

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

**Пояснювальна записка**

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему РОЗРОБКА ВЕБЗАСТОСУНКУ ДЛЯ СТВОРЕННЯ ТА

(назва теми)

КЕРУВАННЯ РОЗКЛАДОМ. FRONTEND ЧАСТИНА  
DEVELOPMENT OF A WEB APPLICATION TO CREATE AND MANAGE A  
SCHEDULE. FRONTEND PART

Виконав(ла): студент(ка) 4 курсу, групи КНТ-219

Спеціальності 122 Комп'ютерні науки

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Комп'ютерні науки

МЄДВЕДЄВ К.В.

(ПРИЗВИЩЕ та ініціали)

Керівник СТЕПАНЕНКО О.О.

(ПРИЗВИЩЕ та ініціали)

Рецензент ГОЛУБ Т.В.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
Національний університет «Запорізька політехніка»

Факультет ФКНТ

Кафедра програмних засобів

Ступінь вищої освіти бакалавр

Спеціальність 122 Комп'ютерні науки

(код і найменування)

Освітня програма (спеціалізація) Комп'ютерні науки

(назва освітньої програми (спеціалізації))

**ЗАТВЕРДЖУЮ**

Завідувач кафедри ПЗ, д.т.н, проф.

**Сергій СУББОТІН**

«\_\_\_\_\_» \_\_\_\_\_ 2023 року

**З А В Д А Н Н Я**  
**НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)**

МЄДВЄДЄВА Кирила Віталійовича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Розробка вебзастосунку для створення та керування розкладом. Frontend частина

Development of a web application to create and manage a schedule. Frontend part

керівник проєкту (роботи) к.т.н., доцент, СТЕПАНЕНКО Олександр Олексійович

(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від «4» квітня 2023 року № 87

2. Строк подання студентом проєкту (роботи) 1 червня 2023 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Матеріали і методи. 3 Опис програми. 4. Експлуатація, тестування та експериментальне дослідження програми.

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

Слайди презентації

## 6. Консультанти розділів проєкту (роботи)

| Розділ              | ПРИЗВИЩЕ, ініціали та посада консультанта | Підпис, дата   |                           |
|---------------------|-------------------------------------------|----------------|---------------------------|
|                     |                                           | завдання видав | прийняв виконане завдання |
| 1-4 Основна частина | СТЕПАНЕНКО О.О., доцент                   |                |                           |
| Нормоконтроль       | БСЛОВА А.В., асистент                     |                |                           |

## 7. Дата видачі завдання «4» квітня 2023 року.

### КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів дипломного проєкту (роботи)                             | Строк виконання етапів проєкту ( роботи ) | Примітка     |
|-------|----------------------------------------------------------------------|-------------------------------------------|--------------|
| 1     | Постановка завдання роботи.                                          | 1 тиждень                                 | Завдання, ТЗ |
| 2     | Аналіз предметної області.                                           | 1 тиждень                                 | Розділ 1     |
| 3     | Вибір мови програмування та інших технологій розробки.               | 2 тиждень                                 | Розділ 2     |
| 4     | Розробка архітектури програми.                                       | 2 тиждень                                 | Розділ 3     |
| 5     | Розробка програми.                                                   | 3-4 тижні                                 | Розділ 3, 4  |
| 6     | Тестування та експериментальне дослідження програмного забезпечення. | 5 тиждень                                 | Розділ 4     |
| 7     | Оформлення пояснювальної записки та документів до неї.               | 6 тиждень                                 | Додатки      |
| 8     | Нормоконтроль та рецензування.                                       | 7 тиждень                                 |              |
| 9     | Захист роботи.                                                       | 8 тиждень                                 |              |

Студент(ка)

\_\_\_\_\_ Кирило МЄДВЕДЄВ  
( підпис ) (Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

\_\_\_\_\_ Олександр СТЕПАНЕНКО  
( підпис ) (Ім'я ПРИЗВИЩЕ)

## РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 139 с., 58 рис., 8 табл., 4 дод., 35 джерел.

СКЛАДАННЯ РОЗКЛАДУ, РОЗКЛАД ЗАНЯТЬ, ГЕНЕТИЧНИЙ АЛГОРИТМ, REACT, JAVASCRIPT.

Об'єкт дослідження – процес складання та оптимізації розкладу занять для університету.

Предмет дослідження – програмні засоби для складання розкладу занять.

Метою роботи є дослідження та розробка програмного забезпечення для складання та керування розкладом.

Матеріали, методи та технічні засоби: мова програмування JavaScript, фреймворк React.

Результати. Розроблено вебзастосунок для автоматизованого складання розкладу занять для університету, яке дозволяє користувачам переглядати розклад, керувати даними та запускати алгоритми для складання.

Висновки. Створено програмний продукт для складання та керування розкладом.

Галузь використання – може бути використане в університеті для планування розкладу навчальних занять. Програмний продукт допоможе забезпечити оптимальне розподілення ресурсів, уникнути перекриття занять та враховувати особливості графіків викладачів і студентів.

## ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 139 pages, 58 figures, 8 tables, 4 appendixes, 35 sources.

SCHEDULING, CLASS SCHEDULE, GENETIC ALGORITHM, REACT, JAVASCRIPT.

The object of research is the process of creating and optimizing class schedules for a university. The subject of the study is software tools for scheduling classes.

The purpose of the study is to research and develop software for creating and managing schedules. Materials, methods and techniques: JavaScript programming language, React framework.

Results. A web application for automated scheduling of classes for a university has been developed that allows users to view the schedule, manage data, and run algorithms for scheduling. Conclusions. A software product for creating and managing schedules has been created.

Application – can be used at the university to plan the schedule of classes. The software product will help to ensure optimal resource allocation, avoid overlapping classes, and take into account the peculiarities of teachers' and students' schedules.

## ЗМІСТ

|                                                                                    |    |
|------------------------------------------------------------------------------------|----|
|                                                                                    | С. |
| Перелік скорочень та умовних позначок .....                                        | 8  |
| Вступ .....                                                                        | 9  |
| 1 Аналіз предметної області .....                                                  | 11 |
| 1.1 Основні поняття .....                                                          | 11 |
| 1.2 Огляд наявних методів .....                                                    | 13 |
| 1.2.1 Алгоритм табу пошуку .....                                                   | 16 |
| 1.2.2 Алгоритм імітації відпалу .....                                              | 17 |
| 1.2.3 Генетичний алгоритм та його модифікації .....                                | 19 |
| 1.2.4 Нейронні мережі .....                                                        | 23 |
| 1.3 Порівняльний аналіз наявних методів вирішення завдання .....                   | 26 |
| 1.4 Огляд наявних систем .....                                                     | 28 |
| 1.4.1 Огляд програми aSc TimeTables .....                                          | 28 |
| 1.4.2 Огляд програми UniTime .....                                                 | 29 |
| 1.4.3 Огляд програми FET .....                                                     | 30 |
| 1.5 Порівняльна характеристики наявних систем .....                                | 31 |
| 1.6 Формулювання завдання до роботи .....                                          | 33 |
| 1.7 Висновок .....                                                                 | 35 |
| 2 Матеріали і методи .....                                                         | 36 |
| 2.1 Вибір мови програмування .....                                                 | 36 |
| 2.1.1 JavaScript .....                                                             | 36 |
| 2.1.2 Python .....                                                                 | 37 |
| 2.1.3 C++ .....                                                                    | 38 |
| 2.2 Порівняння мов програмування та обґрунтування вибору архітектури системи ..... | 39 |
| 2.3 Вибір та порівняння фреймворків для реалізації клієнтської частини ..          | 42 |
| 2.4 Вибір та порівняння протоколів взаємодії між клієнтом та сервером ..           | 44 |
| 2.5 Розробка моделі взаємодії користувача з системою .....                         | 45 |

|                                                                           |     |
|---------------------------------------------------------------------------|-----|
| 2.6 Висновок .....                                                        | 51  |
| 3 Опис програми.....                                                      | 52  |
| 3.1 Структура програмного забезпечення .....                              | 52  |
| 3.2 Функціонування програмного забезпечення.....                          | 62  |
| 3.3 Опис програмної реалізації .....                                      | 66  |
| 3.4 Висновок .....                                                        | 72  |
| 4 Експлуатація, тестування та експериментальне дослідження програми ..... | 73  |
| 4.1 Опис застосування програми.....                                       | 73  |
| 4.2 Налаштування доступу до вебзастосунку .....                           | 74  |
| 4.3 Опис роботи з програмою .....                                         | 74  |
| 4.4 Результати тестування програмного забезпечення .....                  | 86  |
| 4.5 Висновок .....                                                        | 89  |
| Висновки .....                                                            | 91  |
| Перелік джерел посилання .....                                            | 93  |
| Додаток А Технічне завдання .....                                         | 97  |
| Додаток Б Опис програмного продукту .....                                 | 103 |
| Додаток В Текст програми.....                                             | 109 |
| Додаток Г Слайди презентації .....                                        | 131 |

**ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК**

|          |                                                     |
|----------|-----------------------------------------------------|
| API      | – Application Programming Interface;                |
| DNS      | – Domain Name System;                               |
| DOM      | – Document Object Model;                            |
| HTML     | – HyperText Markup Language;                        |
| HTTP     | – HyperText Transfer Protocol;                      |
| IP       | – Internet Protocol;                                |
| JS       | – JavaScript;                                       |
| JSON-RPC | – JavaScript Object Notation Remote Procedure Call; |
| JSX      | – JavaScript Extensible Markup Language;            |
| LSTM     | – Long Short-Term Memory;                           |
| REST     | – Representational State Transfer;                  |
| RNN      | – Recurrent Neural Network;                         |
| SOAP     | – Simple Object Access Protocol;                    |
| SPA      | – Single-Page Application;                          |
| UML      | – Unified Modeling Language;                        |
| URL      | – Uniform Resource Locator;                         |
| XLSX     | – Microsoft Excel Open XML Spreadsheet;             |
| XML      | – Extensible Markup Language;                       |
| БД       | – база даних;                                       |
| ВНЗ      | – вищий навчальний заклад;                          |
| ГА       | – генетичний алгоритм;                              |
| НМ       | – нейронна мережа.                                  |

## ВСТУП

У сучасному світі, коли обсяги інформації швидко зростають, а технології розвиваються швидкими кроками, автоматизація стає ключовим фактором у підвищенні продуктивності роботи в будь-якій галузі людської діяльності. Вища освіта не є винятком і також потребує сучасних інструментів для управління та організації різних процесів.

Університети та інші вищі навчальні заклади стикаються з викликом створення розкладу занять для тисячі студентів та викладачів щороку. Це є складним та трудомістким процесом, який вимагає великої кількості ресурсів, часу та знань з планування. Однак, з розвитком технологій та появою нових інструментів для автоматизації процесів, можна спростити та оптимізувати цей процес [1].

Складання розкладу для ВНЗ є важливим елементом організації навчального процесу. Крім того, при складанні розкладу потрібно враховувати багато факторів, таких як: кількість студентів, кількість викладачів, заняття в аудиторії, вікна між заняттями, а також побажання студентів та викладачів. Тому, розробка системи для автоматизованого складання розкладу може допомогти у полегшенні цієї задачі та забезпечити більш ефективне використання ресурсів закладу [1].

На цей час існує багато різних систем, що допомагають у розв'язанні цієї задачі. Однак, деякі з цих систем мають свої недоліки, такі як складність у використанні або невідповідність потребам користувачів. Тому, актуальним завданням є розробка нової системи для автоматизованого складання розкладу занять для ВНЗ, яка б враховувала всі потреби користувачів та дозволяла знизити час і зусилля, необхідні для складання розкладу.

Об'єктом дослідження є процес складання розкладу для університету, включаючи процес планування, підготовки та оновлення розкладу.

Предметом дослідження є розробка системи для автоматизованого складання розкладу, яка забезпечує оптимальне використання ресурсів,

враховує потреб студентів та викладачів, дозволяє знизити час та уникнути кількості помилок.

Метою роботи є розробка та впровадження системи для автоматизованого складання розкладу для університету з метою забезпечення оптимального та точного складання розкладу занять, зменшення часу на підготовку до навчального семестру та уникнення помилок при створенні розкладу навчального процесу.

Для досягнення поставленої мети у роботі вирішувалися такі завдання:

- аналіз процесу складання розкладу занять університету;
- розробка вимог до системи для автоматизованого складання розкладу занять;
- розробка архітектури та функціональних можливостей системи;
- розробка та реалізація програмного забезпечення системи;
- проведення тестування та оцінка ефективності системи.

Методика дослідження охоплює аналіз літературних джерел, збір вимог до системи від користувачів, розробку прототипу системи, її тестування та оцінку ефективності.

## **1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ**

### **1.1 Основні поняття**

Розклад занять – це план, який відображає розподіл занять протягом семестру. Розклад містить наступну інформацію: день тижня, номер пари, тип пари, інформацію про заняття та місце проведення (аудиторія) [2].

Заняття – це часовий інтервал, під час якого викладач проводить навчальний процес зі студентами з метою вивчення певної дисципліни. Заняття є елементами розкладу та проводяться протягом семестру. Заняття ведеться викладачем для певної групи. До занять входять різні типи викладання: лекції, практичні, лабораторні. В залежності від типу заняття, можуть мати рекомендації до проведення в конкретних, спеціалізованих аудиторіях [2].

Аудиторія – це місце, яке обладнане для проведення занять різних типів. Залежно від призначення аудиторії та кількості груп, які відвідують аудиторію, можуть різнитися обладнання та розміри. Зазвичай, аудиторії закріплюються за кафедрами [2].

Кафедра – це структурний підрозділ вищого навчального закладу, який займається викладанням групи наукових дисциплін та проводить наукові дослідження в цій галузі. Кафедри мають перелік спеціальностей, за якими навчають студентів, та складаються з викладачів, аспірантів, науковців тощо [2].

Спеціальність – це напрям підготовки фахівців для здобуття знань з певної галузі науки та професійних навичок. Спеціальності закріплюються за кафедрами, мають назву та код. Кожна спеціальність передбачає вивчення набору дисциплін студентами для здобуття кваліфікації у відповідній галузі [2].

Дисципліна – це предмет, який вивчається в рамках певної спеціальності та проводиться в певному семестрі [2].

Група – це набір студентів, які навчаються разом в рамках одного курсу та спеціальності [2].

Викладач – це особа, яка має вищу освіту та професійні знання в певній галузі науки, займається викладанням у навчальному закладі [2].

Складання розкладу занять – це процес, який передбачає розподіл занять на певні часові проміжки згідно з навчальним планом, з урахуванням обмежень, для досягнення оптимального розподілу ресурсів. Навчальний план зазначається в бланку доручень викладачам кафедри. Цей бланк містить переліки: дисциплін, які будуть проводитися в семестрі, груп, які будуть відвідувати заняття, відповідні види занять, пропозиції щодо аудиторії для проведення даного виду занять, метод контролю знань, викладачів, кількість годин проведення на тиждень [2].

При складанні розкладу необхідно врахувати умови, які обмежують проведення заняття у певний проміжок часу:

- наявність вікон між парами;
- розподіл занять протягом академічного тижня;
- побажання на проведення заняття в конкретний час та в аудиторії;
- доступність аудиторії;
- врахування кількості студентів та типу занять при виборі аудиторії;
- доступність викладача;
- доступність групи [2].

Існує декілька способів розв'язання задачі складання розкладу, таких як перебір усіх варіантів або використання методів оптимізації. Однак ці методи є складними для розв'язання задачі вручну. Диспетчера з відділу планування складають розклади інтуїтивно з використанням досвіду. Проте, через ймовірність помилок та час, необхідний для створення, доцільно розробити систему автоматизованого складання розкладу [2].

## 1.2 Огляд наявних методів

Задача складання розкладу полягає в створенні оптимального розкладу занять з урахуванням різних обмежень та вимог. Ця задача належить до класу NP-повних задач, тому що являє собою комбінаторну оптимізаційну задачу, в якій кількість рішень зростає зі збільшенням розміру вхідних даних, обмежень та параметрів [1].

Цільова функція для цієї задачі повинна відображати основні критерії для оптимізації. Основними критеріями оцінки для розкладу є:

- мінімізація кількості вікон між парами: зменшити простій часу викладачів та груп;
- рівномірне розподілення навантаження протягом тижня: рівномірно розподілити навантаження на викладачів та студентів для підвищення їх працездатності;
- мінімізація конфліктів та накладань занять: передбачає уникнення перекриття між заняттями, які мають спільних груп, викладачів, або проводяться в одній аудиторії;
- врахування пропозицій щодо розкладу: розклад повинен бути складений таким чином, щоб максимально задовольнити потреби студентів або викладачів [3].

Отже, цільова функція для задачі складання розкладу – це сума штрафних функцій, які показують наскільки порушується обмеження при складанні. Штрафні функції можуть мати різну вагомість, в залежності від того, наскільки важливим є дотримання певного обмеження, Ведемо наступні позначення:

- $n$  – кількість груп;
- $m$  – кількість викладачів;
- $k$  – кількість аудиторій;
- $L$  – кількість занять;
- $D$  – кількість навчальних днів;

- $U$  – максимальна кількість пар в день;
- $Y$  – тип пари;
- $G = [g_1, g_2, \dots, g_n]$  – розклад відносно груп;
- $T = [t_1, t_2, \dots, t_m]$  – розклад відносно викладачів;
- $A = [a_1, a_2, \dots, a_k]$  – розклад відносно аудиторій;
- $S = [s_1, s_2, \dots, s_l]$  – загальний розклад;
- $X$  – розклад окремої групи, викладача або аудиторій.

$f_1(x)$  – функція, яка розраховує вікна між парами та має відповідний ваговий коефіцієнт  $k_1$ . Функція розраховує різницю між наступною парою та попередньою в межах одного дня. В залежності від типу пар будуть розраховуватися наступні різниці: між двома загальними парами, або між загальною та чисельником чи знаменником, між двома парами по чисельнику або знаменнику. В загальному виду цю функцію можна описати наступним чином (1.1):

$$f_1(x) = \sum_{d=1}^D \sum_{u=1}^U \sum_{y=1}^Y x_{d,u,y} - x_{d,u-1,y}, \quad (1.1)$$

$f_2(x)$  – функція, яка розраховує накладання занять та має відповідний ваговий коефіцієнт  $k_2$ . Ця функція перевіряє чи не накладаються заняття в певній групі, аудиторії, викладача. Накладанням вважається, коли два та більше заняття заплановані на один й той самий день тижня та номер пари, або різні заняття проводяться в одній і тій самій аудиторії. Слід зазначити, що якщо два заняття проводяться в одному часовому проміжку, тип пари відповідно чисельник та знаменник, то це не вважається накладанням. Математична модель функції (1.2):

$$f_2(x) = k_2 \cdot \sum_{i=1}^L \sum_{j=i+1}^L w_2(x_i, x_j), \quad (1.2)$$

Функція  $w_2(x_i, x_j)$  булева функція, яка може бути описана наступним чином (1.3):

$$w_2(x_i, x_j) = \begin{cases} 1, \text{ якщо } i = j \\ 0, \text{ інакше} \end{cases}, \quad (1.3)$$

$f_3(x)$  – функція, яка розраховує дотримання рекомендацій щодо часу проведення заняття та має відповідний ваговий коефіцієнт  $k_3$ .

Цільова функція буде виглядати наступним чином (1.4):

$$F(S) = \sum_{i=1}^2 \sum_{j=1}^n k_i \cdot f_i(g_j) + \sum_{i=1}^2 \sum_{j=1}^m k_i \cdot f_i(t_j) + \sum_{j=1}^k k_2 \cdot f_2(a_j) + \sum_{j=1}^l k_3 \cdot f_3(s_j), \quad (1.4)$$

За допомогою вагових коефіцієнтів ми можемо змінювати важливість дотримання відповідного обмеження. Якщо значення коефіцієнта буде високим, то в пріоритеті алгоритм буде мінімізувати саме це обмеження.

Існують різні методи розв’язування задачі складання розкладу. Усі підходи до створення алгоритмів можна розділити на три групи [4].

Перша група включає повний або частковий перебір варіантів, їх аналіз та вибір найкращого. Ці алгоритми можна використовувати для складання розкладу з невеликою кількістю ресурсів. Проте, зі збільшенням вхідних даних застосування цих алгоритмів стає неприйнятним через зростання варіантів експоненційно [4].

Алгоритми другого виду належать до класу евристичних алгоритмів. Вони не враховують всі можливі наслідки своїх рішень для майбутніх призначень. Наприклад, якщо алгоритм призначає заняття на певний час і місце, він не знає, чи це призведе до конфлікту з іншими групами або викладачами, які також потребують цього часу і місця [4].

Алгоритми третього виду використовують методи покрокового конструювання. Основна ідея полягає в тому, що алгоритм будує розклад

поступово, з кроку в крок, додаючи нове заняття й на кожному кроці вибирає найбільш прийнятний варіант. Цей підхід дозволяє зменшити кількість переборів варіантів і забезпечити прийнятну якість розкладу [4].

### 1.2.1 Алгоритм табу пошуку

Алгоритм Табу-пошуку – це метаевристичний алгоритм, який використовується в задачах оптимізації для пошуку найкращого рішення. В основі алгоритму лежить ідея табу, тобто заборони на певні дії в процесі пошуку рішення, які вже були обрані. Алгоритм зберігає список заборонених дій, щоб уникнути попадання в локальні мінімуми [5]-[6].

Параметрами алгоритму є:

- табу-список – список заборонених дій, які не можуть бути вибрані протягом певного періоду часу;
- довжина табу-списку;
- термін дії табу – це кількість ітерацій, протягом яких хід залишається в списку табу;
- кількість сусідніх розв’язків, яку алгоритм будує на кожній ітерації;
- загальна кількість ітерацій [5]-[6].

Основні кроки алгоритму:

- генерація початкового рішення;
- створення набору сусідніх розв’язків для поточного рішення;
- вибір кращого рішення: обирається рішення, яке краще за поточне та яке не міститься в списку табу;
- оновлення списку заборонених дій: видалення ходів, термін дії яких минув, та додавання нових;
- перевірка критеріїв завершення: якщо вони виконані, то виконання алгоритму закінчується, інакше перехід до наступної ітерації [5]-[6].

Алгоритм може використовуватися для задачі складання розкладу занять. На початку, розклад занять може бути згенерований випадковим

чином. Табу-список можна використовувати для зберігання днів тижня та номерів пар, які були використані для певного заняття. Будування сусідніх розкладів можна шляхом зміни параметрів випадкового заняття. Оцінка розкладу буде відбуватися за допомогою визначеною цільової функції.

Перевагами даного алгоритму є:

- алгоритм може бути модифікований для розв’язання різних задач;
- може знаходити оптимальні рішення для складних задач з великою кількістю обмежень;
- завдяки використанню табу-списку, може уникати повторень вже пройдених шляхів [5]-[6].

