**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**
**“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**
**імені ІГОРЯ СІКОРСЬКОГО”**

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

**До захисту допущено:**
**Завідувач кафедри**
_____ Сергій СТІРЕНКО
(підпис)

“__” _____2021 р.

# Дипломний проєкт

**на здобуття ступеня бакалавра**

**за освітньо-професійною програмою “Комп’ютерні системи та мережі”**

**спеціальності 123 “Комп’ютерна інженерія”**

на тему: Веб-платформа для онлайн моніторингу успішності студентів

Виконав : студент 4 курсу, групи ІВ-71

Проценко Андрій Анатолійович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник:

Доцент кафедри ОТ, с.н.с., к.т.н. Антонюк Андрій Іванович _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант (нормоконтроль) д.т.н, проф. Сімоненко В.П _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2021 р.

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**
**“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**
**імені ІГОРЯ СІКОРСЬКОГО”**

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалавр)

Освітньо-професійна програма

“Комп’ютерні системи та мережі”

спеціальність 123 “Комп’ютерна інженерія”

**ЗАТВЕРДЖУЮ**

**Завідувач кафедри**

_____ Сергій СТІРЕНКО

(підпис)

“__” _____ 2021 р.

**ЗАВДАННЯ**

**на бакалаврський дипломний проєкт студента**

Проценка Андрія Анатолійовича

1. Тема роботи *Веб-платформа для онлайн моніторингу успішності студентів*
   керівник проєкту Доцент кафедри ОТ, с.н.с., к.т.н. Антонюк Андрій Іванович

2. Термін здачі студентом закінченого роботи 20 травня 2021 р.

3. Вихідні дані до проєкту *технічна документація, теоретичні та статистичні дані*

4. Зміст пояснювальної записки: проведення аналізу предметної області та існуючих передумов для розробки, проєктування і розробка додатка для моніторингу успішності студентів, провести тестування додатку
5. Перелік графічного матеріалу (з точним позначенням обов’язкових креслень)
6. Консультанти розділів проекту

| Розділ | Консультант | Підпис, дата | |
|---|---|---|---|
| | | Завдання | Завдання прийняв |

| | | | |
|---|---|---|---|
| *Нормоконтроль* | *д.т.н., проф. Сімоненко В. П..* | | |
| | | | |
| | | | |

7. Дата видачі завдання_____

## КАЛЕНДАРНИЙ ПЛАН

| № п/п | Найменування етапів дипломного проєкту | Строк виконання етапів проєкту | Примітки |
|---|---|---|---|
| *1.* | *Затвердження теми проєкту* | *10.12.2021* | |
| *2.* | *Вивчення та аналіз завдання* | *15.12.2021* | |
| *3.* | *Розробка архітектури та загальної структури системи* | *15.03.2021* | |
| *4.* | *Розробка структур окремих підсистем* | *25.03.2021* | |
| *5.* | *Програмна реалізація системи* | *5.04.2021* | |
| 6. | *Оформлення пояснювальної записки* | *15.04.2021* | |
| 7. | *Захист програмного продукту* | *25.04.2021* | |
| *8.* | *Передзахист* | *23.05.2021* | |
| *9.* | *Захист* | *15.06.2021* | |

Студент _____ Андрій ПРОЦЕНКО
(підпис)

Керівник роботи _____ Андрій АНТОНЮК
(підпис)

## Анотація

В даній роботі була розроблена програмна частина сервісу , а також інтерфейс користувача веб-застосунку, який дозволяє користувачу слідкувати за успішністю студентів в режимі онлайн. В ході проектування застосунку та аналізу аналогів було виявлено, що проект є унікальним і схожа функціональність відсутня у більшості проектів. Під час розробки системи була використана клієнт-серверна архітектура та застосована технологія Single Page Application, що забезпечило високу швидкодію розробленої системи, а також кросплатформенність. Також відбувся детальний аналіз області оцінювання та навчання студентів, та її поділ на структурні одиниці в процесі оцінювання. Як результат був розроблений веб-застосунок, що включає в себе весь задекларований функціонал та забезпечує роботу системи з різних платформ. Головна мета програми є надання зручної форми комунікації між викладачем зі студентом та можливість моніторингу результату навчання.



## Annotation

In this work, the program of the service was developed, as well as the user interface of the web application, which can register, report on student progress online. During the design of the application and analysis of analogues, it was found that the project has a unique and similar functionality that is missing in most projects. During the development of the systems, a client-server architecture was used and Single Page Application technology was used, which provided high speed of the developed system, as well as free convenience. There was also a detailed analysis of the field of assessment and training of students, as well as its division into structural units in the assessment process. As a result, a web application was developed, which includes all the declared functionality and support for systems on different platforms. The main goal of the program provides a convenient form of communication between teachers and students and the ability to monitor learning outcomes.

# ВІДОМІСТЬ ДИПЛОМНОГО ПРОЄКТУ

| № з/п | Формат | Позначення | Найменування | Кількість листів | Примітка |
|---|---|---|---|---|---|
| 1 | А4 | | Завдання на дипломний проєкт | 2 | |
| 2 | А4 | ІАЛЦ.467200.001 ВП | Відомість проєкту | 1 | |
| 3 | А4 | ІАЛЦ.467200.002 ТЗ | Технічне завдання | 3 | |
| 4 | А4 | ІАЛЦ.467200.003 ПЗ | Пояснювальна записка | 53 | |
| 5 | А3 | ІАЛЦ.467200.004 Д1 | Сховище даних. Схема даних. | 1 | |
| 6 | А3 | ІАЛЦ.467200.005 Д2 | Схема структурна Діаграма компонентів | 1 | |
| 7 | А4 | ІАЛЦ.467200.006 Д3 | Схема функціональна Блок-схема алгоритму | 1 | |
| 8 | А4 | ІАЛЦ.467200.007 Д4 | Схема алгоритму аутентифікації користувача. | 1 | |
| 9 | А4 | ІАЛЦ.467200.008 Д5 | Лістинг | 22 | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

| Змн | Арк. | № документа | Підпис | Дата | ІАЛЦ.467200.001 ВП | | |
|---|---|---|---|---|---|---|---|
| Розробив | | Проценко А.А. | | | Відомість дипломного проєкту | Літ. | Арк. 1 / Аркушів 1 |
| Перевірив | | Антонюк А.І. | | | | | |
| Норм. контр. | | Сімоненко В.П. | | | | НТУУ "КПІ", ФІОТ, ІВ-71 | |
| Затвердив | | | | | | | |

# ТЕХНІЧНЕ ЗАВДАННЯ

## до дипломного проєкту
## освітньо-кваліфікаційного рівня___бакалавр___

на тему: “Веб-платформа для онлайн моніторингу успішності студентів”

Київ – 2021 року

## ЗМІСТ



| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | ІАЛЦ.467200.002 ТЗ | | |
| Змн | Арк. | № документа | Підпис | Дата | | | |
| Розробив | | Проценко А.А. | | | Веб-платформа для онлайн моніторингу успішності студентів | Літ. | Арк. 1 / Аркушів 3 |
| Перевірив | | Антонюк А.І. | | | | | |
| Норм. контр. | | Сімоненко В.П. | | | Технічне завдання | НТУУ "КПІ", ФІОТ, ІВ-71 | |
| Затвердив | | | | | | | |

## 1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Дане технічне завдання розповсюджується на розробку веб-платформи для онлайн моніторингу успішності студентів.

Область застосування: моніторинг успішності студентів.

## 2. ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки служить покращення процесу моніторингу успішності та покращення комунікації студента з викладачем.

## 3. МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою даного проекту є розробка веб-платформи для онлайн моніторингу успішності студентів.

## 4. ДЖЕРЕЛА РОЗРОБКИ

Джерелами для розробки служать науково-технічна література з комп’ютерних технологій, публікації в періодичних виданнях, довідники з програмованих логічних інтегральних схем, публікації в Інтернеті за даним питанням.

## 5. ТЕХНІЧНІ ВИМОГИ

### 5.1 Вимоги до розроблюваного продукту

- Розробка серверної частини веб-платформи
- Проектування баз даних
- Розробка клієнтської частини.

### 5.2 Вимоги до програмного забезпечення

• Операційна система MS Windows 7, MS Windows 8/8.1, MS Windows 10, GNU/Linux, MacOS

• Python 3.5 і вище

### 5.3 Вимоги до апаратного забезпечення

• Процесор з тактовою частотою не нижче 2.2 ГГц

• Оперативної пам’яті не менше 2 Гбайт

## 6. ЕТАПИ РОЗРОБКИ

| | Дата |
|---|---|
| Вивчення літератури | 10.03.2021 |
| Складання і узгодження технічного завдання | 10.04.2021 |
| Створення модулів системи, що розробляється | 20.04.2021 |
| Тестування окремих модулів системи | 01.05.2021 |
| Допрацювання, налагодження і виправлення помилок | 10.05.2021 |
| Оформлення документації дипломної роботи | 16.05.2021 |

**Пояснювальна записка до дипломного проєкту**

**на тему: “Веб-платформа для онлайн моніторингу успішності студентів”**

Київ - 2021 року

**ЗМІСТ**

# ВСТУП

Одним із методів який застосовують для характеристики якості навчання виступає аналітика контролю знань, де оцінка знань є головним компонентом інструментів підсумування навчальних здобутків студентів.

Під час підготовки майбутніх фахівців у закладах вищої освіти, виникають проблеми з вільним доступом до результатів їх навчання. Особливо гострою ця проблема постає в ситуаціях, пов’язаних з особливостями організації навчального процесу у закладах освіти певного профілю, коли переважна частина навчальних занять проводиться в практичних умовах, віддалених від закладів навчання. Тому батькам часто важко дізнаватись актуальні результати їх навчання, що в деяких випадках стає відомо вже після підписання наказу про відрахування, коли ситуація невідворотна.

З іншого боку для аналітичних досліджень діяльності вузу обов'язково повинні бути використані відомості, які були отримані в процесі здійснення моніторингу успішності. Проведення моніторингу призначене як для забезпечення органів управління ВНЗ надійними і достовірними даними, так і для швидкого виявлення проблем в організації та проведенні навчального процесу.

Таким чином, даний дипломний проєкт присвячений розробці веб платформі, яка дозволить у онлайн режимі надавати певну аналітичну інформацію конкретним особам з аналітичною метою. Дане рішення спростить комунікацію між усіма сторонами навчального процесу і дозволить в автоматичному режимі формувати звітність у конкретному ВНЗ для ефективного моніторингу успішності.

# РОЗДІЛ 1

# ОГЛЯД СИСТЕМ МОНІТОРИНГУ УСПІШНОСТІ СТУДЕНТІВ

## 1.1 Огляд проблеми, яка вирішується ПЗ

Найбільш важливою складовою контролю процесу управління освітньою системою вищого навчального закладу є моніторинг успішності студентів. Для аналітичних досліджень діяльності вузу обов'язково повинні бути використані відомості, які були отримані в процесі здійснення моніторингу успішності. У вузах на кафедрах та інших навчальних підрозділах надаються функції по проведення моніторингу успішності студентів. Проведення моніторингу призначене як для забезпечення органів управління ВНЗ надійними і достовірними даними, так і для швидкого виявлення проблем в організації та проведенні навчального процесу.

Моніторинг успішності студентів повинен проходити поетапно [3, 5]

Перший етап. Збір, аналіз, класифікація і систематизація інформаційних даних і ресурсів з журналів обліку відвідування та успішності студентів на навчальних заняттях. Цей етап є найбільш тривалим за часом, відведеного на моніторинг успішності.

Другий етап. Обчислення ключових показників успішності учнів, а далі їх аналіз.

Третій етап. Формуються виявлені факти та висновки про успішність студентів вузу на базі показників, обчислених на другому етапі. Оброблення значної кількості інформаційних даних в ручному режимі вимагає великої кількості часу і додаткового допоміжного персоналу. На всіх етапах моніторингу при ручній обробці інформації дуже часто з'являються помилки.

Для підвищення продуктивності процесів моніторингу успішності студентів вузу та його якості, тобто безпомилковості, необхідно впровадити в процес моніторингу програмно-технічні засоби автоматизації.

У ВНЗ України програмно-технічні засоби автоматизації моніторингу успішності різнорідні за своїм складом і змістом. Це в основному спеціально налаштовувані файл-серверні системи управління базами даних, які дозволяють в спеціально розроблені форми вводити дані про успішності студента та здійснювати запити з розрахунком показників успішності в автоматичному режимі.

Системи управління базами даних, налаштовані на роботу з моніторингу успішності студентів, мають такі недоліки [8, 10]:

- невелика надійність, тому що система управління базами даних є універсальним засобом програмування, в якій часто відбуваються збої, особливо після оновлення самої системи і баз даних;
- низька розширюваність і удосконалень, тому що функціонал системи визначається можливостями внутрішньої мови програмування, яка є досить вузькою для великих і складних проектів;
- аналіз введених інформаційних даних обмежений вбудованим математичним апаратом, тобто для складного аналізу даних необхідно розробляти додаткові модулі програм;
- взаємозв'язок з іншими інформаційними системами повинна здійснюється за допомогою спроектованих програмних засобів моніторингу успішності студента з обов'язковим мережевим доступом до даними всіх учнів, а також зберіганням інформаційних даних і видачею необхідної інформації в інформаційні системи електронного документообігу вузу. При цьому необхідно враховувати що інфокомунікаційна система моніторингу успішності студентів повинна експлуатуватися в окремій ізольованій комп'ютерної мережі вузу [4, 9, 10].

За якість введення інформаційних даних в такі системи, їх експлуатацію і обслуговування відповідає адміністратор баз даних.

Перевагою застосування систем управління базами даних є порівняльна спрощеність в розробці і виконання програм моніторингу, а також облік успішності студентів з кожної дисципліни, структурному підрозділу вузу і т.д.

За якість введення інформаційних даних в такі системи, їх експлуатацію і обслуговування відповідає адміністратор баз даних, який повинен мати не тільки високу кваліфікацію в програмуванні, але і знати структуру управління ВНЗ [6].

Архітектура доступу до інформаційної системи моніторингу успішності повинна бути побудована на технології доступу - «клієнт-сервер». Клієнт-серверна технологія на відміну від файл-серверної дозволить знизити навантаження інформаційних даних на мережу, підвищить зручність її управління і адміністрування, збільшити її надійність і рівень інформаційної захищеності, а також застосовувати тонкі клієнти, що економічно дуже вигідно при великій кількості точок доступу [5, 8].

Найбільш інноваційним напрямом в побудові локальних мереж є застосування «тонких клієнтів» на стороні клієнта. «Тонкий клієнт» - це клієнтська програма, яка переносить більшість з численних завдань на сервер.

Наприклад, веб-браузер є класичним прикладом функціоналу тонкого клієнта при роботі з веб-додатками. Розробка автоматизованої інфокомунікаційної системи у вигляді веб-додатків, безсумнівно, є зараз найбільш перспективною галуззю завдяки її численним перевагам по порівняно з традиційними клієнтськими додатками.

Кросплатформеність є ключовою перевагою веб-додатків, тому що вони не залежать від виду операційної системи і не вимагають встановлювати додаткове клієнтське програмне забезпечення. Це дає можливість застосовувати для доступу до інформаційних даних крім персонального комп'ютера ще й інші електронні гаджети (планшети і телефони) як по провідних, так і по безпровідних лініях [4, 7].

Мета даної роботи - розробка електронної системи моніторингу успішності з вільним доступом до неї студентів та викладачів.

## 1.2 Опис вимог до розроблюваного ПЗ

Потрібно сформувати ряд вимог, яким має бути веб-платформа для онлайн моніторингу успішності студентів.

Головна вимога, враховуючи сучасні реалії, додаток має бути доступним на ПК не зважаючи операційну систему Windows чи Linux, а також підтримка платформ iOS чи Android.

Уніфікований та інтуєтивно зрозумілий інтерфейс є важливим критерієм. Сервіс має бути зрозумілим для користувачів, в інакшому випадку, якщо інтерфейс матиме складнощі в освоєнні його функціональності, дане програмне рішення не здобуде популярності в широких масах.

Потрібно забезпечити схожий інтерфейс на мобільних і десктопних пристроях для легкого переходу користувачів між цими платформами. В сучасних реаліях присутні шаблони інтерфейсів до яких користувачі вже звикли і використання інших патернів відображення може бути не зручним для користувача.

Використані технології  мають забезпечити коректну роботу системи навіть при слабкому зв’язку або навіть при повній відсутності мережі Інтернет.

Для доступу в систему виділимо такі групи: адміністратор, викладач та студент. Для кожної групи виділимо конкретну функціональність для користування сервісом.

Адміністратор виконує тільки сервісну функцію не маючи доступу до бази даних.

Головна вимога, враховуючи сучасні реалії, додаток має бути доступним на ПК уніфікований та інтуєтивно зрозумілий інтерфейс є важливим критерієм.

Викладачу має бути доступна наступна функціональність:

- Створення класу/групи/напряму доступ до якого надає конкретним студентам, для перегляду оцінок та аналітики.
- Створення таблиці оцінювання для конкретного напряму/групи/класу.
- Можливість редагування власного профілю.
- Переглянути загальну аналітику по системі яка не прив’язана до конкретних прізвищ.

Інтерфейс користувача студента має такі пункти:

- Перегляд власних оцінок в системі.
- Переглянути загальну аналітику по системі яка не прив’язана до конкретних прізвищ .
- Можливість редагування власного профілю.

Система мусить містити інтерфейси авторизації, автентифікації. Так як продукт може використовуватись не тільки україномовними громадянами, на сайті має бути присутня локалізація (українська та англійська мова).

Підсумовуючи, вимоги які були наведені вище до системи приведені наступні вимоги:

1. Багатоплатформність.
2. Робота при слабкому підлюченні до мережі Інтернет.
3. Локалізація.
4. Інтуєтивно зрозумілий інтерфейс[10].
5. Захищеність.

### 1.3 Аналіз існуючих рішень

При розгляді питання про створення системи для моніторингу успішності студентів було виявлено, що найбільш популярним програми, що працюють лише як десктопна програма і не мають нативної підтримки IOS та Android.

Для розуміння функціональності платформи, проаналізовано програмні засоби, розроблені іншими компаніями.

### 1.3.1 Система «Infinite Campus»

Infinite Campus - це комплексна веб-інформаційна система студентів (SIS) з доступом в реальному часі до адміністрації, інструктажів, спілкування, навчальної програми, звітування тощо.

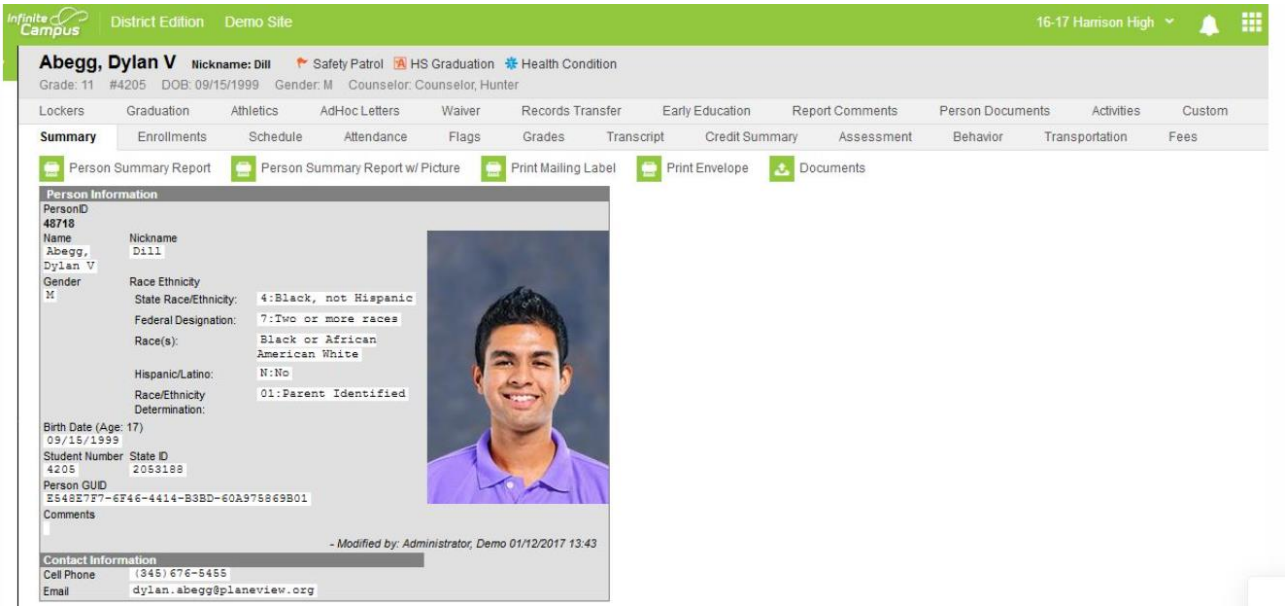



Рис. 1.1 Інтерфейс «Infinite Campus»

Портали Infinite Campus - Campus Parent та Campus Student - надають доступ до інформації для всіх студентів не виходячи з додому. Інформація включає оцінки, відвідуваність, завдання та шкільні оголошення.

