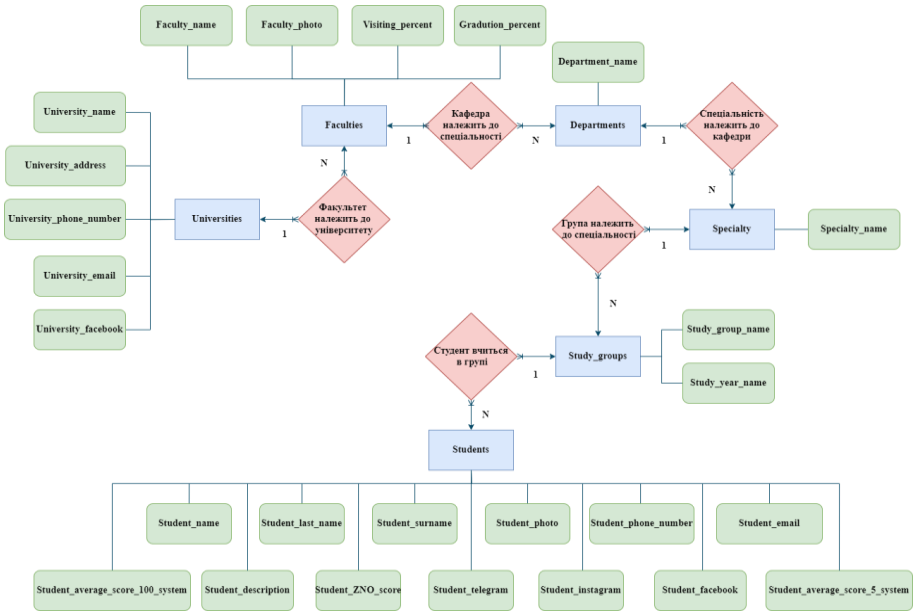


Рисунок 3.1 – Концептуальна база даних

Як видно з концептуальної моделі бази даних, на першому етапі проектування, було продумано: загальний вид бази даних, усі види полів, та об’єктів до яких ці поля будуть застосовані.

Після побудови концептуальної моделі та повного представлення вигляду бд, наступним кроком було побудування логічної моделі бази даних, то повна підготовка усіх необхідних даних системи заради реалізації фізичної моделі бази даних даного проекту.

Логічна модель (рис. 4.2), бази даних є проміжним етапом між концептуальною та фізичною моделями. Вона визначає структуру даних, що зберігаються в базі даних, та спосіб їх організації. Логічна модель надає деталізований опис сутностей, атрибутів та зв’язків між ними, що дозволяє розробити детальний план створення бази даних. Вона служить основою для розробки фізичної моделі бази даних, яка визначає спосіб розміщення даних на фізичних носіях, таких як жорсткий диск. Логічна модель дозволяє розробити точний план створення бази даних, що відповідає вимогам конкретного проекту.

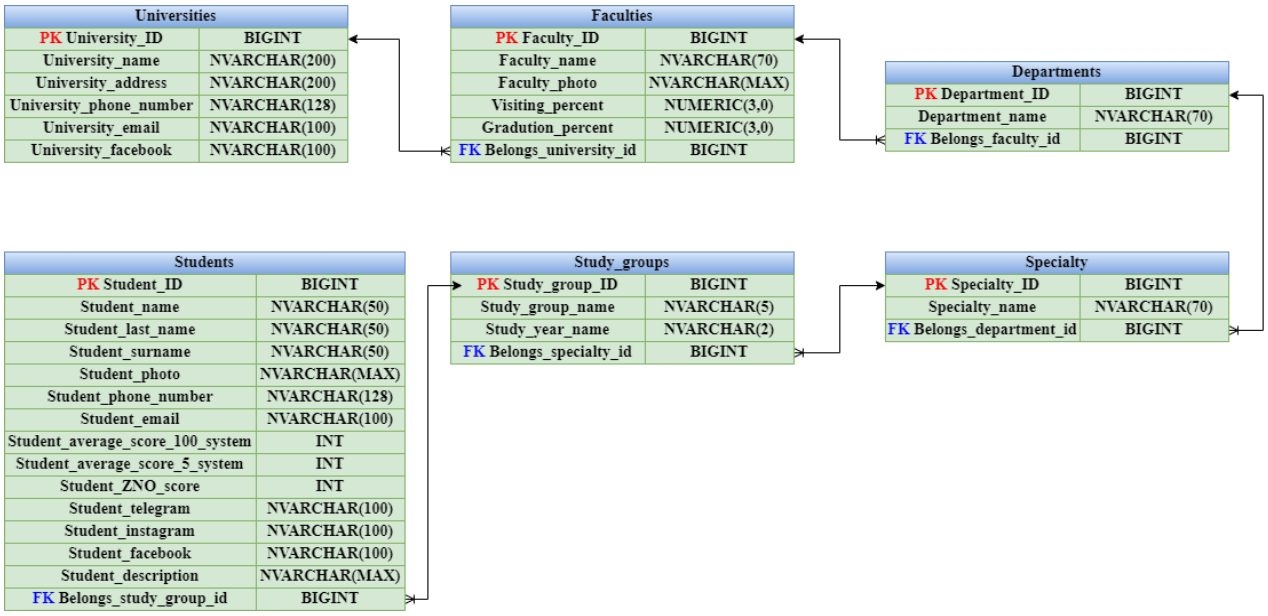


Рисунок 3.2 – Логічна модель бази даних

На момент розробки логічної бази даних, вже було визначено тип даних для полів, які були створені на етапі розробки концептуальної моделі бази даних. Логічна схема бази даних, має більш конкретний вигляд завдяки більш цілісній моделі. Логічна модель бази даних представлена на рисунку 3.2

Далі був написаний скрипт зі створенням таблиць і даних, та розміщення бд в MS SQL Server. Скрипт для фізичної бази даних зазвичай використовується для створення самої бази даних, таблиць, відносин між таблицями, індексів, обмежень на рівні бази даних та інших об’єктів. Цей скрипт містить SQL-команди, які будуть виконані базою даних для створення вказаних об’єктів.

Висновки по розділу

Після завершення роботи над цим розділом було визначено який тип бази даних буде застосовано, а саме реляційна база даних. Був обраний стек технологій який буде використовуватися для розробки бази даних. А також були створені концептуальна, логічна та фізична моделі бази даних.

4 СТВОРЕННЯ ІНТЕРФЕЙСУ

Етап розробки інтерфейсу є важливою частиною створення інформаційної системи, тому що виникає питання, яку обрати колірну гамму та в якому стилі розробляти інтерфейс. Після перегляду аналогів, дослідження їх переваг і недоліків було обрано, що в де більшому будуть використовуватися фіолетовий та чорний кольори. А програма буде представлена у вигляді web-застосунку.

Середовищем розробки інтерфейсу було обрано таке середовище як Figma. В порівняні з іншими середовищами, які використовуються у розробці інтерфейсу, Figma перевершує їх за багатьма показниками.

4.1 Обрання середовища для створення дизайну

Figma – це онлайн-інструмент для дизайну та розробки інтерфейсів, який дозволяє створювати візуальні макети та прототипи. Figma була розроблена з урахуванням потреб дизайнерів та команд розробників, які працюють над великими проектами та мають потребу в спільній роботі над дизайном.

Основна перевага Figma полягає у тому, що вона працює в браузері та може бути використана на будь-якому пристрої з Інтернет-підключенням, не потребуючи встановлення додаткових програм. Крім того, Figma дозволяє дизайнерам працювати в режимі реального часу та здійснювати спільну роботу над проектом з будь-якого місця в світі.

Figma має багатий набір інструментів для дизайну та розробки інтерфейсів, таких як інструменти для малювання, роботи з текстом та зображеннями, векторні форми, компоненти та бібліотеки. Крім того, Figma дозволяє дизайнерам створювати прототипи та анімацію, що дозволяє більш ефективно представляти дизайн та отримувати зворотний зв’язок від команди розробників.

Figma також дозволяє дизайнерам та розробникам спільно працювати над проектом, співпрацюючи над одним та самим документом в режимі реального часу. Це дозволяє командам працювати швидше та більш ефективно, зменшуючи час, необхідний для зворотного зв’язку та погодження змін.

Узагальнюючи, Figma – це потужний інструмент для дизайну та розробки інтерфейсів, який дозволяє дизайнерам та розробникам працювати разом.

Після завершення попереднього дизайну, була проведена зустріч з керівником, обговорені певні моменти, після чого було підтверджено дизайн. Далі показані основні елементи інтерфейсу.

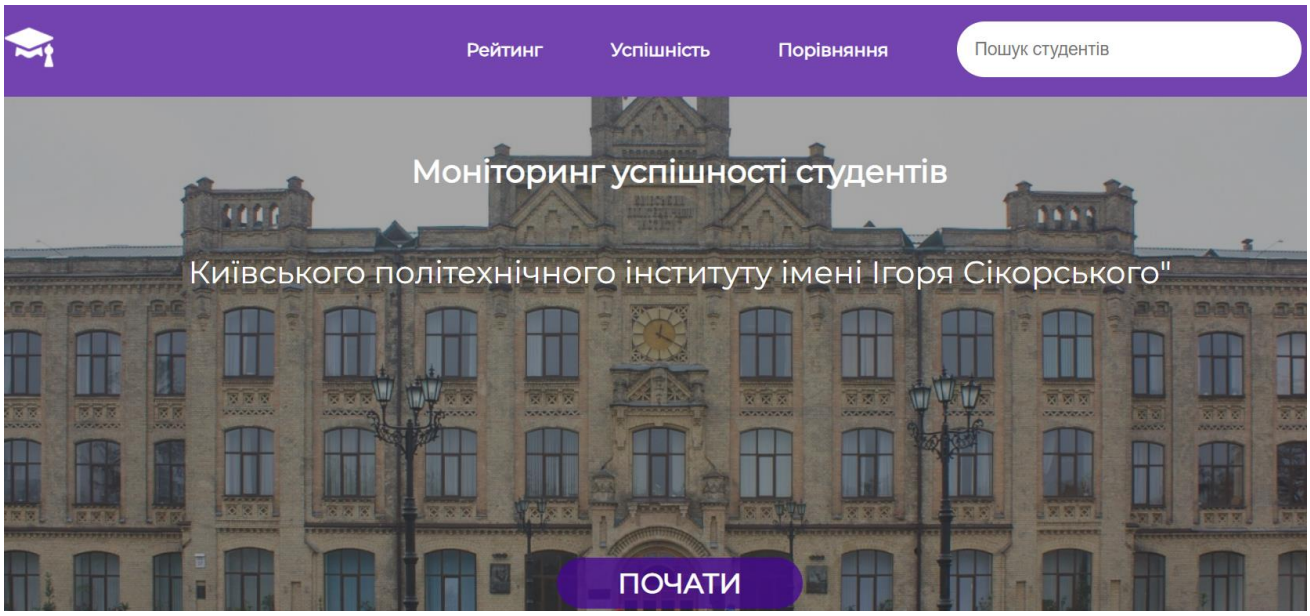
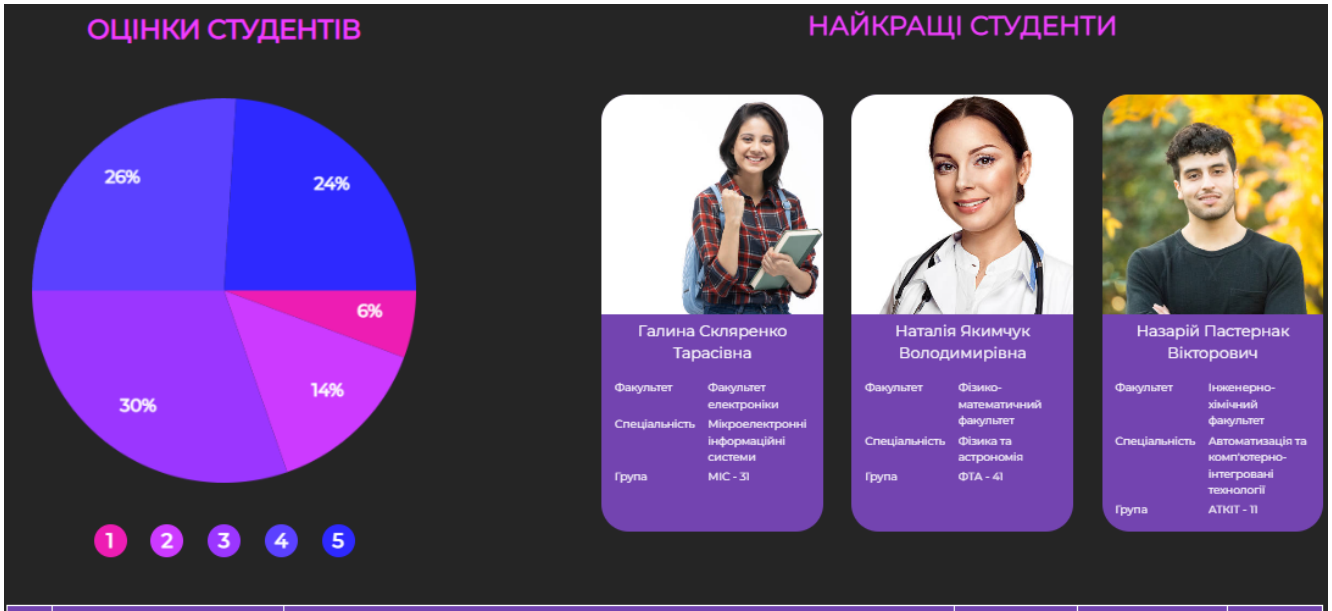