Недоліками алгоритму Табу-пошуку є:

- залежність від початкового рішення;
- багато параметрів: для ефективної роботи алгоритму необхідно налаштування цих параметрів [5]-[6].

### 1.2.2 Алгоритм імітації відпалу

Алгоритм імітації відпалу – це метаевристичний алгоритм оптимізації, який моделює процес нагрівання та охолодження речовини, з метою досягнення глобального оптимуму цільової функції [7].

Ідея алгоритму симуляції відпалу полягає в тому, що він здатний виходити з локального оптимуму й переходити в околиці кращого рішення з певною ймовірністю, яка зменшується з часом. Це дозволяє алгоритму досліджувати більшу область пошуку рішень та знайти глобальний мінімум [7].

Основними параметрами алгоритму є:

- а) початкова висока температура;
- б) кінцева температура (точка зупину);
- в) коефіцієнт зниження температури;

г) коефіцієнт повільного охолодження;

д) функція охолодження – визначає швидкість зменшення температури, поділяється на 3 основних типи, кожне з якого знижує параметр з різною швидкістю:

1) лінійне (1.5):

$$T = t - \alpha, \quad (1.5)$$

де  $t$  – поточна температура;

$\alpha$  – коефіцієнт зниження температури;

$T$  – знижена температура;

2) геометричне (1.6):

$$T = t \cdot \alpha, \quad (1.6)$$

де  $t$  – поточна температура;

$\alpha$  – коефіцієнт зниження температури;

$T$  – знижена температура;

3) повільне зниження (1.7):

$$T = \frac{t}{1 - \beta \cdot t}, \quad (1.7)$$

де  $t$  – поточна температура;

$\beta$  – коефіцієнт повільного охолодження;

$T$  – знижена температура [7].

Загальними кроками алгоритму імітації відпаду є:

- генерація початкового стану;
- генерація нового стану на основі поточного;
- обчислення різниці між енергією поточного стану та нового стану;

- якщо різниця менша або дорівнює нулю, приймаємо новий стан як поточний;
- якщо різниця більша за нуль, обчислюємо ймовірність переходу до нового стану за допомогою формули Больцмана-Гіббса:

$$P = e^{-\frac{\Delta E}{k \cdot t}}, \quad (1.8)$$

де  $e$  – експонента;

$\Delta E$  – різниця енергії станів;

$t$  – температура;

$k$  – постійна Больцмана;

$P$  – ймовірність переходу стану.

- генеруємо випадкове число та приймаємо новий стан з ймовірністю, визначеною на попередньому кроці;
- повторюємо кроки, доки система не досягне заданої кінцевої температури [8].

Переваги алгоритму імітації відпалу це простота реалізації та забезпечення оптимальні рішення широкого кола проблем. Недоліками цього алгоритму є для складних задач процес симуляції може зайняти багато часу, потреба у налаштуванні параметрів, відсутність гарантії знаходження глобальної оптимальності [8].

Алгоритм імітації відпалу можна застосувати для задачі складання розкладу занять, щоб отримати оптимальне рішення яке врахує всі потреби та обмеження [8].

### 1.2.3 Генетичний алгоритм та його модифікації

Генетичний алгоритм – це евристичний алгоритм пошуку, який базується на ідеях природного відбору та генетики. Алгоритм

використовується для створення якісних розв'язань проблем оптимізації та пошуку, спираючись на такі біологічні оператори, як мутація, кросовер і відбір [9].

Генетичні алгоритми моделюють процес природного відбору, тобто ті види, які можуть адаптуватися до змін у навколишньому середовищі, здатні виживати, розмножуватися та переходити до наступного покоління [9].

При описі генетичного алгоритму використовуються визначення, запозичені з генетики. Поняття ГА описані нижче.

Популяція – множина потенційних рішень, яка складається з індивідів [10]-[11].

Індивід – це окреме рішення, яке складається з генів [10]-[11].

Ген – це представлення потенційного рішення у вигляді масиву значень, які можуть бути змінені та комбіновані [10]-[11].

Функція пристосованості (або фітнес-функція) – функція, яка оцінює якість рішення індивіда [10]-[11].

Схрещування – оператор генетичного алгоритму, що застосовується для генерації нових рішень для наступної популяції. Цей процес аналогічний схрещенню в біології, його принцип полягає в тому, що вибирається два батьківських рішення, вони обмінюються генами з метою створення нового нащадку. Зазвичай схрещування відбувається за такими правилами:

- одноточкове схрещування – вибирається випадковий пункт перетину та гени до цього пункту обмінюються між двома батьками;
- двоточкове схрещення – вибираються дві позиції, потім гени між цими частинами обмінюються генами між батьками;
- рівномірне схрещення – вибір випадкових генів в батьківських індивідів та обмін між ними з деякою ймовірністю;
- інші види схрещення – це схрещення різними способами або схрещення модифіковані для певних задач [10]-[11].

Мутація – це процес випадкової зміни гена індивіда з певною ймовірністю. Використання мутації допомагає забезпечити різноманітність у популяції та зменшити попадання в локальні оптимуми [10]-[11].

Селекція – процес відбору кращих кандидатів з популяції для формування наступного покоління. Це дозволяє алгоритму рухатись до кращих розв’язків та забезпечувати збільшення якості популяції з покоління в покоління. Існують наступні методи селекції:

- метод рулеткою – відбір кандидатів наступного покоління на основі ймовірностей, які відповідають їх пристосованості;
- метод турніром – випадковим чином обираються пара рішень, з них вибирається найкраще;
- метод селекції за рангом – метод вибору на основі їх ранжування за пристосованістю [10]-[11].

Параметрами генетичного алгоритму є:

- розмір популяції;
- кількість ітерацій;
- ймовірність мутації;
- ймовірність схрещування;
- ймовірність мутації гена.

Основні кроки класичного ГА наведені нижчі:

- а) ініціалізація популяції;
- б) ініціалізація параметрів алгоритму;
- в) початок основного циклу:

- 1) перевірка умов зупинки алгоритму;
- 2) схрещування;
- 3) мутація;
- 4) оцінка пристосованості;
- 5) селекція;

- г) відбір найкращого розв’язку [10].

Переваги генетичного алгоритму:

- знаходження оптимальних рішень для складних задач, які не можуть бути розв’язані традиційними методами;
- може застосовуватися для різних типів проблем та задач, що робить його універсальним;
- алгоритм може бути модифікованим та адаптованим для конкретних задач;
- знаходження оптимального рішення швидше, ніж інші алгоритми оптимізації;
- шукають найкраще рішення у широкому просторі можливих рішень [12].

Недоліки ГА:

- складність налаштування параметрів, через це залежить швидкість та ефективність роботи алгоритму;
- можуть потрапити в локальні мінімум чи максимум;
- вимагають значних обчислювальних ресурсів та часу [12].

Генетичний алгоритм є досить гнучким і може бути модифікованим для розв’язання певних задач.

Елітарний ГА – це модифікація стандартного ГА, процес побудови нової популяції полягає в тому, щоб дозволити найкращим індивідам переходити із покоління в покоління в незмінному вигляді. Це гарантує, що якість рішення не погіршуватиметься через покоління [9].

Паралельна реалізація стандартного ГА – ідея використання паралельних обчислень для прискорення пошуку оптимального рішення. У паралельній реалізації кожен індивід виконує такі операції, як схрещування, мутації, розрахунку фітнес значення паралельно на кожному вузлі процесора. Індивіди згуртовуються для селекції та формування наступного покоління [9].

Острівна модель ГА – модель паралельного генетичного алгоритму, в якому популяція розбивається на підпопуляції однакового розміру (острови), які можуть використовувати різні налаштування параметрів та розвиваються окремо. Для опрацювання окремого острова застосовується окремий

процесор, але острова можуть обмінюватися між собою індивідами за допомогою міграції для більшої збіжності. В цьому випадку кращі особи мігрують з одного острова на інший через певну кількість ітерацій. Одним з недоліків цього методу є те, що він потребує значних обчислювальних ресурсів [13].

#### 1.2.4 Нейронні мережі

Штучні нейронні мережі – це математичні моделі, які імітують роботу людського мозку. Вони здатні розв’язувати складні задачі шляхом визначення залежностей між великою кількістю вхідних та вихідних даних. Нейронні мережі можуть бути застосовані для розв’язання задач у різних галузях: класифікації, регресії, кластеризації, розпізнавання образів, обробка мови, аналіз даних тощо [14]-[16].

Нейронна мережа складається з нейронів та зв’язків між ними. Кожен нейрон приймає вхідні сигнали від інших нейронів чи зовнішнього середовища, обробляє їх та генерує вихідний сигнал. Зв’язки між нейронами відповідають за передачу сигналів між ними, а їх ваги визначають внесок кожного нейрона у формування вихідного сигналу [14]-[16].

Математична модель нейронної мережі базується на використанні лінійної алгебри та математичного аналізу. Кожен нейрон у мережі можна описати наступною формулою (1.9):

$$z = \sum_{i=1}^n w_i \cdot x_i + b, \quad (1.9)$$

де  $w_i$  – вага, яка присвоюється вхідному сигналу  $x_i$ ;

$x_i$  – значення вхідного сигналу;

$n$  – кількість вхідних сигналів;

$b$  – зсув [14]-[16].

Далі до результату  $z$  застосовується функція активації, яка дозволяє моделювати нелінійні залежності між вхідними та вихідними даними (1.10):

$$y = f(z), \quad (1.10)$$

де  $y$  – вихідний сигнал нейрона;

$f$  – функція активації [14]-[16].

Нейрони групують у шари. Кожен шар містить групу нейронів, які взаємодіють з нейронами попереднього та наступного шарів. Виділяють наступні види шарів:

- вхідний шар: приймає вхідні дані;
- приховані шари: обробляють вхідні дані та передають результати до наступного шару;
- вихідний шар: видає результати роботи мережі [14]-[16].

Нейронні мережі можна класифікувати за різними ознаками. Основні класифікації НМ:

- а) за кількістю шарів:
  - 1) одношарові;
  - 2) багатошарові;
- б) за способом навчання:
  - 1) НМ з учителем;
  - 2) НМ без учителя;
  - 3) НМ з підсиленням;
- в) за типом зв'язків між нейронами:
  - 1) повнозв'язні;
  - 2) неповнозв'язні;
  - 3) ієрархічно інтерактивні;
  - 4) слабозв'язні;
- г) за типом функції активації:
  - 1) лінійна;

- 2) порогова;
- 3) гіперболічний тангенс;
- 4) сигмоїдна логістична;
- 5) косинусоїдальна;
- 6) синусоїдальна [14]-[16].

Найбільш відомі архітектури НМ:

- прямолінійні нейронні мережі;
- згорткові нейронні мережі;
- рекурентна нейронна мережа [14]-[16].

Довга короткочасна пам'ять (Long short-term memor) – це вид рекурентних нейронних мереж, який дозволяє моделювати та працювати з послідовностями даних довільної довжини та здатна зберігати інформацію про довготривалі залежності вхідних даних [17].

Стандартні RNN можуть мати проблеми зі збереженням довготривалих залежностей, які виникають між далекими моментами часу в послідовності вхідних даних. LSTM розв'язує цю проблему за допомогою спеціальної структури, яка дозволяє мережі зберігати попередні значення стану та використовувати їх при обчисленні наступних значень [17].

LSTM, використовуючи спеціальний механізм, дозволяє контролювати потік інформації через різні шари мережі. Він складається з трьох основних блоків: блоку забування, блоку входу та блоку виводу [17].

Блок забування відповідає за розв'язання проблеми зниклого градієнту, дозволяючи мережі забувати непотрібну інформацію. Блок входу контролює, яка нова інформація повинна бути додана до внутрішнього стану мережі, а блок виводу контролює, яку інформацію мережа повинна виводити [17].

LSTM використовується в багатьох задачах обробки природних мов (наприклад, у машинному перекладі, розпізнаванні мовлення), а також в задачах прогнозування часових рядів та аналізу даних [17].

До переваг нейронних мереж можна віднести:

- нейронні мережі можуть навчатися на основі вхідних даних та знаходити неявні взаємозв'язки між ними;
- НМ можуть адаптуватися до змін у вхідних даних;
- можливість ефективно працювати з великими обсягами даних;
- можуть бути застосовані до розв'язання складних задач;
- нейронні мережі можуть робити узагальнення
- нейронні мережі можуть обробляти нечітку та неповну інформацію;
- можуть обробляти багато даних одночасно та забезпечувати високу продуктивність [14]-[16].

Недоліки НМ:

- необхідність великої кількості даних для навчання;
- високі вимоги до обчислювальних ресурсів;
- наявність великої кількості параметрів та складність їх налаштування;
- складність вибору архітектури
- вразливість до шуму даних;
- НМ можуть бути складними для інтерпретації та розуміння [14]-[16].

### **1.3 Порівняльний аналіз наявних методів вирішення завдання**

Ознайомившись з описаними вище алгоритмами було проведено порівняльний аналіз досліджених методів для автоматичного складання розкладу занять (табл. 1.1).

Таблиця 1.1 – Порівняльна характеристика наявних методів для автоматичного складання розкладу занять

| Характеристика<br>/ Назва<br>алгоритму                          | Табу-пошук | Імітація<br>відпалу | Генетичний<br>алгоритм | Штучна<br>нейронна<br>мережа<br>(LSTM) |
|-----------------------------------------------------------------|------------|---------------------|------------------------|----------------------------------------|
| 1                                                               | 2          | 3                   | 4                      | 5                                      |
| Швидкість<br>знаходження<br>оптимального<br>рішення             | Середня    | Середня             | Середня                | Швидка                                 |
| Потрібність у<br>вибірці даних<br>для навчання та<br>тренування | Ні         | Ні                  | Ні                     | Так                                    |
| Залежність від<br>початкового<br>наближення                     | Висока     | Висока              | Середня                | Низька                                 |
| Складність<br>реалізації                                        | Низька     | Низька              | Середня                | Висока                                 |
| Потреба в<br>налаштуванні<br>параметрів                         | Так        | Так                 | Так                    | Так                                    |
| Складність<br>налаштування<br>параметрів                        | Середня    | Низька              | Середня                | Висока                                 |

Кінець таблиці 1.1

| 1                                         | 2  | 3  | 4   | 5   |
|-------------------------------------------|----|----|-----|-----|
| Потреба в великих обчислювальних ресурсах | Ні | Ні | Так | Так |

Під час порівняння алгоритмів для задачі складання розкладу занять було визначено, що генетичний алгоритм є більш прийнятним в порівнянні з табу-пошуком, імітацією відпалу та LSTM. Генетичний алгоритм має кращу здатність до знаходження глобального оптимуму, що є важливою характеристикою для складання розкладу, оскільки це гарантує мінімізацію заданих обмежень. Крім того, цей алгоритм є більш гнучким і адаптивним до змін вхідних даних. Однак, використання ГА може бути часово і ресурсомістким процесом, особливо при великій кількості обмежень та розміру популяцій, що додає до розрахунків.

## **1.4 Огляд наявних систем**

### **1.4.1 Огляд програми aSc TimeTables**

Програма aSc TimeTables – це програмне забезпечення для автоматичного створення розкладів у навчальних закладах (рис. 1.1). Основні функції та характеристики програми включають:

- автоматична генерація розкладу на основі заданих обмежень та вимог;
- ручне редагування розкладу з можливістю перетягування занять, зміни часу проведення та інших параметрів;
- візуалізація розкладу: розклад можна відображати у вигляді таблиці, графіку або різноманітних звітів та статистики;

- імпорт даних;
- експорт даних в різних форматах;
- можливість створювати розклад для різних типів навчальних закладів;
- гнучкі налаштування параметрів розкладу [18].

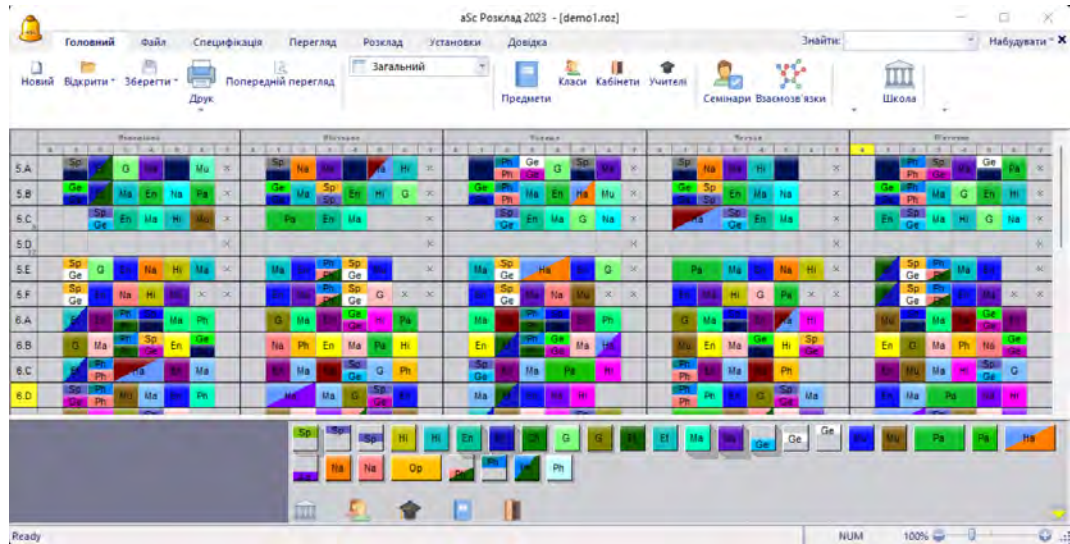


Рисунок 1.1 – Вікно програми aSc TimeTables [18]

Перевагами програми є широкі можливості налаштування, автоматичне створення розкладу та зручний інтерфейс користувача. Недоліком може бути висока вартість ліцензії та необхідність додаткового навчання для користувачів в роботі з програмним забезпеченням.

### 1.4.2 Огляд програми UniTime

UniTime – це відкрите програмне забезпечення для складання розкладу занять (рис. 1.2). Unitime пропонує ряд інструментів для автоматичного створення розкладу занять для викладачів та студентів [19].

Основні функції системи UniTime:

- автоматичне створення розкладу;

- має алгоритми для виявлення конфліктів та пропонує способи їх вирішення;

- можливість завантаження розкладу в різних форматах;

- відстежування прогресу створення розкладу [19].

Переваги UniTime:

- відкрите програмне забезпечення, що значить, що програма є безкоштовна;

- ручне редагування розкладу.

Недоліки:

- вимагає значних налаштувань.

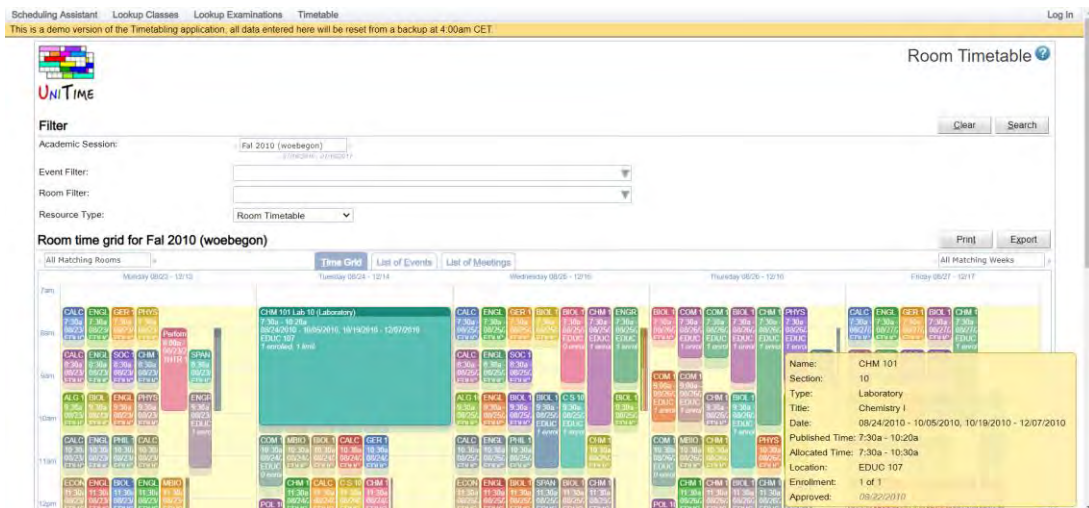


Рисунок 1.2 – Інтерфейс програми UniTime [19]

### 1.4.3 Огляд програми FET

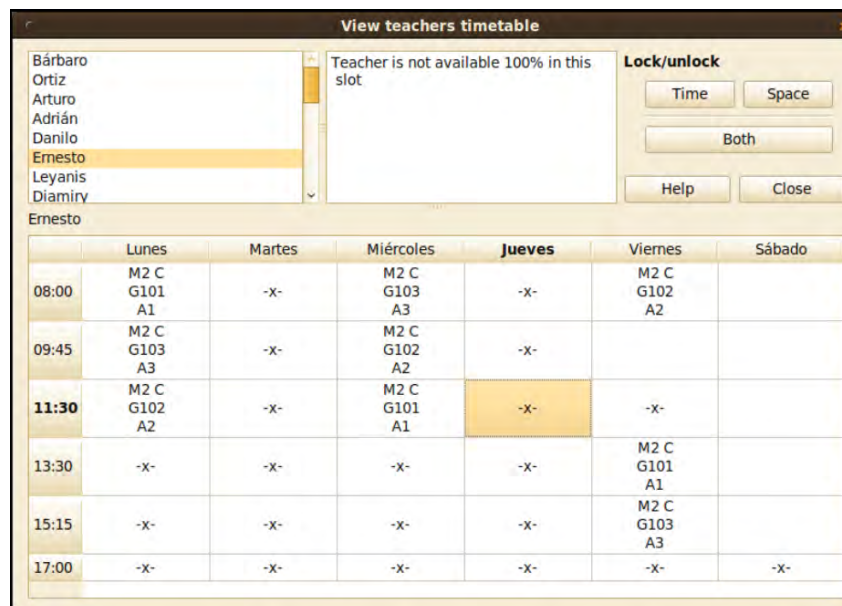
FET (Free Timetabling Software) – це безплатна програма для автоматичного складання розкладу (рис. 1.3) для різних типів навчальних закладів: шкіл, коледжів, університетів [20].

Основними характеристиками FET:

- підтримка різних типів занять;

- можливість задати обмеження на кількість занять у день, на розклад з використанням певних аудиторій, вчителів, груп тощо;
- можливість імпортувати та експортувати розклад у різних форматах;
- формування звітів та статистики про розклад;
- автоматичне розміщення занять [20].