Примітка: Батьки повинні зареєструватися, коли вперше отримують доступ до свого облікового запису.

Усі студенти мають доступ до Campus Student через ClassLink.

Доступний обмін даними через батьківські портали: вивчення досвіду батьків, використання даних і тривала та постійна участь батьків в навчальному процесі[11].

### 1.3.2 Система «PowerSchool»

PowerSchool - це платформа освітніх технологій, що належить компанії PowerSchool Group LLC, штаб-квартира якої знаходиться в місті Фолсом, штат Каліфорнія. Програмні продукти компанії включають інформаційні системи для студентів: онлайн реєстрацію та вступ студентів, оцінювання, аналітичний сервіс, управління спеціальною освітою, планування ресурсів університету та систему управління навчанням[12].

Можливості PowerSchool включають такі основи як звітність, відвідуваність та дисципліна, а також багато інших спеціалізованих можливостей, включаючи батьківський доступ до оцінок учнів в режимі реального часу та підтримкою кількох навчальних закладів на одному сервері.

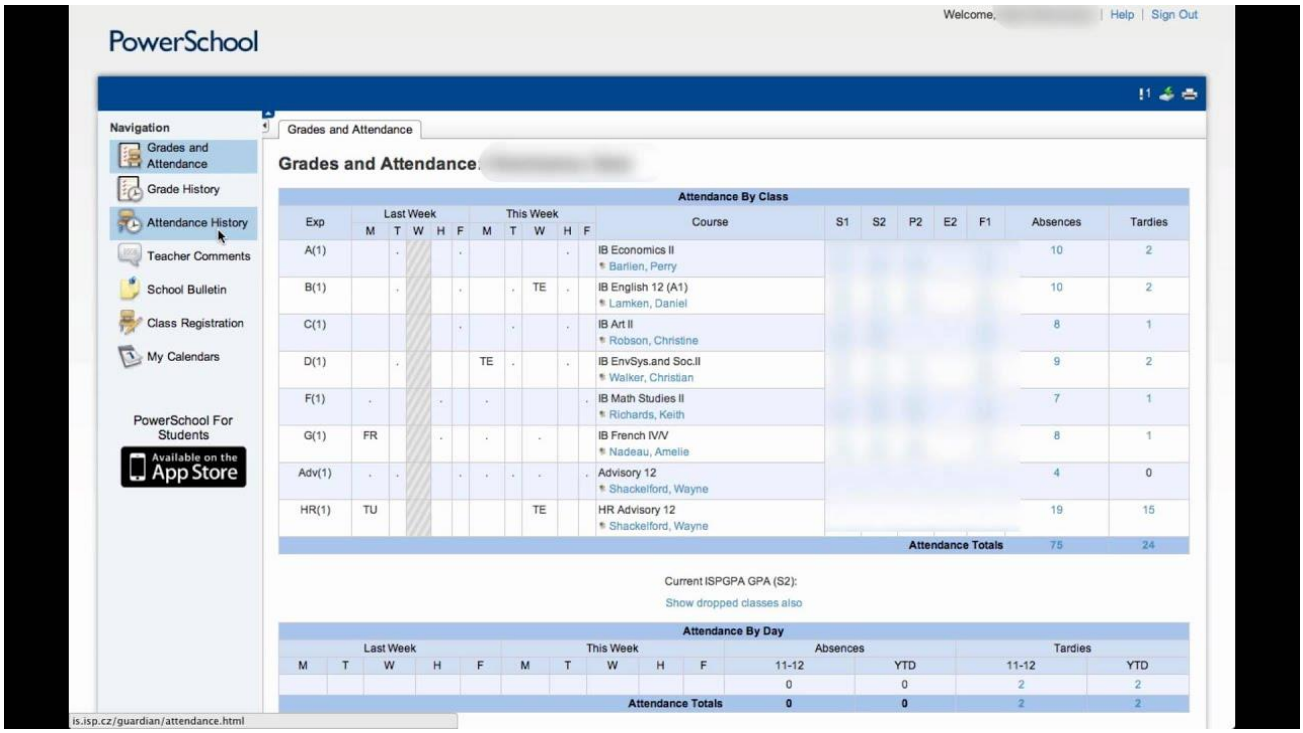



Рис. 1.2 Інтерфейс «PowerSchool»

Всі результати оцінювань та відвідувань перебувають в електронному вигляді, завдяки цьому система досить легко інтегрується в навчальний процес створюючи власний електронний журнал для ведення оцінок та відвідування уроків.

Вся ця інформація передається в електронному вигляді до системи PowerSchool, де авторизовані користувачі можуть отримати до неї доступ.

Це означає, що доступ до даних у режимі реального часу мають не лише адміністратори, а й батьки, вчителі та студенти. За допомогою традиційних методів звітування батьки не мають постійного контролю над успішністю. PowerSchool дозволяє батькам постійно знати, як поводиться їхній учень, використовуючи Інтернет.

PowerSchool може надсилати електронні звіти про прогрес по електронній пошті. Оскільки доступ до системи здійснюється через браузер, тому є можливість використання системи на всіх основних платформах, включаючи мобільні пристрої.

Система написана на мові програмування Java.

### 1.3.3 Система «Student Monitoring System»

Student Monitoring System - новаторський продукт від ASR Digitech. Додаток допомагає інститутам організовувати та вдосконалювати свої оперативні можливості, а отже, пропонувати найкращу якість освіти для своїх студентів[13].

Щоб користуватися цією програмою, потрібно мати студентський акаунт ASR.

Список функцій які доступні:

- Позначення відвідуваності
- Додавання та керування оцінками та оцінками студентів
- Автоматизовані звіти про відвідуваність, оцінки та інші важливі дані
- Підтримка інформаційної панелі батьків, студентів, викладачів та управління
- Повідомлення батьків у режимі реального часу за допомогою SMS / електронної пошти

- Гнучкість інтерфейс для зворотного зв'язку батьків та вчителів
- Галерея закладу
- Розклад занять та підтримка миттєвого сповіщення про зміну розкладу.

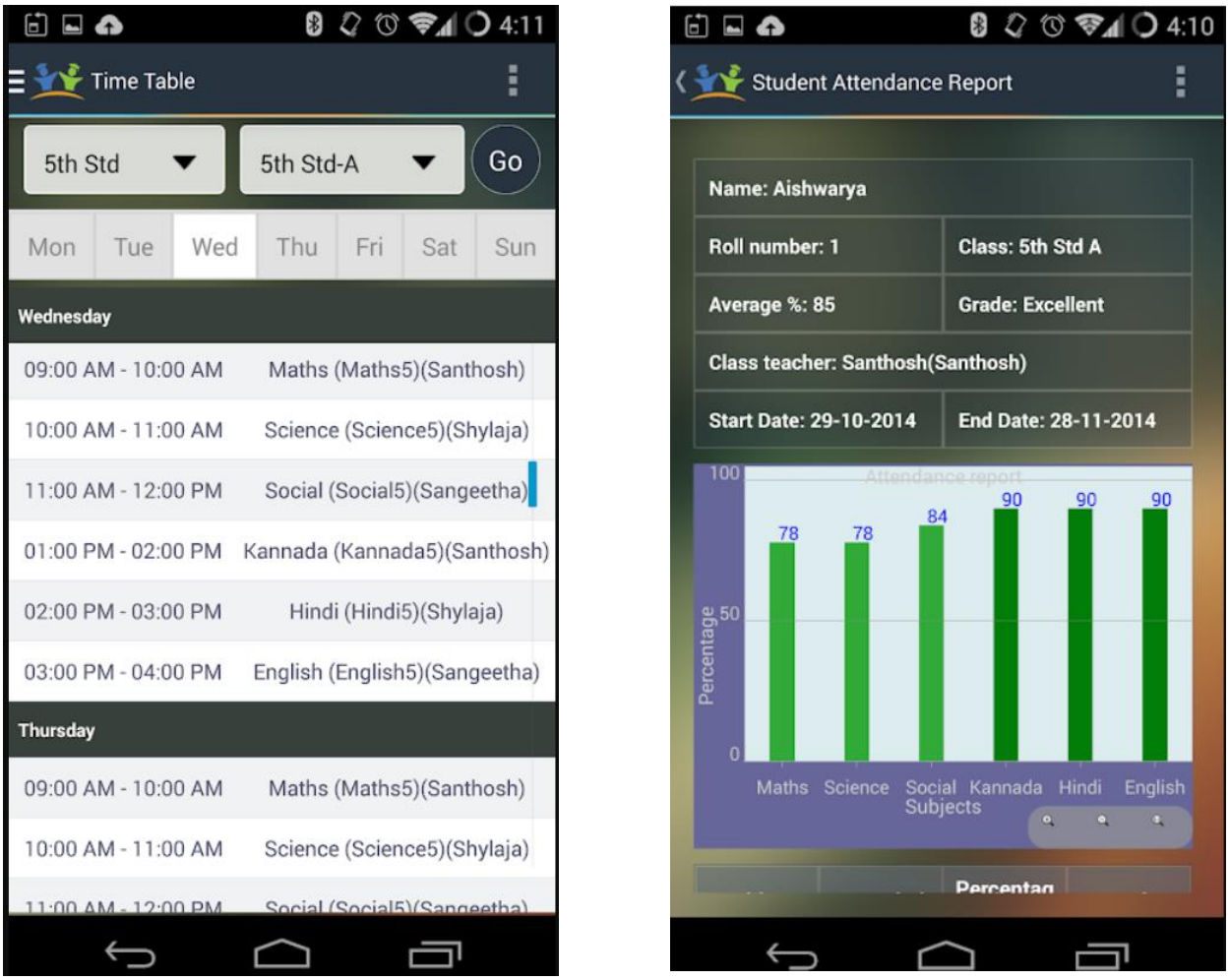



Рис. 1.3 Інтерфейс «Student Monitoring System»

### 1.3.4 Система «GISMO»

GISMO - Graphical Interactive Student Monitoring System.

GISMO - це система моніторингу та відстеження успішності студентів, яка використовує дані з Moodle, перетворює дані у форму, зручну для обробки, та генерує графічні подання, які можна досліджувати та маніпулювати ними для вивчення соціальних, когнітивних та поведінкових аспектів студентів. Ця робота є лише частиною більш широкого Європейського дослідницького проекту [14].

GISMO, витягує дані відстеження з онлайн-курсу, що підтримується Moodle, та генерує графічні зображення, які можуть бути вивчені викладачами курсу. Ця робота є лише частиною більш широкого Європейського дослідницького проекту. Більше візуалізації за словами розробників буде впроваджено найближчим часом, і вони очікують, що цей інструмент буде інтегрований на офіційний сайт розповсюдження Moodle як додатковий модуль, що встановлюється.

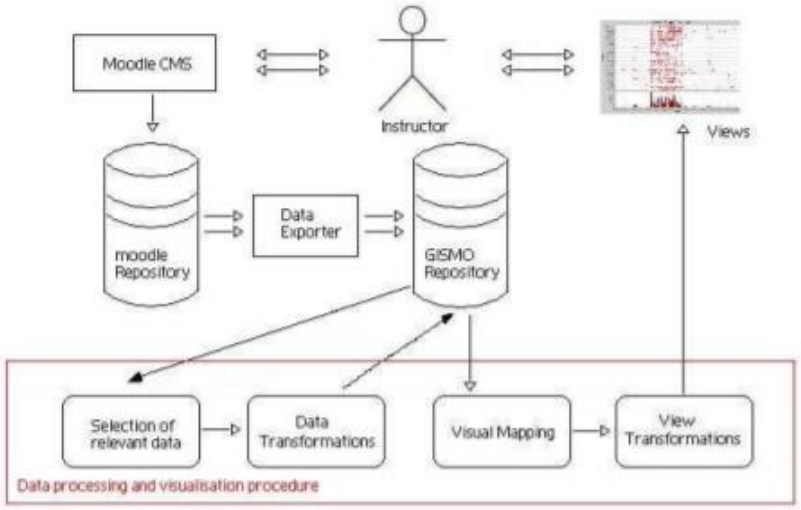



**Fig. 1.** The GISMO system architecture

Рис. 1.4 Архітектура GISMO системи

Доступні 3 режими перегляду:

- Панель діаграм: на цій панелі малюються діаграми.
- Панель списків: містить перелік студентів, груп, ресурсів тощо звичайно. Для кожного списку викладач може вибрати/скасувати вибір даних для візуалізації.
- Часова панель: за допомогою цієї панелі ви можете скоротити час вибору.
- Панель з таблицею: містить перелік студентів, груп, ресурсів тощо звичайно.

Графічні подання, створені за допомогою GISMO, можуть допомогти викладачам для виявлення осіб, які потребують особливої уваги, для виявлення закономірностей в режимі доступу та обговорення, а також обмірковувати свою педагогічну практику. Опис цієї оцінки виходить за межі даної роботи, і подробиці описані в [15].

## ВИСНОВКИ ДО РОЗДІЛУ 1

Після проведення аналізу виходячи з отриманих даних, можна стверджувати, що згадані системи повністю вирішують конкретно поставлені задачі і є комплексним рішенням, яке можна охарактеризувати як комплекс програмного забезпечення в якому присутня аналітична функціональність. Більшість із описаних програм орієнтовані на використання лише на персональних комп'ютерах або мобільних пристроях, деякі лише в браузері, в деяких не вистачає функціоналу, або вони мають проблеми з оновленням.

Отже, в підсумку отримано порівняльну таблицю результатів аналізу програмних комплексів (табл. 1.1).

В кінцевому рахунку, при аналізі систем і характеристики основної функціональності, для програмного рішення задачі моніторингу успішності студентів виділено ряд вимог:

- Локалізація
- Кросплатформеність
- Коректна робота при повільному з'єднанні з інтернетом
- Просте оновлення версій
- Інтуєтивно зрозумілий інтерфейс

Таблиця 1.1 Результати аналізу існуючих програмних рішень

| | Infinite Campus | PowerSchool | Student Monitoring System | GISMO |
|---|---|---|---|---|
| Локалізація | - | - | - | - |
| Кросплатформеність | + | - | - | - |
| Висока швидкодія | + | - | - | + |
| Зрозумілий інтерфейс | - | - | + | + |

Програмне рішення має вирішувати наступні проблеми:

1. Кросплатформеність.
2. Проблему моніторингу успішності студента
3. Аналіз результатів моніторингу.
4. Взаємодія з іншими системами

Отже, після проведення аналізу існуючих систем відповідно до висунутих вимог, маємо наступні елементи для створення програмного продукту відповідно до теми:

1. Розроблення клієнтської частин:

1.1. Створення інтерфейсу

1.2. Кросплатформеність

1.3. Висока швидкодія

2. Створення серверної частини.

# РОЗДІЛ 2

# ОГЛЯД ЗАСОБІВ РЕАЛІЗАЦІЇ

## 2.1 Огляд інструментів для розробки клієнтської частини

Так як рішення є клієнт-серверним, тобто складається з двох самостійних частин - працює в браузері і працює на сервері, то і аналіз доступних мов і інструментів буде розділений на дві частини.

Оскільки наша платформа є веб-орієнтованою, її клієнтська частина повинна працювати у веб-браузері. Незважаючи на те що браузери підтримують мову JavaScript, ця мова не є єдиним використовується в розробці веб додатків. існує безліч мов, розроблених спеціально щоб компілюватиметься в JavaScript, а також безліч інструментів, які дозволяють компілювати програми, написані на інших мовах, в JavaScript. До таких мов відносяться:

- TypeScript
- Elm
- CoffeScript
- Grooscript
- Scala.js
- Emscripten

Серед доступних варіантів найбільш доцільним видається використання мов, спеціально розроблених для компіляції в JavaScript. Розглянемо далі мови, зазначені в перших трьох пунктах.

CoffeScript - найстаріша з наведених мов. Була популярна серед розробників веб-клієнтів які отримують дані з сервера , але з розвитком стандарту ECMAScript, був витіснений звичайним JavaScript. Основні відмінності цієї мови від TypeScript і Elm - то, що вона не має статичної типізації. І потребує технології Babel для компіляції, а також безліч інструментів, які дозволяють компілювати програми в JavaScript. Технології вже більше 10 років.

Elm - функціональна мова програмування, створена спеціально для розробки веб-додатків. Офіційна документація заявляє, що одна з головних особливостей мови в тому, що вона дозволять уникнути runtime помилок. Так само його особливостями можна вважати те, що він має опціональну статичну типізацію, а також компілюється в JavaScript з-за допомогою власного компілятора.

TypeScript - мова програмування, яка розроблена і підтримується Microsoft. Є підмножиною мови JavaScript. Це означає, що будь-яка програма на мові JavaScript є так само коректною програмою на мові TypeScript. Так само, як Elm, ця мова має опціональну статичну типізацію і компілюється в JavaScript з-за допомогою власного компілятора. Тоді як використання TypeScript або Elm можуть істотно полегшити деякі аспекти розробки, наприклад, завдяки наявності статичної типізації, саме по собі додавання інструменту - мови, компільованої в JavaScript, збільшить складність самого процесу розробки, тому в рамках цього проекту буде використаний лише JavaScript.

Сучасна екосистема мови JavaScript надає вибір з багатьох інструментів для розробки клієнтської частини програми.

Розділимо їх на наступні категорії:

- Інструменти управління станом додатки
  - ✓ Flux
  - ✓ Redux
  - ✓ Mobx
  - ✓ Інші
- Інструменти для відображення інтерфейсу
  - ✓ React
  - ✓ Vue
  - ✓ Інші
- Фреймворки, що реалізують обидва завдання
  - ✓ Angular
  - ✓ Ext.js

Порівняємо між собою інструменти з кожної категорії, щоб зрозуміти, чи потрібні вони в даній розробці.

Flux і Redux досить схожі один на одного - обидва реалізують запропоновану Facebook flux архітектуру, альтернативу MVC архітектурі. Особливість архітектури полягає в однобічному потоці даних, тобто для зміни store, який грає роль моделі, використовуються action, що проходять через його dispatch і змінюють стан з допомогою reducer - чистих функцій. Різниця між Flux і Redux ж полягає в тому, що з використанням Flux прийнято розділяти стан додатки на кілька store, тоді як в Redux існує тільки один store, що описує весь стан додатку.

Mobx, згідно з документацією, не є контейнером для зберігання стану програми, хоча може використовувати в якості такого. Mobx дозволяє реактивно управляти станом і не прив'язує розробника до якоїсь певної архітектури.

Так як по постановці завдання проекту всі обчислення будуть проводитися на стороні сервера, клієнтська частина не передбачає наявності будь-яких складних даних і маніпуляцій з ними. Це дозволяє обійтися в даному проекті без використання даних інструментів.

React і Vue - інструменти, створені для завдання відображення інтерфейсу, тоді як Angular і Ext.js поєднують в собі інструменти так і для управління його станом. На відміну від React і Vue, Angular і Ext.js так само змушують розробника використовувати певну архітектуру при написанні програми.

Даний проект має на меті використання складних інтерфейсів, тому для полегшення розробки застосована технологія React.

Обмовимося, що не варто переоцінювати важливість вибору конкретного інструменту - детальне вивчення одного полегшить розуміння інших. Так чи інакше, зрозумілі будуть не тільки особливості, характерні для одного фреймворка, але і загальні концепції, застосовні до веб-розробки в цілому: проектування компонентів, розуміння потоку даних, управління станом, шаблонами і т. д.

### 2.1.1 Бібліотека React

React - це JavaScript бібліотека, яку використовують для проектування та створення веб-сторінок. React створили і перша версія була випущена у 2013 році, на даний час актуальна версія 17.0.2. Створений React відомою компанією Facebook.

На початку React був створений для WEB, для створення веб-сайтів, проте у скорому часі на основі технології, була випущена платформа React Native, яка націлена на створення додатків для мобільних пристроїв. React це ідеальний інструмент для створення масштабованих веб-додатків, це стало можливим завдяки використанню технології Single Page Application (SPA), детальніше про яку буде сказано в наступному розділі.

Для роботи з React, потрібно добре знати JavaScript, HTML5 та CSS. Незважаючи на те, що React не використовує HTML, JSX схожий, тому знання HTML будуть дуже корисні. JSX це щось середнє між JavaScript і XML, що дає нам простий та інтуїтивно зрозумілий спосіб для визначення коду візуального інтерфейсу.

Докладніше пояснення цього в одному з наступних розділів. Також використовуватиметься синтаксис EcmaScript 2015, тому будь-які знання в цій галузі також потрібні.

React - це бібліотека JavaScript, яка використовується для побудови багаторазових компонентів інтерфейсу. Згідно з офіційною документацією React, таке визначення: «React - це бібліотека для побудови компонованих інтерфейсів користувача»[19]. Це заохочує створення багаторазових компонентів інтерфейсу, які представляють дані, які змінюються з часом. Багато людей використовують React як V (View) у MVC. Реагути на елементи від DOM, пропонуючи простішу модель програмування та кращу продуктивність. React також може відображатись на сервері за допомогою Node, а також може подавати власні програми за допомогою React Native.