Рисунок 4.1 – Головна сторінка програми

На головній (рис. 4.1), сторінці веб-застосунку можна побачити фото головного корпусу інституту. Далі в хедері сторінки знаходиться навігація, можна перейти на сторінки рейтингу, успішності, порівняння, а також до пошуку студентів. Якщо перейти на будь-яку сторінку, то натиснувши на іконку в лівому верхньому куті, можна повернутися до головної сторінки. Також кнопка «Почати» одразу переадресує вас на сторінку рейтингу, аби подивитися результати студентів по інституту. На сторінці рейтингу інституту також можна обрати конкретного студента та перейти на його особисту сторінку, для більш детального

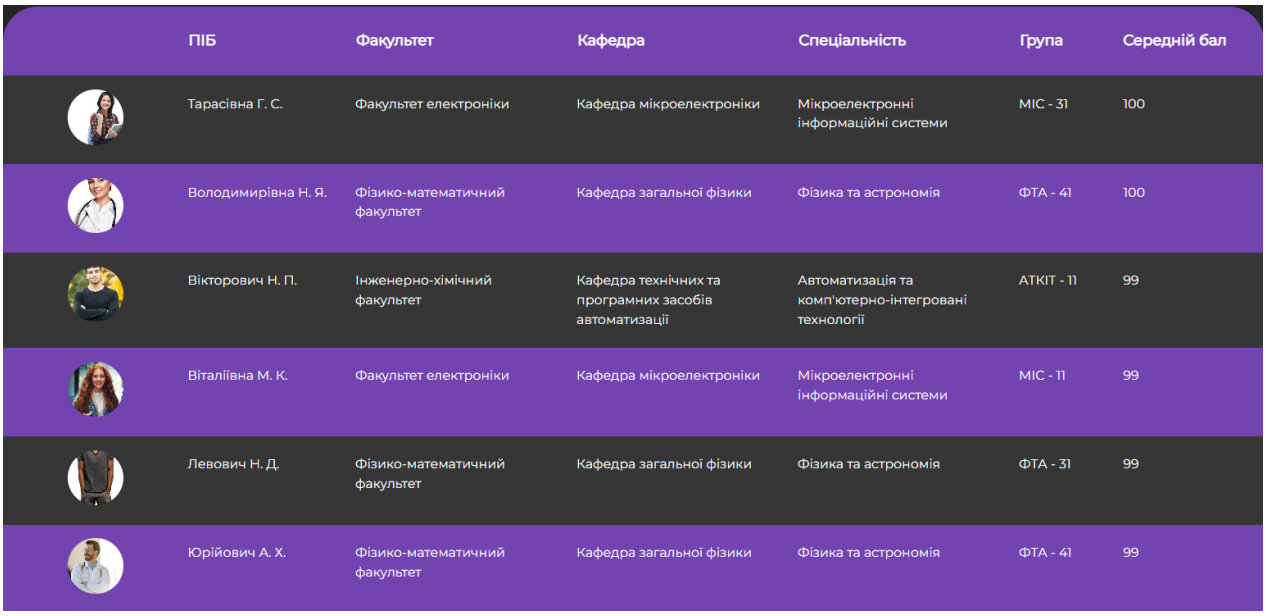
ознайомлення з його інформацією. Можна подивитися його соціальні мережі коротку інформацію вступний бал, а також середній бал навчання.



№ з/п	Прізвище, ім'я, по батькові	Результати навчання										Частка оцінок «відмінно» та «дуже добре»	Частка оцінок «добре», «задовільно» та «достатньо»	Середній бал
		«відмінно» 100-95 балів		«дуже добре» 94-85 балів		«добре» 84-75 балів		«задовільно» 74-65 балів		«достатньо» 64-60 балів				
1	Скляренко Галина Тарасівна	36	55%	19	29%	9	13%	2	3%	0	0%	84%	16%	4,67
2	Якимчук Наталія Володимирівна	25	38%	21	32%	10	15%	10	15%	0	0%	70%	30%	4,46
3	Пастернак Назарій Вікторович	23	35%	15	23%	13	20%	8	12%	7	10%	58%	42%	4,3

Рисунок 4.2 – Розділ головної сторінки, де показані відсоток по 5 бальній системі оцінок, та троє найкращих студентів.

На скріншоті (рис. 4.2) показано розділ головної сторінки на якій можна побачити кругову діаграму, на якій відображаються відсотки відповідно до 5 – бальної системи оцінювання. Кожна частина діаграми відповідає своїй оцінці. Також присутні 3 кращих студента з найвищим середнім балом. Короткий їх опис, та можливість перейти на їх сторінки та ознайомитися з кращими студентами кафедри. З цієї сторінки можна дізнатися шкалу оцінок по університету, зробити висновки який бал здебільшого отримують студенти, а також на основі трьох кращих студентів, їх факультетів, спеціальностей подивитися яких результатів досягають найкращі студенти інституту.



	ПІБ	Факультет	Кафедра	Спеціальність	Група	Середній бал
	Тарасівна Г. С.	Факультет електроніки	Кафедра мікроелектроніки	Мікроелектронні інформаційні системи	МІС - 31	100
	Володимирівна Н. Я.	Фізико-математичний факультет	Кафедра загальної фізики	Фізика та астрономія	ФТА - 41	100
	Вікторович Н. П.	Інженерно-хімічний факультет	Кафедра технічних та програмних засобів автоматизації	Автоматизація та комп'ютерно-інтегровані технології	АТКІТ - 11	99
	Віталіївна М. К.	Факультет електроніки	Кафедра мікроелектроніки	Мікроелектронні інформаційні системи	МІС - 11	99
	Левович Н. Д.	Фізико-математичний факультет	Кафедра загальної фізики	Фізика та астрономія	ФТА - 31	99
	Юрійович А. Х.	Фізико-математичний факультет	Кафедра загальної фізики	Фізика та астрономія	ФТА - 41	99

Рисунок 4.3 – Сторінка де відображається рейтинг студентів, на скріншоті показано у порядку спадання

На даній (рис. 4.3), сторінці можна побачити відображення списку студентів, які йдуть у порядку спадання від найкращого студента до найгіршого. Можна побачити на якому факультеті, кафедрі, спеціальності, групі вчиться студент, а також його середній бал, за яким і формується рейтинг. Кожен з списку студентів є клікабельним, що надає змогу перейти на сторінку студента та ознайомитися з його інформацією (рис. 5.4). Як вже було вказано вище, на основі даної сторінки є можливість зробити висновок які оцінки отримують студенти на тих чи інших факультетах та спеціальностях, також є змога більш детально ознайомитися з інформацією про різних студентів з різних напрямків навчання. Це в свою чергу надає можливість зробити висновки для користувача який хоче порівняти успішність студентів з різних напрямків, особливо корисно дана сторінка буде для тих, хто має намір вступати да інституту і хоче отримати мінімальну інформацію для того щоб обрати на яку кафедру, спеціальність краще всього вступати.

Рисунок 4.4 – Сторінка студента

На сторінці студента (рис. 4.4) відображається така інформація: контактна інформація студента, тобто його телефон, почтова адреса (e-mail) та соц. мережі, такі як telegram, 41nstagram, facebook. Також присутня деяка загальна особиста інформація, а саме: прізвище, ім’я, по-батькові, факультет де навчається студент, його кафедра, спеціальність, група, вступний бал ЗНО, а також середній бал студента в 5-бальній, та в 100-бальній системі. Та його короткий опис в нижньому блоці сторінки студента.

Сторінка студента має в цілому інформативний характер, винесені лише основні дані які можуть знадобитися по студенту в процесі навчання, наприклад соц. мережі слугують здебільшого для можливості підтримки контакту зі студентом, місце навчання – для швидкого пошуку студента за кафедрою, спеціальністю тощо, а така інформація як вступний бал та короткий опис студента про себе, є факторами в першу чергу для викладачів, при виникненні можливо труднощів зі студентом під час навчання, що очікувати від студента, а можливо який підібрати підхід для максимально ефективного періоду викладання для студента.

Рисунок 4.5 – Середня оцінка груп студентів

На сторінці успішності (рис. 4.5) у вигляді гістограми представлений середній бал, за такими розділами як: факультет, кафедра, спеціальність та група. Значення для обрахування беруться з бази даних, обраховуються і парсяться до фронтенду в гістограму. Справа на панелі є кілька кнопок навігації, такі як: приховати підписи, приховати значення та змінити вигляд системи оцінювання. З 5-бальної в 100-бальну і навпаки.

Головна сторінка моніторингу процесу навчання студентів, ця сторінка надає всю необхідну інформацію для розуміння яка ситуація у тій чи іншій групі, на тій чи іншій спеціальності. Подана інформація не викладається у величезних таблицях з оцінками, інформація подана максимально стисло у вигляді діаграм та гістограм, що в свою чергу дає візуальне бачення який напрямок є ефективним по навчанню, а який має проблеми або з подачею матеріалу та його викладанням, або ж проблема полягає у студентах. Це надає можливість доволі швидко прийняти рішення по зміненню програми навчання або в проведені виховної роботи зі студентами.

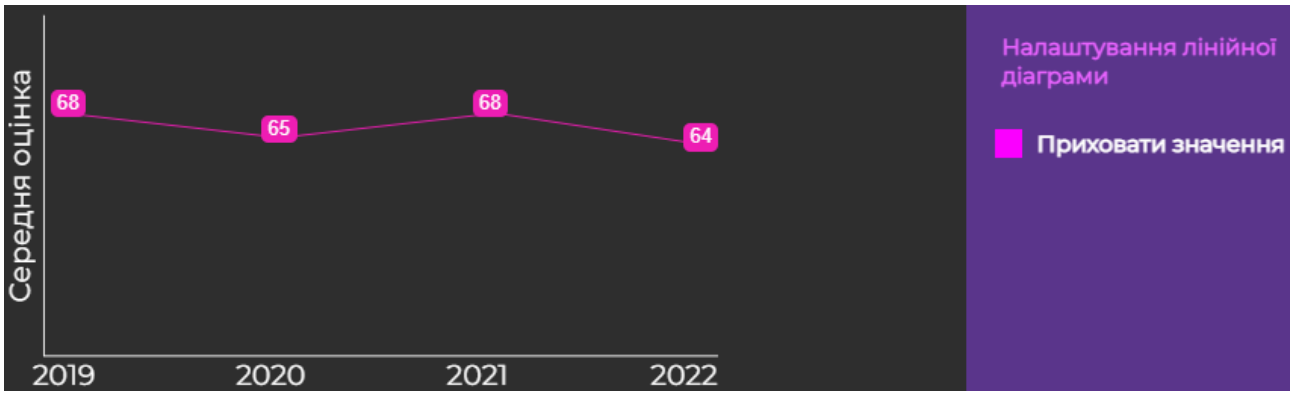


Рисунок 4.6 – Середня оцінка за проміжок часу у 4 роки

Середня оцінка впродовж 4 років навчання (рис. 4.6), аналогічно до гістограми на якій відображається оцінки за даний семестр, на даному графіку відображається середня оцінка впродовж навчання на бакалавра. Дані аналогічно беруться з бази даних та парсяться до графіку на фронт-енд частині. Також на панелі справа можна приховати значення, аби побачити чистий вигляд графіку. Графік змінюється в залежності від того по чому йде сортування, факультет, кафедра, спеціальність, група.