Переваги FET включають безкоштовне використання програми та відкритий код, що дозволяє модифікувати програми під потреби користувачів. З недоліків можна зазначити, що точність складання розкладу може бути нижчою, порівняно з програмами, які мають ліцензію.



|       | Lunes              | Martes | Miércoles          | Jueves | Viernes            | Sábado |
|-------|--------------------|--------|--------------------|--------|--------------------|--------|
| 08:00 | M2 C<br>G101<br>A1 | -x-    | M2 C<br>G103<br>A3 | -x-    | M2 C<br>G102<br>A2 |        |
| 09:45 | M2 C<br>G103<br>A3 | -x-    | M2 C<br>G102<br>A2 | -x-    |                    |        |
| 11:30 | M2 C<br>G102<br>A2 | -x-    | M2 C<br>G101<br>A1 | -x-    | -x-                |        |
| 13:30 | -x-                | -x-    | -x-                | -x-    | M2 C<br>G101<br>A1 |        |
| 15:15 | -x-                | -x-    | -x-                | -x-    | M2 C<br>G103<br>A3 |        |
| 17:00 | -x-                | -x-    | -x-                | -x-    | -x-                | -x-    |

Рисунок 1.3 – Інтерфейс програми FET [20]

## 1.5 Порівняльна характеристики наявних систем

Ознайомившись з декількома програмами для автоматизованого складання розкладу, було створено їх порівняльну характеристику (табл. 1.2).

Таблиця 1.2 – Порівняльна характеристика наявних програмних засобів

| Характеристика                          | FET                      | UniTime          | aSc TimeTables            |
|-----------------------------------------|--------------------------|------------------|---------------------------|
| 1                                       | 2                        | 3                | 4                         |
| Відкритий код                           | Так                      | Так              | Ні                        |
| Операційна система                      | Кросплатформна           | Кросплатформна   | Windows                   |
| Підтримка різних мов                    | Багатомовний             | Багатомовний     | Багатомовний              |
| Наявність ліцензії                      | Ні                       | Ні               | Так                       |
| Адаптивність до різних типів установ    | Так                      | Так              | Так                       |
| Тип програми                            | Десктопна                | Вебдодаток       | Десктопна                 |
| Автоматичне створення розкладу          | Так                      | Так              | Так                       |
| Можливість ручного редагування розкладу | Так                      | Так              | Так                       |
| Можливість імпорту/експорту даних       | Так                      | Так              | Так                       |
| Формат збереження даних                 | Файли з розширенням .fet | База даних MySQL | Формат з розширенням .tsm |

Кінець таблиці 1.2

| 1                              | 2   | 3   | 4   |
|--------------------------------|-----|-----|-----|
| Редагування даних              | Так | Так | Так |
| Фільтрація та сортування даних | Так | Так | Так |
| Формування звітів              | Так | Так | Так |

### 1.6 Формулювання завдання до роботи

Метою роботи є дослідження та реалізація автоматизованої системи для складання розкладу занять для університету.

Вхідними даними для системи будуть:

- дані про аудиторії: назва аудиторії, тип, місткість, закріплені кафедри;
- дані про кафедри: назва кафедри та її скорочена назва;
- дані про спеціальності: назва спеціальності, код, кафедра;
- дані про дисципліни: назва, спеціальність, семестр в якому викладається дисципліна;
- дані про групи: назва групи, кількість студентів, поточний семестр, спеціальність;
- дані про викладачів: ПІБ викладача та кафедра, на якій він працює;
- дані про заняття: дисципліна, тип заняття, кількість занять на тиждень, викладачі, групи, рекомендовані аудиторії, рекомендований час проведення.

Вихідними даними буде оптимальний розклад занять, складений для обраної кафедри, в якому будуть враховуватися рекомендації до проведення та мінімізовано наступні параметри:

- кількість накладань занять для груп, вчителів та аудиторій;
- кількість вікон між заняття для груп та вчителів.

Під час аналізу наявних методів розв'язання проблеми, було встановлено, що найбільш підходящим алгоритмом для складання оптимального розкладу занять є генетичний алгоритм. Тому було вирішено розробити адаптований та модифікований генетичний алгоритм для розв'язання поставленої задачі.

Алгоритм для складання розкладу повинен мінімізувати значення визначеної цільової функції (1.4). Роботи алгоритму можна представити у вигляді наступної формули (1.11):

$$F(S) \rightarrow \min, \quad (1.11)$$

де  $S$  – розклад занять;

$F(S)$  – цільова функція.

Для оцінювання якості результатів роботи алгоритму буде використовуватися описана вище цільова функція, яка показує кількість порушень заданих обмежень.

До системи висуваються наступні вимоги:

- перегляд даних;
- редагування та видалення вхідних даних;
- ручне та автоматичне ведення вхідних даних;
- фільтрація даних;
- вибір та запуск алгоритму;
- налаштування параметрів алгоритму;
- порівняння роботи алгоритму з різними параметрами.

### **1.7 Висновок**

У цьому розділі було проведено аналіз предметної області, визначено основні поняття, пов'язані з предметною областю й об'єктом дослідження та розробки. Було розглянуто проблему складання розкладу занять та методи її вирішення.

Було досліджено деякі методи для складання оптимального розкладу занять та проведено порівняльний аналіз цих методів, який показав перевагу використання генетичного алгоритму.

Проведено порівняльну характеристику наявних програмних засобів та було виявлено основні вимоги до власної системи.

Було сформульовано та поставлено завдання до роботи.

## 2 МАТЕРІАЛИ І МЕТОДИ

### 2.1 Вибір мови програмування

#### 2.1.1 JavaScript

JavaScript – це мова програмування, яка використовується для створення динамічних інтерактивних вебсайтів та додатків. Вона була розроблена в 1995 році Бренданом Айком компанією Netscape Communications і стала однією з найпоширеніших мов програмування в Інтернеті [21].

JavaScript використовується в різних областях: веброзробка, розробка мобільних додатків, розробка настільних додатків, створення розширень для браузерів, розробка серверних додатків та інше [21].

Найпоширеніше застосування JavaScript пов'язане з веброзробкою. JavaScript є основною мовою програмування на стороні клієнта вебсайтів та додатків, які працюють у браузері. Він використовується для динамічного змінення сторінок, анімації, валідації форм, розробки інтерактивних інтерфейсів, створення односторінкових та складних вебзастосунків [21].

JavaScript також використовується для розробки серверних додатків з використанням фреймворків, таких як Node.js. Це дозволяє розробникам створювати повністю функціональні вебдодатки, які можуть обробляти запити користувачів, працювати з базами даних та здійснювати інші операції на сервері [21].

Перевагами JavaScript є:

- може використовуватися як на стороні клієнта, так і на стороні сервера, що дозволяє розробляти повноцінні веб-додатки;
- має простий синтаксис, що дозволяє швидко вивчити мову програмування;
- має багато бібліотек та фреймворків, що спрощують розробку [22].

Недоліками є:

- різні браузери можуть різним чином інтерпретувати JavaScript, що може призвести до проблем зі сумісністю;

- може використовуватися для зловмисних дій, таких як внесення змін до сторінок або виконання шкідливого коду;
- може знижувати продуктивність в деяких випадках через те, що JavaScript не компілюється, а інтерпретується під час виконання [22].

### 2.1.2 Python

Python – це високорівнева мова програмування загального призначення, яка була створена у 1991 році Гвідо ван Россумом. Python має простий синтаксис та підтримує об'єктно-орієнтоване, функціональне та інші підходи до програмування. Вона широко використовується у багатьох сферах, включаючи науку про дані, розробку вебдодатків, машинне навчання, наукові дослідження, розробку ігор та багато іншого [23].

Основні сфери застосування Python:

- веброзробка;
- наукові обчислення;
- розробка ігор;
- автоматизація;
- розробка додатків.

Основними перевагами Python є:

- легкість вивчення та читання коду;
- має велику кількість сторонніх бібліотек та фреймворків;
- платформонезалежність: Python може працювати на різних операційних системах без необхідності внесення значних змін;
- відкритий код [24].

До недоліків цієї мови програмування можна віднести:

- швидкість виконання: Python є інтерпретованою мовою програмування, що може вплинути на швидкість виконання програм;

- обмеження потоків виконання: Python має обмеження щодо паралельного виконання декількох операцій одночасно через глобальний інтерпретатор блокування;
- потребує більше пам'яті: Python має великий обсяг пам'яті, що потрібний для виконання коду, порівняно з іншими мовами програмування [24].

### 2.1.3 C++

C++ – це мова програмування загального призначення, яка виникла як розширення мови C, і є однією з найпопулярніших мов програмування. C++ поєднує в собі можливості мови C та додає об'єктно-орієнтований підхід, що дозволяє створювати ефективні та складні програми для різних областей.

C++ має велику кількість особливостей, таких як низькорівневий доступ до пам'яті, розширені можливості об'єктно-орієнтованого програмування, підтримку шаблонів і поліморфізму. Ці особливості дозволяють розробникам створювати дуже швидкодіючі та ефективні програми.

Основні переваги C++:

- швидкодія: C++ належить до мов нижчого рівня, тобто код виконується на апаратному рівні, що дозволяє досягати високої швидкодії;
- ефективність: C++ дозволяє програмістам ефективно використовувати ресурси системи, такі як пам'ять та процесорний час, що дозволяє створювати додатки з меншим використанням ресурсів;
- розширюваність: C++ підтримує об'єктно-орієнтоване програмування, що дозволяє розробникам створювати складні програми з більшою кількістю функцій та об'єктів;
- мобільність: C++ можна використовувати на багатьох платформах, що дозволяє створювати додатки, які можна використовувати на різних ОС та апаратних платформах;

– множинність парадигм програмування [25]-[26].

Недоліки C++:

– мова C++ є складною для вивчення й компіляції;

– складність у керуванні пам'яттю;

– C++ не має автоматичного збирання сміття це означає, що програміст повинен самостійно контролювати використання пам'яті;

– довгий час розробки: розробка програм на C++ може зайняти більше часу, ніж на інших мовах програмування [25]-[26].

## 2.2 Порівняння мов програмування та обґрунтування вибору архітектури системи

Проаналізувши особливості мов програмування: C++, JavaScript, Python, було проведено порівняльний аналіз (табл. 2.1).

Таблиця 2.1 – Порівняльна характеристика мов програмування

| Критерії порівняння                            | C++      | JavaScript  | Python      |
|------------------------------------------------|----------|-------------|-------------|
| 1                                              | 2        | 3           | 4           |
| Складність синтаксису                          | Складний | Середній    | Простий     |
| Швидкість виконання                            | Швидка   | Помірна     | Повільна    |
| Керованість пам'яті                            | Вручну   | Автоматично | Автоматично |
| Ефективність виконання багатопотокових програм | Висока   | Помірна     | Помірна     |

Кінець таблиці 2.1

| 1                                     | 2                                                                  | 3                                                                 | 4                                                                      |
|---------------------------------------|--------------------------------------------------------------------|-------------------------------------------------------------------|------------------------------------------------------------------------|
| Наявність фреймворків для веброзробки | Більш призначений для системного програмування                     | Багато вебфреймворків, таких як Angular, React, Vue.js та Node.js | Доступний великий вибір вебфреймворків, таких як Django, Flask та інші |
| Кросплатформність                     | Компілятори для цієї мови доступні для багатьох операційних систем | Може працювати на будь-якому пристрої з підтримкою браузера       | Інтерпретатор Python доступний для багатьох операційних систем         |

Отже, порівнявши три мови програмування, було вирішено надати перевагу використанню JavaScript та розробити вебзастосунок. Розробка вебзастосунків мають ряд переваг:

- вебзастосунки можна запускати на будь-якому пристрої з веббраузером і підключенням до Інтернету, це дозволить забезпечити широкую доступність;
- вони не потребують встановлення на пристрої користувача, що значно спрощує їх розгортання та оновлення;
- можна легко масштабувати, додаючи нові сервери або збільшуючи потужність наявних серверів;
- вебзастосунки можна оновлювати на стороні сервера, що забезпечує високу швидкість виправлення помилок і оновлення функціонала;
- можна легко оновлювати і підтримувати без необхідності оновлення кожного клієнта окремо;

– вебзастосунки можуть працювати на різних пристроях з різними характеристиками, тому не потребують потужного обладнання для роботи [27].

Також було обрано трирівневу архітектуру вебдодатка. Трирівнева архітектура додатків – це підхід до розробки програмного забезпечення, який використовує три рівні для розподілу функцій та обов'язків (рис. 2.1) [28].



Рисунок 2.1 – Схема трирівневої архітектури

Клієнтський рівень – це верхній рівень, що забезпечує інтерфейс користувача та взаємодіє з користувачем. Цей рівень зазвичай містить вебсторінки, додатки для мобільних пристроїв, настільні додатки та інші клієнтські програми [28].

Серверний рівень – це проміжний рівень, який забезпечує логіку бізнес-процесів та обробку даних. На цьому рівні зазвичай розміщується сервер, який відповідає за отримання та обробку запитів від клієнтів [28].

Рівень даних – це найнижчий рівень, де знаходяться дані, які зберігаються та обробляються додатком. Це може бути база даних, файлова система або будь-яке інше джерело даних [28].

Основні переваги трирівневої архітектури додатків:

- розділення обов'язків: кожен рівень відповідає за конкретну частину додатка, що дозволяє краще розбити обов'язки між розробниками;
- масштабованість: кожен рівень може бути масштабований окремо, що дозволяє додатку легше адаптуватися до зростання навантаження;
- модульність: кожен рівень може бути розроблений як окремий модуль, що дозволяє повторне використання коду;

- ізоляція помилок: кожен рівень може бути тестований окремо, що дозволяє знайти та виправити помилки без впливу на решту додатка;
- безпека: логічний та рівень доступу до даних можуть бути захищені від несанкціонованого доступу, що покращує безпеку додатка [28].

Недоліки цієї архітектури:

- складність розробки;
- потрібна висока якість зв'язку між різними рівнями;
- складність тестування [28].

## **2.3 Вибір та порівняння фреймворків для реалізації клієнтської частини**

Одними з найпопулярнішими JavaScript фреймворками для реалізації клієнтської частини є: Angular, React, Vue.js.

React – це відкритий фреймворк JavaScript для створення користувацьких інтерфейсів. Розроблений Facebook, React став одним з найпопулярніших фреймворків для розробки фронтенду в останні роки [29].

Основою React є компоненти, які можна розглядати як незалежні блоки, з яких складається користувацький інтерфейс. Кожен компонент може мати свій власний стан та методи для зміни цього стану. React дозволяє писати код з використанням JSX, синтаксису, що поєднує в собі JavaScript та HTML [29].

Однією з ключових особливостей React є віртуальна об'єктна модель документа (DOM), що дозволяє зменшити кількість операцій, необхідних для оновлення сторінки при зміні даних. Замість того, щоб оновлювати кожен елемент окремо, віртуальний DOM оновлює лише змінені елементи, що забезпечує покращення продуктивності та швидкості роботи додатка [29].

Angular є одним з найпопулярніших фреймворків для розробки вебдодатків з використанням мови програмування JavaScript та TypeScript.

Розроблений компанією Google, він має велику підтримку та активну спільноту розробників [30].

Основні принципи Angular – це компонентна структура та шаблони для створення вебдодатків з використанням директив, сервісів та інших функціональних елементів. В основі Angular лежить концепція односторінкових додатків (SPA), що дозволяє створювати динамічні та багатосторінкові вебдодатки, які працюють швидко та ефективно [30].

Vue.js – це прогресивний фреймворк для створення користувацьких інтерфейсів та односторінкових додатків на JavaScript. Він був створений у 2014 році європейським розробником Еваном Ю [31].

Однією з ключових особливостей Vue.js є його простота вивчення та використання. Vue.js має інтуїтивно зрозумілий синтаксис та добре документованій API, що робить його легшим для використання [31].

Порівняльна характеристика цих фреймворків наведена у табл. 2.2.

Таблиця 2.2 – Порівняльна характеристика фреймворків JavaScript

| Характеристика                | React    | Angular  | Vue.js   |
|-------------------------------|----------|----------|----------|
| 1                             | 2        | 3        | 4        |
| Розмір                        | Середній | Великий  | Середній |
| Швидкодія                     | Швидкий  | Помірний | Швидкий  |
| Компонентна архітектура       | +        | +        | +        |
| Складність вивчення           | Середня  | Висока   | Середня  |
| Маршрутизація                 | +        | +        | +        |
| Наявність повної документації | +        | +        | +        |

Порівнявши фреймворки, було вирішено розробити клієнтську частину використовуючи React через його високу гнучкість та простоту.

## **2.4 Вибір та порівняння протоколів взаємодії між клієнтом та сервером**

Протоколи API – це набір правил та стандартів, які визначають, як взаємодіяти з API (прикладний програмний інтерфейс) для отримання інформації або виконання операцій. Вони використовуються для комунікації між клієнтською та серверною частинами. Основні протоколи API:

- REST (Representational State Transfer) – це архітектурний стиль, який використовує HTTP-протокол для передачі даних між клієнтом та сервером. REST є одним з найпопулярніших протоколів API;

- GraphQL – це мова запитів та середовище виконання запитів для серверних API, розроблений Facebook. GraphQL дозволяє клієнту запитувати тільки необхідні дані та визначати структуру відповіді;

- SOAP (Simple Object Access Protocol) – це протокол, який використовує XML-формат для обміну даними між різними пристроями та мережами;

- JSON-RPC (Remote Procedure Call) – це протокол, який використовує JSON для передачі даних та виклику функцій між різними системами [32]-[35].

Порівняння протоколів API наведено в табл. 2.3.

Таблиця 2.3 – Порівняння протоколів API

| Характеристика                   | REST            | GraphQL | SOAP    | JSON-RPC |
|----------------------------------|-----------------|---------|---------|----------|
| Складність використання          | Проста          | Середня | Складна | Проста   |
| Формат передачі даних            | XML, JSON, HTML | JSON    | XML     | JSON     |
| Можливість вказати потрібні дані | Обмежена        | Є       | Є       | Немає    |
| Швидкість передачі даних         | Висока          | Середня | Низька  | Висока   |
| Кешування                        | Так             | Так     | Так     | Ні       |

Порівнявши наступні протоколи API: REST, GraphQL, SOAP, JSON-RPC, було обрано GraphQL через те, що GraphQL дозволяє отримувати точно ті дані, які потрібні клієнту, та надає можливість кешування даних на стороні клієнта, що дозволить зменшити кількість запитів до сервера та покращити швидкість роботи додатка.

## 2.5 Розробка моделі взаємодії користувача з системою

Необхідно розробити систему для складання оптимального розкладу занять, яка дозволить ефективно організувати процес створення розкладу для навчального закладу. Це допоможе зменшити час, який витрачається на розробку розкладу, та покращити якість його планування.

Основні терміни системи:

- розклад;
- групи;
- аудиторії;

- заняття;
- викладачі;
- кафедра;
- спеціальність;
- дисципліни.

В системі передбачені наступні ролі користувачів:

- а) переглядач – неавторизований користувач, який може переглядати, фільтрувати та завантажувати розклад по групах, викладачам, аудиторіям;
- б) адміністратор – авторизований користувач, якому доступні наступні можливості:

- 1) авторизація;
- 2) керування даними про аудиторії: додавання, видалення, редагування, перегляд, фільтрація;
- 3) керування даними про кафедри: додавання, видалення, редагування, перегляд, фільтрація;
- 4) керування даними про спеціальності: додавання, видалення, редагування, перегляд, фільтрація;
- 5) керування даними про дисципліни: додавання, видалення, редагування, перегляд, фільтрація;
- 6) керування даними про заняття: додавання, видалення, редагування, перегляд, фільтрація;
- 7) керування даними про групи: додавання, видалення, редагування, перегляд, фільтрація;
- 8) керування даними про викладачів: додавання, видалення, редагування, перегляд, фільтрація;
- 9) видалення всіх даних;
- 10) завантаження даних з відомостями кафедри;
- 11) перегляд якості поточного розкладу;
- 12) редагування параметрів розкладу;

13) керування алгоритмами: запуск, редагування параметрів алгоритмів.

Для опису взаємодії між системою та користувачами була розроблена діаграма прецедентів. Діаграма прецедентів системи автоматизованого складання розкладу наведена на рис. 2.2 – 2.5.

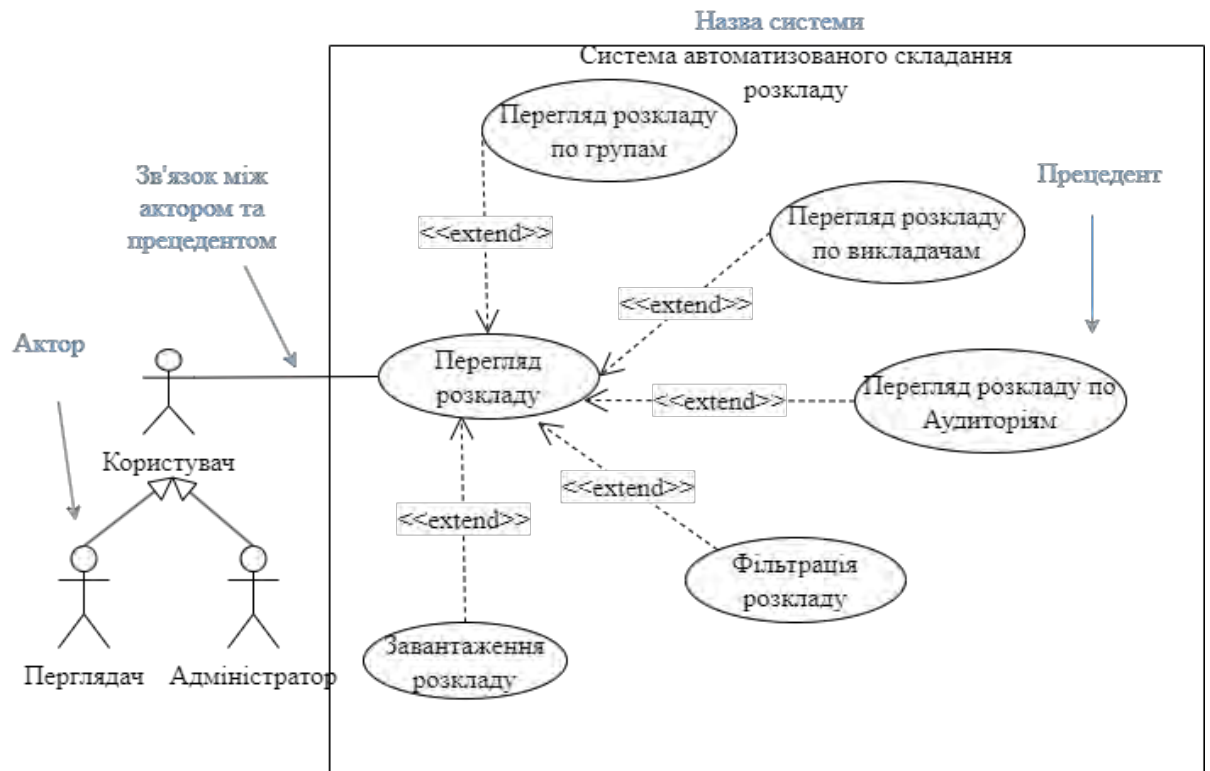


Рисунок 2.2 – Діаграма прецедентів для системи автоматизованого складання розкладу (а)

На рис. 2.2 присутні наступні актори: переглядач, адміністратор, користувач (узагальнення переглядача та адміністратора).