React реалізує односторонній реактивний потік даних, що зменшує шаблон і легше міркувати, ніж традиційне прив'язування даних.

Односпрямований потік даних і потік - React реалізує односторонній потік даних, що полегшує міркування щодо вашого додатка. Flux - це шаблон, який допомагає зберегти односпрямованість ваших даних.

Ліцензія - React ліцензований під Facebook Inc. Документація ліцензована під CC BY 4.0.

Вся структура веб-сторінки може бути представлена за допомогою DOM (Document Object Model) - організація елементів html, якими ми можемо маніпулювати, змінювати, видаляти або додавати нові. Для взаємодії з DOM застосовується мова JavaScript. Однак коли ми намагаємося маніпулювати html-елементами за допомогою JavaScript, то ми можемо зіткнутися зі зниженням продуктивності, особливо при зміні великої кількості елементів. А операції над елементами можуть зайняти деякий час, що неминуче позначиться на призначеному для користувача досвід. Однак якби ми працювали з коду js з об'єктами JavaScript, то операції проводилися б швидше.

Для вирішення проблеми продуктивності якраз і з'явилася концепція віртуального DOM.

Використання віртуального DOM, який є об'єктом JavaScript, покращує продуктивність додатків, оскільки віртуальний DOM JavaScript швидший за звичайний DOM.

React може використовуватися на стороні клієнта та сервера, а також з іншими фреймворками. Компоненти та шаблони даних покращують читабельність, що допомагає підтримувати великі програми. Охоплює лише шар перегляду програми, тому вам все одно потрібно вибрати інші технології, щоб отримати повний набір інструментів для розробки. Використовує вбудовані шаблони та JSX, що може здатися незручним для деяких розробників.

Віртуальний DOM представляє систематизовану копію звичайного DOM.  Саме в React це є важливим елементом, тому що ця бібліотека працює саме з віртуальним DOM, що позначається на більшій швидкості роботи.

Якщо з додатком потрібно дізнатися інформацію про стан елементів, то відбувається звернення до віртуального DOM.

Якщо необхідно змінити елементи веб-сторінки, то зміни спочатку вносяться в віртуальний DOM. Потім новий стан віртуального DOM порівнюється з поточним станом. І якщо ці стани розрізняються, то React знаходить мінімальну кількість маніпуляцій, які необхідні до поновлення реального DOM до нового стану і виробляє їх.

У результаті такий алгоритм взаємодії з елементами веб-сторінки допомагає реагувати набагато швидше і ефективніше, ніж якби застосований був DOM безпосередньо, з коду JavaScript.

Ключова відмінність бібліотеки є поділ на компоненти - ми можемо створити окремі частини (компоненти) і потім їх легко можливо інтегрувати в інший проект.

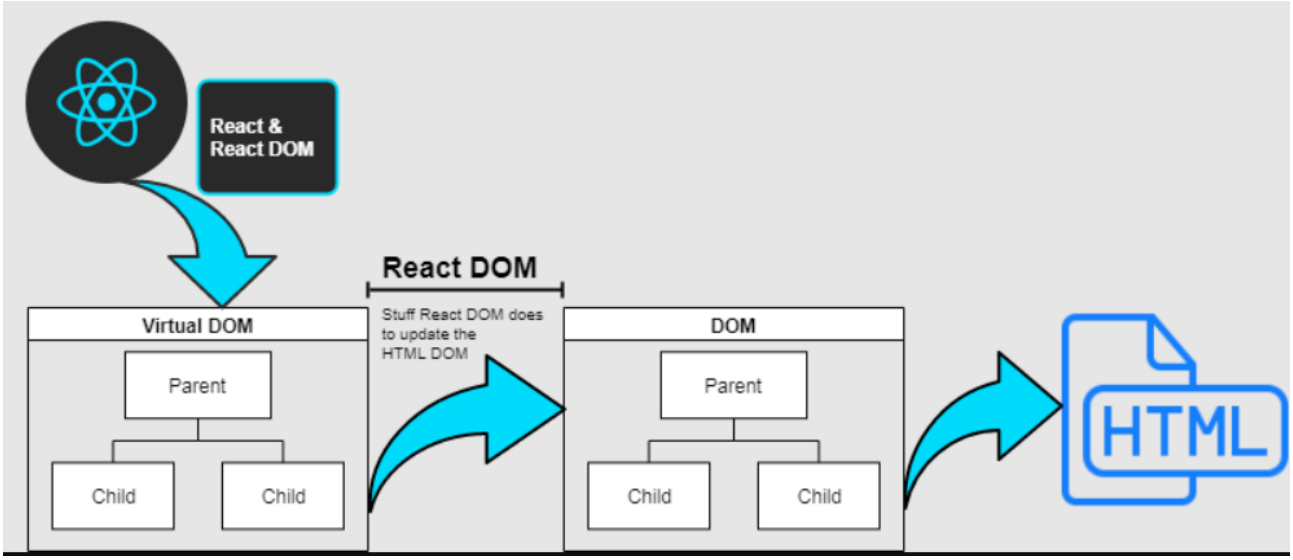



Рис. 2.1 React Virtual DOM

### 2.1.2 SPA

Всі веб-додатки поділяються на односторінкові (SPA) та багатосторінкові (MPA). SPA, Single Page Application, або «додаток однієї сторінки» - це тип веб-додатка, при завантаженні якого необхідне кодування відбувається на одній сторінці. Це дозволяє заощадити час на повторному завантаженні одних і тих же елементів.

В перекладі SPA - це буквально одна сторінка, яка постійно взаємодіє з користувачем, динамічно переписує поточну сторінку та не завантажує цілі нові сторінки із сервера.

Багатосторінковий додаток - це традиційний веб-додаток, у якому із сервера завантажується нова сторінка для відображення кожного разу, коли відбувається обмін даними туди і навпаки. Об'єктом контенту, який вони несуть, великий, тому вони, як правило, багаторівневі, що мають значну кількість посилань та складних користувацьких інтерфейсів.

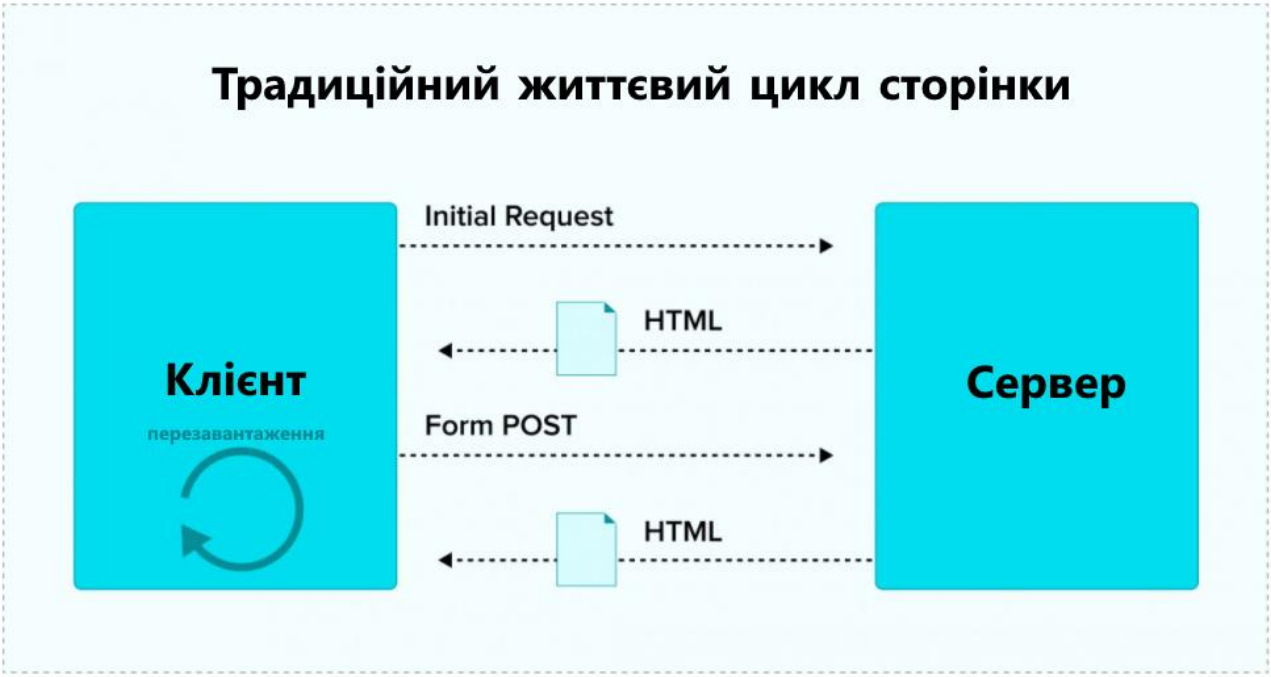



Рис. 2.2 Традиційний цикл життя веб-сторінки

В перекладі SPA - це буквально одна сторінка, яка постійно взаємодіє з користувачем, динамічно переписує поточну сторінку та не завантажує цілі нові сторінки із сервера.

Ми бачимо приклади односторінкових програм кожного дня: Trello, Facebook, Gmail, Twitter - ось кілька прикладів SPA.

Особливість архітектури SPA є в тому, що всі елементи, необхідні для роботи софту, знаходяться на одній сторінці. Вони завантажуються при ініціалізації. Також даний вид додаткового завантаження завантажує додаткові модулі після запрошення від користувача. Люба користувацька активність фіксується для зручності навігації. Це дозволяє скопіювати посилання та відкрити програмне забезпечення на тому же етапі взаємодії на іншій вкладці, браузері чи пристрої.

При завантаженні нових модулів у контент SPA на них оновлюється лише частково, так як немає необхідності повторно завантажувати невідомі елементи. Це збільшує швидкість відкриття та скорочує переданий обсяг даних між браузером та сервером.

Даний вид софту за способами взаємодії з користувачем більше всього схожий на роботу десктопних додатків, але на серверах.

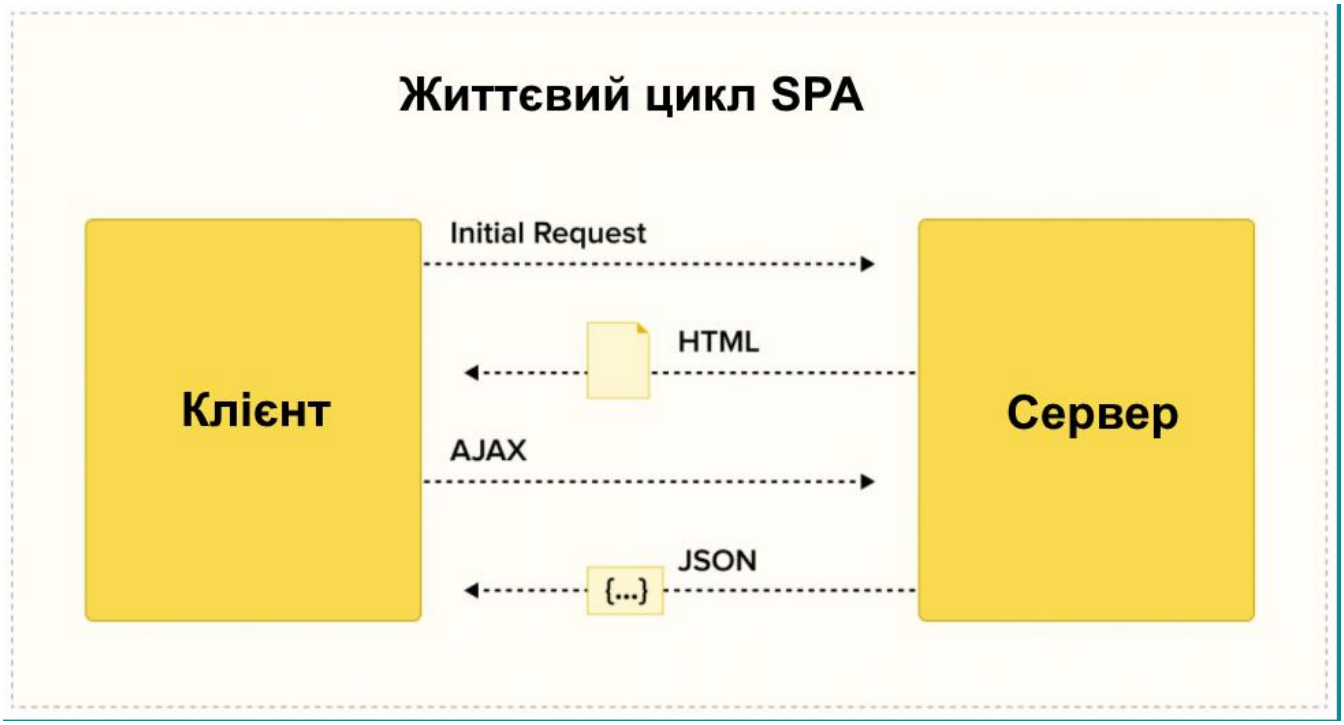



Рис. 2.3 Життєвий цикл SPA

Переваги SPA сторінок:

- Доступність. Ми можемо отримати миттєвий доступ до функціональності будь-якого типу пристрою без проблеми із сумісністю, об'ємом пам'яті, можливостями або часом роботи на установці.
- Універсальність. Використовувати програмне забезпечення можна практично з будь-яким пристроєм, якщо він не доступний в Інтернеті. Якщо розробник інтерфейсу використовує різні варіанти екрану, щоб використовувати SPA одночасно зручно та ПК та смартфон.
- Можливість використання більших обсягів даних. Розмір додатків і використовуваних даних у них не обмежений.

- Швидкість. Одна сторінка всім необхідним не тільки економить час при повторному завантаженні даних, але і підвищує продуктивність роботи.
- Можливості розробки. Розробникам доступні фреймворки, які сприяють створенню архітектурних проектів та надають безліч готових елементів для роботи.

### 2.1.3 JSX

JavaScript XML (JSX) - це розширення синтаксису JavaScript, яке дозволяє використовувати HTML-подібний синтаксис для опису структури інтерфейсу. Як правило, компоненти написані з використанням JSX, але також є можливість використання звичайного JavaScript . JSX нагадує іншу мову, створену в компанії Фейсбук для розширення PHP, XHP.

JSX - це теж вираз. Після компіляції JSX-вирази стають звичайними JavaScript-об'єктами. Це означає, що можливо використовувати JSX-вираз всередині операторів if і циклів for, привласнювати його змінній, приймати в якості аргументу і повертати його з функції.

## 2.2 Огляд інструментів для розробки серверної частини

Для реалізації серверної частини програми існує велика кількість різних мов і інструментів.

Щоб взаємодіяти з клієнтом, сервер розподіляє необхідні ресурси між процесами взаємодії і чекає на запит на з'єднання (або, по суті, запит на надану послугу). Залежно від типу ресурсу, сервер може обслуговувати процеси в одній комп'ютерній системі або процесах на інших комп'ютерах через канали передачі даних або мережні підключення. Форма запиту клієнта і відповідь сервера визначаються протоколом. Специфікації відкритого протоколу описуються відкритими стандартами, наприклад інтернет-протоколи визначені в документах RFC.

Залежно від виконуваних завдань, сервер без запитів на послуги може бути неактивним заздалегідь.

Інші можуть працювати (наприклад, збирати інформацію); На таких серверах робота з клієнтами може бути незначним завданням.

Основними мовами програмування, на яких написано програмне забезпечення на серверах, є:

- C/C ++
- Java
- C#
- JavaScript
- Python

Для цього проекту був обраний Python та його веб-фреймворк Django.

### 2.2.1 Фреймворк Django

Django - це високорівневий Python веб-фреймворк, який дозволяє швидко створювати безпечні і підтримувані веб-сайти. Створений досвідченими розробниками, Django бере на себе більшу частину турбот веб-розробки, тому можливо зосередитися на написанні свого веб-додатку без необхідності винаходити велосипед. Він безкоштовний і з відкритим вихідним кодом, має зростаючу і активну спільноту, відмінну документацію[16].

Фреймворк - це набір інструментів, які допомагають розробляти програму швидше і простіше. Кожного разу при розробці, потрібні схожі набори компонентів: спосіб авторизації користувачів (вхід, вихід, реєстрація), адмін панель управління сайтом, форми, функції для завантаження файлів і т. д. Фреймворки створені, щоб полегшити процес розробки і дозволитяють не винаходити колесо.

Django допомагає писати програмне забезпечення, допомагає уникнути багатьох поширених помилок безпеки, надає безпечний спосіб керування обліковими записами користувачів і паролями, уникаючи поширених помилок, таких як розміщення інформації про сеанс в файли cookie.

Django є інструментом, що допомагає писати програмне забезпечення, яке буде:

1) Повний – Django пропагує і слідує філософії «Все включено» і надає майже все, що найчастіше потрібно розробникам уже «з коробки». Оскільки все, що потрібно, є частиною одного цілісного рішення, все це ідеально працює разом, відповідає найкращим принципам проектування і має велику підтримку і щонайважливіше актуальну документацію.
2) Універсальним - Django може бути використаний для створення майже будь-якого типу веб-сайтів - від систем моніторингу і блогу, до соціальних мереж і систем агрегації новин. Він працює з будь-яким клієнтським середовищем і може обмінюватись інформацією практично в будь-якому форматі (включаючи HTML, RSS, JSON, XML і т. д.).
3) Захищенний - Django допомагає уникнути багатьох поширених помилок безпеки, надаючи фреймворк, розроблений з використанням актуальних технологій для автоматичного захисту сайту. Наприклад, Django надає безпечний спосіб керування обліковими записами користувачів і паролями, уникаючи поширених помилок, таких як розміщення інформації про сеанс в файли cookie, де вона вразлива (замість цього файли cookie мають містити тільки ключ, а дані мають зберігаютися в базі даних) або безпосередньо зберігати паролі замість хеша пароля[17].
4) Django, за замовчуванням, забезпечує захист від багатьох вразливостей, включаючи SQL-ін'єкцію, міжсайтовий скриптинг, підробку міжсайтових запитів і клікджекінг (див. Website security для отримання додаткової інформації про ці атаках).

Django написаний на Python, який працює на багатьох платформах. Це означає, що ви не прив'язані до якої-небудь конкретної серверній платформи і можете запускати додатки на багатьох версіях Linux, Windows і Mac OS X.

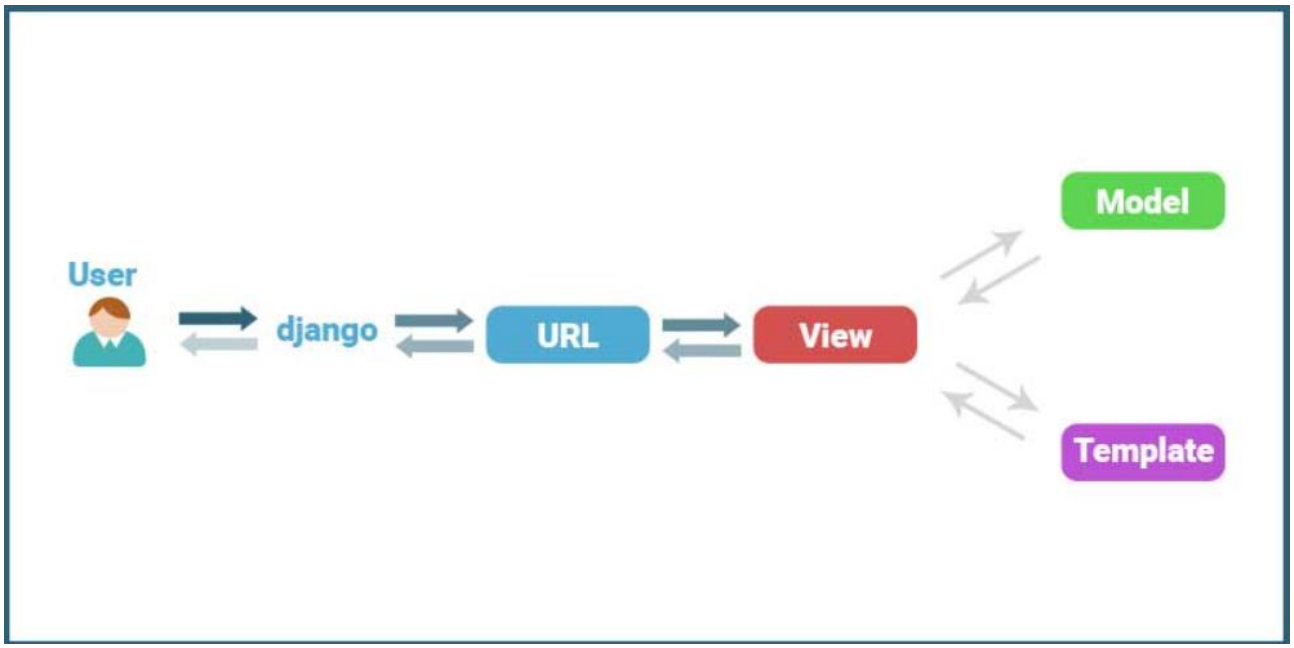



Рис. 2.4 Django MVC

Django добре підтримується багатьма веб-хостингами, які часто надають певну інфраструктуру і документацію для розміщення сайтів Django.