Даний графік є чинником який надає змогу робити аналіз після основного етапу навчання, а саме періоду навчання на бакалавра, бо дані за 4 роки є результатом опираючись на які можна зробити висновки які показала себе обрана схема навчання, як показали себе впроваджені зміни на тих чи інших напрямах і як показали себе студенти за цей час. Бо кожне наступне покоління студентів може погано сприймати матеріал в силу його застарілості. Та модернізація навчального процес може бути ключовим фактором за яким навчальний процес зможе показати максимальну ефективність. Що в свою чергу наддасть на випуску максимальну кількість спеціалістів у багатьох сферах навчання.

4.2 Розробка інтерфейсу

При розробці інтерфейсу (web-частини програми), я використав такий стек технологій: html, css, Java script, Vue js. Трохи про сам стек технологій, який був використаний для створення інтерфейсу.

1. HTML (Hypertext Markup Language):

HTML — це мова розмітки, яка використовується для створення та дизайну веб-сторінок і програм. Він забезпечує структуру та вміст веб-сторінки та працює в поєднанні з іншими технологіями веб-розробки, такими як CSS і JavaScript. HTML використовує теги для визначення та опису вмісту веб-сторінки, такого як заголовки, абзаци, зображення та посилання. Це фундаментальна технологія веб-розробки, яка підтримується всіма основними веб-переглядачами.

2. CSS (Cascading Style Sheets):

CSS — це мова таблиць стилів, яка використовується для опису представлення веб-сторінки, написаної на HTML. CSS дозволяє веб-розробникам відокремити вміст і структуру веб-сторінки від її візуального стилю та макета. Він надає можливість налаштовувати зовнішній вигляд веб-сторінки за допомогою кольорів, шрифтів і методів компонування, таких як системи сіток і flexboxes. CSS можна використовувати для створення адаптивного веб-дизайну, який адаптується до різних розмірів екрана та пристроїв.

3. JavaScript:

JavaScript — це мова програмування високого рівня, яка використовується для створення інтерактивних і динамічних веб-сторінок. Це мова сценаріїв на стороні клієнта, яка працює у веб-браузерах і надає можливість маніпулювати вмістом веб-сторінки, реагувати на взаємодії користувача та взаємодіяти з веб-службами та API. JavaScript використовується для створення анімації, перевірки даних форми та створення інтерфейсу користувача. Його також можна використовувати на стороні сервера з такими технологіями, як Node.js [1, 6, 8].

Запити можуть бути відправлені на сервер за допомогою методів HTTP, таких як GET, POST, PUT або DELETE. На стороні сервера дані оброблюються та повертаються у відповідь на запит.

Окремим інструментом який широко був застосований у дипломній роботі, це фрейворк від JS, Vue Js.

4. Vue Js.

У моїй роботі Vue.js використовується для організації компонентів та директив, які дозволяють зв'язати дані та відображення і створити інтерактивний інтерфейс користувача [2, 4, 11, 12].

На прикладі сторінки успішності, а саме лінійної діаграми на якій відображено середній бал впродовж 4 років навчання (рисунок 3.6). Vue.js використовується для створення компоненту line-chart, який відповідає за відображення лінійної діаграми [14, 16].

Окрім діаграми Vue.js використовується для:

- організації шаблону компоненту line-chart за допомогою директиви template;
- виведення даних на сторінку за допомогою директиви v-for та інтерполяції даних;
- зв'язування атрибутів елементів SVG з даними за допомогою директиви : (двокрапка);
- відслідковування змін даних та автоматичне оновлення відображення за допомогою реактивних властивостей даних та обчислювальних властивостей (computed properties);
- збереження даних на стороні клієнта та отримання даних з сервера за допомогою бібліотеки axios та методу mounted();
- організації взаємодії з користувачем за допомогою методів, які відповідають на події на сторінці, наприклад, метод toggleNames() змінює властивості даних, щоб приховати або показати значення на діаграмі.

Якщо говорити про застосування Vue, на інших сторінках програми, то яскравим використанням vue буде на головній сторінці програми.
Тут використовується Vue Js для розробки клієнтської частини веб-додатка.

Цей код складається з однієї компоненти Vue, яка відображає головну сторінку додатка, яка містить кілька ділянок інформації.

Цей компонент має дані про студентів та їхній успіх, які витягуються з зовнішнього джерела (API). Дані про студентів перебувають в масиві об'єктів students, а targetStudentIds містить ідентифікатори кращих студентів. З цих даних створюється список студентів з їхніми даними, такими як ім'я, прізвище, факультет, спеціальність тощо. Список студентів відображається на сторінці, а також показується діаграма з їхнім успіхом за допомогою компонента PieChart.

Vue Js використовується для реалізації таких функцій, як рендеринг списку студентів, звернення до API, обробка даних та перехід до інших сторінок веб-додатка.

З опису даного стеку, зрозуміло чому було обрано саме ці технології для розробки дизайну.

Наступним кроком було обрано середовище розробки інтерфейсу, вибір впав на Visual Studio Code. Чому саме було обрано дане середовище, бо на мою думку воно є самим зручним та потужним середовищем для розробки інтерфейсу. Ось кілька аргументів чому було обрано саме VS Code.

Visual Studio Code є дуже потужним та популярним середовищем розробки, яке може бути корисним для розробки інтерфейсів.

Visual Studio Code підтримує багато різних мов програмування, включаючи HTML, CSS та JavaScript, що дозволяє розробникам працювати з цими технологіями в одному інтегрованому середовищі.

Він має дуже потужні функції підсвічування синтаксису, автодоповнення та рефакторингу коду, що дозволяє розробникам швидко та легко створювати та редагувати код.

Visual Studio Code також має велику кількість розширень та плагінів, що дозволяє розширювати можливості середовища розробки та додавати нові функції, які допомагають розробникам працювати більш ефективно.
Він має потужний інструментарій для відлагодження коду, що дозволяє розробникам швидко виявляти та виправляти помилки в коді.

Visual Studio Code є безкоштовним та відкритим середовищем розробки, що дозволяє розробникам використовувати його безкоштовно та вносити свій внесок у вдосконалення цього інструменту.

Відзначу, що Visual Studio Code не є єдиним середовищем розробки та вибір залежить від особистих уподобань розробника та вимог проекту. Проте, Visual Studio Code має багато переваг, які дозволяють розробникам працювати швидко та ефективно, тому може бути хорошим вибором для розробки інтерфейсів.

Висновки по розділу

Після завершення роботи с даним розділом було обрано стек технологій які будуть застосовані для розробки інтерфейсу та дизайну. Реалізовані дизайн та інтерфейс, за допомогою html, css, js, та vue js. Виконані елементи інтерфейсу які були обговорені з викладачем, а саме. Відтворення діаграм та гістограм з успішністю студентів, за різними факультетами та спеціальностями. Виведення на графіки результатів середнього балу студентів за період навчання на бакалавра, а саме 4 роки навчання.

5 РОЗРОБКА ПРОГРАМНОГО КОДУ

Враховуючи що програма буде розроблена у вигляді web-застосунку, то для її створення було обрано наступний стек технологій: Python та його фреймворк Django. Переваги цих технологій включають в себе: Python - це потужна та динамічно розвиваюча мова програмування, яка має велику кількість різноманітних бібліотек та фреймворків, що дозволяє ефективно розробляти web-застосунки різного рівня складності [5-9].

Django - це один з найпопулярніших фреймворків для розробки web-застосунків. Він має широкі можливості, включаючи швидку розробку, високу продуктивність, безпеку, масштабованість та розширюваність. Django також має багато документації та велику спільноту розробників, що робить його привабливим для використання у проектах будь-якої складності [10, 13].

Головною метою Django є полегшення створення веб-додатків швидко та ефективно. Нижче наведено детальний опис деяких основних компонентів та можливостей Django.

Моделі (Models): Моделі є в основі Django і використовуються для взаємодії з базою даних. Django використовує ORM (Object-Relational Mapping), щоб забезпечити взаємодію з різними СУБД, такими як SQLite, MySQL, PostgreSQL та іншими. Моделі в Django були широко застосовані у моїй дипломній роботі, тому буде правильно сконцентрувати на них додаткову увагу.

Django використовує ORM (Object-Relational Mapping), щоб забезпечити взаємодію з різними СУБД, такими як SQLite, MySQL, PostgreSQL та іншими. ORM дозволяє працювати з даними, використовуючи об'єктно-орієнтований підхід, що спрощує взаємодію з базою даних та дозволяє змінювати базу даних без зміни коду в додатку.

Для створення моделі у Django потрібно створити клас, який наслідується від класу django.db.models.Model. Кожен атрибут класу представляє поле моделі, яке може бути текстовим, числовим, булевим або більш складним, наприклад,

зображенням або файлом. Крім того, можна використовувати різні параметри, які дозволяють вказати тип даних, валідацію, обмеження тощо.

Основні компоненти моделі в Django:

Поля (Fields): Поля є атрибутами класу моделі та визначають тип та властивості поля в базі даних. Django надає багато вбудованих типів полів, таких як CharField, TextField, IntegerField, BooleanField, DateTimeField, і т. д.

Відносини (Relations): Відносини визначають зв'язки між різними моделями. Django надає можливість використовувати різні типи відносин, такі як ForeignKey, OneToOneField, ManyToManyField, і т. д.

Методи (Methods): Методи визначають поведінку моделі. Django дозволяє створювати власні методи для моделі, що дозволяє додавати додаткову логіку до моделі.

Метадані (Metadata): Метадані містять додаткову інформацію про модель, таку як ім'я таблиці в базі даних, порядок сортування, обмеження тощо.

URL маршрутизація (URL Routing): Django надає механізм URL маршрутизації, що дозволяє легко визначати шляхи запитів, використовуючи правила маршрутизації.

Представлення (Views): Представлення в Django відповідають за обробку запитів та генерацію відповідей. Django надає багато вбудованих функцій, таких як шаблонізація, кешування та інші, що дозволяють легко та ефективно створювати веб-сторінки.

Шаблони (Templates): Django надає механізм шаблонів, що дозволяє створювати веб-сторінки з використанням HTML та інших технологій. Django також підтримує наслідування шаблонів, що дозволяє створювати багато повторюваного коду.

Крім того, Python має простий та зрозумілий синтаксис, що дозволяє швидко розробляти та відлагоджувати код. Він також підтримує багато стандартів, таких як WSGI та ASGI, що дозволяє використовувати його з різноманітними веб-серверами.

Усі ці переваги дозволяють ефективно розробляти web-застосунки з використанням Python та Django, що робить їх привабливими для використання в будь-яких проектах.

5.1 Архітектура серверної частини

Будь-який проект створений за допомогою Django має в основі структуру проекту яка базується на концепції Model-View-Controller (MVC), де моделі (Models) відповідають за роботу з базою даних, представлення (Views) відповідають за відображення сторінок, а контролери (Controllers) відповідають за координацію роботи між моделями та представленнями. Однак, в Django використовується власна реалізація патерну Model-View-Template (MVT), де замість контролерів використовуються шаблони (Templates).

Основні елементи структури проекту:

- manage.py: файл, який використовується для керування Django-проектом, таким як запуск сервера або створення міграцій;
- myproject/: це основна папка проекту, де знаходяться налаштування проекту, URL-адреси та серверний скрипт;
- __init__.py: порожній файл, який дозволяє Python розглядати цю папку як пакет;
- settings.py: файл, який містить налаштування проекту, такі як база даних, шлях до статичних файлів та інші;
- urls.py: файл, який містить список URL-адрес, які користувачі можуть відвідати, та їх зв'язок з відповідними представленнями;
- asgi.py та wsgi.py: файли, які використовуються для запуску сервера.

Папка migrations в Django - це місце, де зберігаються файли міграцій бази даних. Міграції - це спосіб зберігати зміни моделей Django в базі даних.

Кожна міграція представляє собою файл Python, який містить інструкції для зміни бази даних, такі як створення нових таблиць, додавання або видалення стовпців, зміна типу даних тощо.