Користувач переглядає розклад. Додатково він може переглядати розклад по групах, викладачам, аудиторіям, фільтрувати розклад, завантажувати розклад.

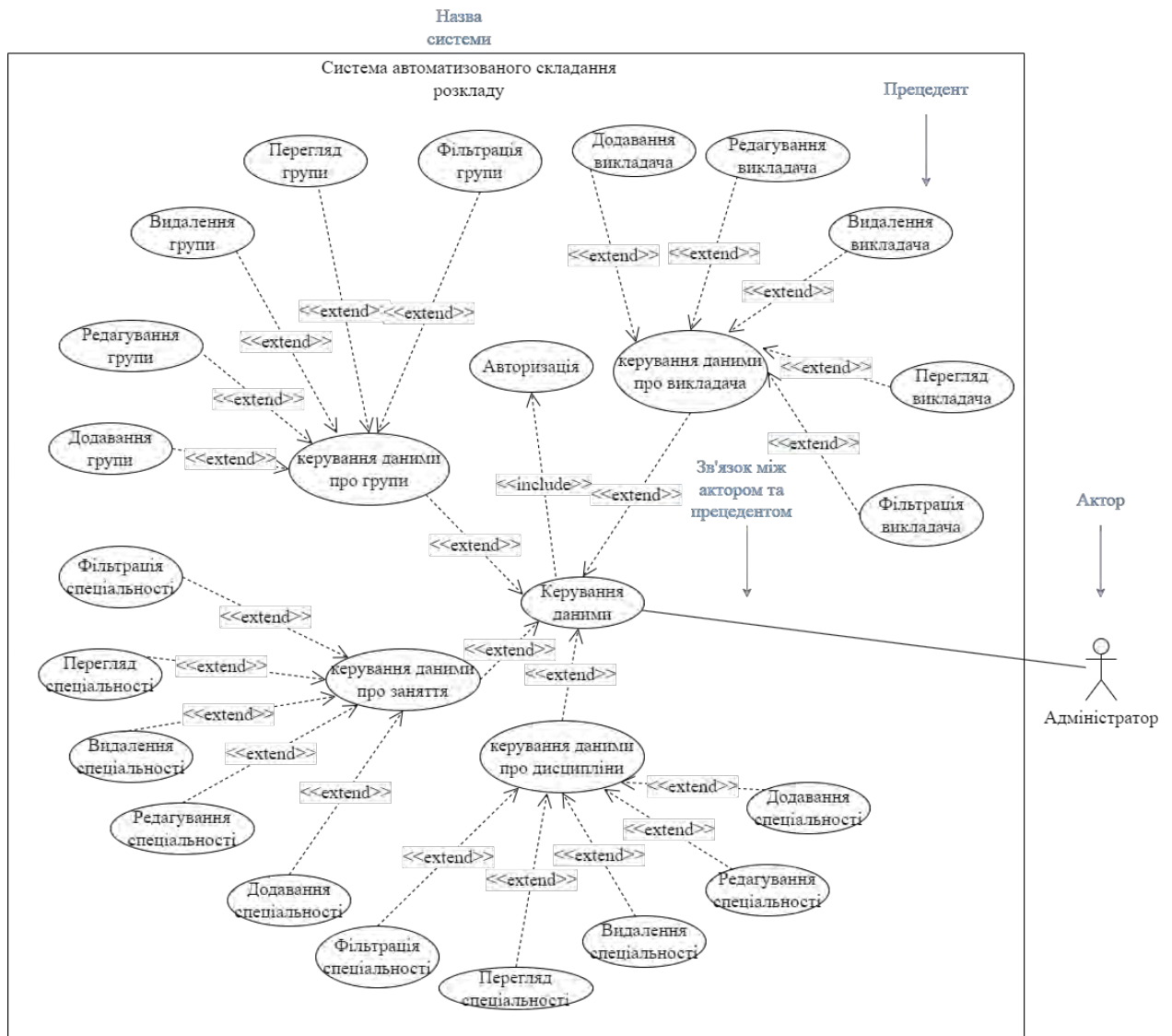


Рисунок 2.3 – Діаграма прецедентів для системи автоматизованого складання розкладу (б)

На рис. 2.3 – 2.4 присутні наступні прецеденти, з якими взаємодіє актор Адміністратор: керування даними, який вимагає виконання прецеденту Авторизація. Виконання прецеденту Керування даними може бути розширене наступними прецедентами: керування даними про дисципліни, заняття, групи, викладача, аудиторії, кафедри, спеціальності. Всі ці прецеденти включають наступні розширення: видалення, додавання, редагування, перегляд, фільтрація даних.



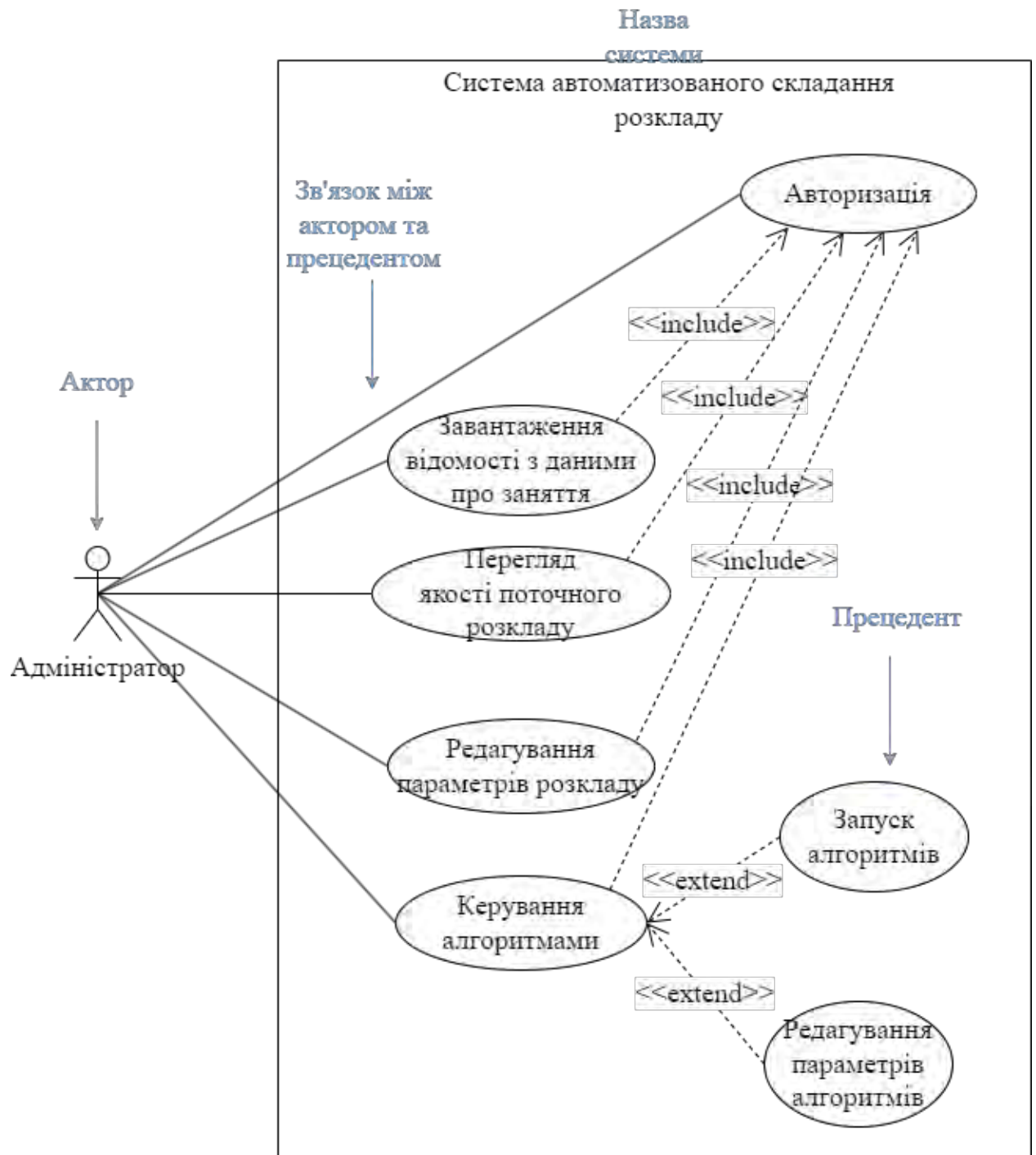


Рисунок 2.5 – Діаграма прецедентів для системи автоматизованого складання розкладу (г)

Отже, була створена модель, яка описує, як користувачі будуть взаємодіяти з системою для автоматичного складання розкладу. Ця модель була представлена у вигляді діаграми прецедентів, яка використовує

стандартну нотацію UML. Дана модель може бути використана для розробки системи для складання розкладу занять в університеті.

## **2.6 Висновок**

В цьому розділі було проведено аналіз та вибір інструментарію розробки ПП та було створено модель взаємодії користувача з програмним забезпеченням.

Було проведено порівняльний аналіз мов програмування та вирішено розробити вебзастосунок на мові JavaScript з трирівневою архітектурою.

Було проведено порівняльний аналіз фреймворків для реалізації клієнтської частини, в ході чого було обрано React через його простоту та гнучкість.

Було також проведено вибір та порівняння протоколів взаємодії між клієнтським та серверним рівнями та вирішено використовувати GraphQL через те, що він дозволяє клієнту маніпулювати даними та отримувати тільки ті, які необхідні.

Створено модель взаємодії користувача з системою, яка подана у вигляді діаграми прецедентів у нотації UML, та дозволяє розроблювати вебзастосунок для автоматизованого складання розкладу.

## 3 ОПИС ПРОГРАМИ

### 3.1 Структура програмного забезпечення

Структуру вебзастосунку автоматизованого складання та керуванням розкладом занять подано у вигляді діаграм в нотатції UML.

Для візуалізації структури класів і взаємозв'язків між ними була розроблена діаграма класів. Загальна діаграма класів вебзастосунку для складання та керуванням розкладом занять зображена на рис. 3.1 – 3.2.

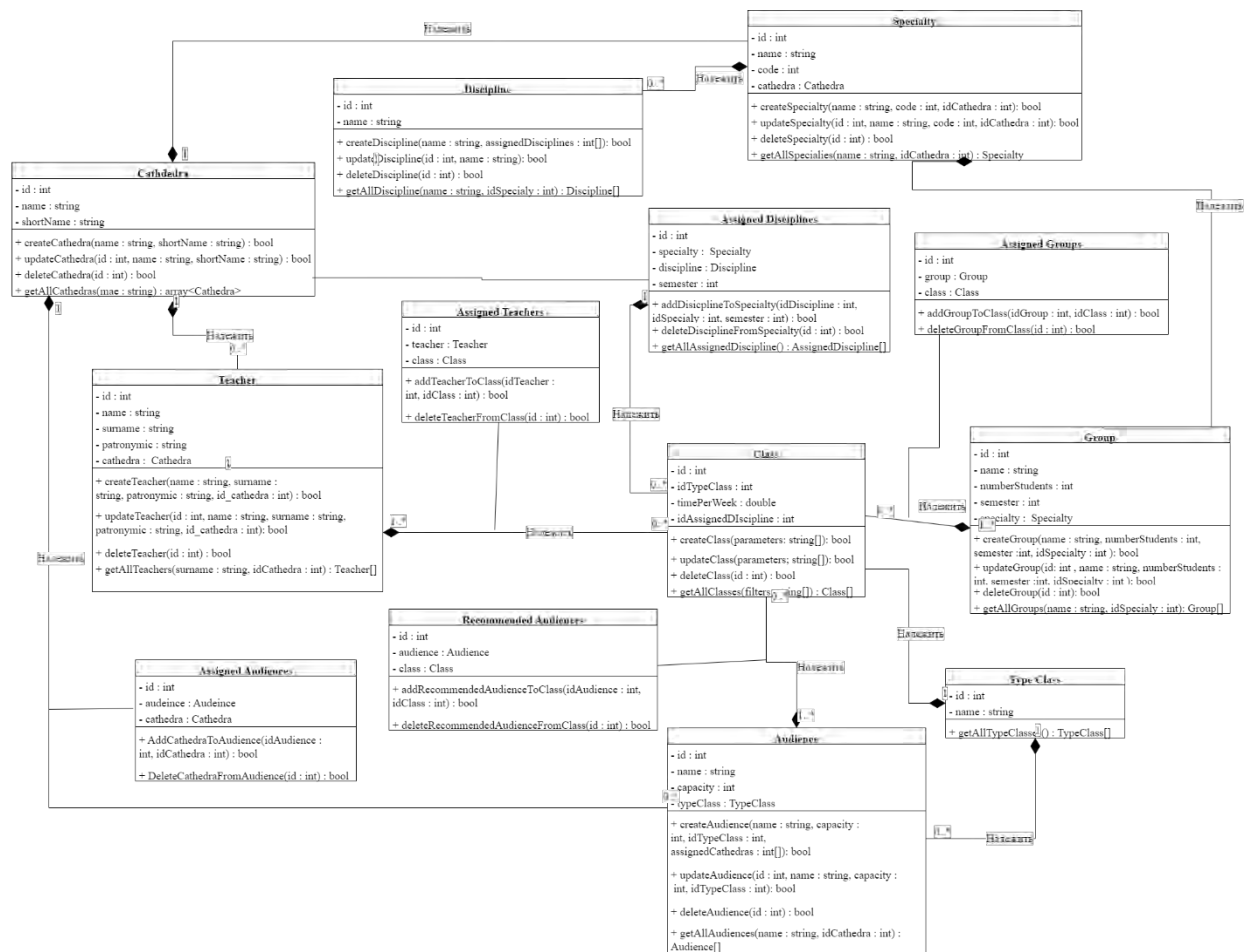


Рисунок 3.1 – Діаграма класів вебзастосунку для складання та керуванням розкладом занять (а)

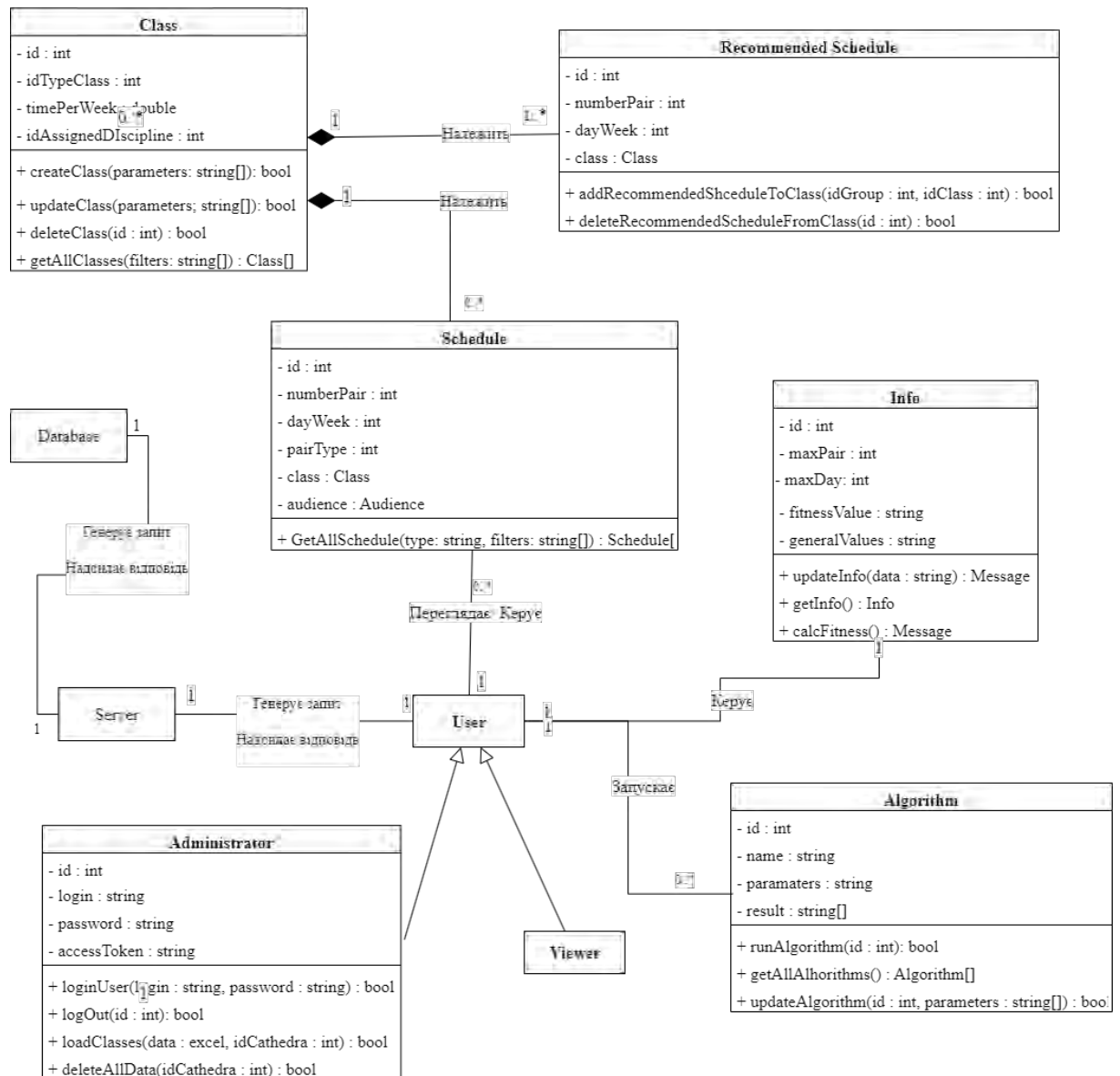


Рисунок 3.2 – Діаграма класів вебзастосунку для складання та керуванням розкладом занять (б)

На рис. 3.1 – 3.2 присутні наступні зв'язки між класами:

а) узагальнення: Користувач є узагальненням Адміністратора та Переглядача;

б) асоціація:

- 1) Користувач переглядає або керує Розкладом;
- 2) Користувач керує Інформацією;
- 3) Користувач запускає Алгоритм;
- 4) Користувач генерує запит до Сервера;

- 5) БД надсилає відповідь Сервера;
- 6) Сервер надсилає відповідь Користувачу;
- 7) Сервер генерує запит до БД;

в) композиція:

- 1) між класами Заняття та Рекомендований розклад, Розклад;
- 2) між класами Група та Заняття (через додатковий клас Закріплені групи);
- 3) між класами Викладач та Заняття (через додатковий клас Закріплені викладачі);
- 4) між класами Заняття та Дисципліна (через додатковий клас Закріплені дисципліни);
- 5) між класами Заняття та Аудиторії (через додатковий клас Рекомендовані аудиторії);
- 6) між класами Кафедра та Аудиторія (через додатковий клас Закріплені аудиторії);
- 7) між класами Група та Спеціальність;
- 8) між класами Спеціальність та Дисципліна;
- 9) між класами Кафедра та Спеціальність.

Опис класів, їх атрибутів, методів наведено в таблиці 3.1.

Таблиця 3.1 – Опис класів, їх атрибутів та методів

| Назва класу   | Атрибути                                                                                                                          | Опис методів                                                                                                                                                          |
|---------------|-----------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1             | 2                                                                                                                                 | 3                                                                                                                                                                     |
| Адміністратор | <ul style="list-style-type: none"> <li>– ідентифікатор;</li> <li>– логін;</li> <li>– пароль;</li> <li>– токен доступу.</li> </ul> | <ul style="list-style-type: none"> <li>– увійти в систему;</li> <li>– вийти із системи;</li> <li>– завантажити дані з файлу;</li> <li>– видалити всі дані.</li> </ul> |
| Переглядач    | –                                                                                                                                 | –                                                                                                                                                                     |

Продовження таблиці 3.1

| 1          | 2                                                                                                                                                                                                           | 3                                                                                                                                                                                   |
|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Користувач | –                                                                                                                                                                                                           | –                                                                                                                                                                                   |
| Алгоритм   | <ul style="list-style-type: none"> <li>– ідентифікатор;</li> <li>– назва;</li> <li>– параметри;</li> <li>– результати.</li> </ul>                                                                           | <ul style="list-style-type: none"> <li>– запустити алгоритм;</li> <li>– отримати всі наявні алгоритми;</li> <li>– оновити дані про алгоритми.</li> </ul>                            |
| Інформація | <ul style="list-style-type: none"> <li>– ідентифікатор;</li> <li>– максимальна кількість пар в день;</li> <li>– кількість навчальних днів;</li> <li>– оцінка розкладу;</li> <li>– загальні дані.</li> </ul> | <ul style="list-style-type: none"> <li>– оновити інформацію;</li> <li>– отримати всю інформацію;</li> <li>– порахувати оцінку розкладу.</li> </ul>                                  |
| Розклад    | <ul style="list-style-type: none"> <li>– ідентифікатор;</li> <li>– номер пари;</li> <li>– день тижня;</li> <li>– тип пари;</li> <li>– заняття;</li> <li>– аудиторія.</li> </ul>                             | <ul style="list-style-type: none"> <li>– отримати весь розклад по типу та за фільтрами.</li> </ul>                                                                                  |
| Заняття    | <ul style="list-style-type: none"> <li>– ідентифікатор;</li> <li>– тип заняття;</li> <li>– кількість разів проведення на тиждень;</li> <li>– дисципліна, яку викладають.</li> </ul>                         | <ul style="list-style-type: none"> <li>– створити заняття;</li> <li>– оновити заняття;</li> <li>– видалити заняття;</li> <li>– отримати список всіх занять за фільтрами.</li> </ul> |

Продовження таблиці 3.1

| 1                      | 2                                                                                                                                              | 3                                                                                                                                                                                                                          |
|------------------------|------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Рекомендований розклад | <ul style="list-style-type: none"> <li>– ідентифікатор;</li> <li>– номер пари;</li> <li>– день тижня;</li> <li>– заняття;</li> </ul>           | <ul style="list-style-type: none"> <li>– додати рекомендований розклад к заняттю;</li> <li>– видалити рекомендований розклад із заняття.</li> </ul>                                                                        |
| Групи                  | <ul style="list-style-type: none"> <li>– ідентифікатор;</li> <li>– назва;</li> <li>– кількість студентів;</li> <li>– спеціальність.</li> </ul> | <ul style="list-style-type: none"> <li>– створити групу;</li> <li>– оновити групу;</li> <li>– видалити групу;</li> <li>– отримати список всіх груп.</li> </ul>                                                             |
| Закріплені групи       | <ul style="list-style-type: none"> <li>– ідентифікатор;</li> <li>– група;</li> <li>– заняття</li> </ul>                                        | <ul style="list-style-type: none"> <li>– додати групу до заняття;</li> <li>– видалити групу з заняття.</li> </ul>                                                                                                          |
| Дисципліна             | <ul style="list-style-type: none"> <li>– ідентифікатор;</li> <li>– назва.</li> </ul>                                                           | <ul style="list-style-type: none"> <li>– створити дисципліну;</li> <li>– оновити дисципліну;</li> <li>– видалити дисципліну;</li> <li>– отримати список всіх дисциплін.</li> </ul>                                         |
| Закріплені дисципліни  | <ul style="list-style-type: none"> <li>– ідентифікатор;</li> <li>– спеціальність;</li> <li>– дисципліна;</li> <li>– семестр.</li> </ul>        | <ul style="list-style-type: none"> <li>– додати дисципліну до спеціальності в певному семестрі;</li> <li>– видалити дисципліну з спеціальності;</li> <li>– отримати всі дисципліни закріплені за спеціальністю.</li> </ul> |
| Кафедра                | <ul style="list-style-type: none"> <li>– ідентифікатор;</li> <li>– назва;</li> <li>– коротка назва.</li> </ul>                                 | <ul style="list-style-type: none"> <li>– створити кафедру;</li> <li>– оновити кафедру;</li> <li>– видалити кафедру;</li> <li>– отримати список всіх кафедр.</li> </ul>                                                     |

Кінець таблиці 3.1

| 1                       | 2                                                                                            | 3                                                                                                                           |
|-------------------------|----------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------|
| Викладач                | – ідентифікатор;<br>– ПІБ;<br>– кафедра.                                                     | – додати викладача;<br>– оновити дані про викладача;<br>– видалити викладача;<br>– отримати список всіх викладачів кафедри. |
| Закріплені викладачі    | – ідентифікатор;<br>– викладач;<br>– заняття.                                                | – додати викладача до заняття;<br>– видалити викладача з заняття.                                                           |
| Аудиторія               | – ідентифікатор;<br>– назва;<br>– місткість;<br>– тип заняття для яких призначена аудиторія. | – додати аудиторію;<br>– оновити дані про аудиторію;<br>– отримати список аудиторій.                                        |
| Закріплені аудиторії    | – ідентифікатор;<br>– аудиторія;<br>– кафедра.                                               | – додати аудиторію до кафедри;<br>– видалити аудиторію з кафедри.                                                           |
| Рекомендовані аудиторії | – ідентифікатор;<br>– аудиторія;<br>– заняття.                                               | – додати аудиторію до заняття;<br>– видалити аудиторію з заняття.                                                           |
| Тип заняття             | – ідентифікатор;<br>– назва.                                                                 | – отримати всі можливі типи занять.                                                                                         |
| Сервер                  | –                                                                                            | –                                                                                                                           |
| База даних              | –                                                                                            | –                                                                                                                           |

Для візуалізації організації класів в системі розроблено діаграму пакетів. Можна виділити наступні пакети:

- клієнтський пакет;
- серверний пакет;
- пакет бази даних.