Логіка доступу до даних, логіка виконання і логіка надання даних - складають концепт, який зазвичай називають концепцією Model-View-Controller (MVC) архітектури додатків. У цій схемі «Модель» належить до структури даних, «Представлення» - відноситься до частини, яка відповідає за вибірку того, що і як відображати, а «Контролер» - відноситься до частини, яка визначає яке уявлення використовувати, наприклад - в залежності від введення даних користувача. Django слідує концепції MVC досить строго, що б його можна було називати «MVC фреймворк»[18].

### 2.2.2 GraphQL

GraphQL спочатку створений Facebook, але в даний час розробляється в рамках GraphQL Foundation, GraphQL - це мова запитів і середовище виконання, яка дозволяє нам отримувати і маніпулювати даними.

Веб-API - це двигуни, на яких засновано більшість наших додатків. Протягом багатьох років REST був домінуючою архітектурою для API, але з часом у нього стало проявлятися безліч недоліків.

На заміну REST було розроблено GraphQL. За допомогою REST API ви зазвичай створюєте кілька URL для кожного доступного об'єкта даних. GraphQL це синтаксис, який описує як запитувати дані і, в переважній частоті, використовується для завантаження даних з сервера.

Можливо дати 3 основні характеристики:

- Дозволяє клієнту точно вказати, які дані йому потрібні.
- Полегшує збір даних з різних джерел.
- Присутня систему типів для взаємодії з даними.

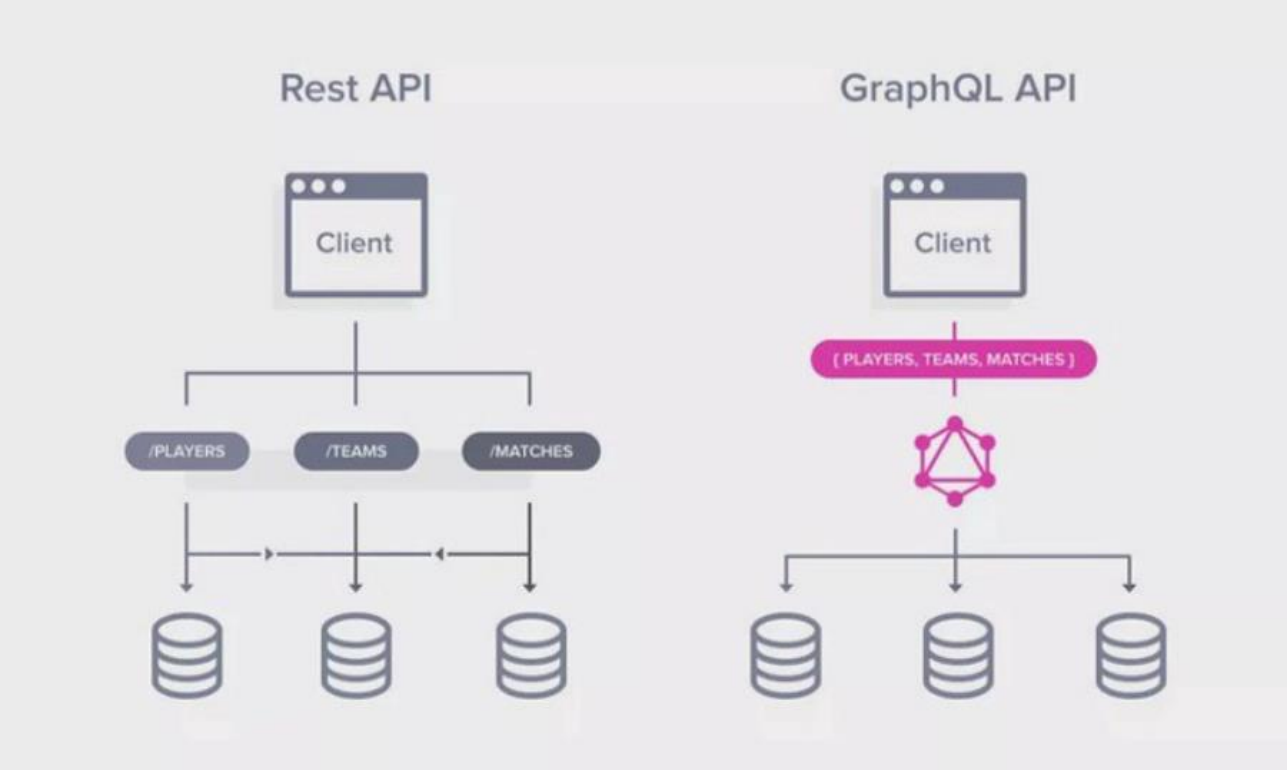



Рис. 2.5 Відмінності в структурі REST API і GraphQL API

GraphQL - це технологія на стороні сервера з відкритим кодом, розроблена Facebook оптимізувати виклики RESTful API. Це механізм виконання та мова запитів даних. У цьому розділі ми обговоримо переваги використання GraphQL.

Нижче наведено переваги які будуть використанні при розробці мовою запитів GraphQL:

- Отримуємо те, що хочемо - і змінюємо

- Надішлимо запит GraphQL своєму API і отримаємо саме те, що нам потрібно. Завдання GraphQL завжди повернути передбачувані результати. Додатки, що використовують GraphQL, швидкі та стабільні. На відміну від RESTfull служби, ці програми можуть обмежувати дані, які слід отримувати з сервера.

Наступні приклади допоможуть нам краще зрозуміти це.

API RESTful дотримуються чіткого та добре структурованого підходу, орієнтованого на ресурси. Однак коли дані ускладнюються, маршрути стають довшими. Іноді це неможливо отримати дані з одним запитом. Тут GraphQL стане в нагоді. Дані структури GraphQL у вигляді графіка з потужним синтаксисом запитів для обходу, отримання та зміни даних.

Розглянемо бізнес-об’єкт Student з атрибутами id, firstName, lastName та collegeName. Мобільний додаток повинен отримувати лише firstName, lastName. Якщо ми розробити кінцеву точку REST, як / api / v1 / students, в кінцевому підсумку буде отримано дані для всіх полів для об’єкта студента. Це означає, що дані надмірно отримуються службою RESTful. Ця проблема можна вирішити за допомогою GraphQL.

Розглянемо наведений нижче запит GraphQL:

```
{
 students {
        id
        firstName
         }
}
```

Ми можемо отримайте багато ресурсів за один запит.

Запити GraphQL допомагають плавно отримувати пов’язані бізнес-об’єкти, тоді як типовий REST API вимагають завантаження з декількох URL-адрес. API API GraphQL отримують усі дані вашого додатка необхідність в одному запиті.

Цей запит поверне значення лише для полів id та firstname. Запит не отримує значень для інших атрибутів об’єкта студента. Відповідь буде проілюстрований нище:

```
{
    "data": {
    "students": [
        {
            "id": "S1001",
            "firstName": "Mohtashim"
        },
    {
        "id": "S1002",
        "firstName": "Kannan"
    }
    ]
    }
}
```

Програми, що використовують GraphQL, можуть бути швидкими навіть на повільних мобільних пристроях підключення до мережі.

Розглянемо ще один бізнес-об'єкт, коледж, який має атрибути: ім'я та Розташування. Студент як бізнес-об’єкт пов’язаний із кафедрою об'єкт. Якби ми використовували REST API для того, щоб отримати деталі студентів та їхніх коледжу, ми в підсумку зробимо два запити до сервера як / api / v1 / студентів та / api / v1 / коледжів. Це призведе до недостатнього отримання даних за допомогою кожен запит. Тож мобільні програми змушені робити кілька дзвінків на сервер, щоб отримати потрібні дані. Запити GraphQL допомагають плавно отримувати пов’язані бізнес-об’єкти, тоді як типовий REST API вимагають завантаження з декількох URL-адрес.

Наша система може отримувати деталі як для студентів, так і для об'єктів коледжу в один запит за допомогою GraphQL.

Далі подано запит GraphQL для отримання даних:

```
{
        students{
                id
                firstName
                lastName
                college{
                name
                location
        }
 }
```

Вихідні дані вищезазначеного запиту містять саме ті поля, які ми запитували як показано нижче:

```
{
  "data": {
  "students": [
        {
         "id": "S1001",
         "firstName": "Mohtashim",
         "lastName": "Mohammad",
         "college": {
         "name": "CUSAT",
         "location": "Kerala"
        }
 ]
 }
 }
```

Можливо на перший погляд здається, що цим система GraphQL сильно навантажений, але запити базуються на полях та пов’язаних з цими даних типами, тому все працює швидко.

Якщо в запиті GraphQL є невідповідність типів, серверні програми повертають очищення та корисні повідомлення про помилки. Це допомагає плавно налагоджувати та легко виявляти помилки клієнтські програми. GraphQL також пропонує бібліотеки на стороні клієнта, які можуть допомогти зменшити явне перетворення та аналіз даних.

Тип даних студентів та університету наведено нижче:

```
type Query {
        students:[studentId]
        }
 type Student {
        id:ID!
        firstName:String
        lastName:String
        fullName:String
        university:College
 } type University {
        id:ID!
        name:String
        city:String
        mark:Float
        students:[studentId]
}
```

Ми можемо рухатись швидше за допомогою потужних інструментів розробника. GraphQL надає багаті інструменти нам як розробникам для документації та тестування запитів. GraphQL це чудовий інструмент, який генерує документацію щодо запиту та його схеми. Він також дає редактор запитів для тестування API GraphQL та можливості інтелектуального заповнення коду побудованих запитів.

## ВИСНОВКИ ДО РОЗДІЛУ 2

Після проведення аналізу виходячи з отриманих даних, підсумуємо що система складатиметься з 2-х основних частин:

• Клієнт

• Сервер

Сервер - це частина за якою користувач безпосередньо не працює, але вона вагомою частиною системи і вся аналітична функціональність працюватиме саме тут.

Клієнт - це система з якою в першу чергу користувач контактує, і відправляє запити на сервер.

Для розробки сервера були обрані:

1. Мова програмування Python, перевага якої є кросплатформенність і легкість роботи.
2. Для розробки на даній мові використовуватиметься фреймворк Django, так як він дає широкий вибір готових рішень і в ньому присутня безпекова функціональність, що полегшує розробку надійної системи
3. Для зберігання даних обрана база даних PostgreSQL, проте Django дозволяє абстрагуватись від взаємодії з базою даних, тому обрана БД є умовністю
4. Для взаємодії між клієнт сервером буде використана технологія GraphQL, замість стандартної REST. Це збільшить швидкодію і навантаження на сервер оскільки дозволяє сегрегувати і отримати лише потрібні дані.
5. Для клієнтської частини були обрані технології які забезпечуюють швидку швидкість розробки:

1. Бібліотека React
2. Мова JavaScript

# РОЗДІЛ 3

# РЕАЛІЗАЦІЯ

## 3.1 Призначення системи

В даній роботі буде розроблено кроссбраузерна система для моніторингу успішності студентів. Дана платформа повинна працювати на десктопі та мобільних пристроях. У системі повинен надаватись інтерфейс користувача для викладача, адміна та студента.

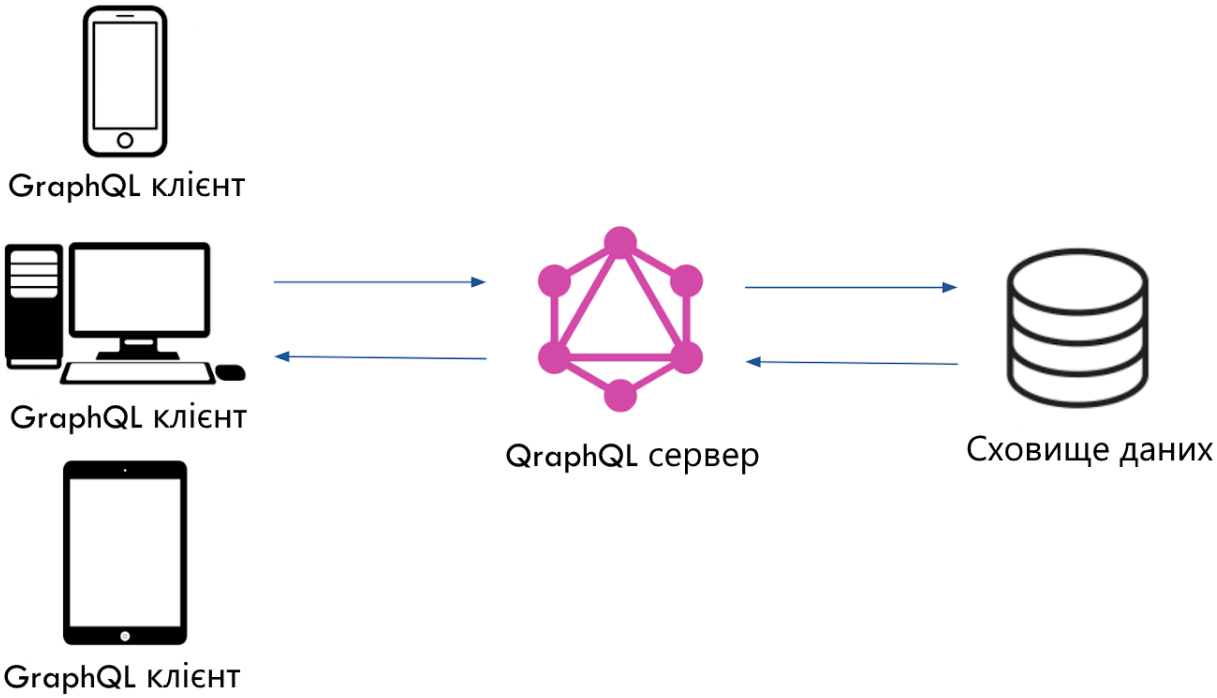
GraphQL запити та мутації дають доступ для читання та редагування до основних типів даних на сервері.

Представлені вимоги які присутні у функціоналі клієнтів можуть бути поділенні на наступні пункти:

1) перегляд списку студентів;
2) форма залишення відгуку;
3) прив’язка предмета до студента
4) завантаження результатів оцінювання
5) сортування студентів за певними критеріями, курс, предмет, успішність
6) форма створення предмету
7) форма логіну

Користувачам буде надана можливість авторизації. Авторизовані студенти можуть залишати коментарі, завантажувати та переглядати свою успішність. Викладачі в свою чергу можуть створювати групи, створювати предмети, по яким будуть опрацюватись та систематизуватись дані, залишати коментарі студентам.

Система буде складатись з 2 частин та зв’язків між ними, детальна структура та зв’язки зображені на рисунку 3.1.




3.1. Взаємодія елементів верхнього рівня системи

## 3.2 Розробка клієнтської частини

Враховуючи, що в розділі 2 було вирішено, що найдоцільніший метод це використання React для сторення інтерфейсу, застосуємо основні підходи які притаманні цій бібліотеці.

Для реалізації системи нам необхідно зробити інтерфейс користувача (браузерне вікно і з'єднати компоненти з сервером для роботи).

Оскільки ми використуваємо React то елементи ми визначемо як компоненти. Це нам допоможе в майбутньому використувавати код єдиноразово не повторюючи його. Для створення компонентів використаємо JSX.

Використовуючи його ми зможемо створити базові HTML сторінки для:

- Логіну
- Списку студентів
- Сторінку моніторингу конкретного студента
- Списку предметів

Структура представлення має бути відділена від логіки отримання та обробки даних, так звані, компонети(представлення) та контейнерів(логіка).

Є простий шаблон, який є дуже корисним при написанні додатків на React..

Буде набагато простіше повторно використовувати компоненти, якщо ми їх розділимо на дві категорії. Найчастіше їх називають Контейнер і Презентаційний компоненти. Вони не зовсім ідентичні, але в основі лежить одна ідея.

Презентаційні компоненти:

- Цікавить як частини виглядають.
- Можуть містити як презентаційний, так і контейнери-компоненти всередині, і зазвичай містять деяку власну розмітку DOM і стилі.
- Часто дає можливість звернутися через this.props.children.
- Чи не залежить від іншої частини програми, таких як дії Flux або Store.
- Не використовується для завантаження або зміни даних.
- Отримує дані і функції зворотного виклику тільки через props.
- Рідко зберігає свій стан (зазвичай вони відносяться до інтерфейсу, ніж до інших даних).
- Написані як функціональні компоненти до тих пір поки не потребують стані, використанні в життєвому циклі або оптимізації продуктивності.

Наприклад, Page, Sidebar, Story, UserInfo, List.

Компоненти-контейнери:

- Цікавить як частини працюють.
- Можуть містити як презентаційний, так і контейнери-компоненти ** всередині, але зазвичай не мають розмітки DOM, за виключення деяких блоків div як обгортки, і не мають ніяких стилів.

- Містять дані або поведінку презентаційних або інших компонентів-контейнерів.
- Викликають дії Flux і передають їх в якості функцій зворотного виклику в презентаційні компоненти.
- Часто мають стан і служать джерелом даних.
- Найчастіше створюються з використанням компонентів високого порядку, таких як connect () в React Redux, createContainer () в Relay або Container.create () в Flux Utils, ніж пишуться вручну.

Наприклад, UserPage, FollowersSidebar, StoryContainer, FollowedUserList.

Використовуються різні папки, щоб зробити це відмінність наочним.

Переваги такого підходу це найкраще поділ у вирішенні проблем. Зрозуміти додаток краще при написанні компонентів слідуючи підходу.

Краще повторне використання. Можливо використовувати один і той же компонент з абсолютно різними джерелами даних і перетворити їх в окремий компонент-контейнер, який можна використовувати повторно.

Презентаційний компоненти є "палітрою" програми. Можливо вставити їх на окрему сторінку і є можливість налаштувати дизайн все їх варіації минаючи логіку.

Це змушує витягти "компоненти макета" як Sidebar, Page, ContextMenu і використовувати this.props.children замість дублювання однаковою розмітки і верств в декількох компонентах-контейнерах.

Компоненти не повинні викидати DOM. Вони необхідні тільки для забезпечення структури кордонів в проблемі інтерфейсу.

Почнемо побудову застосування з простих презентаційних компонентів. У процесі зрозуміло що передаємо велику кількість параметрів props через проміжні компоненти. Коли помітитимо, що деякі не використовують props, а лише перенаправляють їх вниз і необхідно перемонтувати всі ці проміжні компоненти в будь-який час, коли дочірнім компонентів необхідна нова порція даних, то це вдалий час ввести деякий компонент-контейнер.

Компонент дозволить отримати дані і поведінку props до компонентів сторінки не зачіпаючи компоненти в середині дерева.

Рефакторинг здійснюється постійно, тому не намагаємось написати все відмінно з першого разу. Експериментуючи з цим паттерном, можливо розвинути інтуїтивне розуміння, коли необхідно ввести контейнер, також, як коли потрібно витягти функцію.

Важливо розуміти, що поділ між компонентами не технічна, а радше пов'язане з їх цілями.

І навпаки, є кілька пов'язаних (але різних!) Технічних відмінностей:

І хоча в даному випадку все буде працювати так, як очікується, якщо є, команда поверне весь цей код рекомендується укласти подібні конструкції в круглі скобки та застосовувати ще одне погодження, прийняте в React при форматуванні програмного забезпечення. Воно замикається в тому, що окремі елементи розміщують на окремих строках і відповідним чином виражають. В результаті застосування вищеописаних ідей код нашого функціонального компонента буде виглядає так:

```
import React
function MyMonitoringApp() {
return (
<ul>
        <li>1 - One</li>
        <li>2 - Two</li>
        <li>3 - Three</li>
</ul>
)}
```

Класи і Функції. Починаючи з React 0.14, компоненти можуть бути оголошені як класи або функції. Основні компоненти наведені в таблиці 3.1 Функціональні компоненти простіше визначити, але їм не вистачає деяких функцій доступних класів. Деякі з них можуть забрати в майбутньому, але зараз вони є.

Оскільки функціональні компоненти легше зрозуміти, краще використовувати з поки ви не будете потребувати зберіганні стану, участі в життєвому циклі або оптимізації продуктивності, які доступні в даний момент для компонентів утворених з класів.

При такому підході повернена з компонента розмітка виявляється дуже схожим на звичайний HTML-код.

Зі станом і Без. Деякі компоненти використовують в React метод setState (), а деякі ні. Якщо контейнери схильні до наявності стану, а презентаційні компоненти немає, то це не є жорстким правилом. Контейнер не може мати стану, а презентаційний компонент навпаки.

Таблиця 3.1 – Основні компоненти

| Назва компонента | Представлення |
|---|---|
| PaginationFooter() | Представляє інтерфейс взаємодії з поділом сторінки на дозовані елементи інформації, має методи переходу на попередню та наступну сторінку. |
| StudentsPage() | Контейнер для отримання списку студентів та перевірку інформації на коректність перед представленням |
| StudentsView({get_n, get_page}) | Представлення сторінки зі списком студентів на конкретній сторінці в конкретному порядку |
| HomeView() | Домашня сторінка для авторизації в системі |
| GroupsPage() | Контейнер для отримання інформації про групу ти перевірку інформації на коректність перед представленням |
| StudentPage({id}) | Відображення інформації про конкретного студента |

### 3.2.1 Кроссплатформенність

SPA (Single Page Applications; Простий сторінковий додаток) - це веб сторінка, яка поєднують у собі властивості сайту і мобільного застосування.