Папка migrations зазвичай містить два підкаталоги: initial та __pycache__, а також файл __init__.py, який дозволяє Python трактувати папку як пакет. Кожна міграція зберігається в окремому файлі з іменем, що містить послідовність номерів та описовій назві, наприклад 0001_initial.py.

Папка migrations створюється автоматично при створенні нового Django проекту та містить міграції для стандартних модулів Django, таких як auth та admin. Крім того, кожна нова модель, створена в проекті Django, має свій власний файл міграції в цій папці.

Папка migrations є важливим елементом процесу розробки в Django, оскільки вона дозволяє зберігати та відстежувати зміни в базі даних, що зроблені під час розробки проекту

5.2 Взаємодія серверної та користувацької частини

За реалізацію взаємодії між серверною та клієнтською частиною в програмі відповідають SQL-запити з бекенду до бази даних. Всі підрахунки, виведення даних до користувацького інтерфейсу, передача інформації до діаграм та гістограм побудовано на запитах. Кілька аргументів чому було обрано саме таку реалізацію взаємодії двох головних складових частин програми.

- Безпека даних: Виконання запитів до бази даних з серверної частини дозволяє контролювати доступ до даних і застосовувати необхідні правила безпеки. Це дозволяє уникнути прямого доступу до бази даних з клієнтської сторони, що може бути небезпечним з погляду зловмисників;
- Контроль бізнес-логіки: Виконання запитів на серверній стороні дозволяє зберігати бізнес-логіку програми на сервері і забезпечувати її цілісність та

консистентність. Це дозволяє уникнути розсіювання бізнес-логіки по різних клієнтських пристроях і забезпечити однорідність обробки даних;

- Оптимізація трафіку: Використання запитів до бази даних з серверної сторони дозволяє виконувати операції обробки даних на сервері і передавати лише необхідну інформацію клієнту. Це допомагає зменшити обсяг передачі даних по мережі і покращити продуктивність програми, особливо при великому обсязі даних;

- Масштабованість: Використання запитів до бази даних на серверній стороні дає можливість гнучко налаштовувати та масштабувати систему. Наприклад, можна використовувати кешування результатів запитів, розподілені бази даних або інші методи оптимізації для забезпечення високої продуктивності і швидкодії;

- Спрощення клієнтської частини: Використання запитів до бази даних на серверній стороні дозволяє клієнтській стороні бути більш простою і мінімізувати відповідальність за обробку даних. Клієнтська сторона може отримувати готові результати запитів і просто їх відображати або виконувати інші нескладні операції.

Усі ці фактори враховуються при розробці програмного забезпечення, і використання запитів до бази даних на серверній стороні може бути ефективним підходом для забезпечення безпеки, продуктивності та розширюваності програми.

Нижче будуть продемонстровані запити які будують робочу архітектуру всієї програми.

Перший запит "UniversityInfoView" запит для отримання даних про інститут.

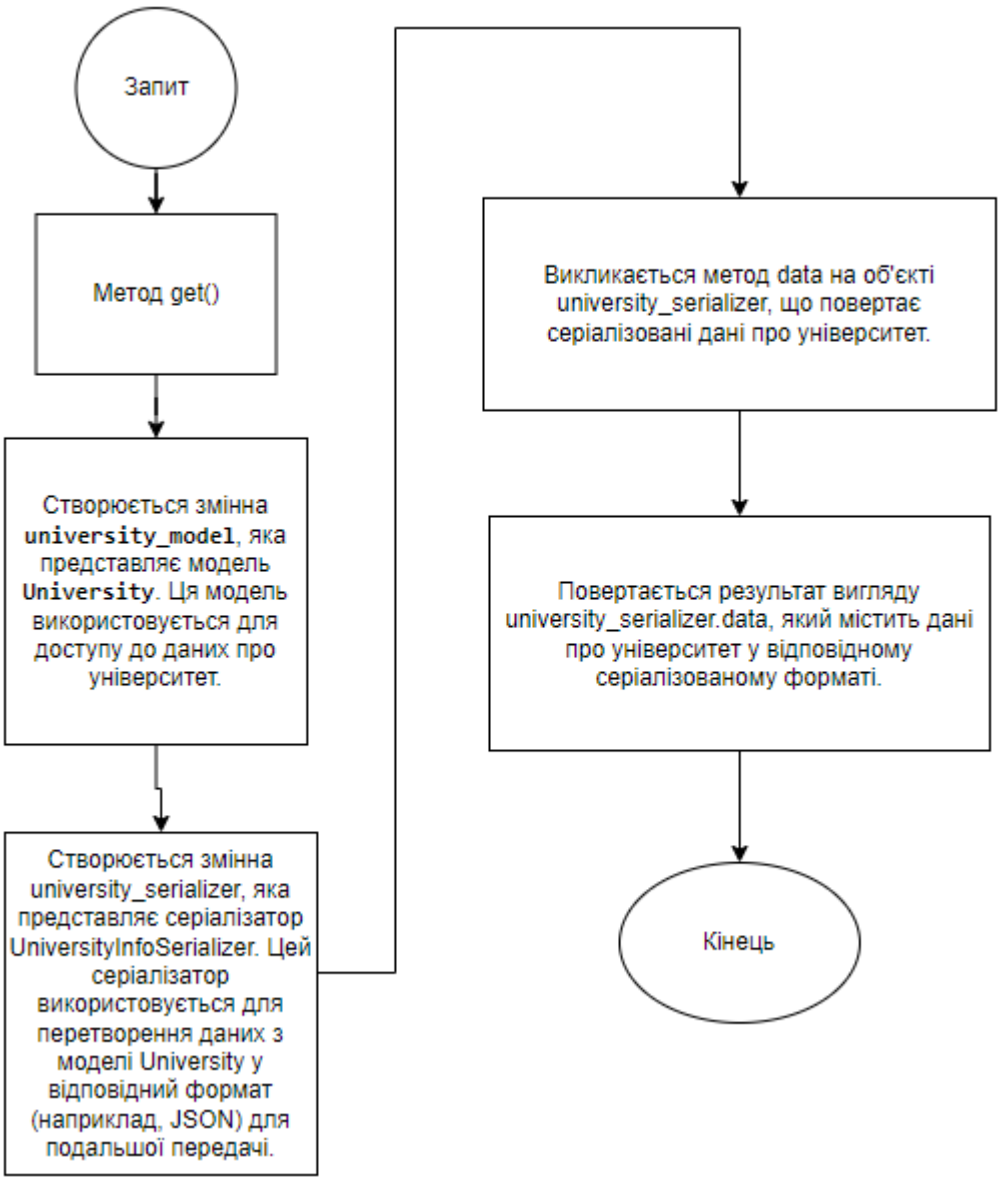


Рисунок 5.1 – запит для отримання даних про інститут

Даний запит (рис. 5.1) визначає клас UniversityInfoView, який містить статичний метод get(). При виклику цього методу, відбувається наступне:

- Створюється змінна university_model, яка представляє модель University. Ця модель використовується для доступу до даних про університет;
- Створюється змінна university_serializer, яка представляє серіалізатор UniversityInfoSerializer. Цей серіалізатор використовується для перетворення даних з моделі University у відповідний формат (наприклад, JSON) для подальшої передачі;

• Викликається метод data на об'єкті university_serializer, що повертає серіалізовані дані про університет;

• Повертається результат вигляду university_serializer.data, який містить дані про університет у відповідному серіалізованому форматі.

Отже, цей код отримує дані про університет з використанням моделі University, серіалізує ці дані за допомогою серіалізатора UniversityInfoSerializer і повертає серіалізовані дані про університет.

Другий запит "Top3StudentInfoView", слугує для виведення трьох кращих студентів кафедри до головної сторінки застосунку.

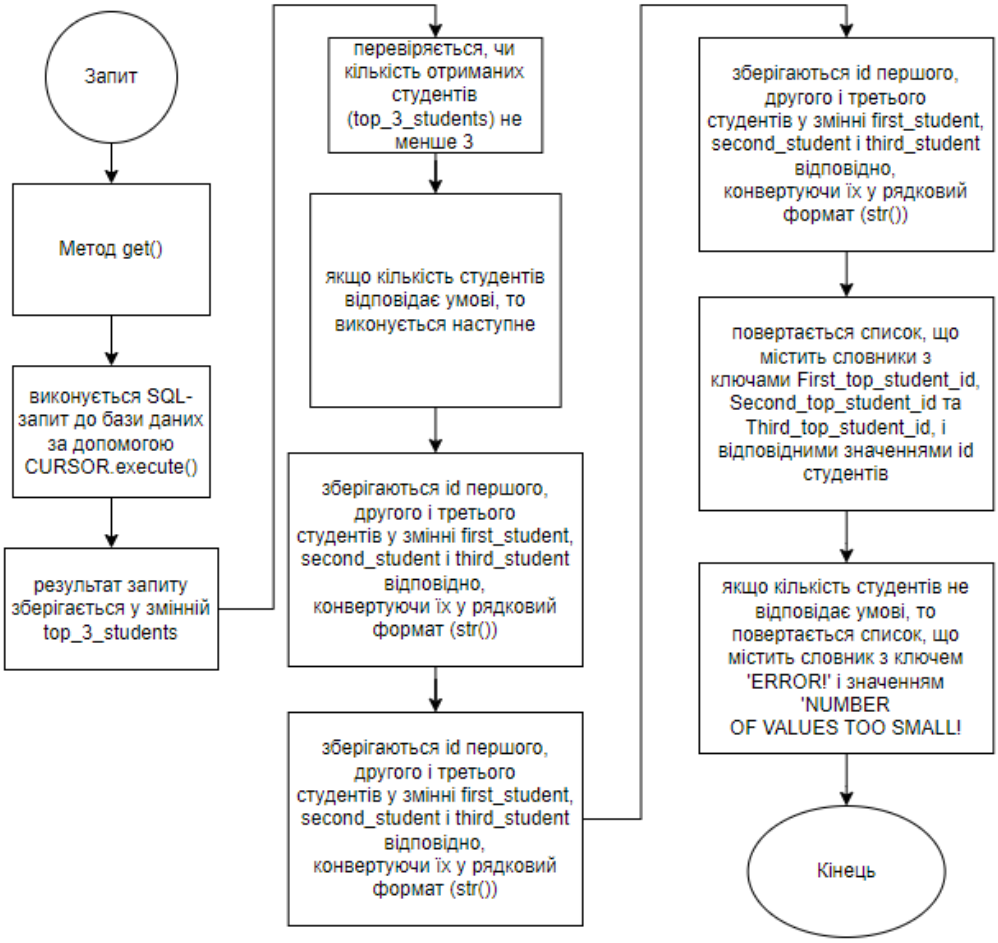


Рисунок 5.2 – запит на виведення трьох кращих студентів

Даний запит (рис. 5.2) визначає клас Top3StudentInfoView, який містить статичний метод get(). При виклику цього методу, відбувається наступне:

- Виконується SQL-запит до бази даних за допомогою CURSOR.execute(). Запит вибирає топ-3 студентів (id) з таблиці Students за умови сортування за зниженням середнього балу студента (Student_average_score_100_system);
- Результат запиту зберігається у змінній top_3_students;
- Перевіряється, чи кількість отриманих студентів (top_3_students) не менше 3;
- Якщо кількість студентів відповідає умові, то виконується наступне;
- Зберігаються id першого, другого і третього студентів у змінні first_student, second_student і third_student відповідно, конвертуючи їх у рядковий формат (str());
- Повертається список, що містить словники з ключами First_top_student_id, Second_top_student_id та Third_top_student_id, і відповідними значеннями id студентів;
- Якщо кількість студентів не відповідає умові, то повертається список, що містить словник з ключем 'ERROR!' і значенням 'NUMBER OF VALUES TOO SMALL!'.

Отже, цей код виконує SQL-запит до бази даних для отримання топ-3 студентів за середнім балом, обробляє результати запиту та повертає інформацію про цих студентів у вигляді списку словників або повідомлення про помилку, якщо кількість значень недостатня.

Третій запит "GradesDiagramInfoView(ModelViewSet)" відповідає за виведення кругової діаграми з оцінками.