Клієнтський пакет – це пакет, з яким взаємодіє користувач. До цього пакету належать такі класи: Алгоритм, Інформація, Розклад, Заняття, Рекомендований розклад, Групи, Закріплені групи, Дисципліна, Закріплені дисципліни, Кафедра, Викладач, Закріплені викладачі, Аудиторія, Закріплені аудиторії, Рекомендовані аудиторії, Тип заняття.

Серверний пакет – це пакет, який надає виконання сервісних функцій за запитом клієнта. До нього належать такі класи: Сервер.

До пакета бази даних належить клас База даних.

Ці пакети пов'язані між собою зв'язком Запит - Відповідь, клієнт відправляє запит на сервер, де він обробляється, і готовий результат відправляється клієнтові. На рис. 3.3 зображено загальну діаграму пакетів системи складання та керування розкладом занять.

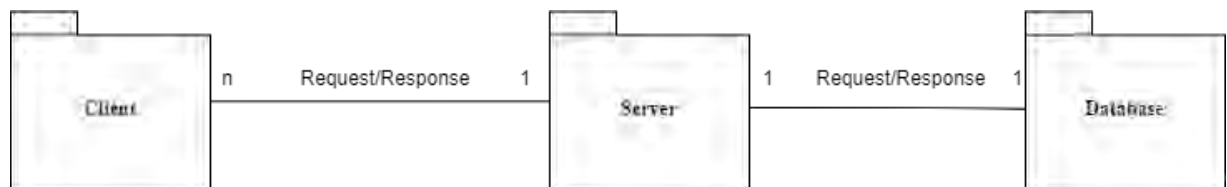


Рисунок 3.3 – Діаграма пакетів системи складання та керування розкладом занять

Для представлення архітектури системи була розроблена компонента діаграма (рис. 3.4 – 3.6).

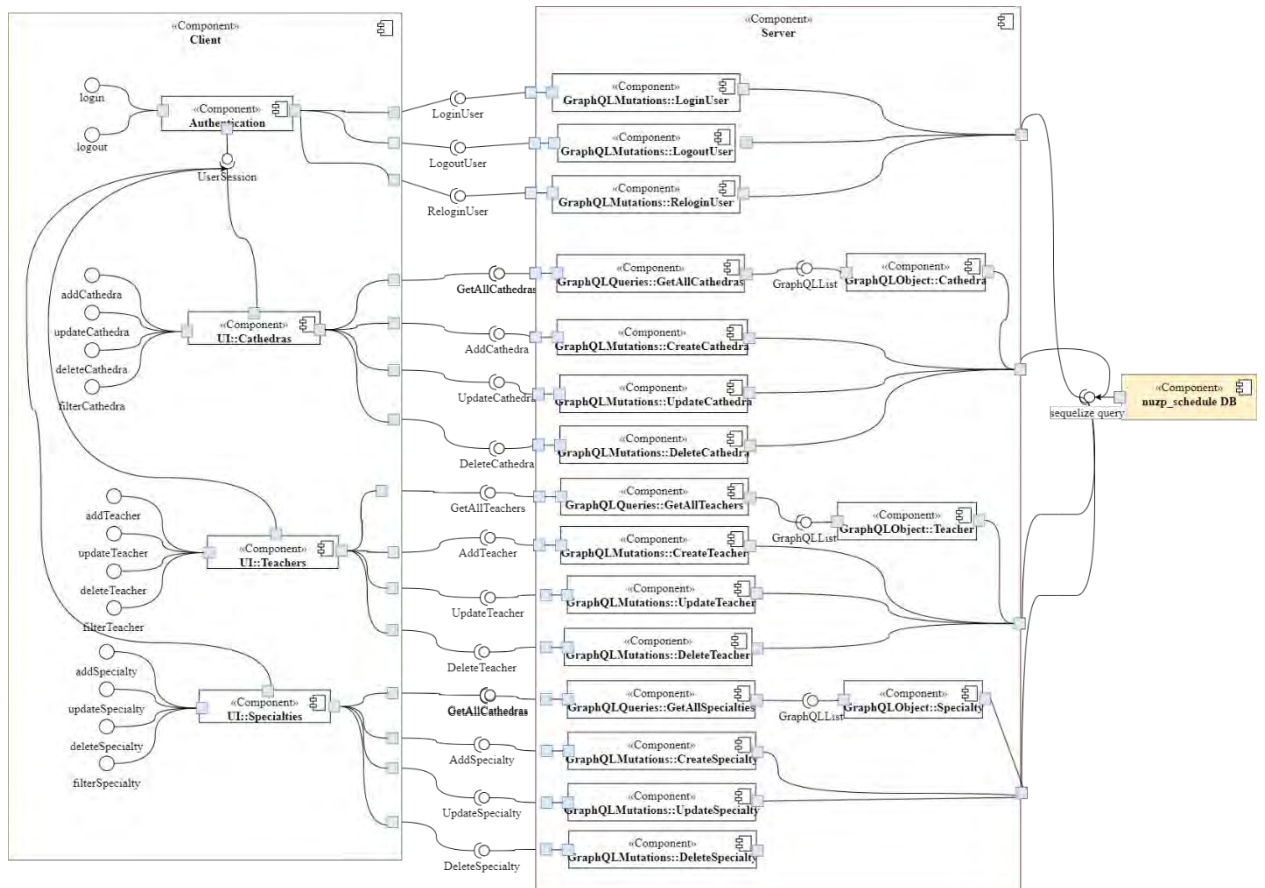


Рисунок 3.4 – Діаграма компонентів системи керування та складання розкладу (а)

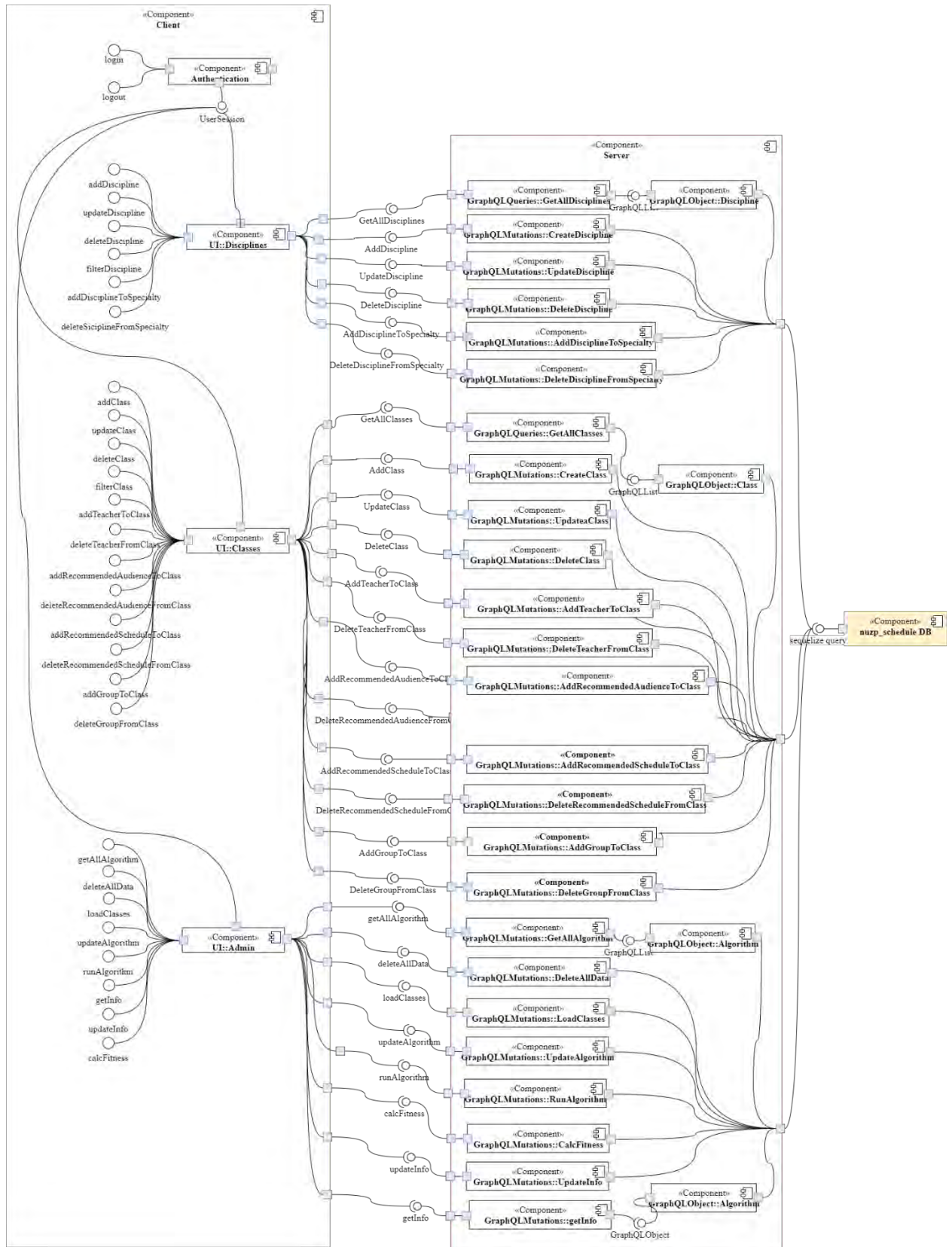


Рисунок 3.5 – Діаграма компонентів системи керування та складання розкладу (б)

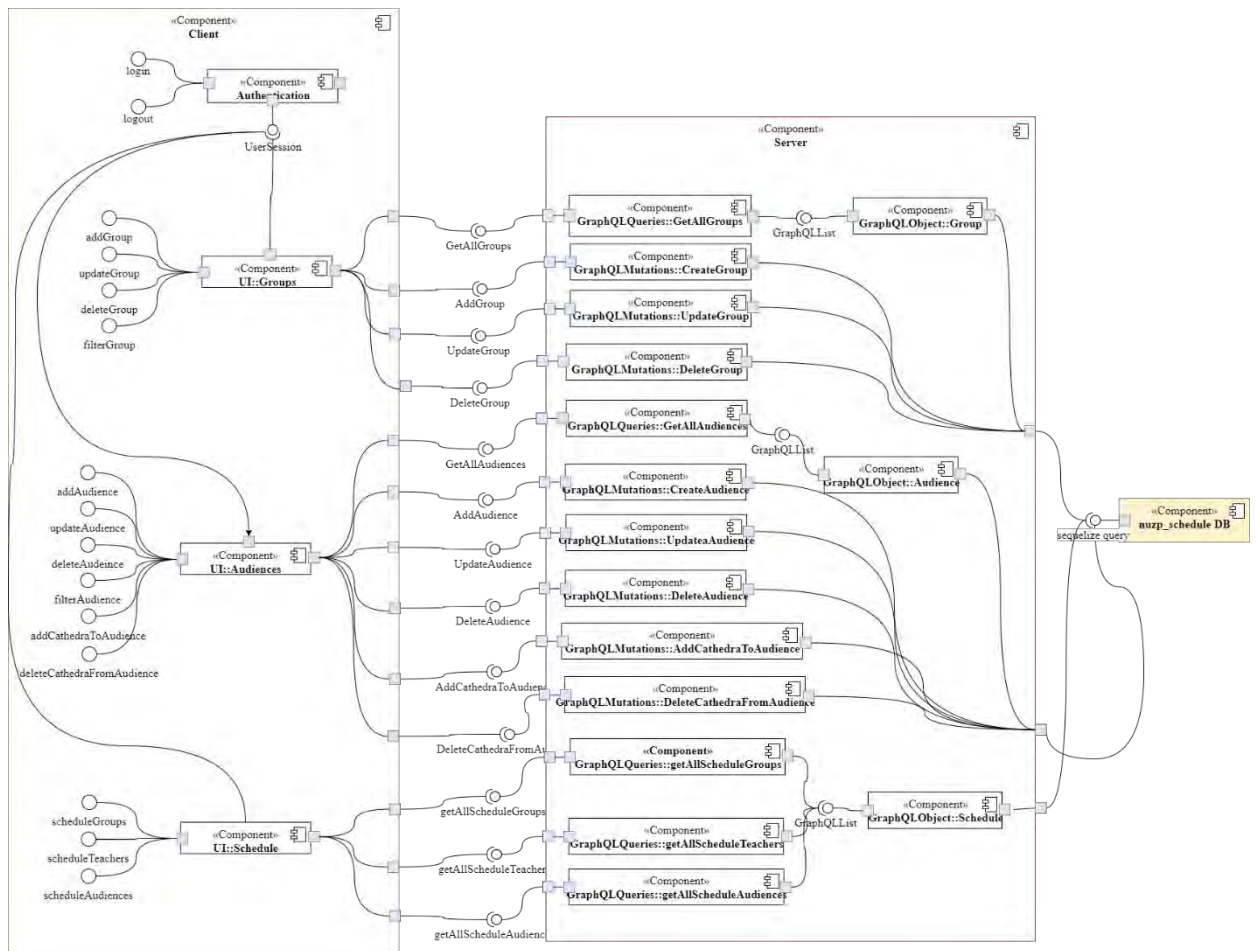


Рисунок 3.6 – Діаграма компонентів системи керування та складання розкладу (в)

На рис. 3.4 – 3.6 присутні наступні компоненти:

- Клієнт;
- Сервер;
- База даних.

Компонент Клієнт – це графічний інтерфейс, в якому реалізовано наступні компоненти: Розклад, Групи, Аудиторії, Авторизація, Кафедри, Викладачі, Спеціальності, Заняття, Адміністрування, Дисципліни. Ці компоненти надають наступні інтерфейси:

- компонент Кафедри: додавання, оновлення, видалення, фільтрування записів кафедри;

- компонент Вчителі: додавання, оновлення, видалення, фільтрування записів вчителів;
- компонент Спеціальності: додавання, оновлення, видалення, фільтрування записів спеціальностей;
- компонент Авторизація: вхід, вихід;
- компонент Дисципліни: додавання, оновлення, видалення, фільтрування записів дисципліни, додавання та видалення дисципліни до спеціальності;
- компонент Заняття: додавання, оновлення, видалення, фільтрування записів заняття, додавання та видалення до заняття викладача, групу, рекомендований розклад та аудиторію;
- компонент Адміністрування: видалення всіх даних, завантаження відомостей, оновлення параметрів алгоритмів, запуск алгоритмів, редагування загальної інформації для алгоритмів, розрахування оцінки розкладу;
- компонент Групи: додавання, оновлення, видалення, фільтрування записів групи;
- компонент Аудиторії: додавання, оновлення, видалення, фільтрування записів аудиторії, додавання та видалення аудиторії до кафедри;
- компонент Розклад: перегляд розкладу по заданим фільтрам.

Через GraphQL API компоненти взаємодіють з компонентом Сервер, який надає відповідні серверні послуги, такі як додавання, видалення, оновлення, вибір даних з БД, запуск алгоритмів.

### **3.2 Функціонування програмного забезпечення**

Набір функцій програмного забезпечення залежить від ролі користувача.

Для всіх користувачів доступно перегляд розкладу з можливістю відбирати за критеріями: вид розкладу, кафедра, за певною групою або

викладачем, або аудиторією, спеціальність та семестр для груп. Також можливо завантажити таблицю з розкладом.

Для того, щоб отримати повну функціональність додатка, потрібно авторизуватись. Слід зазначити, що не передбачена функція реєстрації для забезпечення контролю доступу та зменшення ризику помилкових даних. Отримати облікові дані можливо від адміністратора, який має доступ до БД. Алгоритм авторизації:

- введення облікових даних;
- перевірка логіна та паролю;
- якщо дані користувача підтверджені, то він отримує токен для створення сесії;
- перевірка авторизації при доступі: коли користувач закрив вкладку, а потім знову відкрив сайт, то токен перевіряється, якщо він є дійсним, то користувачу надається доступ до адміністрування.

Для збереження токена доступу він зберігається в файлі Cookie, який розташований на пристрої користувача. Коли користувач успішно авторизується на вебсайті, сервер виділяє токен та зберігає його в куках на боці користувача. При кожному наступному запиті сервер може перевірити цей токен для підтвердження авторизації та ідентифікації користувача без необхідності повторного введення облікових даних.

Після авторизації користувачу доступні наступні функції:

- ведення даних;
- редагування даних;
- видалення даних;
- запуск алгоритмів.

Якщо користувач хоче додати нові дані або редагувати наявні він заповнює форму даних та натискає відповідну кнопку. Далі введені дані перевіряються. Процес перевірки та валідації введених даних можна описати за допомогою наступного алгоритму:

- дані зчитуються з форми;

- перевірка, чи всі обов'язкові поля заповнені;
- перевірити формат введених даних для кожного поля;
- якщо будь-яка перевірка не пройшла успішно, відображається повідомлення про помилку;
- якщо всі дані успішно пройшли перевірку та валідацію, то вони відправляються на сервер.

Для уникнення введення помилкових даних були застосовані для текстових полів спадні списки з множинним вибором, окрім полів для ведення власних назв, та поля для ведення числових даних з обмеженням.

Центральним елементом системи є розклад занять. Діаграма станів для класу Розклад зображена на рис. 3.7.



Рисунок 3.7 – Діаграма станів для класу Розклад

На рис. 3.7 розклад може знаходитись в наступних станах:

- старий розклад, той який був до запуску алгоритму для складання;
- початковий розклад, той який з'являється в процесі складання;
- новий, той який замінює старий розклад.

Для візуалізації алгоритму запуску складання розкладу були розроблені діаграма послідовностей та кооперації. Загалом алгоритм запуску складання розкладу можна описати наступним чином:

- користувач обирає алгоритм, налаштовує параметри та запускає його;

- сервер приймає запит на запуск та викликає обраний метод для генерації розкладу;
- після генерація, сервер замінює старий розклад на новий та відправляє повідомлення про результати складання.

Діаграма послідовностей процесу запуску алгоритму зображена на рис. 3.8.

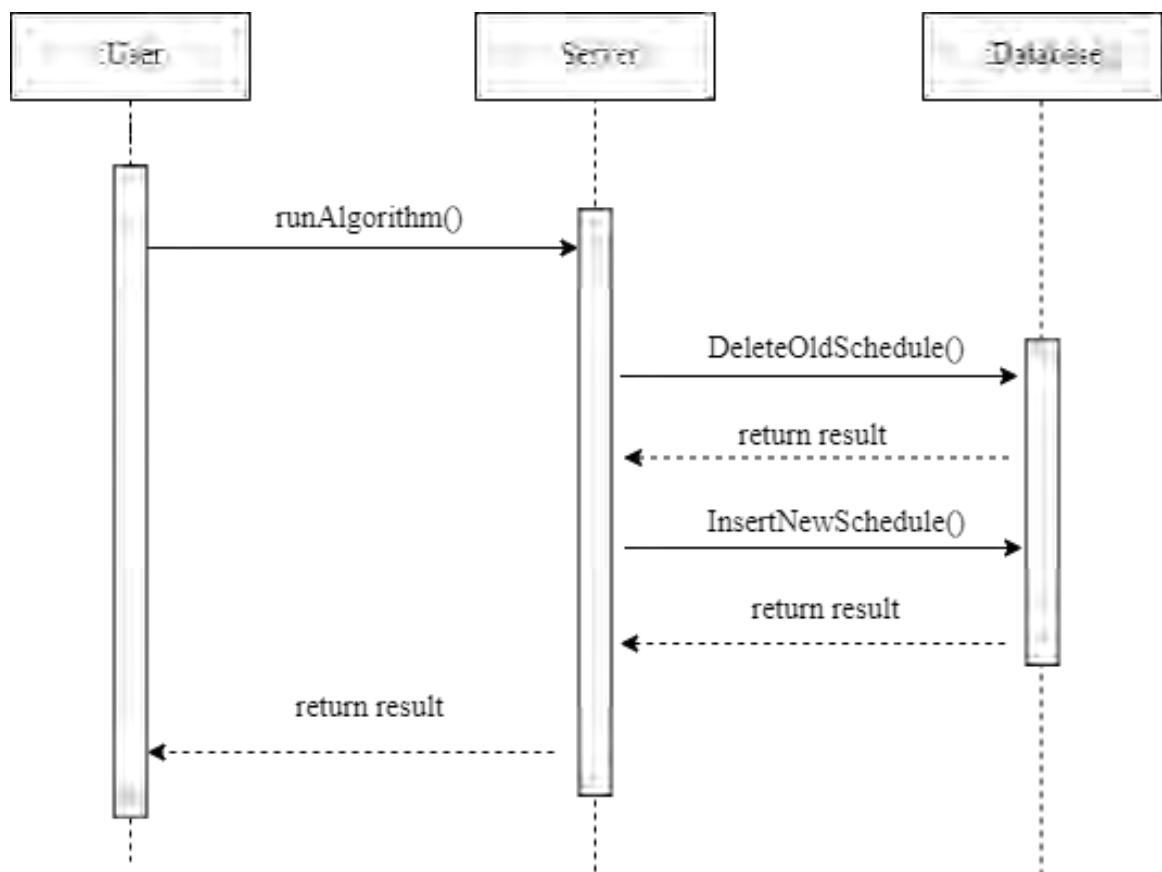


Рисунок 3.8 – Діаграма послідовностей процесу запуску алгоритму

Діаграма кооперації процесу запуску алгоритму зображена на рис 3.9.