Створити SPA нам допоможе React.

У корені проекту створюємо папку **single-spa**. У нього додамо 2 файли.

1. route-reuse-strategy.ts - файл маршрутизації наших мікросервісов.

Якщо дочірнє додаток виконує маршрутизацію всередині себе, це додаток інтерпретує це як зміна маршруту.

За замовчуванням це призведе до знищення поточного компонента і заміні його новим екземпляром того ж компонента spa-host.

Ця стратегія повторного використання маршруту дивиться на routeData.app, щоб визначити, чи повинен новий маршрут бути обробляється як той же маршрут, що і попередній, гарантуючи, що ми не перемонтіруем дочірнє додаток, коли вказане дочірнє додаток маршрути всередині себе.

```
// file route-reuse-strategy.ts

import { RouteReuseStrategy, ActivatedRouteSnapshot,
DetachedRouteHandle }
import { Injectable }
export class App extend RouteReuseStrategy {
    shouldDetachs(): boolean {
        return false;
    }
    store(): void { }
    shouldsAttach(): boolean {
        return false;
    }
    retrieve(): DetachednewComponentsRouteHandle {
        return null;
    }
    shouldReuseRoute(future: ActivatedRoute, curr:
ActivatedRouteSnapshoting): boolean {
        return future.routeConfig ===
current.routeConfig || (future.data.app &&
(future.data.app === current.data.myapp));
    }
    }
```

2. Сервіс single-spa.service.ts

У сервісі буде зберігатися метод монтування (mount) і демонтажу (unmount) фронтенд додатку див. Таблиця 3.2.

Таблиця 3.2 – Методи SPA

| Назва методу | Функція |
|---|---|
| bootstrap (mounter, bus) | Викликається після завантаження сервісу, скаже в який елемент дому потрібно монтуватися, дасть йому шину повідомлень на яку мікросервіс у себе підпишеться і зможе слухати і посилати запити та команди. |
| mount () | монтувати додаток в дом |
| unmount () | демонтаж додатку |
| unload () | вивантаження додатку |

## 3.3 Серверна частина

На основі аналізу який був здійснений в попередніх розділах, був створений прототип системи, розроблений з-за допомогою мови програмування Python і бази даних для зберігання даних та інформації, створеної на SQLite3.

Для даної системи було використано архітектурний патерн проєктування MVC. Він описує простий спосіб побудови архітектури додатка, метою якого є розмежування контролера та представлння. В результаті, система стає зрозуміла то прогнозована, легше масштабується, тестування здійснються легше, підримується легше.

Під час розробки ми використовували Django і веб-сервер розробки Django для обслуговування наших статичних файлів (CSS, JavaScript і т. Д.). У

виробничому середовищі замість цього ми зазвичай обслуговуємо статичні файли з мережі доставки контенту (CDN) або веб-сервера.

Основні методи які застосовуються на сервері та їх функціонал наведені в таблиці 3.3.

Таблиця 3.3 – Методи сервера

| Назва методу | Функція |
|---|---|
| resolve_student(id) | Повертає інформацію про судента за його id |
| resolve_students(page) | Повертає список студентів які відфільтровані по фільтру на сторінці page |
| resolve_teachers(page) | Повертає список викладачів які відфільтровані по фільтру на сторінці page |
| main() | Запуск сервера |
| _genarate_people() | Скрипт який генерує випадковий набір людей для тестування роботу системи. |

## ВИСНОВКИ ДО РОЗДІЛУ 3

У даному розділі було виконана розробка програмного забезпечення, створені основні компоненти системи моніторингу успішності студентів, розроблені методи та засоби взаємодії компонентів, а також показані деталі реалізації системи.

# РОЗДІЛ 4

# ОПИС ІНТЕРФЕЙСУ СИСТЕМИ

## 4.1 Запуск програмного забезпечення

Для запуску серверної частини платформи, необхідно Python, який може бути встановлений з офіційного сайту для необхідної ОС.

Для запуску браузерної частини платформи, необхідно Node.js версії не нижче 14+, який може бути встановлений з офіційного сайту для необхідної ОС.

## 4.2 Огляд інтерфейсу

Перейдемо до безпосереднього до інтерфейсу користувача (UI), і опишемо основні елементи та їх функціонал.

Першочергова задача було створити кроссплатформенну систему.

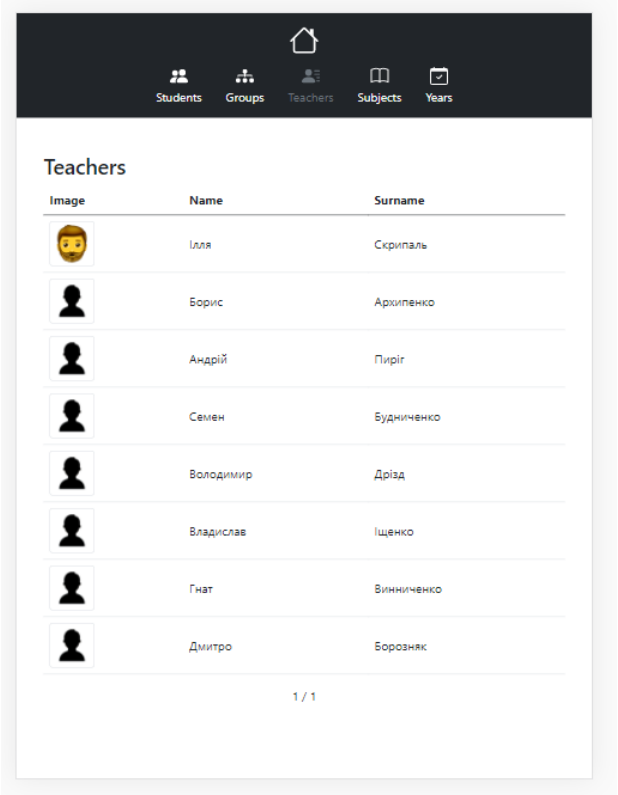



Рисунок 4.1 Представлення інтерфейсу iPad

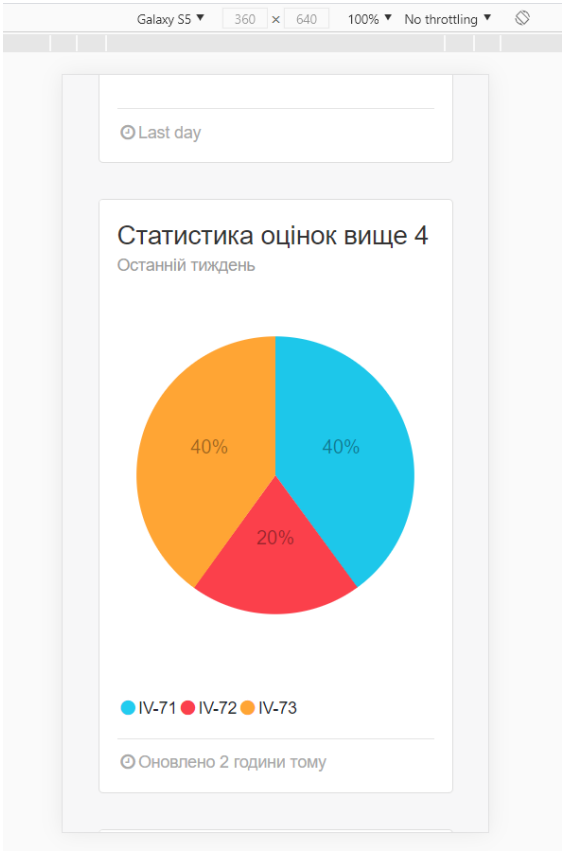



Рисунок 4.2 Представлення інтерфейсу Android

При початковому доступі до платформи відображається сторінка входу користувача яка зображена на рис. 4.3. Користувач має ввести дані для входу і отримання доступу до системи.

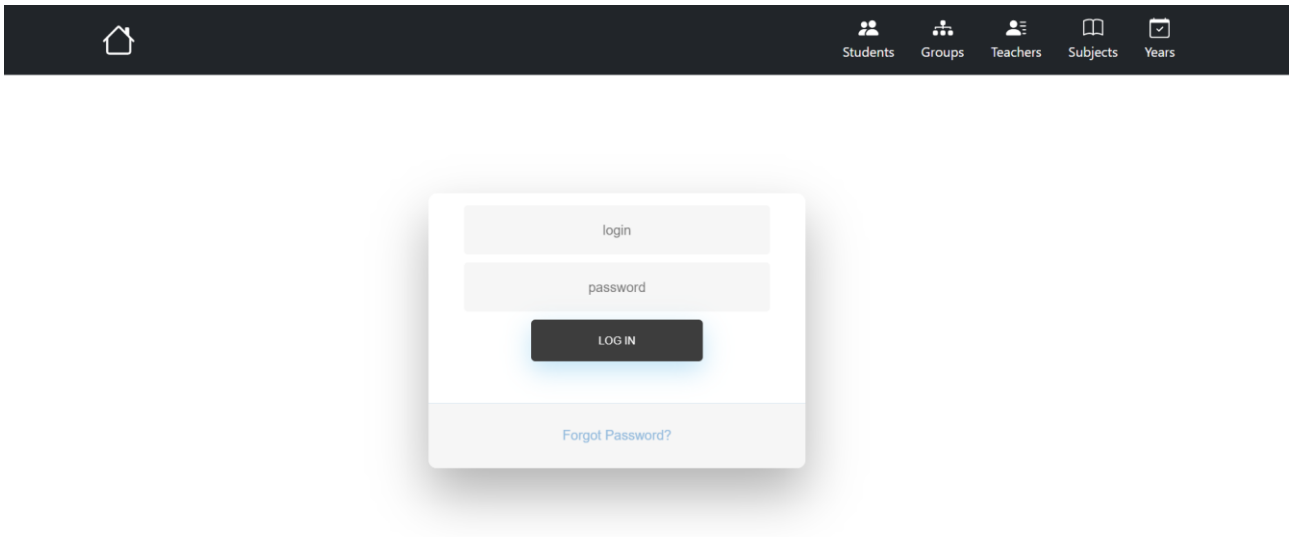



Рисунок 4.3 Сторінка входу

При успішному вході стає доступна навігація рис. 4.4

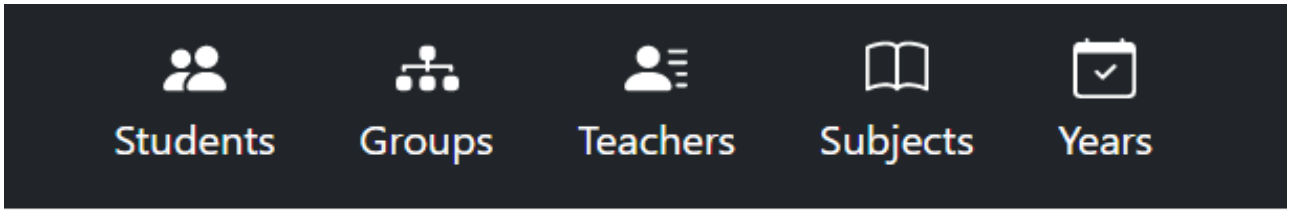



Рисунок 4.4 Навігація

На рис. 4.5, наведена сторінка із списком студентів, їхня група та середня оцінка.

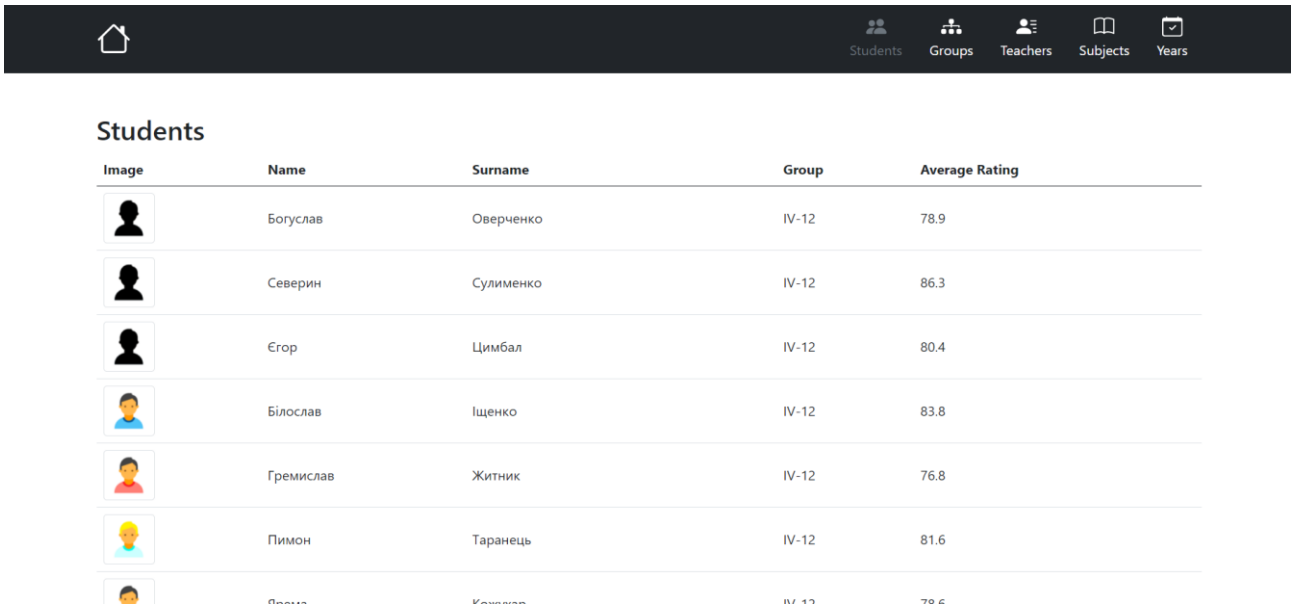



Рисунок 4.5 Сторінка зі студентами

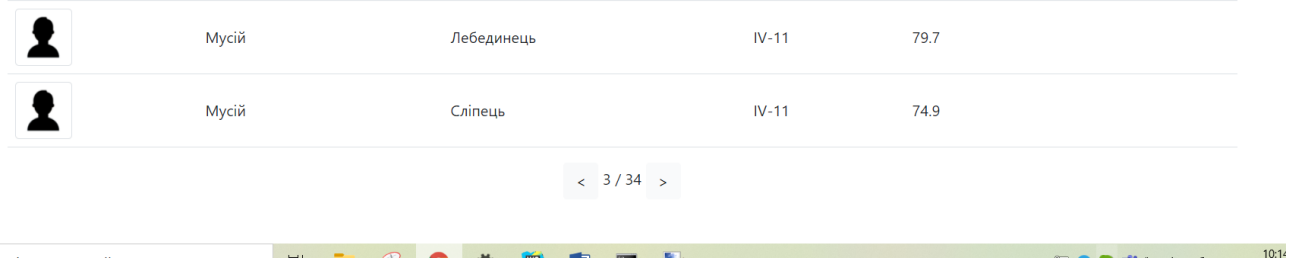



Рисунок 4.6 Навігація між сторінками

На рис. 4.7, наведена сторінка із списком викладачів. На цій сторінці також присутня навігація по сторінкам.

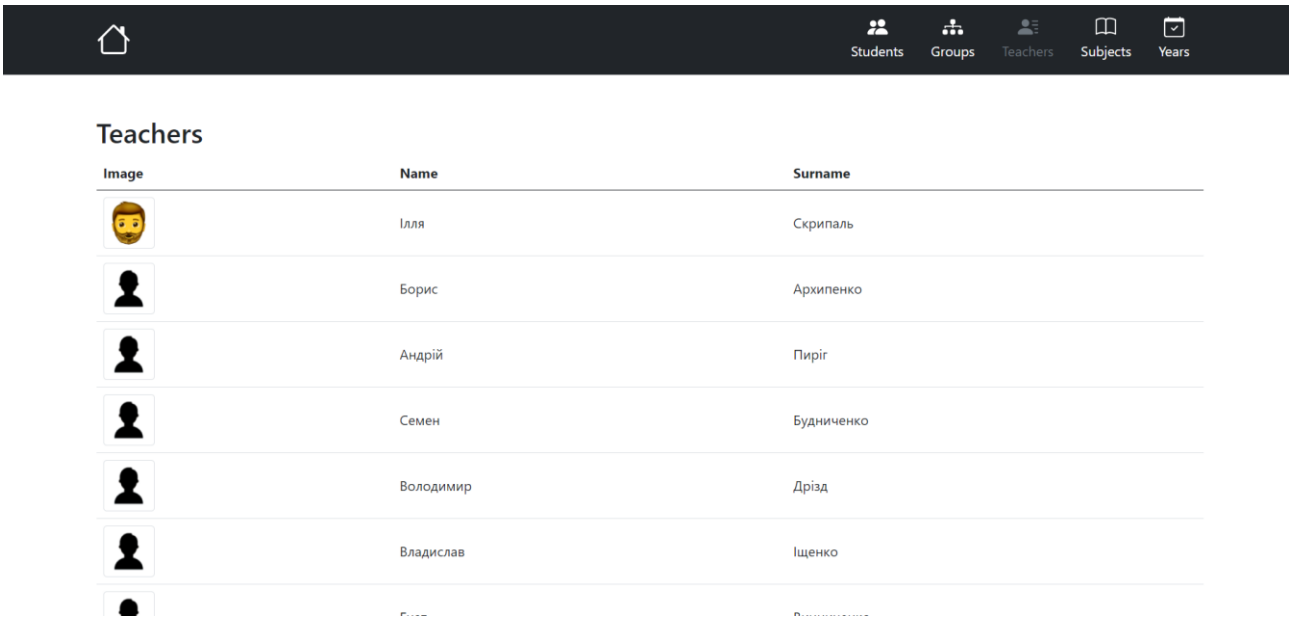



Рисунок 4.7 Сторінка із викладачами

При переході на сторінку з предметами рис. 4.8, буде показана сітка з доступними для аналізу дисципланами.

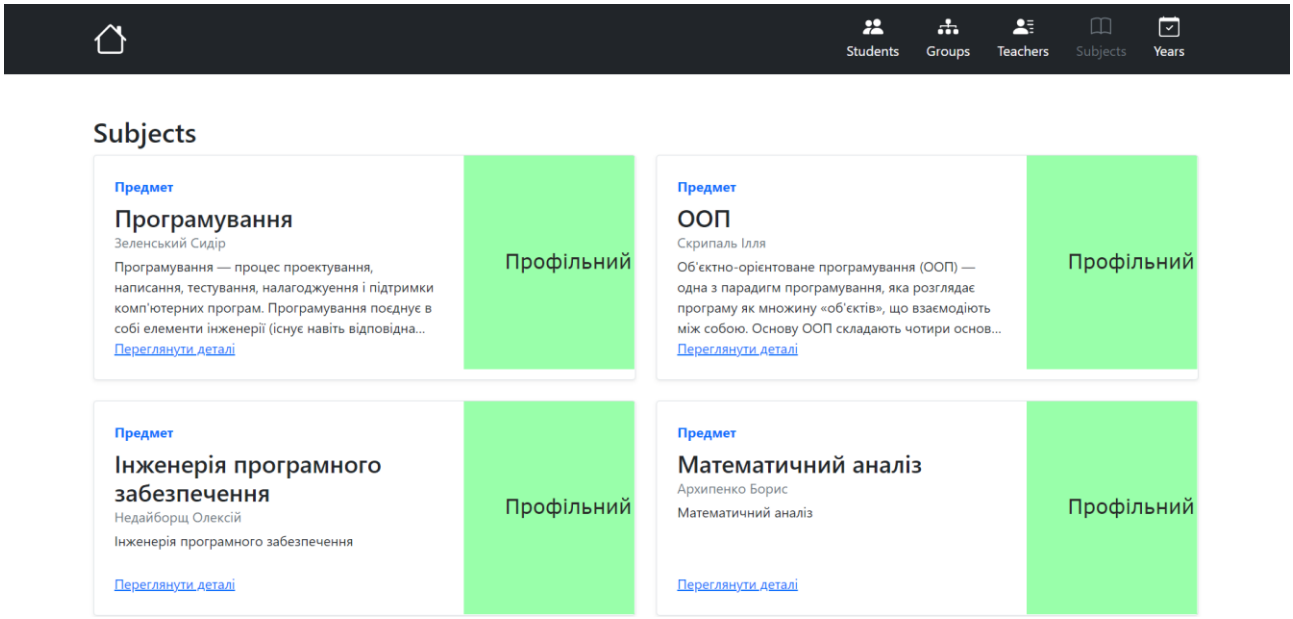



Рисунок 4.8 Сторінка з предметами

При перегляді деталей для конкретної дисципліни, буде показана статистика рис. 4.9.