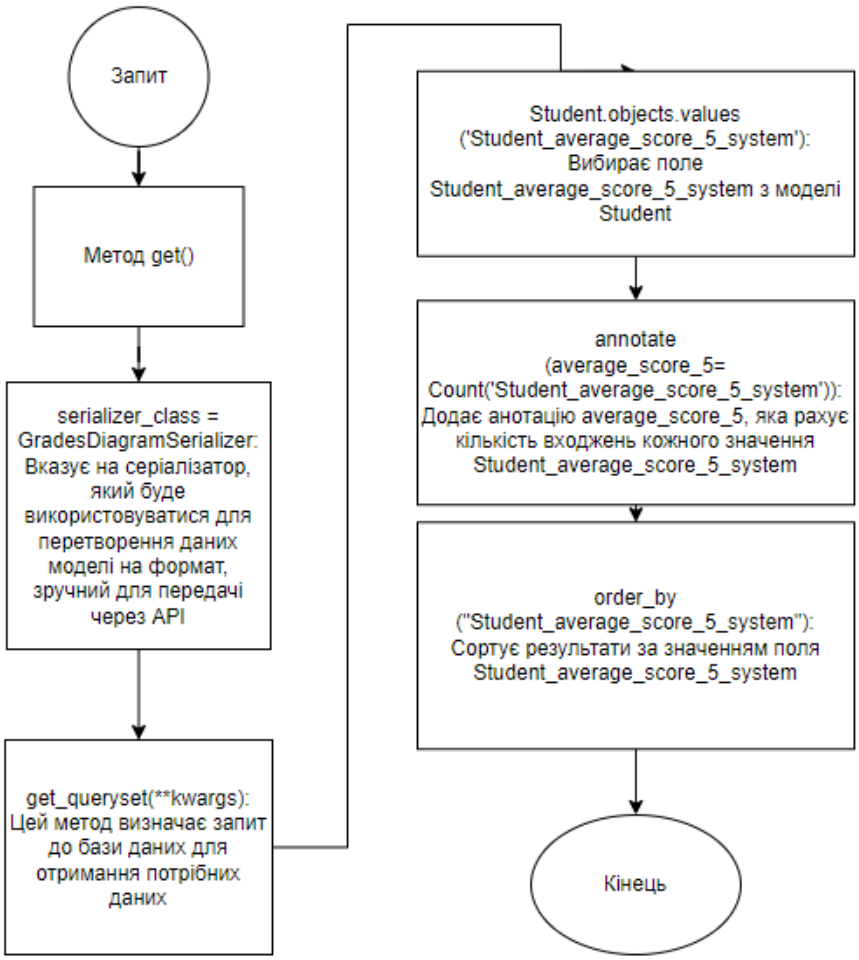


Рисунок 5.3 – запит на виведення кругової діаграми

Даний запит (рис. 5.3) представляє клас GradesDiagramInfoView, який є підкласом ModelViewSet. Він використовує серіалізатор GradesDiagramSerializer для обробки даних моделі.

- serializer_class = GradesDiagramSerializer: Вказує на серіалізатор, який буде використовуватися для перетворення даних моделі на формат, зручний для передачі через API;
- get_queryset(**kwargs): Цей метод визначає запит до бази даних для отримання потрібних даних. В даному випадку, метод використовує модель Student і виконує наступні дії;
- Student.objects.values('Student_average_score_5_system'): Вибирає поле Student_average_score_5_system з моделі Student;

• annotate(average_score_5=Count('Student_average_score_5_system')): Додає анотацію average_score_5, яка рахує кількість входжень кожного значення Student_average_score_5_system;

• order_by("Student_average_score_5_system"): Сортує результати за значенням поля Student_average_score_5_system.

Отримані дані (рис. 5.4) відповідатимуть наступній структурі:

```
[{'Student_average_score_5_system': value1, 'average_score_5': count1},
 {'Student_average_score_5_system': value2, 'average_score_5': count2},
 ...]
```

Рисунок 5.4 – Структура даних в запиті

де value1, value2 - унікальні значення Student_average_score_5_system, а count1, count2 - кількість входжень цих значень в базі даних.

Таким чином, цей код визначає запит до бази даних для отримання даних студентів з їх середніми оцінками у п'ятибальній системі, а також кількістю студентів з кожною оцінкою.

Четвертий запит виводить середній бал по всіх групах студентів.

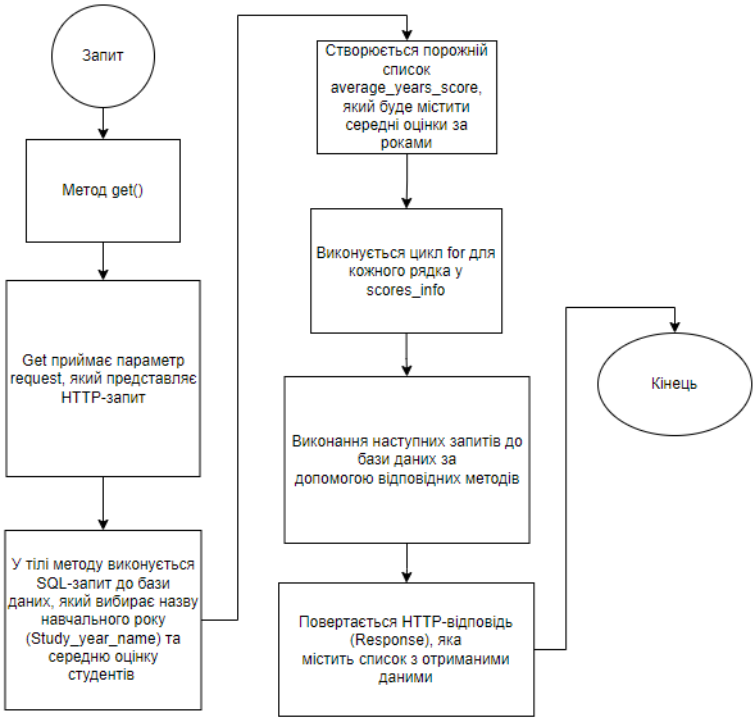


Рисунок 5.5 – запит на виведення середнього балу

Даний запит (рис. 5.5) представляє клас AverageGradesByStudyGroup, який має статичний метод get(). Цей метод використовується для отримання середніх оцінок за групами навчання.

- study_group_model = StudyGroup.objects: Створення змінної study_group_model, яка представляє модель StudyGroup. Ця модель використовується для отримання даних про групи навчання;
- study_group_serializer = Average Grades By Study Group Serializer (study_group_model, many=True): Створення змінної study_group_serializer, яка ініціалізує об'єкт серіалізатора AverageGradesByStudyGroupSerializer. При ініціалізації передається модель study_group_model і параметр many=True, що вказує, що серіалізатор буде обробляти кілька об'єктів моделі;
- return study_group_serializer.data: Повертає дані, отримані з серіалізатора. Це може бути список словників, кожен з яких містить серіалізовані дані про кожну групу навчання. Структура даних залежить від того, як визначено серіалізатор AverageGradesByStudyGroupSerializer.

Отже, цей код використовує серіалізатор AverageGradesByStudyGroupSerializer для обробки даних про групи навчання і повертає серіалізовані дані відповідно до визначеного серіалізатора.

П'ятий та останній з головних запитів це вивід графіку з оцінками за період навчання в 4 роки.

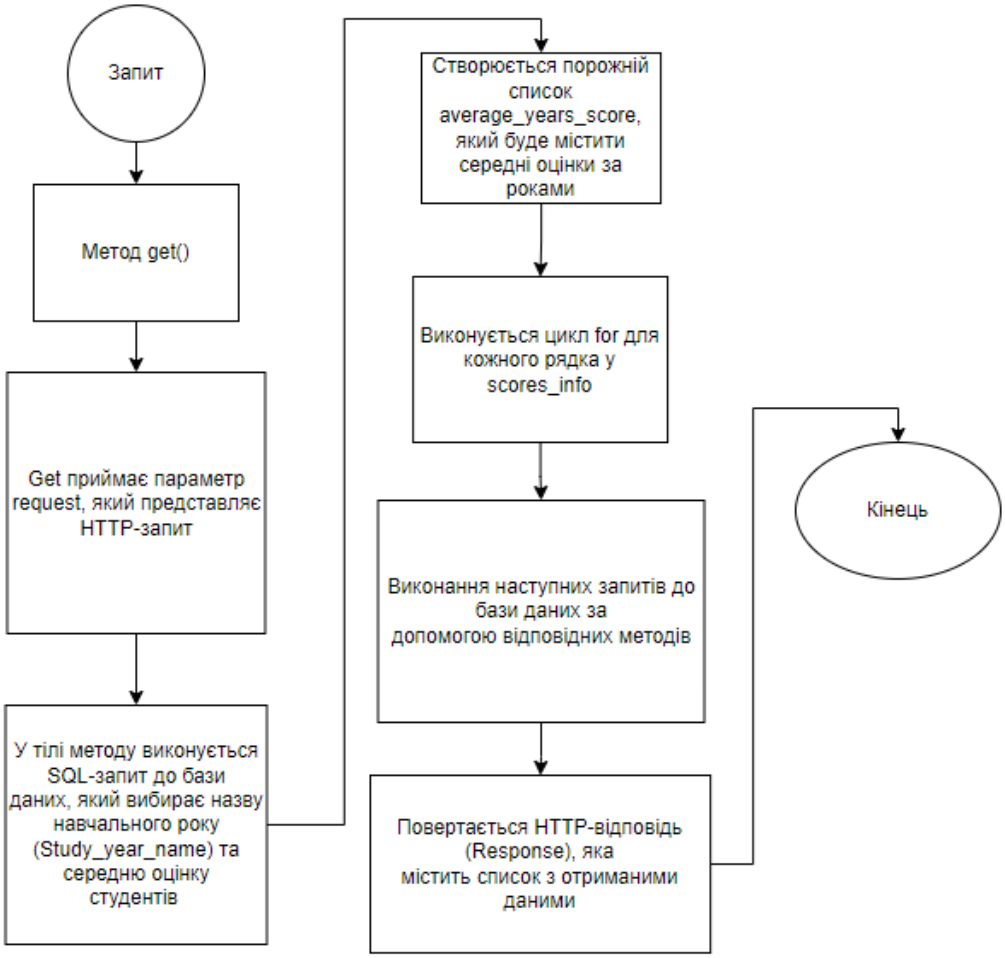


Рисунок 5.6 – запит на вивід оцінок за чотири роки навчання

Даний запит (рис. 5.6) визначає клас GraphsPageView, який є підкласом APIView з Django REST Framework. Цей клас використовується для обробки HTTP-запиту методом GET на сторінці з графіками.

У класі GraphsPageView визначено наступне:

1. Метод get, який виконується при HTTP-запиті методом GET. Він приймає параметр request, який представляє HTTP-запит.

2. У тілі методу виконується SQL-запит до бази даних, який вибирає назву навчального року (Study_year_name) та середню оцінку студентів (AVG(Student_average_score_100_system)) для кожної групи зі сполученої таблиці Study_groups та Students. Результат запиту отримується за допомогою CURSOR.fetchall() та зберігається у змінній scores_info.

3. Створюється порожній список average_years_score, який буде містити середні оцінки за роками.

4. Виконується цикл for для кожного рядка у scores_info. У циклі перевіряється значення year[0] (назва навчального року) і на основі цього значення встановлюється відповідне значення year_name (рік) у змінну average_years_score. Якщо значення year[0] відповідає '11', то year_name встановлюється на 2022, якщо '21', то на 2021, і так далі. Цей рік та відповідна середня оцінка (year[1]) додаються у словник у вигляді {'Study_year': рік, 'Average_year_score': середня оцінка} та додаються до списку average_years_score.

5. Виконуються наступні запити до бази даних за допомогою відповідних методів:

- grades_diagram_info = GradesDiagramInfoView.get_queryset(): Отримання діаграми оцінок за допомогою методу get_queryset() з класу GradesDiagramInfoView.
- faculties_average_grades = AverageGradesByFaculties.get(): Отримання середніх оцінок за факультетами за допомогою методу get() з класу AverageGradesByFaculties.
- departments_average_grades = Average Grades By Departments.get(): Отримання середніх оцінок за кафедрами за допомогою методу get() з класу AverageGradesByDepartments.
- specialty_average_grades = Average Grades By Specialty.get(): Отримання середніх оцінок за спеціальностями за допомогою методу get() з класу AverageGradesBySpecialty.
- study_group_average_grades = Average Grades By Study Group.get(): Отримання середніх оцінок за групами за допомогою методу get() з класу AverageGradesByStudyGroup.