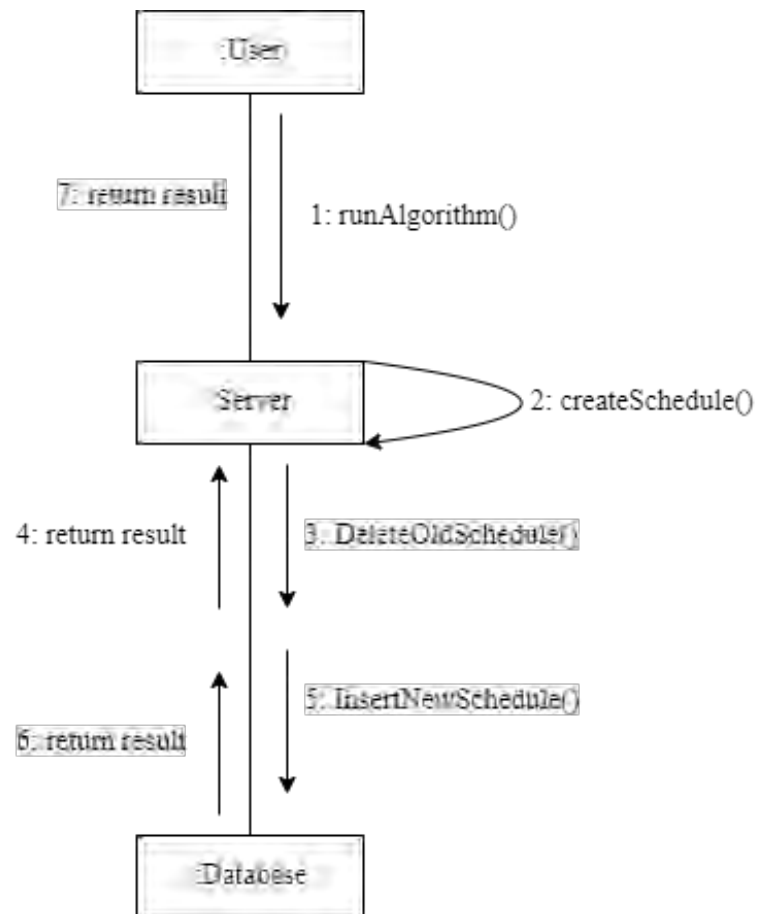


Рисунок 3.9 – Діаграма кооперації процесу запуску алгоритму

### 3.3 Опис програмної реалізації

Фізична структура проекту наведена на рис. 3.10.

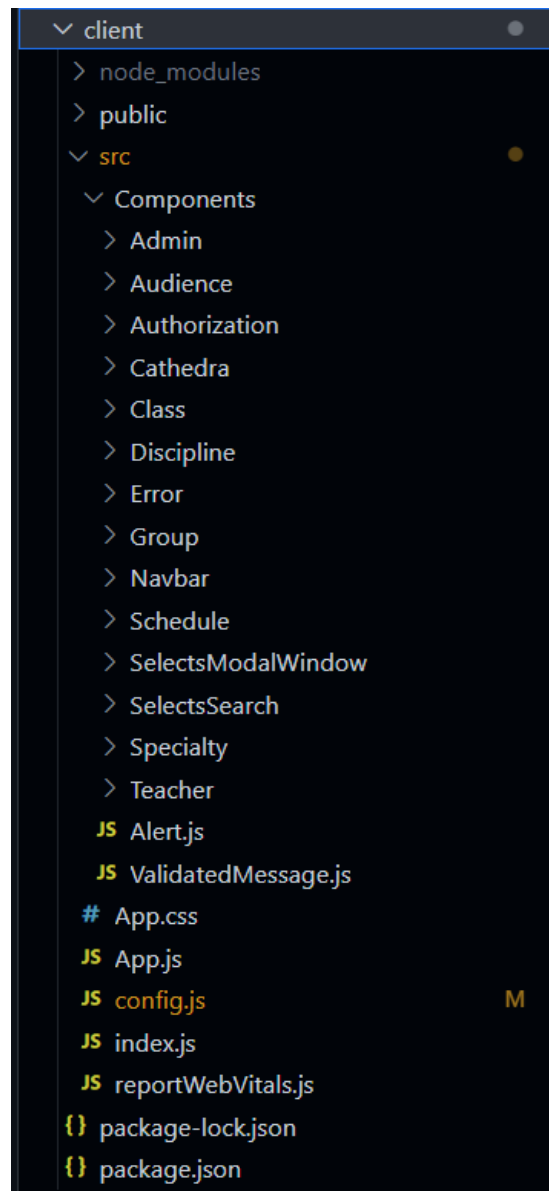


Рисунок 3.10 – Фізична структура проєкту

Директорія «src» – це головна директорія проєкту, де знаходиться весь вихідний код React-додатку.

В директорії «public» зберігається статичний файл `index.html` – це кореневий HTML-файл, містить кореневий DOM-елемент з ідентифікатором `root`, до якого React-додаток монтується за допомогою методу `ReactDOM.render()`.

`index.js` є вхідним файлом, з якого починається виконання застосунку, де імпортується основний компонент `App.js` і монтується до DOM. Також файл містить налаштування зв'язку з сервером через GraphQL, використовуючи

бібліотеку Apollo Client. Apollo Client – це бібліотека для керування станом та виконання запитів у вебзастосунках, які використовують GraphQL.

Хост та порт, куди будуть спрямовані запити, зазначені в config.js.

Всі компоненти в проєкті реалізовані як React-компоненти – класи, що розширюють базовий клас React.Component. Вони використовуються для реалізації сторінок або окремих вебелементів, які мають внутрішній стан (state) та здатні до роботи з методами життєвого циклу React. Класові компоненти мають метод render(), який повертає JSX-код для відображення компонента.

App.js – це основний компонент, де реалізовано архітектуру додатку та визначена навігація між сторінками за допомогою бібліотеки react-router-dom. react-router-dom – це бібліотека для навігації та маршрутизації вебдодатків, розроблених на React. Вона надає можливості для організації маршрутів, відображення різних компонентів на основі URL та забезпечення навігації між різними сторінками. Навігація в вебдодатку описана в табл. 3.2. В директорії Components розташовані основні React-компоненти.

Таблиця 3.2 – Навігація в вебдодатку

| Шлях | Відповідний компонент | Призначення компонента                                                                                                                                                                                                           |
|------|-----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1    | 2                     | 3                                                                                                                                                                                                                                |
| /    | Schedule.js           | Являє собою сторінку для відображення розкладу. Сторінка містить блок з фільтрами даних (вид розкладу, кафедра, фільтр по групі, викладачу або аудиторії), кнопку для завантаження розкладу в форматі XLSX, таблицю з розкладом. |

Продовження таблиці 3.2

| 1            | 2             | 3                                                                                                                                                                                                               |
|--------------|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| /cathedras   | Cathedra.js   | Являє собою сторінку для відображення кафедр. Містить пошук кафедри за назвою, кнопку для виклику модального вікна для додавання кафедри, таблицю з кафедрами.                                                  |
| /audiences   | Audience.js   | Являє собою сторінку для відображення аудиторій. Містить пошук аудиторії за назвою та кафедрою, кнопку для виклику модального вікна для додавання аудиторії, таблицю з даними.                                  |
| /specialties | Specialty.js  | Являє собою сторінку для відображення спеціальностей. Містить пошук спеціальностей за назвою та кафедрою, кнопку для виклику модального вікна для додавання спеціальності, таблицю з даними.                    |
| /disciplines | Discipline.js | Являє собою сторінку для відображення дисциплін. Містить пошук дисципліни за назвою та спеціальністю, кнопку для виклику модального вікна для додавання дисципліни, таблицю з даними.                           |
| /classes     | Class.js      | Являє собою сторінку для відображення занять. Містить пошук занять за спеціальністю, дисципліною, групою, викладачем та семестром, кнопку для виклику модального вікна для додавання заняття, таблицю з даними. |

Продовження таблиці 3.2

| 1                        | 2                  | 3                                                                                                                                                                                                     |
|--------------------------|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| /groups                  | Group.js           | Являє собою сторінку для відображення групи. Містить пошук групи за назвою та спеціальністю, кнопку для виклику модального вікна для додавання групи, таблицю з даними.                               |
| /teachers                | Teacher.js         | Являє собою сторінку для відображення викладача. Містить пошук викладача за прізвищем та кафедрою, кнопку для виклику модального вікна для додавання дисципліни, таблицю з даними.                    |
| /admin_general_values    | GeneralValues.js   | Являє собою сторінку для відображення форми з загальними даними для алгоритмів (значення ваг, кількість днів тижня та занять на день), також містить кнопку для оновлення даних.                      |
| /admin_management_data   | ManagementData.js  | Являє собою сторінку для видалення всіх даних за кафедрою, містить вибір кафедри та кнопку Видалити.                                                                                                  |
| /admin_submit_data_sheet | SubmitDataSheet.js | Являє собою сторінку для завантаження відомостей даних для кафедри, містить вибір файлу, вибір кафедри, вибір листу таблиці, кнопку для завантаження даних. Можна також завантажити шаблон відомості. |

Кінець таблиці 3.2

| 1                    | 2               | 3                                                                                                                                                                                                  |
|----------------------|-----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| /admin_schedule_data | ScheduleData.js | Являє собою сторінку для відображення форми з даними про поточний розклад – це оцінка поточного розкладу, також містить кнопку для рахування фітнес значення.                                      |
| /admin_algorithms    | Algorithms.js   | Являє собою сторінку для вибору та запуску алгоритму для складання розкладу. Містить вибір кафедри, вибір алгоритму, редагування параметрів алгоритму та графіки з результатами роботи алгоритмів. |
| *                    | Error.js        | Являє собою сторінку з помилкою для невизначених шляхів.                                                                                                                                           |

Для розробки інтерфейсу користувача використовувалася бібліотека React-Bootstrap. React-Bootstrap є бібліотекою, яка надає набір готових компонентів Bootstrap для використання в додатках, розроблених з використанням React. Вона містить набір готових графічних елементів для створення адаптивного та функціонального інтерфейсу. Також для створення розкривних списків, які дозволяють вибирати значення зі списку або шукати значення за допомогою пошуку, використовувалася бібліотека react-select.

Для створення сповіщень і повідомлень була застосована бібліотека react-notifications. Вона надає прості та зручні компоненти, які дозволяють виводити різноманітні повідомлення, такі як спливні повідомлення, сповіщення про успіх, помилки, попередження.

Для візуалізації даних у вигляді лінійних графіків була використана бібліотека Recharts. Вона дозволяє створювати різноманітні типи графіків, додавати до них інтерактивність та анімацію, також дозволяє налаштувати їх стиль.

Для роботи з файлами в форматі XLSX застосовував бібліотеку `xlsx-colog`. Вона надає можливість читати, записувати, редагувати та обробляти дані в електронних таблицях з використанням JavaScript. Вона також підтримує функції створення нових аркушів, редагування наявних аркушів, форматування комірок, вставку зображень та багато іншого.

### **3.4 Висновок**

Розроблено вебзастосунок для керування та складання розкладу занять. Для відображення структури програмного забезпечення було розроблено класову діаграму, компоненту діаграму в нотації UML.

Описано функціонування програмного забезпечення в нотації UML основі діаграм станів, діаграм послідовностей та діаграм кооперації.

Описано програмну реалізацію React-додатка: файлову структуру, призначення та функції розроблених React-компонентів, допоміжні бібліотеки.

## **4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ**

### **4.1 Опис застосування програми**

Вебзастосунок для складання та керування розкладу занять є інструментом для ефективного планування розкладу занять в університеті. Ця програма допомагає автоматизувати процес створення розкладу, забезпечуючи оптимальну організацію часу та ресурсів. Програма виконує такі функції:

- перегляд розкладу занять;
- авторизація користувача для адміністрування даних;
- ручне управління даними про групи, аудиторії, дисципліни, кафедри, спеціальності, викладачів, заняття, алгоритми;
- запуск алгоритмів для складання розкладу;
- автоматичне завантаження даних з навчальних відомостей;
- перегляд результатів роботи алгоритмів у вигляді графіків.

Ця система призначена для університетів. Вона дозволяє автоматично генерувати розклад занять на основі введених даних. Застосовувані методи включають алгоритми оптимізації, графічні інтерфейси для взаємодії з користувачами.

Обмеження для застосування системи включають необхідність наявності базових даних про курси, викладачів, аудиторії та їхню доступність у системі. Також необхідно мати достатню обчислювальну потужність технічних засобів, щоб забезпечити швидку та ефективну роботу системи при обробці великого обсягу даних.

Мінімальна конфігурація технічних засобів для використання системи включає комп'ютер або сервер з достатньою оперативною пам'яттю та процесором для обробки даних, а також наявність необхідного програмного забезпечення, зокрема веббраузера для користувачів та серверного програмного забезпечення для запуску системи.

Ця автоматизована система складання розкладу допомагає зекономити час та зусилля, що зв'язані з ручним складанням розкладу занять. Вона спрощує процес планування, забезпечуючи оптимальну організацію навчального процесу та підвищуючи його ефективність.

#### **4.2 Налаштування доступу до вебзастосунку**

Для доступу до вебзастосунку через домен було виконано наступні дії:

- реєстрація домену на сервісі <https://nic.ua/>: доменна зона .pp.ua надає безплатну реєстрацію домену в Україні на 1 рік;
- налаштування DNS записів: у панелі керування домену на <https://nic.ua/> додав записи типу A для вказання IP-адреси вашого сервера, на якому розміщений вебзастосунок;
- конфігурація Nginx: на сервері, де розміщений сайт, налаштував конфігураційний файл Nginx `nginx.conf`, для того, що сервер Nginx правильно обробляв запити до застосунку, які надходять через зареєстрований домен;
- перезавантаження Nginx: після збереження змін у конфігураційному файлі Nginx, перезавантажив вебсервер, щоб застосувати налаштування.

Після налаштування, доступ до вебзастосунку можна отримати через <http://www.nuzp.pp.ua/>.

#### **4.3 Опис роботи з програмою**

Для того, щоб почати роботу з вебзастосунком, треба перейти за адресою <http://www.nuzp.pp.ua/>.

Для всіх користувачів відображається сторінка для перегляду розкладу (рис. 4.1).

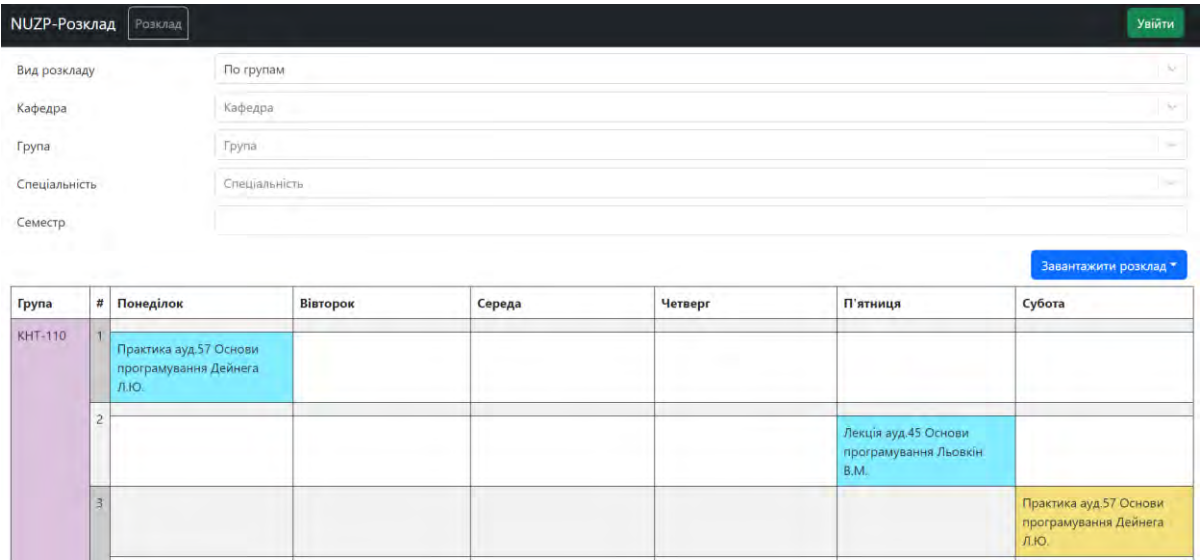


Рисунок 4.1 – Сторінка перегляду розкладу

На сторінці перегляду розкладу можна змінити вид розкладу та відфільтрувати. Також можна завантажити розклад у форматі `xlsx`, натиснувши відповідну кнопку.

Для керування та адміністрування даними треба авторизуватися. Для цього треба натиснути кнопку Увійти. Після натиснення з'явиться модальне вікно (рис. 4.2).

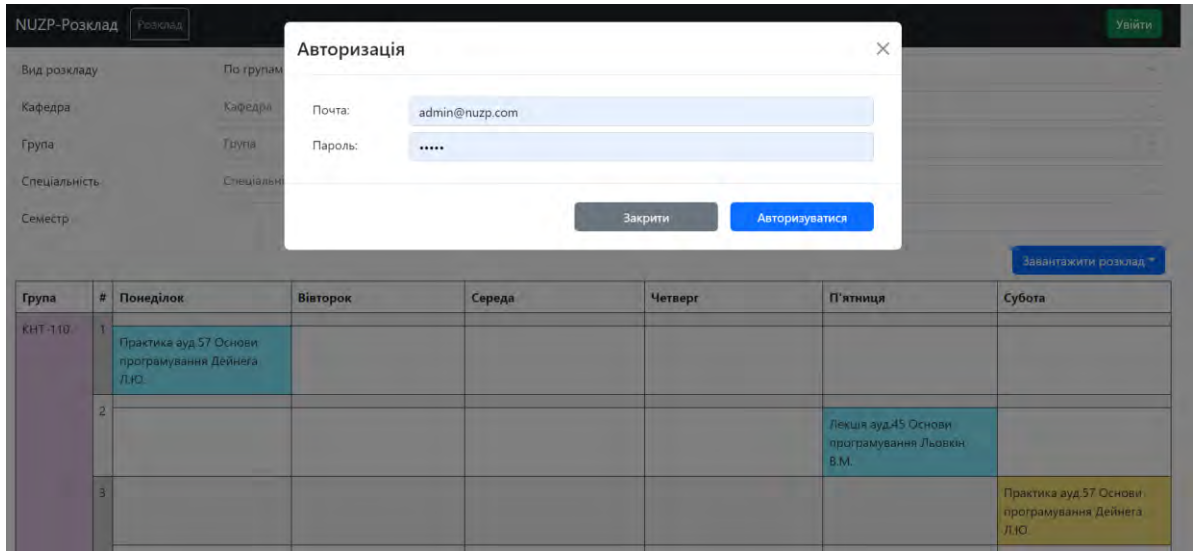


Рисунок 4.2 – Вікно для авторизації

Щоб авторизуватись треба заповнити форму даних. В вебзастосунку для сповіщення про результати виконання певних дій використовуються сповіщення. Приклад сповіщення про ведення неправильних даних при авторизації (рис. 4.3). Повідомлення містить результат виконання (успіх або помилку, текстове повідомлення).

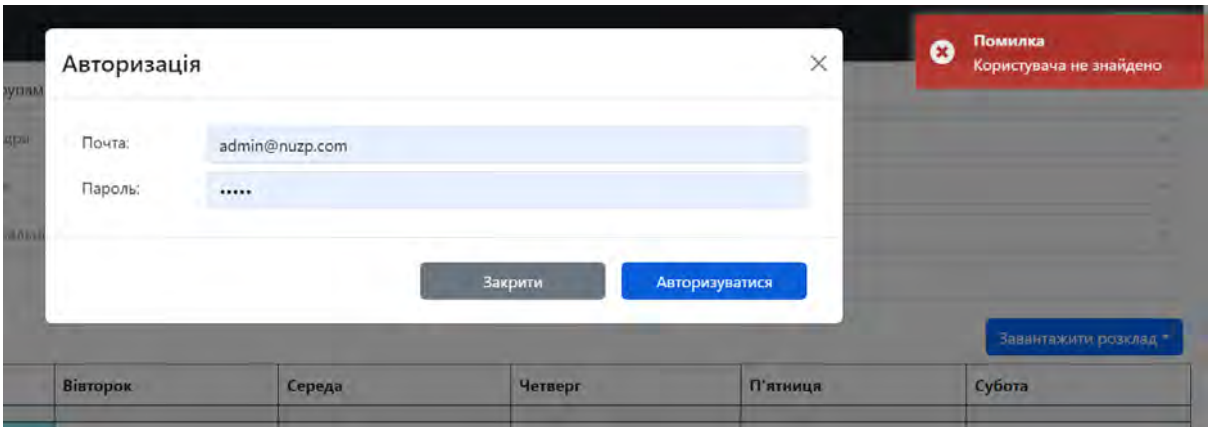


Рисунок 4.3 – Ведення неправильних даних

Після успішної авторизації з’явиться відповідне повідомлення та сторінки для керування даними (рис. 4.4).