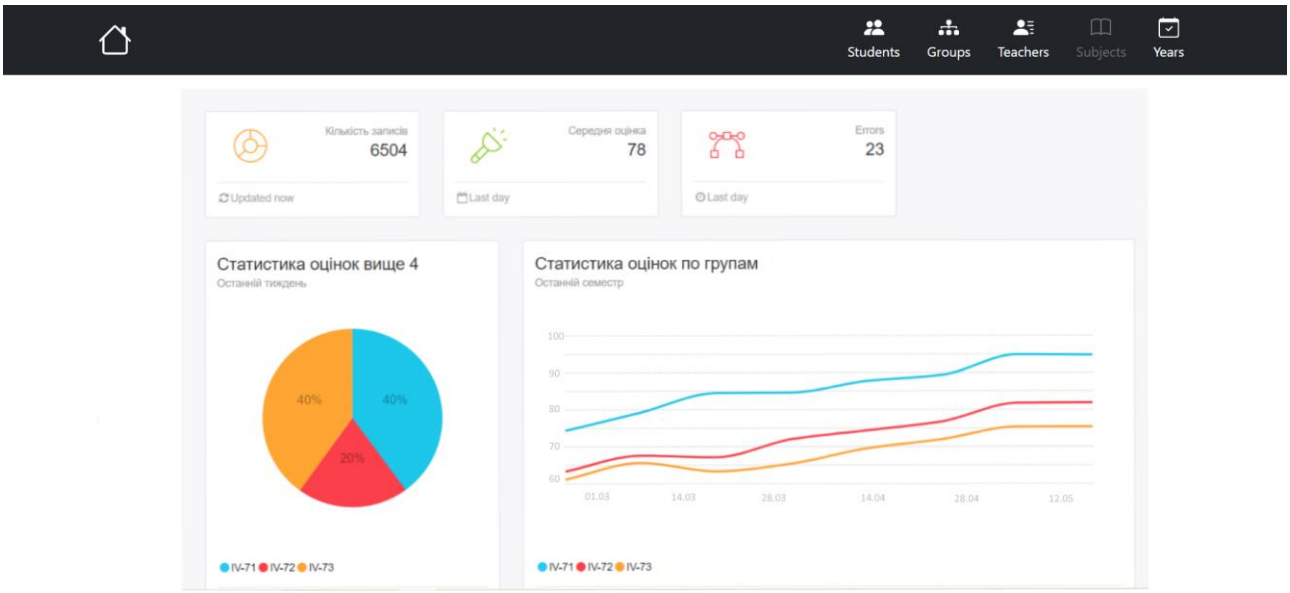



Рисунок 4.8 Статистика

Ця ж сторінка у мобільному поданні рис. 4.9.

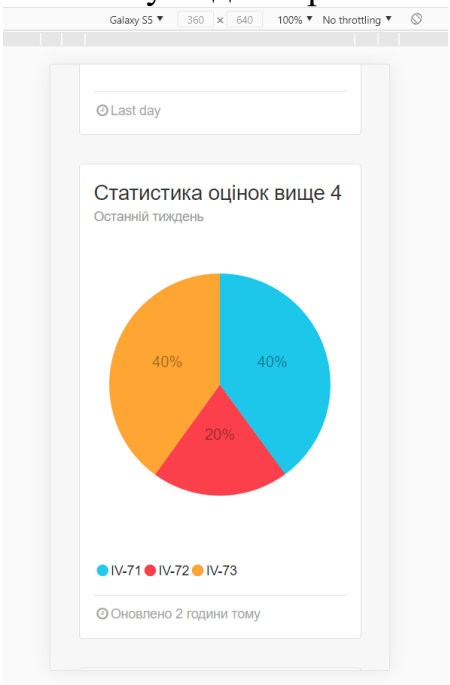



Рисунок 4.9 Статистика на мобільному

## ВИСНОВКИ ДО РОЗДІЛУ 4

У даному розділі наведено основні вимоги до розгортання системи, а також був проведений огляд інтерфейсу та приклади взаємодії користувача з платформою.

# ВИСНОВКИ

Метою даного дипломного проєкту було створення платформа для онлайн моніторингу успішності студента, який би був кроссплатформенним, допомагав в комунікації між викладачем та студентом, вирішував проблему відкритості даних.

Під час виконання дипломної роботи була детально вивчена література і технонології на тему створення веб-систем, серверної і клієнтської його частин.

Були розглянуті сучасні фреймворки, які використовуються в створенні високонавантажених додатків з високим ступенем швидкодії. Беручи до уваги вивчений матеріал, була спроектовані і розроблені серверна і клієнтська частина системи, реалізований сучасний графічний інтерфейс.

Проведений аналіз існуючих програмних рішень, аналізовані переваги наведених підходів, показав доцільність створення даного проєкту.

Для розробки були застосовані сучасні технології, які в рази пришвидшили швидкість розробки і забезпечили високу надійність і захищенність системи. Для серверної частини була застосована мова програмування Python та бібліотека Django та застосована технологія GraphQL для зменшення кількості даних для пересилання та оперування ними на сервері та клієнті.

Клієнтська частина була написана на JavaScript, бібліотеці React з присутньою технологією SPA.

Зазначені вище технології допомогли реалізувати дану платформу та забезпечили стабільну її роботу .

Розроблена система: працює на всіх платформах, підтримує локалізацію забезпечує моніторинг успішності студентів, забезпечує комунікацію між викладачем та студентом.

В підсумку, проект реалізовано у повному обсязі, відповідає всім вимогам наведеним вище та протестоване.

## Використані джерела


1.Аверченков В.И. Системы организационного управления / В.И. Аверченков, В.В. Ерохин. – Брянск: БГТУ, 2012. – 208 с.

2.Белкин В.Н. Методика преподавания: оценка профессиональных компетенций у студентов. – М.: Юрайт, 2019. – 212 с.

3. Kay A. User interface: A personal view //The art of human-computer interface design. – 1990. – С. 191-207.

4. Ерохин В.В. Применение мобильного обучения в учебном процессе вузов // Тенденции развития науки и образования. – 2018. – №43, Ч.6. – С. 76-79.

5. Ерохин В.В. Проектирование систем электронного документооборота вуза //Тенденции развития науки и образования. – 2018. – №42, Ч.5. – С. 8-11.

6. Жетесова Г.С., Ерназарова М.А. Автоматизация контроля знаний студентов примодульной форме обучения на основе программного обеспечения // Фундаментальные исследования. – 2014. – № 6-2. – С. 355-359.

7. Киселев Г.М. Информационные технологии в педагогическом образовании / Г.М. Киселев, Р.В. Бочкова. – М.: Дашков и К, 2013. – 308 с.

8. Притчина Л.С. Цифровизация и новое экономическое образование // Педагогическое образование и наука. – 2018. – № 2. – С. 120-122.

9. Рябова Н.В. Компьютерные технологии в мониторинге профессиональной деятельности становления студентов // Стандарты и мониторинг в образовании. –2007. – № 1. – С. 23–27.

10. Чуйко О. И., Ешенко Р. А. Электронный журнал: анализ применения в школах иперспективы внедрения в вузах // Международный академический вестник. – 2014.– № 6. – С. 27-31.

11. Campus, Infinite. *Company history: About Infinite Campus.* 2012.

12. SCHOOLS, Charlotte Mecklenburg. PowerSchool. *What is PowerSchool*, 2012.

13. Reinders H. Big brother is helping you: Supporting self-access language learning with a student monitoring system //System. – 2007. – Т. 35. – №. 1. – С. 93-111.

14. Mazza R., Milani C. Gismo: a graphical interactive student monitoring tool for course management systems //International Conference on Technology Enhanced Learning, Milan. – 2004. – С. 1-8.

15. Mazza R. Using information visualisation to facilitate instructors in web-based distance learning : дис. – Università della Svizzera italiana, 2004.

16. Django введение - Изучение веб-разработки | MDN [Електронний ресурс] // MDN. – 2021. – Режим доступу до ресурсу: https://developer.mozilla.org/ru/docs/Learn/Server-side/Django/Introduction.

17. LIN, C.-W.; TSAI, C.-S.; HWANG, M.-S. A new strong-password authentication scheme using one-way hash functions. *Journal of Computer and Systems Sciences International*, – 2006, – С. 623-626.

18. Annenkov D. V., Cherkashin E. A. Generation technique for Django MVC web framework using the stratego transformation language //2013 36th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). – IEEE, 2013. – С. 1084-1087.

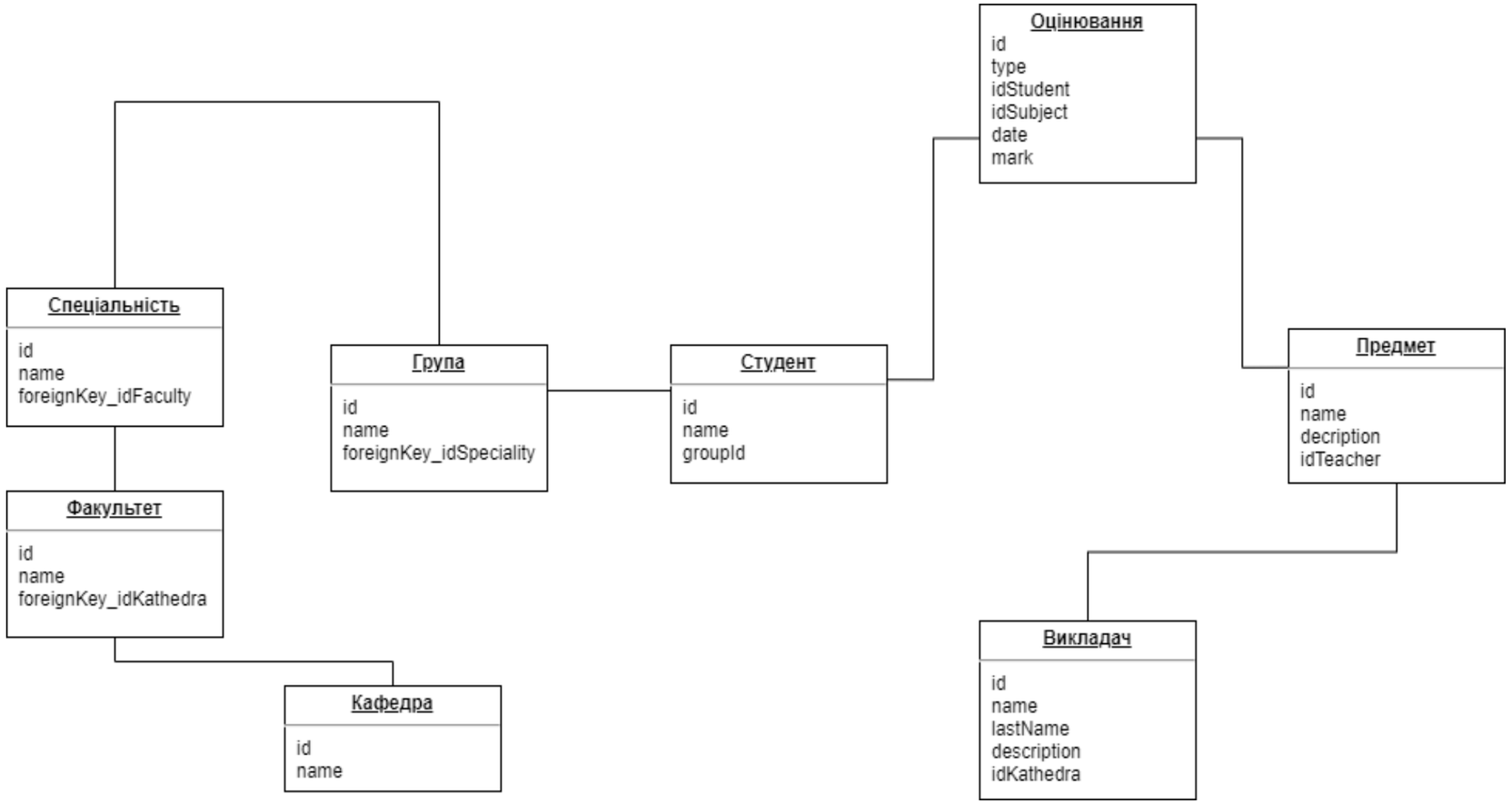
19. Gackenheimer C. Introduction to React. – Apress, 2015.

ДОДАТОК 1
Система моніторингу успішності студентів

# Сховище даних.
# Схема даних
ІАЛЦ.467200.004 Д1

Аркушів 1

Київ 2021 р.



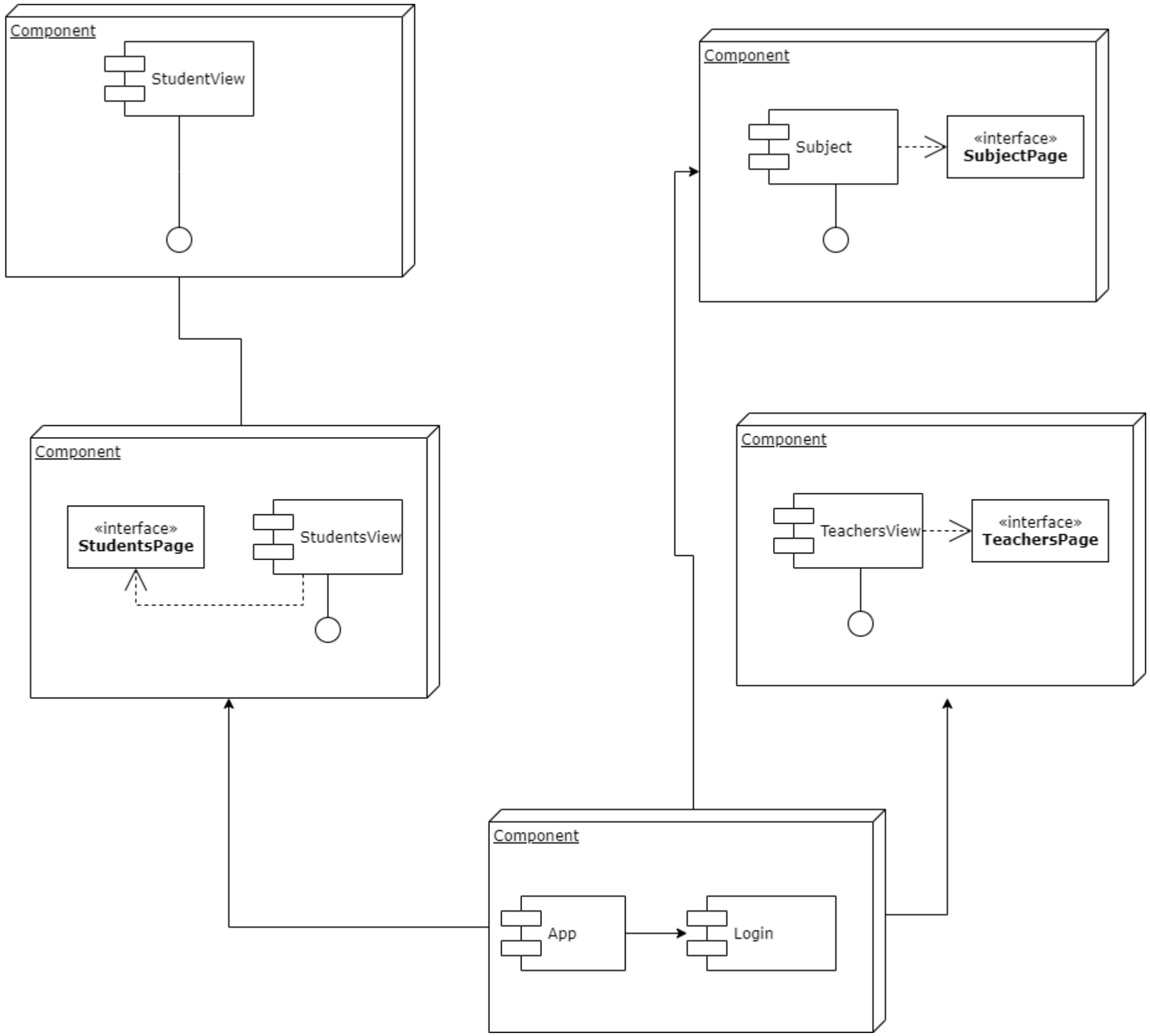

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | ІАЛЦ.467200.004 Д1 | | |
| Змн | Арк. | № документа | Підпис | Дата | | | |
| Розробив | | Проценко А.А. | | | Сховище даних. Схема даних. | Літ. | Арк. 1 / Аркушів 1 |
| Перевірив | | Антонюк А.І. | | | | | |
| Норм. контр. | | Сімоненко В.П. | | | | НТУУ "КПІ", ФІОТ, ІВ-71 | |
| Затвердив | | | | | | | |

ДОДАТОК 2
Система моніторингу успішності студентів

**Схема структурна**
**Діаграма компонент**
ІАЛЦ.467200.005 Д2

Аркушів 1

Київ 2021 р.



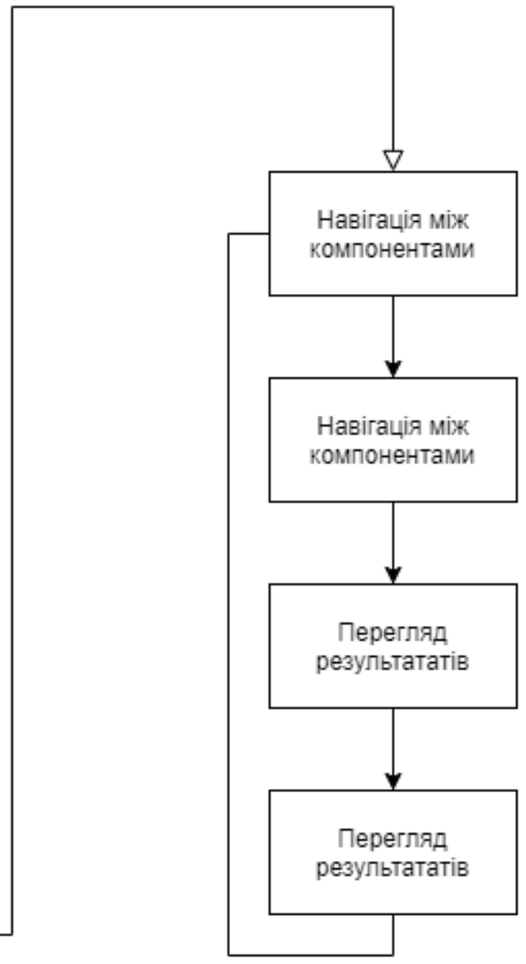

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | ІАЛЦ.467200.005 Д2 | |
| Змн | Арк. | № документа | Підпис | Дата | | |
| Розробив | | Проценко А.А. | | | Схема структурна<br>Діаграма компонент | Літ. / Арк. 1 / Аркушів 1 |
| Перевірив | | Антонюк А.І. | | | | НТУУ "КПІ", ФІОТ, ІВ-71 |
| | | | | | | |
| Норм. контр. | | Сімоненко В.П. | | | | |
| Затвердив | | | | | | |

ДОДАТОК 3
Система моніторингу успішності студентів

**Схема функціональна**
**Блок-схема алгоритму**
ІАЛЦ.467200.006 Д3

Аркушів 1

Київ 2021 р.


| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | ІАЛЦ.467200.006 ДЗ | |
| Змн | Арк. | № документа | Підпис | Дата | | |
| Розробив | | Проценко А.А. | | | Схема функціональна Блок-схема алгоритму | Літ. / Арк. 1 / Аркушів 1 |
| Перевірив | | Антонюк А.І. | | | | |
| Норм. контр. | | Сімоненко В.П. | | | | НТУУ "КПІ", ФІОТ, ІВ-71 |
| Затвердив | | | | | | |