6. Повертається HTTP-відповідь (Response), яка містить список з отриманими даними. До цього списку додаються:

- grades_diagram_info: діаграма оцінок,

- faculties_average_grades: середні оцінки за факультетами,
- departments_average_grades: середні оцінки за кафедрами,
- specialty_average_grades: середні оцінки за спеціальностями,
- study_group_average_grades: середні оцінки за групами,
- average_years_score.__reversed__(): список середніх оцінок за роками, який перевертається.

Отриманий список даних передається у відповідь на HTTP-запит.

5.3 Застосування програмного коду

Головні виконавчі файли бекенд частини містяться в папці, яка створена в папці "main_app" за допомогою Django [15-17, 18-20].

Перший виконавчий файл admin.py:

В даному коді використовується модуль admin Django для реєстрації моделей (models), які раніше були визначені в іншому файлі.

Імпортовано всі необхідні моделі з файлу models.py. Далі, використовуючи admin.site.register(), реєструються ці моделі в панелі адміністратора Django. Це дозволить адміністратору змінювати, видаляти і додавати об'єкти цих моделей через панель адміністратора Django.

В даному коді зареєстровані моделі: University, Faculty, Department, Specialty, StudyGroup та Student. Тобто адміністратор має можливість додавати, видаляти та змінювати об'єкти усіх цих моделей в панелі адміністратора Django.

Наступний файл це apps.py:

Цей код створює об'єкт конфігурації додатка Django. Клас AppConfig знаходиться у модулі django.apps.

MainAppConfig - це наша власна конфігурація додатка, де MainApp - це назва додатка.

За допомогою цього класу ми можемо змінювати налаштування нашого додатку. В даному випадку, ми використовуємо атрибут default_auto_field, щоб вказати Django, яке поле автоматично зберігатиме ідентифікатори в базі даних за замовчуванням. Значення django.db.models.BigAutoField означає, що Django використовуватиме велике автоінкрементне поле для зберігання ідентифікаторів.

Також використовується атрибут name, де main_app - це назва нашого додатка, яку ми вказали при створенні додатка. Цей атрибут дозволяє Django знайти наш додаток та імпортувати всі моделі, види, шляхи та інші ресурси, пов'язані з нашим додатком.

Клас MainAppConfig дозволяє нам виконати всі налаштування та підготовчі дії, щоб наш додаток був готовим до роботи під час запуску додатку.

Наступний файл це models.py:

В даному файлі знаходиться опис моделей даних для бази даних, яку використовує Django. В даному коді створено 6 моделей: University, Faculty, Department, Specialty, StudyGroup, Student.

Кожна з цих моделей описує певний елемент освітньої системи - університет, факультет, кафедру, спеціальність, групу студентів і студента. Для кожної моделі визначено її поля та типи даних, які можна зберігати в цих полях. Наприклад, модель University має такі поля, як University_name, University_address, University_phone_number тощо.

Також для кожної моделі визначено її властивості Meta, що вказують на назву таблиці, в якій будуть зберігатися дані, порядок сортування, що буде використовуватися за замовчуванням, та те, чи може Django керувати таблицею.

Цей код дозволяє описати структуру бази даних, яку Django використовує для зберігання даних про освітні заклади, факультети, кафедри, спеціальності, групи студентів та студентів. Крім того, цей код дозволяє виконувати операції збереження, оновлення, видалення та отримання даних з цих таблиць бази даних за допомогою Django ORM.

Наступний файл views. py:

Цей код - фрагмент веб-додатку, побудованого на фреймворку Django, який забезпечує підключення до бази даних і відображення різноманітної інформації про університет.

Код імпортує різні модулі, такі як pyodbc, django.db.models, rest_framework.response, rest_framework.views і rest_framework.viewsets.

Далі він імпортує моделі бази даних та серіалізатори, необхідні для отримання даних з цих моделей.

За допомогою методу connect з модуля pyodbc, код створює з'єднання з базою даних.

Клас UniversityInfoView містить метод get(), який повертає інформацію про університет за допомогою серіалізатора UniversityInfoSerializer.

Клас Top3StudentInfoView містить метод get(), який отримує три студентів з найвищим балом за допомогою запиту до бази даних.

Клас GradesDiagramInfoView містить метод get_queryset(), який отримує інформацію про бали студентів та кількість студентів з кожним балом, а потім серіалізує цю інформацію за допомогою серіалізатора GradesDiagramSerializer.

Клас StudentInfoView містить метод get(), який повертає інформацію про факультети за допомогою серіалізатора FacultiesListSerializer.

Клас MainPageView містить метод get(), який повертає інформацію про університет, топ-3 студентів, графік балів студентів та інформацію про факультети за допомогою вищезгаданих класів.

Клас Top30StudentInfoView містить метод get(), який отримує тридцять студентів з найвищим балом за допомогою запиту до бази даних.

Клас RatingPageView містить метод get(), який повертає інформацію про топ-30 студентів та інформацію про факультети за допомогою вищезгаданого класу.

Наступний файл serializer.py

Цей код - це модуль Django, що працює з базою даних SQL Server (за допомогою драйвера pyodbc). У ньому використовуються серіалайзери Django

REST framework (DRF), які забезпечують взаємодію з даними, що знаходяться в базі даних.

У даному коді визначаються наступні класи серіалайзерів:

UniversityInfoSerializer - серіалайзер для моделі University, який описує поля, які будуть використовуватися для відображення інформації про університет (адреса, телефон, електронна пошта, facebook);

GradesDiagramSerializer - серіалайзер для моделі Student, який описує поля, що будуть використовуватися для побудови діаграми оцінок (середній бал в системі оцінювання за 5-бальною шкалою);

StudentsListSerializer - серіалайзер для моделі Student, який описує всі поля цієї моделі;

StudyGroupsListSerializer - серіалайзер для моделі StudyGroup, який описує поля, які будуть використовуватися для відображення інформації про групу (назва групи, назва року навчання) та поля Students_info - список студентів групи, який формується за допомогою серіалайзера StudentsListSerializer;

SpecialtyListSerializer - серіалайзер для моделі Specialty, який описує поля, які будуть використовуватися для відображення інформації про спеціальність (назва спеціальності) та поля Study_groups_info - список груп спеціальності, який формується за допомогою серіалайзера StudyGroupsListSerializer;

DepartmentsListSerializer - серіалайзер для моделі Department, який описує поля, які будуть використовуватися для відображення інформації про відділення (назва відділення) та поля Specialty_info - список спеціальностей відділення, який формується за допомогою серіалайзера SpecialtyListSerializer.

Кожен з серіалайзерів описує структуру вихідного JSON, який буде повернутий з API. Вони використовуються для серіалізації моделей Django в формат JSON, щоб передавати дані від веб-застосунку до клієнта.

В коді визначаються 4 класи серіалайзерів, які опрацьовують дані з різних рівнів моделі: факультети, кафедри, спеціальності та групи. Кожен з серіалайзерів використовує статичний метод для отримання середнього балу студентів з

відповідних рівнів, за допомогою виконання запитів до бази даних. Потім серіалайзер повертає вихідний JSON з даними.

Також дуже важливим елементом розробки програми був Django Rest Framework .

Головна задача DRF (Django Rest Framework) – у програмі була наступна: DRF в програмі використовується як бібліотека для створення API для бази даних. Вона надає зручний і простий спосіб створення CRUD операцій (створення, читання, оновлення та видалення) для моделей бази даних, які можуть бути використані в зовнішніх додатках. Також Django Rest Framework дозволяє встановлювати авторизацію, валідацію даних та інші налаштування API.

Окрім цього у програмі DRF використовується для визначення серіалізаторів для моделей Django, які визначають, як дані перетворюються на JSON або інші формати для передачі через мережу. Серіалізатори визначають поля, які мають бути включені в серіалізоване представлення моделей.

Ось огляд ролей DRF у коді:

Серіалізатори: DRF надає клас ModelSerializer, який використовується для визначення серіалізаторів для моделей (UniversityInfoSerializer, GradesDiagramSerializer, StudentsListSerializer тощо). Серіалізатори визначають модель і поля, які будуть включені в серіалізоване представлення. Вони обробляють перетворення екземплярів моделі в JSON і навпаки, забезпечуючи легку серіалізацію та десеріалізацію даних.

Поля серіалізатора: DRF надає різні поля серіалізатора (наприклад, IntegerField, SerializerMethodField), які дозволяють налаштувати процес серіалізації. Ці поля визначають типи даних для серіалізації та надають можливості перевірки та перетворення.

Вкладена серіалізація: DRF підтримує вкладену серіалізацію, що дозволяє включати пов’язані моделі в серіалізоване представлення. У коді це демонструють такі серіалізатори, як StudyGroupsListSerializer, SpecialtyListSerializer, DepartmentsListSerializer, FacultiesListSerializer та інші.

Вони включають пов’язані моделі, вказуючи вкладені класи серіалізатора за допомогою атрибута related_name.

Спеціальні методи серіалізатора: DRF дозволяє визначати спеціальні методи серіалізатора за допомогою SerializerMethodField. У коді такі методи, як _get_average_score_5 і _get_average_score_100, обчислюють середні бали на основі запитів SQL. Ці методи використовуються для включення додаткових обчислених полів у серіалізоване представлення.

Використовуючи Django Rest Framework, код може легко визначати та налаштовувати серіалізатори, обробляти вкладені зв’язки та виконувати додаткові маніпуляції з даними під час серіалізації. Це спрощує процес створення API і гарантує, що дані представлені в структурованому та стандартизованому вигляді.

Висновки по розділу

По завершенню роботи з програмним кодом, був повністю написаний бекенд до програми, реалізовані усі зв’язки з базою даних та фронтендом, реалізовано весь функціонал програми який користувач може використовувати для користування системою моніторингу успішності студентів. Реалізовано бекенд частину було завдяки такому стеку технологій як: Python, Django, Django Rest Framework та використання багатьох бібліотек пайтону. Самими яркими представниками слугують бібліотека Pandas, для обчислення даних, а в даному випадку оцінок студентів.

Другим великим розділом є написання API, для взаємодії з базою даних для витягування даних та парсингу цих даних на фронт частину та застосування у даній частині розробки Django Rest Framework і його реалізація при взаємодії з базою даних та API. Це гарно продемонстровано на прикладі діаграм та гістограм, де після обрахунку даних, відбувається парсинг даних до графіків, гістограм та діаграм.

ВИСНОВКИ

В результаті розробки створено якісний web-додаток, який може використовуватися для моніторингу успішності студентів та допоможе покращити якість освіти та навчального процесу. Створені відповідні сторінки для моніторингу основних елементів навчання таких як: загальний рейтинг студентів, середній бал кожного студента, середній бал кафедри, середній бал груп, спеціальностей, напрямків, які є на кафедрі, а також графіки середнього балу за певний проміжок часу.

Створені сторінки які матимуть можливість моніторингу таку інформацію: загальний рейтинг студентів, середній бал кожного студента, середній бал кафедри, середній бал груп, спеціальностей, напрямків, які є на кафедрі, а також графіки середнього балу за певний проміжок часу.

Використаний сучасний стек технологій, що включає Frontend-інструменти html, css, js, vue js, node js modules і Backend- фреймворк Python з використанням Django, що дозволило створити високопродуктивну і надійну систему.

По мірі завершення розробки досягнуто всіх поставлених цілей та завдань, включаючи створення зручного інтерфейсу для користувачів, реалізацію системи відстеження рівня навчання студентів та моніторинг їх успішності з метою покращення результатів, та внесення правок у майбутні навчальні програми.