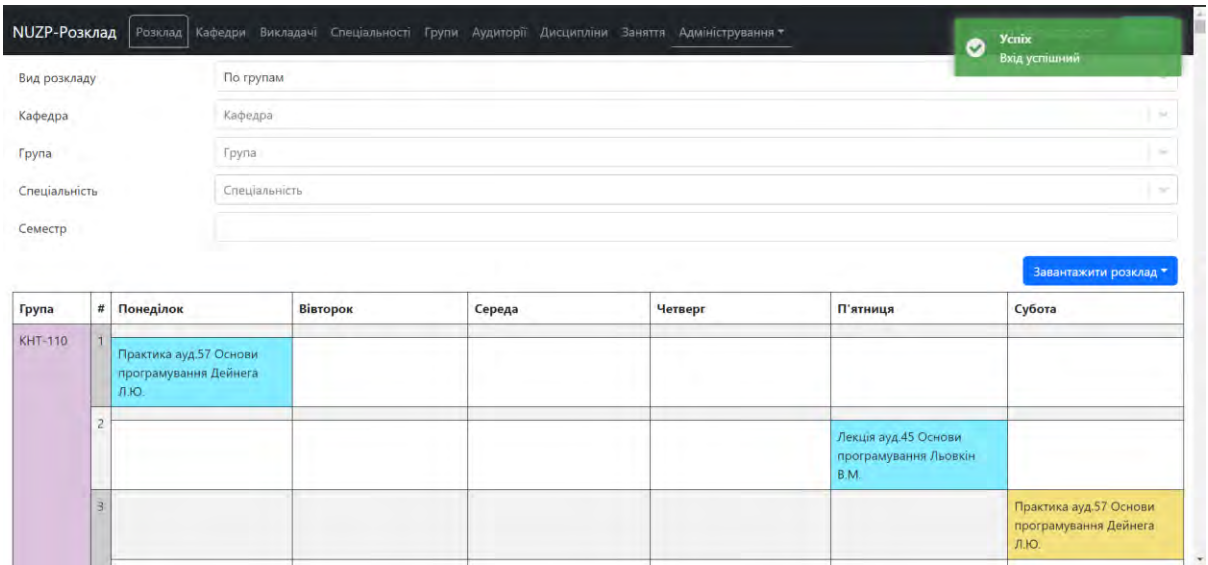


Рисунок 4.4 – Успішна авторизація

У вебзастосунку присутні такі сторінки:

- а) Кафедра;
- б) Викладачі;
- в) Спеціальності;
- г) Групи;
- д) Аудиторії;
- е) Дисципліни;
- ж) Заняття;
- з) Адміністрування:
  - 1) Запуск алгоритмів;
  - 2) Оцінка поточного розкладу;
  - 3) Загальні дані;
  - 4) Видалення даних;
  - 5) Завантаження відомостей.

Для додавання, редагування, видалення записів з бази даних треба перейти на сторінку а) – ж). Розглянемо на прикладі керування записами про наявні дисципліни тому, що інші сторінки мають схожу структуру та функціональність.

Для керування даними про дисципліни треба перейти на сторінку Дисципліни (рис. 4.5).

| Дисципліна                                           | Спеціальність - Семестер |                                                              |
|------------------------------------------------------|--------------------------|--------------------------------------------------------------|
| CAD/CAM/CAE-системи                                  | КНТ - 0 - 2              | <input checked="" type="checkbox"/> <input type="checkbox"/> |
|                                                      | КНТ - 0 - 1              | <input type="checkbox"/> <input type="checkbox"/>            |
|                                                      | КНТ - 2 - 14             | <input type="checkbox"/> <input type="checkbox"/>            |
|                                                      | КНТ - 1 - 14             | <input type="checkbox"/> <input type="checkbox"/>            |
| Scada-системи                                        | КНТ - 1 - 14             | <input checked="" type="checkbox"/> <input type="checkbox"/> |
| Алгоритми та структури даних                         | КНТ - 1 - 8              | <input checked="" type="checkbox"/> <input type="checkbox"/> |
|                                                      | КНТ - 1 - 6              | <input type="checkbox"/> <input type="checkbox"/>            |
| Алгоритмізація та програмування                      | КНТ - 2 - 6              | <input checked="" type="checkbox"/> <input type="checkbox"/> |
| Аналіз вимог до програмного забезпечення             | КНТ - 1 - 10             | <input checked="" type="checkbox"/> <input type="checkbox"/> |
| Архітектура та проектування програмного забезпечення | КНТ - 1 - 10             | <input checked="" type="checkbox"/> <input type="checkbox"/> |

Рисунок 4.5 – Сторінка Дисципліни

Сторінка містить блок пошуку дисципліни за фільтрами, кнопку для додавання запису, таблицю за наявними записами, де можна видалити або відредагувати окремий рядок.

Для додавання запису треба натиснути кнопку Додати дисципліну. Після натискання з'явиться вікно для заповнення даних (рис. 4.6).

| Дисципліна                                           | Семестр                    | Дії |
|------------------------------------------------------|----------------------------|-----|
| CAD/CAM/CAE-системи                                  |                            | ✓ ⓧ |
| Scada-системи                                        | КНТ – 1 – 14               | ✓ ⓧ |
| Алгоритми та структури даних                         | КНТ – 1 – 8<br>КНТ – 1 – 6 | ✓ ⓧ |
| Алгоритмізація та програмування                      | КНТ – 2 – 6                | ✓ ⓧ |
| Аналіз вимог до програмного забезпечення             | КНТ – 1 – 10               | ✓ ⓧ |
| Архітектура та проектування програмного забезпечення | КНТ – 1 – 10               | ✓ ⓧ |

Рисунок 4.6 – Створення запису дисципліни

При введенні пустих або неправильних даних з'явиться повідомлення біля відповідного поля (рис. 4.7).

Створити запис дисципліни

Назва дисципліни:

Закріплені спеціальності:

Назва спеціальності:

Семестр:

Пусте поле назви дисципліни

Рисунок 4.7 – Приклад перевірки форми

Щоб додати дисципліну треба ввести її назву та вказати на якій спеціальності та в якому семестрі вона буде викладатися (рис. 4.8).

Рисунок 4.8 – Заповнення форми

Після додавання дисципліни з’явиться відповідне повідомлення про успіх (рис. 4.9).

NUZP-Розклад

Розклад

Кафедри

Викладачі

Спеціальності

Групи

Аудиторії

Дисципліни

Заняття

Адміністрування

Успіх  
Запис дисципліни успішно створено і додані зв'язки

Назва дисципліни

Спеціальність

Додати дисципліну

| Дисципліна                               | Спеціальність - Семестер |                                                              |
|------------------------------------------|--------------------------|--------------------------------------------------------------|
| CAD/CAM/CAE-системи                      | КНТ - 0 - 1              | <input checked="" type="checkbox"/> <input type="checkbox"/> |
|                                          | КНТ - 2 - 14             | <input type="checkbox"/>                                     |
|                                          | КНТ - 0 - 2              | <input type="checkbox"/>                                     |
|                                          | КНТ - 1 - 14             | <input type="checkbox"/>                                     |
| Scada-системи                            | КНТ - 1 - 14             | <input checked="" type="checkbox"/> <input type="checkbox"/> |
| Алгоритми                                | КНТ - 0 - 1              | <input checked="" type="checkbox"/> <input type="checkbox"/> |
| Алгоритми та структури даних             | КНТ - 1 - 8              | <input checked="" type="checkbox"/> <input type="checkbox"/> |
|                                          | КНТ - 1 - 6              | <input type="checkbox"/>                                     |
| Алгоритмізація та програмування          | КНТ - 2 - 6              | <input checked="" type="checkbox"/> <input type="checkbox"/> |
| Аналіз вимог до програмного забезпечення | КНТ - 1 - 10             | <input checked="" type="checkbox"/> <input type="checkbox"/> |

Рисунок 4.9 – Успішне додавання даних

Для редагування даних треба натиснути першу кнопку в останньому стовпцю відповідного рядка. Відобразиться форма з даними (рис. 4.10).

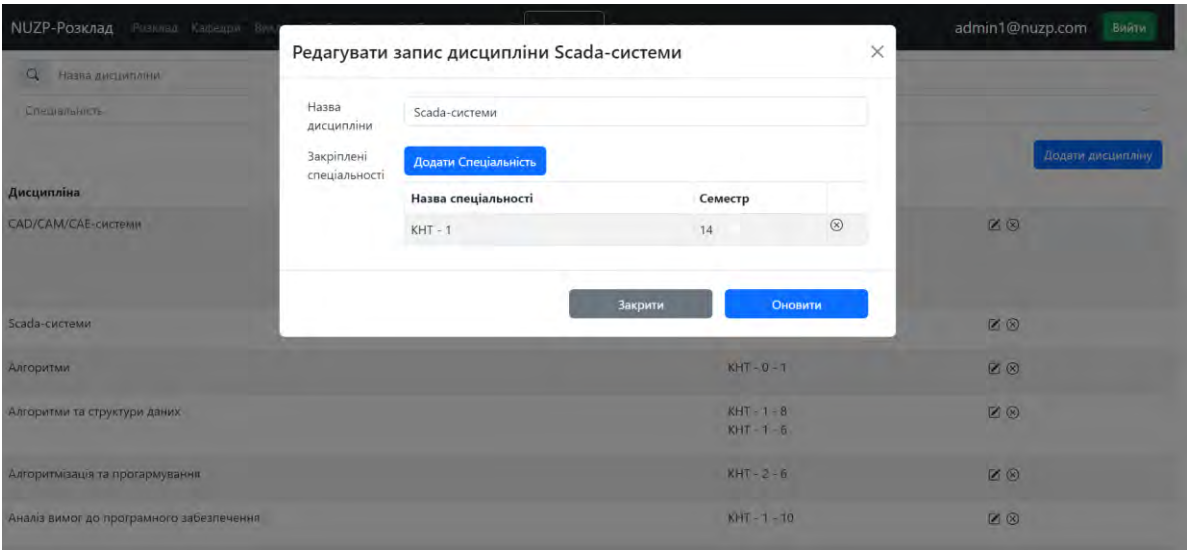


Рисунок 4.10 – Редагування обраного запису

Для видалення треба натисну другу кнопку в останньому стовпцю відповідного рядка. Відобразиться вікно для підтвердження дій (рис. 4.11).

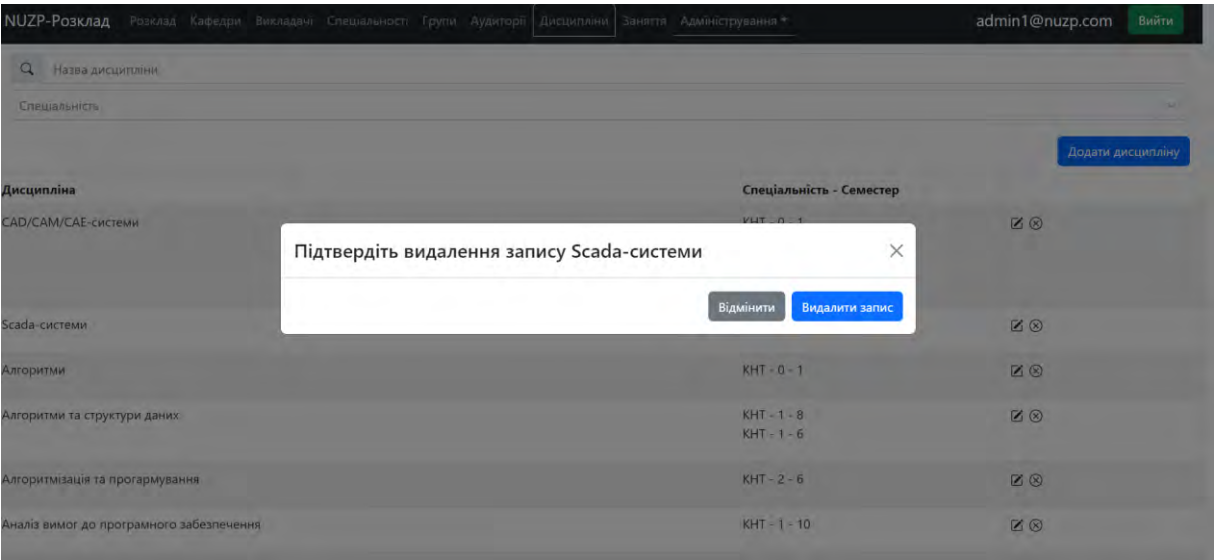


Рисунок 4.11 – Вікно для підтвердження дій

Для запуску алгоритмів, автоматичного завантаження даних, видалення усіх даних, перегляду оцінки поточного розкладу та зміна

загальних даних для алгоритмів треба обрати відповідну сторінку в спадному списку Адміністрування (рис. 4.12).

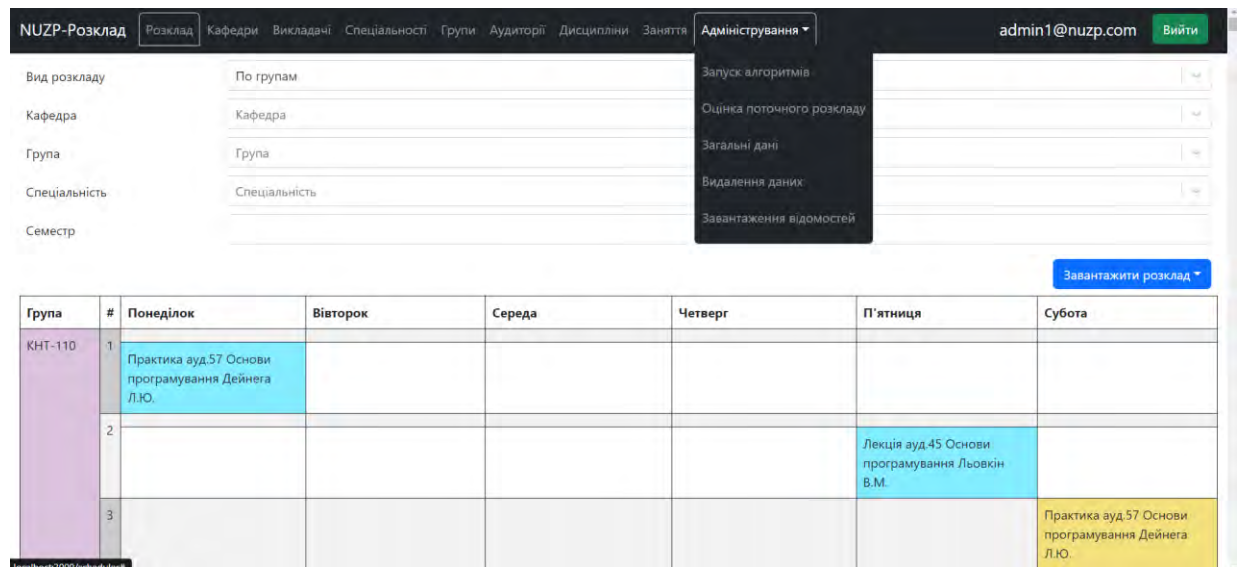


Рисунок 4.12 – Список Адміністрування

Для запуску алгоритмів треба перейти на сторінку Запуск алгоритмів (рис. 4.13).

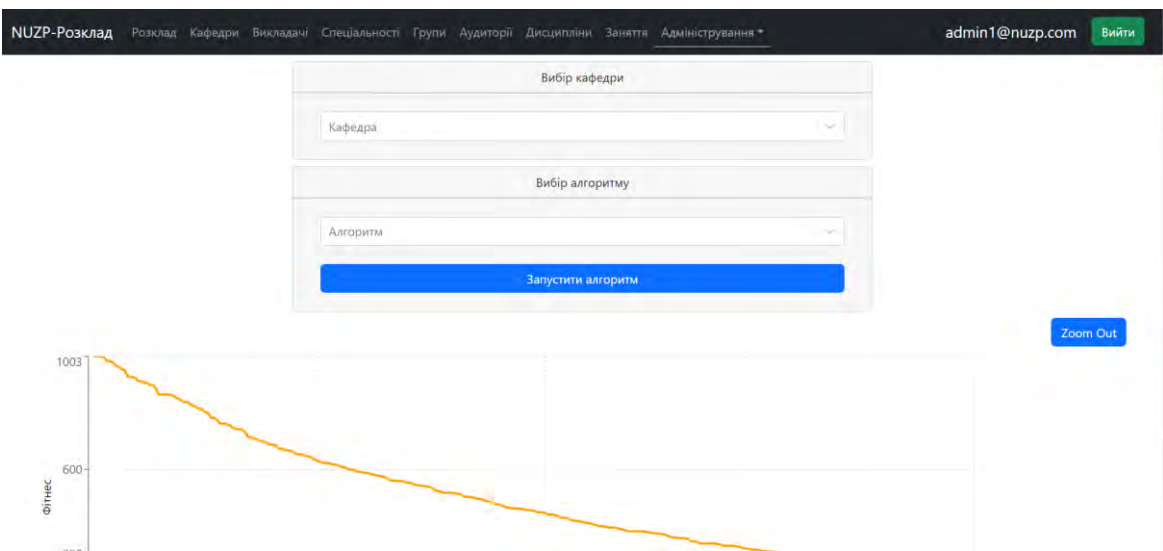


Рисунок 4.13 – Сторінка Запуск алгоритмів

Для того, щоб запустити алгоритм: треба обрати кафедру, обрати алгоритм, налаштувати необхідні параметри, оновити параметри та запустити алгоритм (рис. 4.14).

Рисунок 4.14 – Запуск алгоритму

Швидкість роботи алгоритму залежить від його параметрів та обчислювальної машини, тому потрібно почекати кінця роботи (рис. 4.15).

Рисунок 4.15 – Робота алгоритму

Після виконання алгоритму відобразиться графік з результатами конкретного алгоритму з різними параметрами (рис. 4.16).

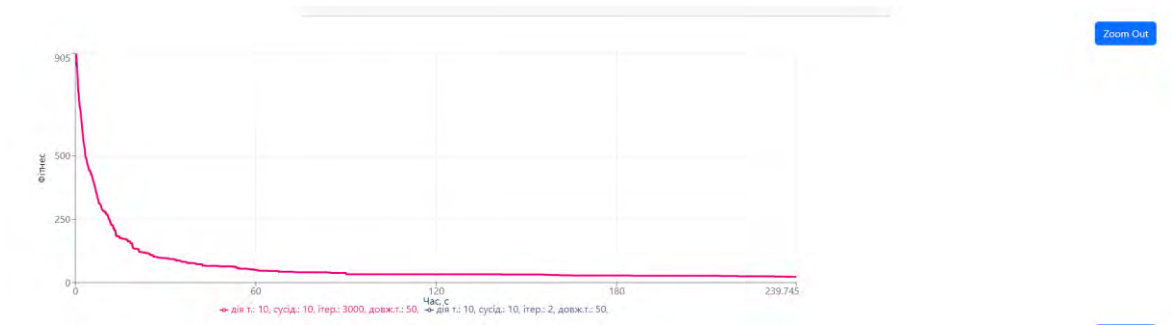


Рисунок 4.16 – Графік роботи алгоритму з різними параметрами

Також доступно загальний графік для порівняння різних алгоритмів (рис. 4.17).

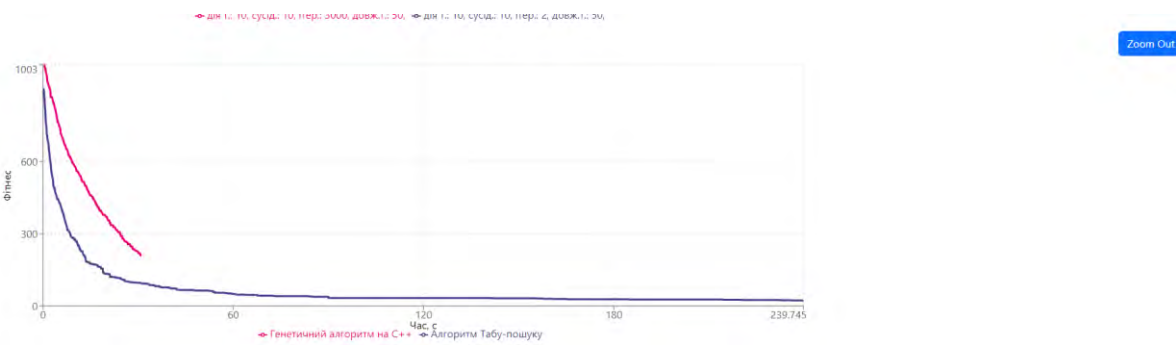


Рисунок 4.17 – Загальний графік

Кожний графік можна масштабувати, потрібно виділити область на графіку (рис. 4.18). Для встановлення початкового виду треба натиснути кнопку Zoom out.

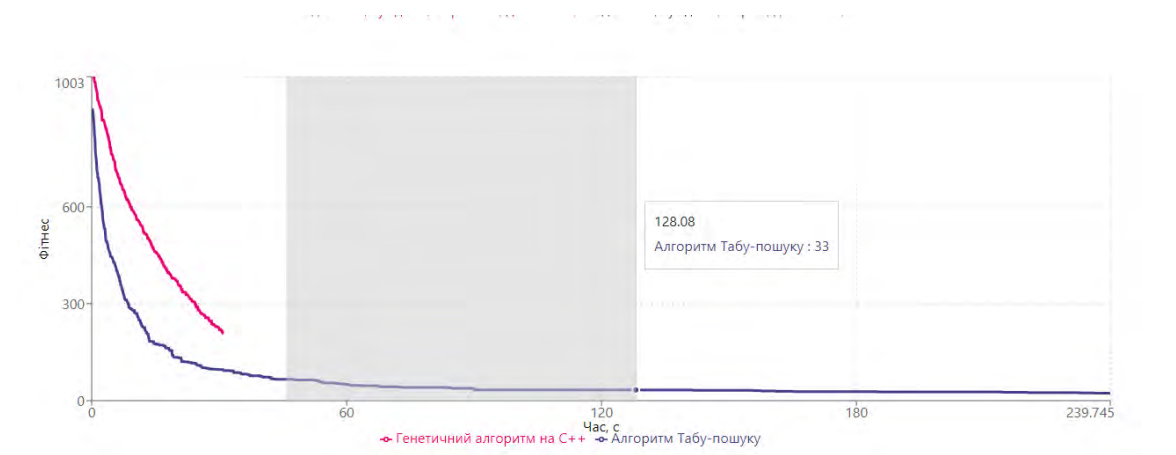


Рисунок 4.18 – Масштабування графіків

Сторінка Оцінка розкладу призначена для перевірки якості складеного розкладу (рис. 4.19). Можна також перерахувати значення фітнес, натиснувши кнопку в формі даних.

| Оцінка поточного розкладу |     |
|---------------------------|-----|
| Загальний фітнес          | 835 |
| Фітнес рек. час           | 0   |
| Фітнес груп               |     |
| Загальний                 | 388 |
| Вікна                     | 188 |
| Накладання занять         | 200 |
| Фітнес викладачів         |     |
| Загальний                 | 167 |
| Вікна                     | 62  |
| Накладання занять         | 105 |
| Фітнес аудиторій          |     |
| Накладання занять         | 280 |

Поррахувати значення фітнес

Рисунок 4.19 – Сторінка Оцінка поточного розкладу

Для зміни значень ваг для різних обмежень, встановлення кількості навчальних днів та пар на день, потрібно перейти до сторінки Загальні дані (рис. 4.20).

| Загальні дані для алгоритмів |   |
|------------------------------|---|
| Вага: вікна групи            | 1 |
| Вага: вікна викладачів       | 1 |
| Вага: пізні заняття          | 2 |
| Вага: рівномірний розклад    | 2 |
| Вага: накладання занять      | 5 |
| Вага: співпадіння занять     | 5 |
| Кількість днів               | 6 |
| Кількість занять             | 6 |

Оновити дані

Рисунок 4.20 – Сторінка Загальні дані

Сторінка Видалення даних призначена для видалення усіх даних кафедри (рис. 4.21).