ДОДАТОК 4
Система моніторингу успішності студентів

**Текст програми**
ІАЛЦ.467200.007 Д4

Аркушів 22

Київ 2021 р.

```
import React from 'react';
import {StudentPage} from "./Students/StudentPage";
import {Header} from "./Header";
import {BrowserRouter, Route, Switch} from "react-router-
dom";
import {GroupsPage} from "./Groups/GroupsPage";
import {TeachersPage} from "./Teachers/TeachersPage";
import {SubjectsPage} from "./Subjects/SubjectsPage";
import {YearsView} from "./Years/YearsView";
import {HomeView} from "./Home/HomeView";
import './App.css'
import {StudentDetail} from "./Students/StudentDetail";

export function App() {
    return(
        <div>
            <BrowserRouter>
                <Header/>
                <div className="container my-5">
                    <Switch>
                        <Route path="/students">
                            <StudentPage/>
                        </Route>
                        <Route path="/students/1">
                            <StudentDetail/>
                        </Route>
                        <Route path="/groups">
                            <GroupsPage/>
                        </Route>
                        <Route path="/teachers">
                            <TeachersPage/>
                        </Route>
                        <Route path="/subjects">
                            <SubjectsPage/>
                        </Route>
                        <Route path="/years">
                            <YearsView/>
                        </Route>
                        <Route path="/">
                            <HomeView/>
                        </Route>
                    </Switch>
                </div>
            </BrowserRouter>
        </div>
```

```
    )
}
import React from 'react';
import { gql } from '@apollo/client';
import {useQuery} from "@apollo/client";

const PEOPLE = gql`
    query {
        students {
            id
            name
            lastName
            img
        }
    }
`;

export function YearsView() {
    const { loading, data } = useQuery(PEOPLE);
    return (
        <div>
            year
        </div>
    );
}
import React, {useState} from 'react';
import {useQuery} from "@apollo/client";
import {Loading} from "../Components/Loading";



export const PaginationFooter = ({query, page, setPage}) =>  {
    const {error, loading, data} = useQuery(query);
    return (
        <div>
            {error ? <span>error</span> : loading ? <Loading/> :
                <div className="d-flex justify-content-around">
                    <span>
                        {page === 1 ? <span/> :
                            <button type="button" onClick={() =>
setPage(page - 1)} className="btn btn-light">{`<`}</button>}
                        <span>   {page} / </span>
                        <span>{data.teacherN}   </span>
                        {page === data.teacherN ? <span/> :
                            <button type="button" onClick={() =>
setPage(page + 1)} className="btn btn-light">{`>`}</button>}
                    </span>
                </div>}
        </div>

    )
}
import React from 'react';
import {TeacherView} from "./TeacherView";
import {gql} from "@apollo/client";
import {StudentView} from "../Students/StudentView";
```

```
const GET_PAGE_OF_TEACHERS = gql`
    query getPageOfTeachers($page: Int!){
        teachers(page: $page){
            name
            lastName
            img
        }
    }
`;

const GET_TEACHERS_N = gql`
    query {
        teacherN
    }
`;


export function TeachersPage() {
    return (
        <div>
            <h2>Teachers</h2>
            <TeacherView get_page={GET_PAGE_OF_TEACHERS}
get_n={GET_TEACHERS_N}/>
        </div>
    )
}
import React, {useState} from 'react';
import { gql } from '@apollo/client';
import {useQuery} from "@apollo/client";
import {Loading} from "../Components/Loading";
import {SmallImg} from "../Components/SmallImg";
import {PaginationFooter} from "./PaginationFooter";

export function TeacherView({get_n, get_page}) {
    const [page, setPage] = useState(1)
    const { loading, error, data } = useQuery(get_page,{
        variables: { page },
    });
    return (
        <div>
            <table className="table">
                <thead className="thead-dark">
                <tr>
                    <th scope="col">Image</th>
                    <th scope="col">Name</th>
                    <th scope="col">Surname</th>
                </tr>
                </thead>
                <tbody>
                {error ? <p>Error</p> : loading ? <Loading/> :
                    data.teachers.map(item => {
                        return (
                            <tr style={{verticalAlign: 'middle'}}
key={item.id}>
                                <td><SmallImg src={item.img}/></td>
                                <td>{item.name}</td>
                                <td>{item.lastName}</td>
                            </tr>
```

```
                        )
                    })
                }
                </tbody>
            </table>
            <PaginationFooter query={get_n} page={page}
setPage={setPage}/>
        </div>

    );
}
import React from 'react';
import {useQuery} from "@apollo/client";
import {Loading} from "../Components/Loading";
import {base_subject} from "../variable";


const Item = ({item}) => {
    return (
        <div className="col-md-6">
            <div
                className="row g-0 border rounded overflow-hidden flex-md-
row mb-4 shadow-sm h-md-250 position-relative">
                <div className="col p-4 d-flex flex-column position-
static">
                    <strong className="d-inline-block mb-2 text-
primary">Предмет</strong>
                    <h3 className="mb-0">{item.name}</h3>
                    <div className="mb-1 text-
muted">{item.teacher.lastName} {item.teacher.name}</div>
                    <p className="card-text mb-auto
overflow4line">{item.description}</p>
                    <a href="#" className="stretched-link">Переглянути
деталі</a>
                </div>
                <div className="col-auto d-none d-lg-block">
                    {item.name in base_subject ?
                        <svg className="bd-placeholder-img" width="200"
height="250"
                             xmlns="http://www.w3.org/2000/svg" role="img"
aria-label="Placeholder: Thumbnail"
                             preserveAspectRatio="xMidYMid slice"
focusable="false"><title>Placeholder</title>
                            <rect width="100%" height="100%"
fill="#ff7073"></rect>
                            <text fontFamily='Verdana' fontSize='24'
x="29%" y="50%" fill="#e3e8e7" dy=".3em">Базовий</text>

                        </svg>
                        :
                        <svg className="bd-placeholder-img" width="200"
height="250"
                             xmlns="http://www.w3.org/2000/svg" role="img"
aria-label="Placeholder: Thumbnail"
                             preserveAspectRatio="xMidYMid slice"
focusable="false"><title>Placeholder</title>
                            <rect width="100%" height="100%"
fill="#99ffaa"></rect>
```

```
                                <text fontFamily='Verdana' fontSize='24'
x="24%" y="50%" fill="#282828" dy=".3em">Профільний</text>

                            </svg>
                        }


                    </div>
                </div>
            </div>
        )
}

export function SubjectView({get_page}) {
    const { loading, error, data } = useQuery(get_page)
    debugger
    return (
        <div className="row mb-2">
            {error ? <p>Error</p> : loading ? <Loading/> :
                data.subjects.map(item => {
                    return <Item item={item}/>
                })
            }
        </div>
    );
}
import React from 'react';
import {SubjectView} from "./SubjectView";
import {gql} from "@apollo/client";


const GET_PAGE_OF_SUBJECTS = gql`
    query GetSubjects {
        subjects {
            name
            description
            teacher {
                name
                lastName
            }
        }
    }`;



export function SubjectsPage() {
    return (
        <div>
            <h2>Subjects</h2>
            <SubjectView get_page={GET_PAGE_OF_SUBJECTS}/>
        </div>

    )
}
import React, {useState} from 'react';
import {useQuery} from "@apollo/client";
import {Loading} from "../Components/Loading";
import {SmallImg} from "../Components/SmallImg";
import {PaginationFooter} from "./PaginationFooter";

const getAverageMark = (n) => {
```

```
    debugger
    let total = 0;
    for(let i = 0; i < n.length; i++) {
        total += n[i].mark;
    }
    return  total / n.length;
}

export function StudentView({get_n, get_page}) {
    const [page, setPage] = useState(1)
    const { loading, error, data } = useQuery(get_page,{
        variables: { page },
    });
    return (
        <div>
            <table className="table">
                <thead className="thead-dark">
                <tr>
                    <th scope="col">Image</th>
                    <th scope="col">Name</th>
                    <th scope="col">Surname</th>
                    <th scope="col">Group</th>
                    <th scope="col">Average Rating</th>
                </tr>
                </thead>
                <tbody>
                {error ? <p>Error</p> : loading ? <Loading/> :
                    data.students.map(item => {
                        return (
                            <tr style={{verticalAlign: 'middle'}}
key={item.id}>
                                <td><SmallImg src={item.img}/></td>
                                <td>{item.name}</td>
                                <td>{item.lastName}</td>
                                <td>{item.group.groupName}</td>
                                <td>{getAverageMark(item.sessionSet)}</td>
                            </tr>
                        )
                    })
                }
                </tbody>
            </table>
            <PaginationFooter query={get_n} page={page}
setPage={setPage}/>
        </div>
    );
}
export function StudentView({get_n, get_page}) {
    const [page, setPage] = useState(1)
    const { loading, error, data } = useQuery(get_page,{
        variables: { page },
    });
    return (
        <div>
            <table className="table">
                <thead className="thead-dark">
                <tr>
                    <th scope="col">Image</th>
                    <th scope="col">Name</th>
```

```
                    <th scope="col">Surname</th>
                    <th scope="col">Group</th>
                    <th scope="col">Average Rating</th>
                </tr>
                </thead>
                <tbody>
                {error ? <p>Error</p> : loading ? <Loading/> :
                    data.students.map(item => {
                        return (
                            <tr style={{verticalAlign: 'middle'}}
key={item.id}>
                                <td><SmallImg src={item.img}/></td>
                                <td>{item.name}</td>
                                <td>{item.lastName}</td>
                                <td>{item.group.groupName}</td>
                                <td>{getAverageMark(item.sessionSet)}</td>
                            </tr>
                        )
                    })
                }
                </tbody>
            </table>
            <PaginationFooter query={get_n} page={page}
setPage={setPage}/>
        </div>
    );
}
import React from 'react';
import {StudentView} from "./StudentView";
import {gql} from "@apollo/client";


import React from 'react';
import {StudentView} from "./StudentView";
import {gql} from "@apollo/client";

const GET_PAGE_OF_STUDENTS = gql`
    query getPageOfStudents($page: Int!){
        students(page: $page){
            name
            lastName
            group {
                groupName
            }
            sessionSet {
                mark
            }
            img
        }
    }`;

const GET_STUDENTS_N = gql`
    query {
        studentN
    }
`;
```

```
export function StudentPage() {
    return (
        <div>
            <h2>Students</h2>
            <StudentView get_page={GET_PAGE_OF_STUDENTS}
get_n={GET_STUDENTS_N}/>
        </div>
    )
}
 import React, {useState} from 'react';
 import {useQuery} from "@apollo/client";
 import {Loading} from "../Components/Loading";



 export const PaginationFooter = ({query, page, setPage}) =>  {
     const {error, loading, data} = useQuery(query);
     return (
         <div>
             {error ? <span>error</span> : loading ? <Loading/> :
                 <div className="d-flex justify-content-around">
                     <span>
                         {page === 1 ? <span/> :
                             <button type="button" onClick={() => setPage(page -
 1)} className="btn btn-light">{`<`}</button>}
                         <span>   {page} / </span>
                         <span>{data.studentN}   </span>
                         {page === data.studentN ? <span/> :
                             <button type="button" onClick={() => setPage(page +
 1)} className="btn btn-light">{`>`}</button>}
                     </span>
                 </div>}
         </div>

     )
 }
 import React from 'react';


 export function HomeView() {
     return (
         <>
             <div className="container login-container">
                 <div className="row">
                     <div className="col-md-6 login-form-1">
                         <h3>Login for Form 1</h3>
                         <form>
                             <div className="form-group">
                                 <input type="text" className="form-control"
 placeholder="Your Email *" value=""/>
                             </div>
                             <div className="form-group">
                                 <input type="password" className="form-control"
 placeholder="Your Password *" value=""/>
                             </div>
                             <div className="form-group">
                                 <input type="submit" className="btnSubmit"
 value="Login"/>
                             </div>
                             <div className="form-group">
                                 <a href="#" className="ForgetPwd">Forget
 Password?</a>
                             </div>
```

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

    )
}

import React from 'react';
import { gql } from '@apollo/client';
import {useQuery} from "@apollo/client";

const PEOPLE = gql`
    query {
        students {
            id
            name
            lastName
            img
        }
    }
`;

export function GroupView() {
    const { loading, data } = useQuery(PEOPLE);
    return (
        <div>
            group
        </div>
    );
}
import React from 'react';
import {GroupView} from "./GroupView";

export function GroupsPage() {
    return (
        <GroupView/>
    )
}
import React from 'react';


export const Loading = () =>  {
    return (
         <div className="spinner-border" role="status">
             <span className="sr-only"> </span>
         </div>
     )
}
import React, {useState} from 'react';


export const SmallImg = ({src}) =>  {
    const [Img, setImg] = useState(src)

    return (
        <img src={Img.valueOf()} onError={() =>
setImg('personUnknown.png')} width="60" className="img-thumbnail"
alt='img'/>
    )
}
```

```
import React from 'react';
import {NavLink} from "react-router-dom";
import './Header.css'

export function Header() {
    return (
        <div>
            <header className="px-3 py-2 bg-dark text-white">
                <div className="container">
                    <div className="d-flex flex-wrap align-items-center
justify-content-center justify-content-lg-start">
                        <NavLink to="/"
                           className="d-flex align-items-center my-2 my-
lg-0 me-lg-auto text-white text-decoration-none">
                            <svg xmlns="http://www.w3.org/2000/svg"
width="40" height="40" fill="currentColor"
                                 className="bi bi-house" viewBox="0 0 16
16">
                                <path fillRule="evenodd"
                                      d="M2 13.5V7h1v6.5a.5.5 0 0 0
.5.5h9a.5.5 0 0 0 .5-.5V7h1v6.5a1.5 1.5 0 0 1-1.5 1.5h-9A1.5 1.5 0 0 1 2
13.5zm11-11V6l-2-2V2.5a.5.5 0 0 1 .5-.5h1a.5.5 0 0 1 .5.5z"/>
                                <path fillRule="evenodd"
                                      d="M7.293 1.5a1 1 0 0 1 1.414
0l6.647 6.646a.5.5 0 0 1-.708.708L8 2.207 1.354 8.854a.5.5 0 1 1-.708-
.708L7.293 1.5z"/>
                            </svg>
                        </NavLink>

                        <ul className="nav col-12 col-lg-auto my-2
justify-content-center my-md-0 text-small">
                            <li>
                                <NavLink to="/students" className="nav-
link" activeClassName="text-secondary">
                                    <svg
xmlns="http://www.w3.org/2000/svg" width="24" height="24"
fill="currentColor"
                                         className="bi d-block mx-auto
mb-1" viewBox="0 0 16 16">
                                        <path d="M7 14s-1 0-1-1 1-4 5-4 5
3 5 4-1 1-1 1H7zm4-6a3 3 0 1 0 0-6 3 3 0 0 0 0 6z"/>
                                        <path fill-rule="evenodd"
d="M5.216 14A2.238 2.238 0 0 1 5 13c0-1.355.68-2.75 1.936-3.72A6.325
6.325 0 0 0 5 9c-4 0-5 3-5 4s1 1 1 1h4.216z"/>
                                        <path d="M4.5 8a2.5 2.5 0 1 0 0-5
2.5 2.5 0 0 0 0 5z"/>
                                    </svg>
                                    Students
                                </NavLink>
                            </li>
                            <li>
                                <NavLink to="/groups" className="nav-
link" activeClassName="text-secondary">
                                    <svg
xmlns="http://www.w3.org/2000/svg" width="24" height="24"
fill="currentColor"
                                         className="bi d-block mx-auto
mb-1" viewBox="0 0 16 16">
```

```
                                        <path fill-rule="evenodd" d="M6
3.5A1.5 1.5 0 0 1 7.5 2h1A1.5 1.5 0 0 1 10 3.5v1A1.5 1.5 0 0 1 8.5
6v1H14a.5.5 0 0 1 .5.5v1a.5.5 0 0 1-1 0V8h-5v.5a.5.5 0 0 1-1 0V8h-
5v.5a.5.5 0 0 1-1 0v-1A.5.5 0 0 1 2 7h5.5V6A1.5 1.5 0 0 1 6 4.5v-1zm-6
8A1.5 1.5 0 0 1 1.5 10h1A1.5 1.5 0 0 1 4 11.5v1A1.5 1.5 0 0 1 2.5 14h-
1A1.5 1.5 0 0 1 0 12.5v-1zm6 0A1.5 1.5 0 0 1 7.5 10h1a1.5 1.5 0 0 1 1.5
1.5v1A1.5 1.5 0 0 1 8.5 14h-1A1.5 1.5 0 0 1 6 12.5v-1zm6 0a1.5 1.5 0 0 1
1.5-1.5h1a1.5 1.5 0 0 1 1.5 1.5v1a1.5 1.5 0 0 1-1.5 1.5h-1a1.5 1.5 0 0 1-
1.5-1.5v-1z"/>
                                    </svg>
                                    Groups
                                </NavLink>
                            </li>
                            <li>
                                <NavLink to="/teachers" className="nav-
link" activeClassName="text-secondary" href="/teachers" className="nav-
link">
                                    <svg
xmlns="http://www.w3.org/2000/svg" width="24" height="24"
fill="currentColor"
                                         className="bi d-block mx-auto
mb-1" viewBox="0 0 16 16">
                                        <path d="M6 8a3 3 0 1 0 0-6 3 3 0
0 0 0 6zm-5 6s-1 0-1-1 1-4 6-4 6 3 6 4-1 1-1 1H1zM11 3.5a.5.5 0 0 1 .5-
.5h4a.5.5 0 0 1 0 1h-4a.5.5 0 0 1-.5-.5zm.5 2.5a.5.5 0 0 0 0 1h4a.5.5 0 0
0 0-1h-4zm2 3a.5.5 0 0 0 0 1h2a.5.5 0 0 0 0-1h-2zm0 3a.5.5 0 0 0 0
1h2a.5.5 0 0 0 0-1h-2z"/>
                                    </svg>
                                    Teachers
                                </NavLink>
                            </li>
                            <li>
                                <NavLink to="/subjects" className="nav-
link" activeClassName="text-secondary">
                                    <svg
xmlns="http://www.w3.org/2000/svg" width="24" height="24"
fill="currentColor"
                                         className="bi d-block mx-auto
mb-1" viewBox="0 0 16 16">
                                        <path d="M1 2.828c.885-.37 2.154-
.769 3.388-.893 1.33-.134 2.458.063 3.112.752v9.746c-.935-.53-2.12-.603-
3.213-.493-1.18.12-2.37.461-3.287.811V2.828zm7.5-.141c.654-.689 1.782-
.886 3.112-.752 1.234.124 2.503.523 3.388.893v9.923c-.918-.35-2.107-.692-
3.287-.81-1.094-.111-2.278-.039-3.213.492V2.687zM8 1.783C7.015.936
5.587.81 4.287.94c-1.514.153-3.042.672-3.994 1.105A.5.5 0 0 0 0
2.5v11a.5.5 0 0 0 .707.455c.882-.4 2.303-.881 3.68-1.02 1.409-.142
2.59.087 3.223.877a.5.5 0 0 0 .78 0c.633-.79 1.814-1.019 3.222-.877
1.378.139 2.8.62 3.681 1.02A.5.5 0 0 0 16 13.5v-11a.5.5 0 0 0-.293-.455c-
.952-.433-2.48-.952-3.994-1.105C10.413.809 8.985.936 8 1.783z"/>
                                    </svg>
                                    Subjects
                                </NavLink>
                            </li>
                            <li>
                                <NavLink to="/years" className="nav-link"
activeClassName="text-secondary">
                                    <svg
xmlns="http://www.w3.org/2000/svg" width="24" height="24"
```

```
fill="currentColor"
                                                 className="bi d-block mx-auto
mb-1" viewBox="0 0 16 16">
                                                <path
                                                    d="M10.854 7.146a.5.5 0 0 1 0
.708l-3 3a.5.5 0 0 1-.708 0l-1.5-1.5a.5.5 0 1 1 .708-.708L7.5
9.793l2.646-2.647a.5.5 0 0 1 .708 0z"/>
                                                <path
                                                    d="M3.5 0a.5.5 0 0 1
.5.5V1h8V.5a.5.5 0 0 1 1 0V1h1a2 2 0 0 1 2 2v11a2 2 0 0 1-2 2H2a2 2 0 0
1-2-2V3a2 2 0 0 1 2-2h1V.5a.5.5 0 0 1 .5-.5zM1 4v10a1 1 0 0 0 1 1h12a1 1
0 0 0 1-1V4H1z"/>
                                            </svg>
                                            Years
                                        </NavLink>
                                    </li>
                                </ul>
                            </div>
                        </div>
                    </header>
                </div>

            )
        }
        from django.db import models


        class Group(models.Model):
            group_name = models.TextField()
            description = models.TextField()
            # specialty

            def __str__(self):
                return self.group_name


        class Teacher(models.Model):
            name = models.TextField()
            last_name = models.TextField()
            img = models.TextField()

            # subjects =
            def __str__(self):
                return self.name + " " + self.last_name


        class Subject(models.Model):
            name = models.TextField()
            teacher = models.ForeignKey(Teacher, on_delete=models.CASCADE)
            description = models.TextField()

            def __str__(self):
                return self.name


        class Student(models.Model):
            name = models.TextField()
            last_name = models.TextField()
```

```
    img = models.TextField()
    group = models.ForeignKey(Group, on_delete=models.CASCADE)

    # subjects =
    def __str__(self):
        return self.name + " " + self.last_name


class Session(models.Model):
    student = models.ForeignKey(Student, on_delete=models.CASCADE)
    subject = models.ForeignKey(Subject, on_delete=models.CASCADE)
    mark = models.IntegerField()
    date = models.DateField()

    def __str__(self):
        return "{} {} {}".format(self.student.__str__(),
self.subject.name, self.mark)
import graphene
from django.core.paginator import Paginator

from graphene_django.types import DjangoObjectType, ObjectType

from student.models import Student, Teacher, Subject, Group, Session

N = 20  # number on page


class StudentType(DjangoObjectType):
    class Meta:
        model = Student


class TeacherType(DjangoObjectType):
    class Meta:
        model = Teacher


class GroupType(DjangoObjectType):
    class Meta:
        model = Group


class SessionType(DjangoObjectType):
    class Meta:
        model = Session


class SubjectType(DjangoObjectType):
    class Meta:
        model = Subject


class Query(ObjectType):
    student = graphene.Field(StudentType, id=graphene.Int())
    studentN = graphene.Field(graphene.Int())
    teacher = graphene.Field(TeacherType, id=graphene.Int())
    teacherN = graphene.Field(graphene.Int())
    subject = graphene.Field(SubjectType, id=graphene.Int())
    students = graphene.List(StudentType, page=graphene.Int())
```

```
    teachers = graphene.List(TeacherType, page=graphene.Int())
    subjects = graphene.List(SubjectType)

    def resolve_student(self, info, **kwargs):
        id = kwargs.get('id')
        if id is not None:
            return Student.objects.get(pk=id)
        return None

    def resolve_studentN(self, info, **kwargs):
        return Paginator(Student.objects.all(), N).num_pages

    def resolve_teacher(self, info, **kwargs):
        id = kwargs.get('id')
        if id is not None:
            return Teacher.objects.get(pk=id)
        return None

    def resolve_teacherN(self, info, **kwargs):
        return Paginator(Teacher.objects.all(), N).num_pages

    def resolve_subject(self, info, **kwargs):
        id = kwargs.get('id')
        if id is not None:
            return Subject.objects.get(pk=id)
        return None

    def resolve_subjects(self, info, **kwargs):
        return Subject.objects.all()

    def resolve_students(self, info, **kwargs):
        page = int(kwargs.get('page'))
        return Paginator(Student.objects.all(), N).page(page).object_list

    def resolve_teachers(self, info, **kwargs):
        try:
            page = int(kwargs.get('page'))
        except:
            page = 1
        return Paginator(Teacher.objects.all(), N).page(page).object_list


schema = graphene.Schema(query=Query)
"""
Django settings for backend project.