Загалом, розробка програми "інформаційна система моніторингу успішності студентів" є успішною і дозволить ефективно керувати інформацією про студентів та їхню успішність.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Flanagan, D. JavaScript: The Definitive Guide. O'Reilly Media. 2017. 1000с.
2. Holmes, K. Vue.js in Action. Manning Publications. 2018. 500с.
3. Welling, L., & Thomson, L. Learning PHP, MySQL & JavaScript: With jQuery, CSS & HTML5. O'Reilly Media. 2017. 800с.
4. Lutz, M. Learning Python. O'Reilly Media. 2019. 833с.
5. VanderPlas, J. Python Data Science Handbook: Essential Tools for Working with Data. O'Reilly Media. 2016. 548с.
6. Freeman, E., & Robson, E. Head First JavaScript Programming. O'Reilly Media. 2014. 704с.
7. Mitchell, R. Web Scraping with Python: Collecting More Data from the Modern Web. O'Reilly Media. 2018. 255с.
8. Duckett, J. JavaScript and JQuery: Interactive Front-End Web Development. Wiley. 2014. 640с.
9. Bucky, R. Python Programming for the Absolute Beginner. Course Technology. 2015. 480с.
10. Документація. Django for Beginners. (n.d.). Retrieved from https://djangoforbeginners.com/ (дата звернення 08.05.2023).
11. Vue.js Developers. (n.d.). Vue.js Tutorials. Retrieved from https://vuejsdevelopers.com/tutorials/ (дата звернення 08.05.2023).
12. The Vue Handbook. (n.d.). Retrieved from https://vuehandbook.com/ (дата звернення 08.05.2023).
13. Django for APIs. (n.d.). Retrieved from https://www.django-rest-framework.org/tutorial/quickstart/ (дата звернення 08.05.2023).
14. Brown, D. Learning Vue.js 2. Packt Publishing. 2019. 334с.
15. Roy, S. (2020). Learning Django: Web Development in Python. Packt Publishing.
16. Vue.js Up and Running. O'Reilly Media. 2018. 219с.

17. Django for Professionals. (n.d.). Retrieved from https://djangoforprofessionals.com/ (дата звернення 06.04.2023).
18. Daniel Roy Greenfeld, Audrey Roy Greenfeld. Two Scoops of Django 3.x: Best Practices for the Django Web Framework. 2020. 475c.
19. Gaston C. Hillar. Django Restful Web Services: The easiest way to build Python RESTful APIs and web services with Django. 2018. 416c.
20. William S. Vincent. Django REST Framework: Build powerful RESTful APIs with Django and Python. 2018. 522c.

ДОДАТОК А

Інформаційна система контролю виконання доручень для управління діяльністю кафедри

Лістинг програми

Аркушів 11

Київ - 2023

models.py

```
from django.db import models
from phonenumber_field.modelfields import PhoneNumberField
from django.core.validators import MinValueValidator, MaxValueValidator
from decimal import Decimal

PERCENTAGE_VALIDATOR = [MinValueValidator(0), MaxValueValidator(100)]


class University(models.Model):
    University_name = models.CharField(max_length=200)
    University_address = models.CharField(max_length=200)
    University_phone_number = PhoneNumberField(null=False, blank=False, unique=True)
    University_email = models.EmailField(max_length=100)
    University_facebook = models.CharField(max_length=100)

    class Meta:
        db_table = 'Universities'
        ordering = ['University_name']
        managed = True

    def __str__(self):
        return self.University_name


class Faculty(models.Model):
    Belongs_university = models.ForeignKey(University, on_delete=models.CASCADE,
related_name='Faculties_info')
    Faculty_name = models.CharField(max_length=70)
    Faculty_photo = models.TextField()
    Visiting_percent = models.DecimalField(max_digits=3, decimal_places=0, default=Decimal(0),
                                           validators=PERCENTAGE_VALIDATOR)
    Graduation_percent = models.DecimalField(max_digits=3, decimal_places=0, default=Decimal(0),
                                             validators=PERCENTAGE_VALIDATOR)

    class Meta:
        db_table = 'Faculties'
        ordering = ['Faculty_name']
        managed = True

    def __str__(self):
        return self.Faculty_name
```

```
class Department(models.Model):
    Belongs_faculty = models.ForeignKey(Faculty, on_delete=models.CASCADE,
related_name='Departments_info')
    Department_name = models.CharField(max_length=70)

    class Meta:
        db_table = 'Departments'
        ordering = ['Department_name']
        managed = True

    def __str__(self):
        return self.Department_name


class Specialty(models.Model):
    Belongs_department = models.ForeignKey(Department, on_delete=models.CASCADE,
related_name='Specialty_info')
    Specialty_name = models.CharField(max_length=70)

    class Meta:
        db_table = 'Specialty'
        ordering = ['Specialty_name']
        managed = True

    def __str__(self):
        return self.Specialty_name


class StudyGroup(models.Model):
    Belongs_specialty = models.ForeignKey(Specialty, on_delete=models.CASCADE,
related_name='Study_groups_info')
    Study_group_name = models.CharField(max_length=5)
    Study_year_name = models.CharField(max_length=2)

    class Meta:
        db_table = 'Study_groups'
        ordering = ['Study_group_name']
        managed = True

    def __str__(self):
        return self.Study_group_name


class Student(models.Model):
    Belongs_study_group = models.ForeignKey(StudyGroup, on_delete=models.CASCADE,
related_name='Students_info')
    Student_name = models.CharField(max_length=50)
```

```
    Student_last_name = models.CharField(max_length=50)
    Student_surname = models.CharField(max_length=50)
    Student_photo = models.TextField()
    Student_phone_number = PhoneNumberField(null=False, blank=False, unique=True)
    Student_email = models.EmailField(max_length=100)
    Student_average_score_100_system = models.IntegerField(validators=[MinValueValidator(0),
MaxValueValidator(100)])
    Student_average_score_5_system = models.IntegerField(validators=[MinValueValidator(0),
MaxValueValidator(100)])
    Student_ZNO_score = models.IntegerField(validators=[MinValueValidator(100),
MaxValueValidator(200)])
    Student_telegram = models.CharField(max_length=100)
    Student_instagram = models.CharField(max_length=100)
    Student_facebook = models.CharField(max_length=100)
    Student_description = models.TextField()

    class Meta:
        db_table = 'Students'
        ordering = ['Student_name']
        managed = True

    def __str__(self):
        return self.Student_name
```

views.py

```
from pyodbc import connect
from django.db.models import Count
from rest_framework.response import Response
from rest_framework.views import APIView
from rest_framework.viewsets import ModelViewSet
from .models import University, Faculty, Department, Specialty, StudyGroup, Student
from .serializer import UniversityInfoSerializer, \
    GradesDiagramSerializer, \
    FacultiesListSerializer, \
    AverageGradesByFacultiesSerializer, \
    AverageGradesByDepartmentsSerializer, \
    AverageGradesBySpecialtySerializer, \
    AverageGradesByStudyGroupSerializer, \
    FacultyComparisonPageSerializer

DB_CONNECT = connect('Driver={SQL Server};'
                     'Server=DESKTOP-U50U5FE\SQLEXPRESS;'
```

```
                'Database=Students_success_DB;'
                'Trusted_Connection=yes;')

CURSOR = DB_CONNECT.cursor()


class UniversityInfoView:
    @staticmethod
    def get():
        university_model = University.objects
        university_serializer = UniversityInfoSerializer(university_model, many=True)
        return university_serializer.data


class Top3StudentInfoView:
    @staticmethod
    def get():
        CURSOR.execute('SELECT TOP 3 id FROM Students ORDER BY
Student_average_score_100_system DESC')
        top_3_students = CURSOR.fetchall()
        if len(top_3_students) >= 3:
            first_student = str(top_3_students[0][0])
            second_student = str(top_3_students[1][0])
            third_student = str(top_3_students[2][0])
            return [{f'First_top_student_id': first_student,
                     f'Second_top_student_id': second_student,
                     f'Third_top_student_id': third_student}]
        else:
            return [{f'ERROR!': 'NUMBER OF VALUES TOO SMALL!'}]


class Top3StudentGradesInfoView:
    @staticmethod
    def get():
        first_top_student_grades_information = {'Student_number': '1',
                                                'Name': 'Скляренко Галина Тарасівна',
                                                'Excellent': '36',
                                                'Excellent_percent': '55%',
                                                'Very_good': '19',
                                                'Very_good_percent': '29%',
                                                'Good': '9',
                                                'Good_percent': '13%',
                                                'Good_enough': '2',
                                                'Good_enough_percent': '3%',
                                                'Enough': '0',
                                                'Enough_percent': '0%',
                                                'Excellent_and_very_good_grades_percent': '84%',
```

```
                                'Good_good_enough_and_enough_grades_percent': '16%',
                                'Average_score': '4,67'}

        second_top_student_grades_information = {'Student_number': '2',
                                'Name': 'Якимчук Наталія Володимирівна',
                                'Excellent': '25',
                                'Excellent_percent': '38%',
                                'Very_good': '21',
                                'Very_good_percent': '32%',
                                'Good': '10',
                                'Good_percent': '15%',
                                'Good_enough': '10',
                                'Good_enough_percent': '15%',
                                'Enough': '0',
                                'Enough_percent': '0%',
                                'Excellent_and_very_good_grades_percent': '70%',
                                'Good_good_enough_and_enough_grades_percent': '30%',
                                'Average_score': '4,46'}

        third_top_student_grades_information = {'Student_number': '3',
                                'Name': 'Пастернак Назарій Вікторович',
                                'Excellent': '23',
                                'Excellent_percent': '35%',
                                'Very_good': '15',
                                'Very_good_percent': '23%',
                                'Good': '13',
                                'Good_percent': '20%',
                                'Good_enough': '8',
                                'Good_enough_percent': '12%',
                                'Enough': '7',
                                'Enough_percent': '10%',
                                'Excellent_and_very_good_grades_percent': '58%',
                                'Good_good_enough_and_enough_grades_percent': '42%',
                                'Average_score': '4,3'}

        return [first_top_student_grades_information,
                second_top_student_grades_information,
                third_top_student_grades_information]


class GradesDiagramInfoView(ModelViewSet):
    serializer_class = GradesDiagramSerializer

    @staticmethod
    def get_queryset(**kwargs):
        return Student.objects.values('Student_average_score_5_system'). \
```

```
annotate(average_score_5=Count('Student_average_score_5_system')).order_by("Student_average_score_5_system")


class StudentInfoView:
    @staticmethod
    def get():
        faculty_model = Faculty.objects
        faculty_serializer = FacultiesListSerializer(faculty_model, many=True)
        return faculty_serializer.data


class MainPageView(APIView):
    @staticmethod
    def get(request):
        university_info = UniversityInfoView.get()
        top_3_student_info = Top3StudentInfoView.get()
        top_3_student_grades = Top3StudentGradesInfoView.get()
        grades_diagram_info = GradesDiagramInfoView.get_queryset()
        student_info = StudentInfoView.get()
        return Response([university_info] +
                        [top_3_student_info] +
                        [top_3_student_grades] +
                        [grades_diagram_info] +
                        [student_info])


class Top30StudentInfoView:
    @staticmethod
    def get():
        CURSOR.execute('SELECT TOP 30 id FROM Students ORDER BY Student_average_score_100_system DESC')
        top_30_students = CURSOR.fetchall()
        result = dict()
        count = 1
        if len(top_30_students) >= 1:
            for elem in top_30_students:
                result[str(count)+'_top_student_id'] = str(elem[0])
                count += 1
            return [result]
        else:
            return [{f'ERROR!': 'NUMBER OF VALUES TOO SMALL!'}]


class RatingPageView(APIView):
    @staticmethod
    def get(request):
```

```
        top_30_student_info = Top30StudentInfoView.get()
        student_info = StudentInfoView.get()
        return Response([top_30_student_info] + [student_info])


class AverageGradesByFaculties:
    @staticmethod
    def get():
        faculty_model = Faculty.objects
        faculty_serializer = AverageGradesByFacultiesSerializer(faculty_model, many=True)
        return faculty_serializer.data


class AverageGradesByDepartments:
    @staticmethod
    def get():
        department_model = Department.objects
        department_serializer = AverageGradesByDepartmentsSerializer(department_model,
many=True)
        return department_serializer.data


class AverageGradesBySpecialty:
    @staticmethod
    def get():
        specialty_model = Specialty.objects
        specialty_serializer = AverageGradesBySpecialtySerializer(specialty_model, many=True)
        return specialty_serializer.data


class AverageGradesByStudyGroup:
    @staticmethod
    def get():
        study_group_model = StudyGroup.objects
        study_group_serializer = AverageGradesByStudyGroupSerializer(study_group_model,
many=True)
        return study_group_serializer.data


class GraphsPageView(APIView):
    @staticmethod
    def get(request):
        CURSOR.execute('SELECT Study_year_name, AVG(Student_average_score_100_system)'
                       'FROM Study_groups JOIN Students ON (Study_groups.id =
Students.Belongs_study_group_id) '
                       'GROUP BY Study_year_name '
                       'ORDER BY Study_year_name')
```

```
        scores_info = CURSOR.fetchall()
        average_years_score = []

        for year in scores_info:
            if year[0] == '11':
                year_name = 2022
            elif year[0] == '21':
                year_name = 2021
            elif year[0] == '31':
                year_name = 2020
            elif year[0] == '41':
                year_name = 2019
            else:
                year_name = 2000

            average_years_score.append({'Study_year': str(year_name), 'Average_year_score': year[1]})

        grades_diagram_info = GradesDiagramInfoView.get_queryset()
        faculties_average_grades = AverageGradesByFaculties.get()
        departments_average_grades = AverageGradesByDepartments.get()
        specialty_average_grades = AverageGradesBySpecialty.get()
        study_group_average_grades = AverageGradesByStudyGroup.get()
        return Response([grades_diagram_info] +
                        [faculties_average_grades] +
                        [departments_average_grades] +
                        [specialty_average_grades] +
                        [study_group_average_grades] +
                        [average_years_score.__reversed__()])


class ComparisonPageView(ModelViewSet):
    serializer_class = FacultyComparisonPageSerializer

    def get_queryset(self):
        return Faculty.objects.all()


class StudentPersonalInfoPageView(ModelViewSet):
    serializer_class = FacultiesListSerializer

    def get_queryset(self):
        return Faculty.objects.all()
```

serializer.py

```
class AverageGradesByFacultiesSerializer(ModelSerializer):
    Average_score_in_5_system = SerializerMethodField('_get_average_score_5')
    Average_score_in_100_system = SerializerMethodField('_get_average_score_100')

    @staticmethod
    def _get_average_score_5(faculty_object):
        CURSOR.execute('SELECT AVG(Student_average_score_5_system) FROM Students '
                       'JOIN Study_groups ON (Students.Belongs_study_group_id = Study_groups.id)'
                       'JOIN Specialty ON (Study_groups.Belongs_specialty_id = Specialty.id)'
                       'JOIN Departments ON (Specialty.Belongs_department_id = Departments.id)'
                       'JOIN Faculties ON (Departments.Belongs_faculty_id = Faculties.id)'
                       f'WHERE Faculties.id = {faculty_object.id}')
        return str(CURSOR.fetchall()[0][0])

    @staticmethod
    def _get_average_score_100(faculty_object):
        CURSOR.execute('SELECT AVG(Student_average_score_100_system) FROM Students '
                       'JOIN Study_groups ON (Students.Belongs_study_group_id = Study_groups.id)'
                       'JOIN Specialty ON (Study_groups.Belongs_specialty_id = Specialty.id)'
                       'JOIN Departments ON (Specialty.Belongs_department_id = Departments.id)'
                       'JOIN Faculties ON (Departments.Belongs_faculty_id = Faculties.id)'
                       f'WHERE Faculties.id = {faculty_object.id}')
        return str(CURSOR.fetchall()[0][0])

    class Meta:
        model = Faculty
        fields = ['Faculty_name',
                  'Average_score_in_5_system',
                  'Average_score_in_100_system']
```

settings.py

```
from pathlib import Path

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent


# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/4.1/howto/deployment/checklist/
```

```
# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'django-insecure-_g5_xgzt4j2f8@9=kp4*13_1blz3a_!#qwtc%@#1m7+)yjd&@p'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []


# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'main_app',
    'phonenumber_field',
    'rest_framework',
    'corsheaders',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'corsheaders.middleware.CorsMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

CORS_ORIGIN_ALLOW_ALL = True

ROOT_URLCONF = 'web_application.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
```

```
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'web_application.wsgi.application'


# Database
# https://docs.djangoproject.com/en/4.1/ref/settings/#databases

DATABASES = {
    "default": {
        "ENGINE": "mssql",
        "NAME": "Students_success_DB",
        "HOST": "DESKTOP-U50U5FE\SQLEXPRESS",
        "PORT": "",
        "OPTIONS":
            {"driver": "ODBC Driver 17 for SQL Server",
             'isolation_level': 'READ UNCOMMITTED'  # To prevent deadlocks
             },
    },
}


# Password validation
# https://docs.djangoproject.com/en/4.1/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]
```

```
# Internationalization
# https://docs.djangoproject.com/en/4.1/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'Europe/Kyiv'

USE_I18N = True

USE_TZ = True


# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/4.1/howto/static-files/

STATIC_URL = 'static/'

# Default primary key field type
# https://docs.djangoproject.com/en/4.1/ref/settings/#default-auto-field

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
```

ДОДАТОК Б

Інформаційна система контролю виконання доручень для управління діяльністю кафедри

Апробація

Аркушів 2

Київ – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Одеський національний технологічний університет
Університет Інформатики і прикладних знань, м.Лодзь, Польща
Національний технічний університет України «Київський політехнічний інститут»
Навчально-науковий інститут комп'ютерних систем і технологій «Індустрія 4.0» ім. П.М. Платонова

XXIII Всеукраїнська науково-технічна конференція молодих вчених, аспірантів та студентів

«СТАН, ДОСЯГНЕННЯ ТА ПЕРСПЕКТИВИ ІНФОРМАЦІЙНИХ СИСТЕМ І ТЕХНОЛОГІЙ»

Матеріали конференції

Одеса

20-21 квітня 2023 р.

На сьогоднішній день в силу великого поширення онлайн навчання система для моніторингу навчального процесу, а зокрема успішності студентів у новому для світу форматі навчання є доволі зручним, гнучким та корисним інструментом. Впровадження інформаційної системи для моніторингу прогресу учнів має потенціал для покращення результатів навчання, надаючи вчителям і адміністраторам своєчасні, точні та практичні дані про успішність учнів.

Враховуючи тенденції у світі було проведено велику адаптацію під онлайн-навчання, але не зважаючи на це, дуже велика кількість інститутів по всьому світі, надає перевагу паперовим журналам. Наразі у світі є технології які уявити буквально 5 років назад, було нереально, проте деякі винаходи перегорнули уявлення про світ технологій с ніг на голову і показали до чого дійшов прогрес, тому впровадження електронних систем не тільки моніторингу, а й ведення звітів, розкладу та графіків занять є необхідним, особливо на сьогоднішній день у світі навчання.

Переваги інформаційної системи для моніторингу успішності учнів: одна з головних переваг інформаційної системи для моніторингу прогресу студенів полягає в тому, що вона може надавати викладач дані про успішність учнів у реальному часі. Ці дані можуть бути використані для ідентифікації студентів, які можуть відчувати труднощі з окремими предметами, і для здійснення цілеспрямованого втручання, щоб допомогти цим учням покращити свої знання. Наприклад, якщо інформаційна система показує, що велика кількість учнів відчуває труднощі з вищою математикою, викладач може надати додаткову підтримку з цього предмету або може порекомендувати учням отримати додаткове репетиторство чи підтримку поза парами.

Ще одна перевага інформаційної системи для моніторингу прогресу студентів полягає в тому, що вона може допомогти викладачам відстежувати прогрес студентів з часом. Збираючи дані про успішність студентів на регулярній основі, інформаційна система може допомогти визначити тенденції та закономірності в

успішності студентів, які можуть бути використані для інформування про практику викладання та розробку навчальних програм. Наприклад, якщо інформаційна система показує, що студенти постійно відчувають труднощі з певним предметом та набуттям навичок з цього предмету, викладачі можуть скоригувати свої методи викладання або навчальний план, щоб краще вирішити цю слабку сферу.

На меті було створити програму яка надавала б можливість швидко та легко переглядати успішність студентів інституту можливість новим користувачам продивлятися та відстежувати середні бали різних факультетів, це орієнтовано на вступників, а також порівнювати між собою ту або іншу спеціальність.

При розробці було визначено кілька основних пунктів для створення системи.

1. Система буде представлена у вигляді веб-додатку.
2. В системі буде можливість перегляду оцінок студентів, перегляд середнього балу по таким розділам як: факультети, кафедри, спеціальність групи.
3. Порівняння показників різних факультетів між собою.

Дизайн виконано в максимально інформаційному стилі в якому кожна деталь надає інформацію про один з елементів інституту. Наприклад головна сторінка.

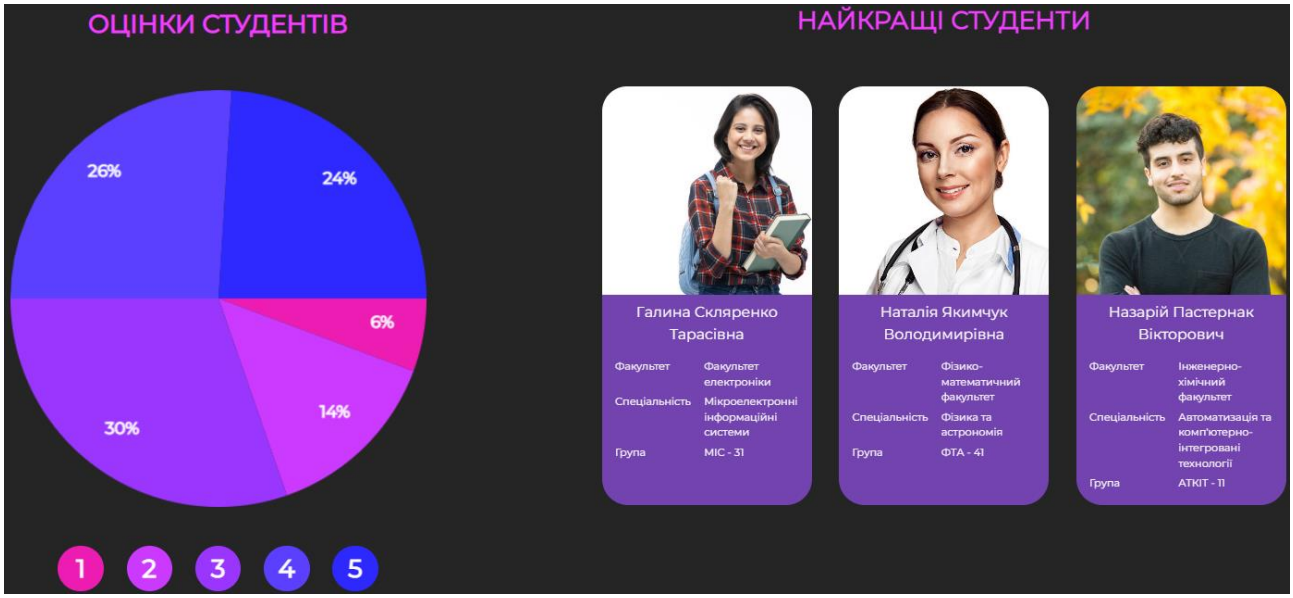


Рис. 1 – головна сторінка

Як можна побачити на скріншоті на головній сторінці користувач може отримати інформацію про трьох найкращих студентів, а також відповідний відсоток отримання оцінок в шкалі від 1 до 5.

Хоча інформаційна система моніторингу успішності студентів має багато потенційних переваг, є також кілька проблем, які слід врахувати. Однією з головних проблем є необхідність забезпечити точність і надійність даних, зібраних системою. Якщо дані неповні або неточні, це може призвести до неправильних висновків щодо успішності студентів і призвести до невідповідних втручань або стратегій навчання.

Підсумовуючи, впровадження інформаційної системи для моніторингу успішності студентів має потенціал для покращення результатів навчання, надаючи викладачам своєчасні, точні та дієві дані про успішність студентів. Хоча, безперечно, існують проблеми та обмеження, які необхідно вирішити, щоб зробити таку систему ефективною, переваги впровадження такої системи очевидні. Надаючи викладачам інформацію, необхідну для визначення областей, де студентам може знадобитися додаткова підтримка, і допомагаючи відстежувати прогрес студентів з часом.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. D. Roy Greenfeld and A. Roy Greenfeld, "Two Scoops of Django 3.x: Best Practices for the Django Web Framework," Two Scoops Press, 2020.
2. Baker and G. N. Chalkiadakis, "Building Web Applications with Django," in IEEE Software, vol. 27, no.
3. J. Zawinski, "Learning Django Web Development," Packt Publishing, 2015.