Generated by 'django-admin startproject' using Django 3.2.

For more information on this file, see
https://docs.djangoproject.com/en/3.2/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/3.2/ref/settings/
"""

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/3.2/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'django-insecure--
!_)h(^k0qxyy+zsnc!9t1o1ie0a0x_0^pcts8g69%q8n!gu=s'

# 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',
    'corsheaders',
    'student',
    'graphene_django',
]

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

]

ROOT_URLCONF = 'backend.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 = 'backend.wsgi.application'


# Database
# https://docs.djangoproject.com/en/3.2/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}


# Password validation
# https://docs.djangoproject.com/en/3.2/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/3.2/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_L10N = True

USE_TZ = True


# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/3.2/howto/static-files/
```

```
STATIC_URL = '/static/'

# Default primary key field type
# https://docs.djangoproject.com/en/3.2/ref/settings/#default-auto-field

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'

GRAPHENE = {
    'SCHEMA': 'backend.schema.schema',
}

CORS_ORIGIN_ALLOW_ALL = True
# -*- coding: cp1251 -*-
import json
import random
import random
import time
import datetime

names = ['Азар', 'Анатолій', 'Андрій', 'Антін', 'Антон', 'Аркадій', 'Артем', 'Архип',
'Бажан', 'Біловид', 'Біломир',
    'Білослав', 'Благомир', 'Богдан', 'Богуслав', 'Божен', 'Борис', 'Борислав',
'Боронислав', 'Братислав',
    'Буревій', 'Вадим', 'Валентин', 'Василь', 'Велемир', 'Валерій', 'Вишеслав',
'Віктор', 'Віорослав', 'Віталій',
    'Владислав', 'Володимир', 'Володислав', 'Всеволод', "В'ячеслав",
'Гаврило', 'Гарнослав', 'Гнат', 'Гордислав',
    'Гордій', 'Горислав', 'Гремислав', 'Григорій', 'Данило', 'Даромир', "Дем'ян",
'Дмитро', 'Добриня', 'Добровіст',
    'Добромир', 'Доброслав', 'Євген', 'Євгеній', 'Єгор', 'Живослав', 'Захар',
'Звенислав', 'Зиновій', 'Златомир',
    'Зоремир', 'Зореслав', 'Зорян', 'Іван', 'Ігор', 'Ілля', 'Кий', 'Кирило', 'Корнило',
'Корній', 'Костянтин',
    'Кузьма', 'Ладомир', 'Левко', 'Лесь', "Лук'ян", 'Любодар', 'Любозар',
'Любомир', 'Любослав', 'Макар', 'Максим',
    'Марко', "Мар'ян", 'Матвій', 'Мечислав', 'Микита', 'Микола', 'Мирон',
'Мирослав', 'Михайло', 'Мстислав',
    'Мусій', 'Назар', 'Назарій', 'Немир', 'Непобор', 'Нестор', 'Никифор',
'Никодим', 'Олег', 'Олекса', 'Олександр',
    'Олексій', 'Олесь', 'Омелян', 'Опанас', 'Орест', 'Остап', 'Острозор',
'Остромир', 'Остромов', 'Павло', 'Панас',
    'Пересвіт', 'Петро', 'Пилип', 'Пимон', 'Полян', 'Потап', 'Предислав',
'Радило', 'Радим', 'Радован', 'Радомир',
    'Радослав', 'Ратислав', 'Родіон', 'Родослав', 'Роман', 'Ростислав', 'Свирид',
'Світ', 'Світлогор', 'Світлодар',
```

'Світозар', 'Світомир', 'Світослав', 'Святогор', 'Святослав', 'Северин', 'Семен', 'Семибор', 'Семирад',

'Серафим', 'Сидір', 'Сергій', 'Силантій', 'Силослав', 'Славобор', 'Славомир', 'Сніжан', 'Сонцевид', 'Сонцедар',

'Станислав', 'Степан', 'Тарас', 'Твердислав', 'Творимир', 'Творислав', 'Терентій', 'Тимофій', 'Тихон', 'Устим',

'Федір', 'Федот', 'Хвалимир', 'Хранимир', 'Честислав', 'Щастибог', 'Щастислав', 'Юрій', 'Юхим', 'Яків', 'Ян',

'Ярема', 'Ярило', 'Яровид', 'Яромир', 'Ярослав', 'Ясновид', 'Ясногор', 'Яснозір', '']

surname = ['Андрейчин', 'Андрієнко', 'Андрійченко', 'Андріїв', 'Андріїшин', 'Андрухів', 'Антипенко', 'Антоненко',

'Артемівець', 'Артюхів', 'Архипенко', 'Архипчук', 'Бабай', 'Бабак', 'Бабець', 'Бабій', 'Багрій', 'Балаба',

'Балицький', 'Бедрій', 'Бездух', 'Березень', 'Березин', 'Березовець', 'Бистрицький', 'Білий', 'Білик',

'Біличак', 'Біличенко', 'Білогай', 'Білозір', 'Білокінь', 'Білуха', 'Благовісний', 'Блажко', 'Богданець',

'Богданів', 'Богомолець', 'Богуславець', 'Божко', 'Бойків', 'Бойко', 'Бойчук', 'Болибік', 'Болотницький',

'Бондар', 'Борівець', 'Бородайко', 'Бородій', 'Борозняк', 'Борщ', 'Брага', 'Бражник', 'Братків',

'Брилинський', 'Брюховець', 'Бугаєць', 'Бугрій', 'Будій', 'Будник', 'Будниченко', 'Букрій', 'Булавин',

'Булигин', 'Бурдій', 'Бутенко', 'Буцько', 'Бучма', 'Вакулик', 'Ванчишин', 'Ваньо', 'Варивода', 'Василець',

'Василів', 'Васильків', 'Васійчук', 'Васьо', 'Ваютин', 'Вдовиченко', 'Ведмідь', 'Вербицький', 'Вересень',

'Верещака', 'Вертій', 'Верходай', 'Веснин', 'Винник', 'Винниченко', 'Виноградів', 'Витвицький', 'Вівсяник',

'Вівчаренко', 'Вовк', 'Вовняр', 'Водовіз', 'Водяник', 'Воловик', 'Волоський', 'Волохач', 'Волошин',

'Волошко', 'Воротило', 'Гаврилець', 'Гавриличенко', 'Гаврилишин', 'Гаврилів', 'Гаврись', 'Гавришко', 'Галан',

'Галицький', 'Галунько', 'Гамалій', 'Гамарник', 'Ганич', 'Гармаш', 'Глібчук', 'Говоруха', 'Голик', 'Гончар',

'Горбай', 'Горобець', 'Гречкосій', 'Гриневич', 'Гринів', 'Грицай', 'Гришковець', 'Громів', 'Грушевський',

'Губанів', 'Губич', 'Гуляйвітер', 'Гутенюк', 'Ґава', 'Ґалаґан', 'Ґонтар', 'Ґедзь', 'Ґудзь', 'Даниленко',

'Данилець', 'Данилишин', 'Дармограй', 'Дарницький', 'Дахно', 'Дацько', 'Двірниченко', "Дем'яненко",

"Дем'янів", 'Денисенко', 'Денисюк', 'Дереворіз', "Дерев'яненко", "Дерев'янко", 'Дереш', 'Дігтяр', 'Дідик',

'Дмитрів', 'Довгай', "Довгоп'ят", 'Довженко', 'Долинний', 'Долинюк', 'Дорошенко', 'Дорощук', 'Доскоч',

'Древницький', 'Дрізд', 'Дріт', 'Дроботько', 'Дубина', 'Дубинський', 'Дубовик', 'Дудниченко', 'Думич',

'Дяченко', 'Євдокимів', 'Єдинець', 'Ємець', 'Єреміїв', 'Єрмолаїв', 'Жвавець', 'Живицький', 'Житник', 'Жихар',

'Жовницький', 'Жовнір', 'Жовтоніг', 'Журило', 'Забужко', 'Завалій', 'Завірюха', 'Загребельний', 'Задерій',

'Задніпрянець', 'Заєць', 'Заїка', 'Закусило', 'Залипський', 'Заліський', 'Запоріжець', 'Захариченко',

'Зеленський', 'Земляний', 'Зименко', 'Зимогляд', 'Зозулич', 'Зозуля', 'Золотар', 'Зубенко', 'Зубко',

'Іваненко', 'Іванець', 'Іваничко', 'Іванишин', 'Іванів', 'Іздрик', 'Ілик', 'Іщенко', 'Кабанець', 'Каленик',

'Калина', 'Калинчук', 'Калинюк', 'Кальницький', 'Карбівник', 'Карпів', 'Карпів', 'Карпюк', 'Касіян',

'Квасниця', 'Кивало', 'Кивко', 'Кириленко', 'Кирилів', 'Кириченко', 'Киричук', 'Кирпань', 'Киселиця',

'Киселюк', 'Кисіль', 'Кислиця', 'Кияшко', 'Кінь', 'Кіт', 'Клименко', 'Климів', 'Клочко', 'Кобилинський',

'Коваленко', 'Ковалець', 'Ковалик', 'Ковалів', "Кожем'яка", 'Кожухар', 'Козачок', 'Козачук', 'Козубів',

'Колісник', 'Колісниченко', 'Коломиєць', 'Колотуха', 'Коляда', 'Коноплицький', 'Кормилець', 'Косенко',

'Косів', 'Котвицький', 'Котів', 'Кочерга', 'Красій', 'Краснокутський', 'Краснощок', 'Красюк',

'Кременчуцький', 'Кремінь', 'Кривицький', 'Кривоніс', 'Криворучко', 'Кривохвіст', 'Кримський', 'Криницький',

'Кропивницький', 'Крутиголова', 'Крушельницький', 'Кубич', 'Куйбіда', 'Кулик', 'Куликівський', 'Кулій',

'Куницький', 'Курило', 'Кухар', 'Куцовіл', 'Кучма', 'Кушнір', 'Лаврів', 'Ладницький', 'Лазарець', 'Лаптій',

'Лахно', 'Лебединець', 'Лебединський', 'Левченко', 'Лемчик', 'Лесик', 'Лесів', 'Леськів', 'Лещенко', 'Лимар',

'Линдюк', 'Линник', 'Липинський', 'Лисенко', 'Лисогір', 'Литвин', 'Литвинець', 'Литвинчук', 'Лісненко',

'Лісницький', 'Логвиненко', 'Лопатин', 'Лубенець', 'Лукаш', "Лук'яненко", 'Луценко', 'Любич', 'Лях', 'Ляшко',

'Мазанець', 'Мазур', 'Макаренко', 'Маковій', 'Макоїд', 'Максименко', 'Максимів', 'Максимюк', 'Малетич',

'Малик', 'Малицький', 'Маловічко', 'Мальованик', 'Мандрика', 'Манилик', 'Маньо', 'Мартиненко', 'Мартинів',

'Марусин', 'Марчук', 'Маслій', 'Масний', 'Матвієнко', 'Матвіїв', 'Мацько', 'Микитенко', 'Микитченко',

'Миколаїв', 'Микулець', 'Миловар', 'Мимохід', 'Мисюра', 'Миськів', 'Михайлик', 'Мицик', 'Мицько', 'Мищук',

'Мірошниченко', 'Міщенко', 'Могильницький', 'Мокрій', 'Монич', 'Морквин', 'Мотиль', 'Музика', 'Музиченко',

'Мусенко', 'Мусій', "М'ясоїд", 'Надточій', 'Наріжний', 'Наумець', 'Небийголова', 'Невінчаний', 'Негрій',

'Недайборщ', 'Неїжмак', 'Немирич', 'Ненчин', 'Неплюй', 'Несвітайло', 'Нестелій', 'Нетудихата', 'Нехай',

'Никифоренко', 'Никінчук', 'Новицький', 'Носенко', 'Нудьга', 'Оберемок', 'Овдієнко', 'Оверченко',

'Овраменко', 'Огій', 'Огорілко', 'Одарченко', 'Одиниця', 'Одноріг', 'Олексієнко', 'Оленич', 'Олеськів',

'Олійник', 'Олійниченко', 'Омеляненко', 'Онисенко', 'Орлик', 'Оселедько', 'Осика', 'Осипенко', 'Остапенко',

'Остапів', 'Осьмака', 'Павличко', 'Павлишин', 'Павлів', 'Павлюх', 'Пазій', 'Палагнюк', 'Палисвіт',

'Палійчук', 'Паляниця', 'Панчишин', 'Панчук', 'Парубій', 'Пасічник', 'Пашковець', 'Пелех', 'Перебийніс',

'Передерій', 'Перець', 'Петренко', 'Петриця', 'Петриченко', 'Петровський', 'Петрунів', 'Пилипенко',

'Пилипець', 'Пиріг', 'Півторацький', 'Підгірний', 'Піддубний', 'Підлісний', 'Плаксій', 'Плахотнюк',

'Поберій', 'Повалій', "Подерв'янський", 'Подоляк', 'Покотило', 'Поляник', 'Пономарів', 'Поперека',

'Порошенко', "Потоп'як", 'Почекайло', 'Прилипко', 'Приходько', 'Прокопів', 'Проценко', 'Пташник', 'Процько',

'Пухлій', 'Пушкар', 'Пшеничний', 'Равлик', 'Радевич', 'Радченко', 'Редько', 'Решетило', 'Решітник', 'Рибник',

'Різник', 'Різниченко', 'Робітько', 'Роздобудько', 'Романець', 'Ротань', 'Рудий', 'Рудник', 'Рукодій',

'Рябець', 'Рябовіл', 'Рябокінь', 'Рябуха', 'Савич', 'Савченко', 'Савчин', 'Салабай', 'Салоїд', 'Самійленко',

'Самокиш', 'Сахно', 'Свириденко', 'Світайло', 'Свічкар', 'Святенко', 'Семенець', 'Семенів', 'Семеренко',

'Семченко', 'Семчишин', 'Сенько', 'Сидоренко', 'Сидоренко', 'Сидорець', 'Сидорів', 'Симоненко', 'Синельник',

'Сироїд', 'Ситник', 'Сіроштан', 'Сковорода', 'Скориченко', 'Скринник', 'Скрипаль', 'Славницький', 'Слинько',

```
        'Сліпець', 'Сліпченко', 'Слободяник', 'Смаглій', 'Смолій', 'Смотрич',
    'Смуток', 'Сокирко', 'Сокіл',
        'Соколик', 'Соловей', 'Соловиченко', 'Соломаха', 'Соляник', 'Сосюра',
    'Співак', 'Стадник', 'Стадниченко',
        'Стахів', 'Стеблюк', 'Стельмащук', 'Степанишин', 'Степанів', 'Стець',
    'Стовпюк', 'Стрельник', 'Стрижко',
        'Стрілецький', 'Стрільниченко', 'Сулименко', 'Сурмій', 'Сухомлин',
    'Сухоніс', 'Танцюра', 'Таранець',
        'Твердик', 'Твердохліб', 'Телиця', 'Теличко', 'Телятник', 'Темник',
    'Терешко', 'Тесленко', 'Теслюк',
        'Теслярчук', 'Тимків', 'Тимофіїв', 'Тимошенко', 'Тимчик', 'Тимчишин',
    'Титаренко', 'Титко', 'Титовський',
        'Тичина', 'Тищенко', 'Ткаченко', 'Ткачів', 'Товстуха', 'Тонконіг',
    'Топольницький', 'Топчій', 'Торкало',
        'Тригуб', 'Трохименко', 'Трохимчук', 'Трубій', 'Труш', 'Турутько',
    'Тушницький', 'Тютюнник', 'Тюхтій',
        'Тягнибік', 'Убийкінь', 'Удовицький', 'Ужвій', 'Українець', 'Уманець',
    'Усик', 'Устименко', 'Федорець',
        'Федорик', 'Фесенко', 'Філюк', 'Франко', 'Фурманюк', 'Харко', 'Хвостик',
    'Хижко', 'Хилобік', 'Хилько',
        'Хильчук', 'Хмельницький', 'Холодницький', 'Цвіркун', 'Цебань',
    'Цегелик', 'Цимбал', 'Чаплій', 'Часник',
        'Червонощок', 'Черевичник', 'Черевченко', 'Чередниченко', 'Черкаський',
    'Черниченко', 'Чорний', 'Чорниш',
        'Чорновіл', 'Шаблій', 'Шамрай', 'Шашкевич', 'Швайка', 'Швець', 'Шевців',
    'Шевченко', 'Шевчук', 'Шепелюк',
        'Шеремет', 'Шинкар', 'Шкіряк', 'Шмагайло', 'Шмалій', 'Шраменко',
    'Шовкопляс', 'Штовхань', 'Шульга',
        'Шульженко', 'Щербина', 'Щербоніс', 'Щурко', 'Щуцький', 'Юзько',
    'Юрашко', 'Юрків', 'Юхименко', 'Юхимець',
        'Юхимів', 'Ющенко', 'Яблонський', 'Яворницький', 'Якименко',
    'Якимець', 'Яковенко', 'Ялиця', 'Яресько',
        'Ярош', 'Яхно', 'Яценко', 'Яцьків', '']
    subject = ["Програмування", "ООП", "Інженерія програмного забезпечення",
    "Математичний аналіз", "Аналітична геометрія",
        "Фізика", "Сопромат", "Архітектура комп'ютерів", "Системне
    програмування", "Обчислювальні системи"]

    d = []
    t = 0

def str_time_prop(start, end, format, prop):
    """Get a time at a proportion of a range of two formatted times.
```

```
    start and end should be strings specifying times formated in the
    given format (strftime-style), giving an interval [start, end].
    prop specifies how a proportion of the interval to be taken after
    start.  The returned time will be in the specified format.
    """

    stime = time.mktime(time.strptime(start, format))
    etime = time.mktime(time.strptime(end, format))

    ptime = stime + prop * (etime - stime)

    return time.strftime(format, time.localtime(ptime))


def random_date(start="2017-1-1", end="2019-1-1", prop=random.random()):
    return str_time_prop(start, end, '%Y-%d-%m', prop)


for i in range(len(surname)):
    temp = i//30
    data = {}
    data['model'] = "student.student"
    data['pk'] = i
    data['fields'] = ({
        'name': str(random.sample(names, 1)[0], ),
        'last_name': random.sample(surname, 1)[0],
        'group': temp,
        'img': "person{}.png".format(i % 31)
    })
    d.append(data)

for i in range(0, int(len(surname)/30)+1,):
    temp = i
    data = {}
    data['model'] = "student.group"
    data['pk'] = temp
    data['fields'] = ({
        'group_name': 'IV-{}'.format(temp+10),
        'description': random.sample(subject, 1)[0]+" наше все!!!!!",
    })
    d.append(data)
```

```
for i in range(len(subject)):
    data = {}
    data['model'] = "student.teacher"
    data['pk'] = i
    data['fields'] = ({
            'name': str(random.sample(names, 1)[0], ),
            'last_name': random.sample(surname, 1)[0],
            'img': "personOld{}.png".format(i % 31)
        })
        d.append(data)

    # for i in range(len(subject)):
    #     data = {}
    #     data['model'] = "student.subject"
    #     data['pk'] = i
    #     data['fields'] = ({
    #         'name': subject[i],
    #         'description': subject[i],
    #         'teacher_id': i
    #     })
    #     d.append(data)

    for i in range(len(surname)):
        for j in range(10):
            t = random_date(prop=random.random()).split('-')
            y = int(t[0])
            m = int(t[2])
            day = int(t[1])
            data = {}
            data['model'] = "student.session"
            data['pk'] = int("{}{}".format(i, j))
            data['fields'] = ({
                'mark': random.sample(range(60, 100), 1)[0],
                'date': datetime.date(y, m, day).__str__(),
                'student_id': i,
                'subject_id': j,
            })
            d.append(data)

    with open('data.json', 'w', encoding='utf-8') as outfile:
        json.dump(d, outfile, ensure_ascii=False)
```