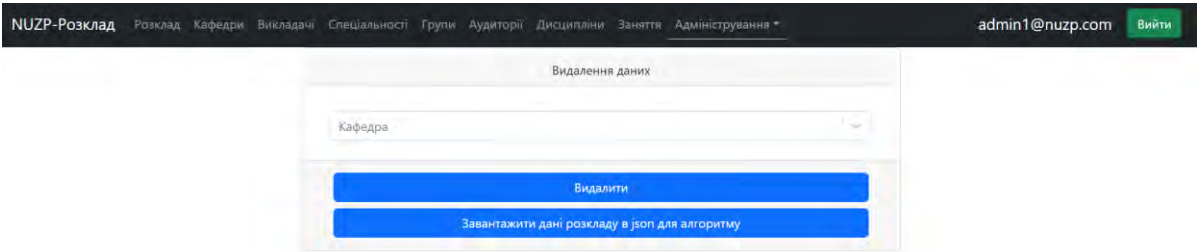


Рисунок 4.21 – Сторінка видалення даних

Для завантаження даних кафедри треба перейти на сторінку Завантаження відомостей, потім обрати файл, кафедру, обрати аркуш таблиці та натиснути кнопку (рис. 4.22).

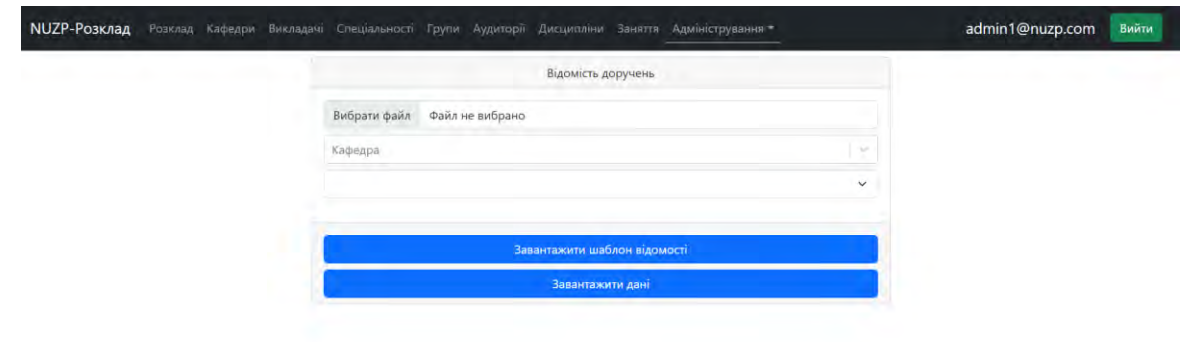


Рисунок 4.22 – Завантаження відомостей доручень

Також на цій сторінці можна завантажити шаблон відомості.

#### 4.4 Результати тестування програмного забезпечення

Для перевірки верстки та функціонування вебзастосунку було проведено тестування. Після проходження тестування був складений чек-лист наведений в табл. 4.1.

Таблиця 4.1 – Чек-лист тестування вебзастосунку

| Вид перевірки                                                           | Windows 11<br>Google Chrome | Windows 7<br>Opera | Android 13<br>Google Chrome |
|-------------------------------------------------------------------------|-----------------------------|--------------------|-----------------------------|
| 1                                                                       | 2                           | 3                  | 4                           |
| Авторизація наявного користувача                                        | Пройдено                    | Пройдено           | Пройдено                    |
| Авторизація не наявного користувача                                     | Пройдено                    | Пройдено           | Пройдено                    |
| Користувач із введеним логіном існує в системі, але пароль неправильний | Пройдено                    | Пройдено           | Пройдено                    |
| Фільтр Розкладу                                                         | Пройдено                    | Пройдено           | Пройдено                    |
| Пошук Розкладу                                                          | Пройдено                    | Пройдено           | Пройдено                    |
| Завантаження таблиці з розкладом                                        | Пройдено                    | Пройдено           | Пройдено                    |

Продовження таблиці 4.1

| 1                         | 2        | 3        | 4        |
|---------------------------|----------|----------|----------|
| Пошук аудиторії           | Пройдено | Пройдено | Пройдено |
| Фільтр аудиторії          | Пройдено | Пройдено | Пройдено |
| Додавання аудиторії       | Пройдено | Пройдено | Пройдено |
| Редагування аудиторії     | Пройдено | Пройдено | Пройдено |
| Видалення аудиторії       | Пройдено | Пройдено | Пройдено |
| Додавання кафедри         | Пройдено | Пройдено | Пройдено |
| Видалення кафедри         | Пройдено | Пройдено | Пройдено |
| Пошук кафедри             | Пройдено | Пройдено | Пройдено |
| Фільтр спеціальності      | Пройдено | Пройдено | Пройдено |
| Пошук спеціальності       | Пройдено | Пройдено | Пройдено |
| Додавання спеціальності   | Пройдено | Пройдено | Пройдено |
| Редагування спеціальності | Пройдено | Пройдено | Пройдено |
| Видалення спеціальності   | Пройдено | Пройдено | Пройдено |
| Пошук дисципліни          | Пройдено | Пройдено | Пройдено |
| Фільтр дисципліни         | Пройдено | Пройдено | Пройдено |
| Додавання дисципліни      | Пройдено | Пройдено | Пройдено |

Продовження таблиці 4.1

| 1                        | 2        | 3        | 4        |
|--------------------------|----------|----------|----------|
| Редагування дисципліни   | Пройдено | Пройдено | Пройдено |
| Видалення дисципліни     | Пройдено | Пройдено | Пройдено |
| Пошук групи              | Пройдено | Пройдено | Пройдено |
| Фільтр групи             | Пройдено | Пройдено | Пройдено |
| Додавання групи          | Пройдено | Пройдено | Пройдено |
| Редагування групи        | Пройдено | Пройдено | Пройдено |
| Видалення групи          | Пройдено | Пройдено | Пройдено |
| Пошук викладача          | Пройдено | Пройдено | Пройдено |
| Фільтр викладача         | Пройдено | Пройдено | Пройдено |
| Додавання викладача      | Пройдено | Пройдено | Пройдено |
| Редагування викладача    | Пройдено | Пройдено | Пройдено |
| Видалення викладача      | Пройдено | Пройдено | Пройдено |
| Пошук заняття            | Пройдено | Пройдено | Пройдено |
| Фільтр заняття           | Пройдено | Пройдено | Пройдено |
| Додавання заняття        | Пройдено | Пройдено | Пройдено |
| Редагування заняття      | Пройдено | Пройдено | Пройдено |
| Видалення заняття        | Пройдено | Пройдено | Пройдено |
| Оновлення значень Фітнес | Пройдено | Пройдено | Пройдено |

Кінець таблиці 4.1

| 1                                                   | 2        | 3        | 4        |
|-----------------------------------------------------|----------|----------|----------|
| Оновлення загальних даних                           | Пройдено | Пройдено | Пройдено |
| Завантаження відомостей                             | Пройдено | Пройдено | Пройдено |
| Видалення даних                                     | Пройдено | Пройдено | Пройдено |
| Запуск алгоритмів                                   | Пройдено | Пройдено | Пройдено |
| Вихід з облікового запису                           | Пройдено | Пройдено | Пройдено |
| Адаптивність верстки                                | Пройдено | Пройдено | Пройдено |
| Не авторизований користувач не може редагувати дані | Пройдено | Пройдено | Пройдено |
| Збереження сесії після закриття вкладки             | Пройдено | Пройдено | Пройдено |

Вебзастосунок пройшов успішне тестування і відповідає встановленим вимогам та специфікаціям. Він продемонстрував стабільну та надійну роботу, а також забезпечує зручну та коректну взаємодію з користувачем. Рекомендується продовжити роботу з налагодженням дрібних покращень та вирішенням незначних недоліків.

#### 4.5 Висновок

У цьому розділі було описано програмне забезпечення для керування та складання розкладу. Було розглянуто призначення та характеристики

вебзастосунку. Описано процес налаштування доступу до сайту через виділений домен. Було описано роботу з програмою: основні функції, призначення сторінок, опис функціональності, призначення елементів вебінтерфейсу. Було проведено тестування застосунку з використанням різних веббраузерів та на різних операційних системах. Тестування показало стабільну та коректну роботу додатка.

## ВИСНОВКИ

В процесі виконання кваліфікаційної роботи бакалавра було розроблено систему для автоматизованого складання та керування розкладом занять для університету.

Було проведено аналіз предметної області, визначено основні поняття, пов'язані з предметною областю й об'єктом дослідження та розробки. Було досліджено основні методи оптимізації цільової функції для задачі складання оптимального розкладу занять та проведено порівняння цих методів, що показав перевагу використання генетичних алгоритмів для розв'язання поставлених задач. Було досліджено та проведено порівняльну характеристику наявних програмних засобів для визначення функціональності та вимог.

Було проведено вибір мови програмування для розроблюваного забезпечення. В ході порівняння мов було вирішено використовувати JavaScript та фреймворк React.js. Було визначено архітектуру додатка та протокол взаємодії клієнтської та серверної частини. За допомогою діаграми прецедентів у нотації UML було створено модель взаємодії користувача з системою.

Було надано структуру програмного забезпечення у вигляді класової та компонентної діаграм. За допомогою діаграм прецедентів, діаграми станів і діаграми послідовностей надано функціонування вебзастосунку. Було описано фізичну структуру додатка, надано опис призначення файлів проєкту та використаних допоміжних бібліотек.

В останньому розділі було описано застосування програми, її призначення та обмеження. Було описано налаштування проєкту та робота з програмою. Також було проведено тестування за допомогою чек-листу.

Було розроблено систему для керування та складання розкладу навчальних занять. Розроблене програмне забезпечення може бути використане в навчальних закладах для автоматизації процесу складання

розкладу занять. Рекомендується його впровадження для покращення ефективності та зменшення затрат часу на складання розкладу.

## ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Гайтан О. М. Автоматизація генерації розкладу навчального процесу університету / О. М. Гайтан // Інформатика, обчислювальна техніка та автоматизація. – Том 31 (70) Ч. 1. – № 2. – 2020. – С. 58–66.
2. Снитюк В. Є. Технологія еволюційного формування розкладів у закладах вищої освіти: монографія / В. Є. Снитюк, О. М. Сіпко. – Київ : Видавець ФОП Піча Ю.В., 2022. – 136 с.
3. Мулява І. Я. Вирішення задачі автоматизованого формування розкладу навчального закладу за допомогою генетичних алгоритмів / І. Я. Мулява // Міжнародний науковий журнал "Інтернаука". – № 9(1). – 2018. – С. 77-83.
4. Кисіль В. В. Модель задачі складання та оптимізації розкладу занять за умови задоволення об'єктивних та суб'єктивних вимог навчального закладу / В. В. Кисіль, І. В. Драч, Т. М. Кисіль // Інформатика, обчислювальна техніка та автоматизація. – Том 30 (69) Ч. 1. – № 6. – 2019. – С. 65–70.
5. Optimization Techniques — Tabu Search [Electronic resource]. – Access mode: <https://towardsdatascience.com/optimization-techniques-tabu-search-36f197ef8e25>.
6. Слесарєв В. В. Оптимізація розкроювання продукції прокатного виробництва з використанням методу пошуку із заборонами / В. В. Слесарєв, Т. А. Желдак, Д. М. Гаранжа, О.Д. Станіна // Збірник наукових праць Національного гірничого університету. – 2010. – № 35(2). – С. 41-50.
7. Optimization Techniques — Simulated Annealing [Electronic resource]. – Access mode: <https://towardsdatascience.com/optimization-techniques-simulated-annealing-d6a4785a1de7>.
8. Томашевський В. М. Складання розкладів занять у дистанційних системах навчання / В. М. Томашевський, Ю. Л. Новіков, П. А. Камінська // Вісник Національного технічного університету України "КПІ". Інформатика, управління та обчислювальна техніка. – 2010. – Вип. 52. – С. 118–130.

9. Genetic Algorithm in Machine Learning [Electronic resource]. – Access mode: <https://www.javatpoint.com/genetic-algorithm-in-machine-learning>.
10. Генетичні алгоритми. Ключові поняття і методи реалізації [Електрон. ресурс]. – Режим доступу: [http://www.znannya.org/?view=ga\\_general](http://www.znannya.org/?view=ga_general).
11. An Illustrated Guide to Genetic Algorithm [Electronic resource]. – Access mode: <https://towardsdatascience.com/an-illustrated-guide-to-genetic-algorithm-ec5615c9ebe>.
12. Genetic Algorithm – Advantages & Disadvantages [Electronic resource]. – Access mode: <https://electricalvoice.com/genetic-algorithm-advantages-disadvantages/>.
13. Семенчук В. М. Особливості використання острівної моделі генетичних алгоритмів / В. М. Семенчук. – Тернопільський національний технічний університет імені Івана Пулюя. – УДК 004.021. – 2020. – С. 114.
14. Субботін, С. О. Нейронні мережі : навчальний посібник / С. О. Субботін, А. О. Олійник ; під заг. ред. проф. С. О. Субботіна. – Запоріжжя : ЗНТУ, 2014. – 135 с.
15. Добровська Л. М. Теорія та практика нейронних мереж : навч. посіб. / Л. М. Добровська, І. А. Добровська. – К. : НТУУ «КПІ» Вид-во «Політехніка», 2015. – 396 с. – Бібліогр. : с. 385–387. – 55 пр.
16. Новотарський М. А. Штучні нейронні мережі: обчислення / М. А. Новотарський, Б. Б. Нестеренко // Праці Інституту математики НАН України. – Т50. – Київ: Ін-т математики НАН України, 2004. – 408 с.
17. Long short-term memory [Electronic resource]. – Access mode: [https://en.wikipedia.org/wiki/Long\\_short-term\\_memory](https://en.wikipedia.org/wiki/Long_short-term_memory).
18. aSc TimeTables [Електрон. ресурс]. – Режим доступу: <https://www.asctimetables.com>.
19. University Timetabling Comprehensive Academic Scheduling Solutions [Electronic resource]. – Access mode: <https://www.unitime.org/>.

20. Безкоштовне програмне забезпечення для складання розкладу FET [Електрон. ресурс]. – Режим доступу: <https://www.lalescu.ro/liviu/fet/features.html>.
21. Вступ до JavaScript [Електрон. ресурс]. – Режим доступу: <https://uk.javascript.info/intro>.
22. Advantages and Disadvantages of JavaScript [Electronic resource]. – Access mode: <https://www.tutorialspoint.com/advantages-and-disadvantages-of-javascript>.
23. Мова програмування python – з чого розпочати навчання? [Електрон. ресурс]. – Режим доступу: <https://edu.cbsystematics.com/ua/blog/python-start-blog>.
24. Python Language advantages and applications [Electronic resource]. – Access mode: <https://www.geeksforgeeks.org/python-language-advantages-applications/>.
25. Огляд і основи мови програмування C++ [Електрон. ресурс]. – Режим доступу: [http://www.znannya.org/?view=Cpp\\_basics](http://www.znannya.org/?view=Cpp_basics).
26. Advantages and Disadvantages of C++ [Electronic resource]. – Access mode: <https://www.geeksforgeeks.org/advantages-and-disadvantages-of-cpp/>.
27. Переваги та недоліки web-додатків [Електрон. ресурс]. – Режим доступу: [https://stud.com.ua/97611/informatika/perevagi\\_nedoliki\\_dodatktiv](https://stud.com.ua/97611/informatika/perevagi_nedoliki_dodatktiv).
28. Триярусна архітектура [Електрон. ресурс]. – Режим доступу: [https://uk.wikipedia.org/wiki/Триярусна\\_архітектура](https://uk.wikipedia.org/wiki/Триярусна_архітектура).
29. Вступ в React, якого нам не вистачало [Електрон. ресурс]. – Режим доступу: <https://devzone.org.ua/post/vstup-v-react-yakogo-nam-ne-vistachalo>.
30. IT технології: Angular [Електрон. ресурс]. – Режим доступу: <https://apeirondb.com/ua/blog/it-technologies-angular>.
31. Vue.js [Electronic resource]. – Access mode: [https://www.tutorialspoint.com/vuejs/vuejs\\_overview.htm](https://www.tutorialspoint.com/vuejs/vuejs_overview.htm).

32. What is Rest API? Features, Principles, And Challenges [Electronic resource]. – Access mode: <https://www.knowledgehut.com/blog/programming/rest-api>.

33. GraphQL [Electronic resource]. – Access mode: <https://www.atatus.com/glossary/graphql/>.

34. SOAP (Simple Object Access Protocol) [Electronic resource]. – Access mode: <https://www.techtarget.com/searchapparchitecture/definition/SOAP-Simple-Object-Access-Protocol>.

35. What is JSON RPC? [Electronic resource]. – Access mode: <https://www.wallarm.com/what/what-is-json-rpc>.

**ДОДАТОК А**  
**Технічне завдання**

## **Вступ**

Найменування – вебдодаток для керування та складання розкладу занять.

Об'єктом дослідження є процес планування та складання розкладу для університету, включаючи всі етапи підготовки, виконання та оновлення розкладу.

Предметом дослідження є розробка системи для автоматизованого складання розкладу, що має на меті оптимальне використання наявних ресурсів, враховує потреби студентів та викладачів, та дозволяє скоротити час, а також уникнути помилок, пов'язаних зі складанням розкладу.

Метою роботи є розробка та впровадження системи для автоматизованого складання розкладу для університету. Основною метою системи є забезпечення оптимального та точного складання розкладу занять, зменшення часу, необхідного для підготовки до навчального семестру, а також уникнення помилок при створенні розкладу навчального процесу. Система має на меті полегшити роботу адміністраторів та забезпечити ефективний розподіл ресурсів для забезпечення якісної освіти для студентів та оптимальних умов для викладачів.

### **А.1 Підстави для розробки**

Підставою для виконання розробки є завдання на дипломний проєкт від 4 квітня 2023 року на тему «Розробка вебзастосунку для створення та керування розкладом. Frontend частина», затверджене наказом №87 по НУ «Запорізька політехніка».

## **A.2 Призначення розробки**

Вебдодаток для складання та керування розкладом призначений для ефективного планування розкладу занять в університеті згідно з навчальним планом. Вона надає інструменти для автоматичного генерування розкладу, враховуючи наявні ресурси, обмеження, побажання вчителів та учнів. Система забезпечує оптимізацію розкладу з урахуванням різних факторів, таких як доступність приміщень, розподіл навчальних годин, накладання занять та інші.

## **A.3 Вимоги до програмного продукту**

### **A.3.1 Вимоги до функціональних характеристик**

Вимоги до функціональних характеристик:

- повинно бути реалізовано перегляд та фільтрування даних у вигляді таблиць;
- повинно бути реалізовано ведення вхідних даних;
- повинно бути реалізовано редагування даних;
- повинно бути реалізовано видалення даних;
- повинно бути реалізовано налаштування параметрів алгоритму;
- повинно бути реалізовано виведення графіку роботи алгоритму;

### **A.3.2 Вимоги до надійності**

Система повинна забезпечувати стабільну та безперебійну роботу протягом усього навчального семестру. Вона має бути стійкою до впливу непередбачених ситуацій, таких як відмови обладнання, програмних збоїв або мережевих перебоїв.

Система повинна забезпечувати контроль правильності та цілісності вхідних даних. Вона має перевіряти вхідні дані на відповідність заданим форматам та правилам.

Система повинна мати вбудовані механізми безпеки, щоб запобігти несанкціонованому доступу до інформації про розклад та можливості його зміни. Це включає авторизації, шифрування даних та контроль доступу до системних ресурсів.

### **А.3.3 Умови експлуатації**

Система має працювати від стандартної мережі електроживлення з напругою 220 В та частотою 50 Гц. Вона повинна бути стійкою до нормальних коливань напруги та забезпечувати безперебійну роботу під час невеликих перерв у постачанні електроенергії.

Для користування системою не потрібна спеціальна кваліфікація. Користувачі можуть бути навчені простому та зрозумілому інтерфейсу, що дозволяє зручно та ефективно складати розклади.

### **А.3.4 Вимоги до складу та параметрів технічних засобів**

Мінімальна конфігурація технічних засобів для використання системи включає комп'ютер або сервер з достатньою оперативною пам'яттю та процесором для обробки даних, а також наявність необхідного програмного забезпечення, зокрема веббраузера для користувачів та серверного програмного забезпечення для запуску системи.

### **А.3.5 Вимоги до маркування й упакування**

Програмне забезпечення автоматизованої системи не вимагає додаткових процесів або кроків під час маркування та упакування.

### **A.3.6 Вимоги до транспортування та збереження**

Вимоги до транспортування й зберігання програмного забезпечення відповідають вимогам, які стосуються самого носія, на якому програма знаходиться.

### **A.4 Вимоги до програмної документації**

Склад програмної документації повинен містити:

- технічне завдання;
- текст програми.

### **A.5 Вхідні та вихідні дані**

Вхідними даними для системи будуть:

- дані про аудиторії;
- дані про кафедри;
- дані про спеціальності;
- дані про дисципліни;
- дані про групи;
- дані про викладачів;
- дані про заняття.

Вихідними даними буде оптимальний розклад занять, складений для обраної кафедри, в якому будуть враховуватися рекомендації до проведення та мінімізовано наступні параметри:

- кількість накладань занять для груп, вчителів та аудиторій;
- кількість вікон між заняття для груп та вчителів.

## **А.6 Використання програмного продукту**

Для того, щоб почати роботу з вебзастосунком треба перейти за адресою <http://www.nuzp.pp.ua/>.

**ДОДАТОК Б**  
**Опис програмного продукту**

## **Б.1 Загальні відомості**

Назва програмного продукту – «Вебзастосунок для керування та складання розкладу».

Вебдодаток для складання та керування розкладом призначений для ефективного планування розкладу занять в університеті згідно з навчальним планом. Вона надає інструменти для автоматичного генерування розкладу, враховуючи наявні ресурси, обмеження, побажання вчителів та учнів. Система забезпечує оптимізацію розкладу з урахуванням різних факторів, таких як доступність приміщень, розподіл навчальних годин, накладання занять та інші.

Мінімальна конфігурація технічних засобів для використання системи включає комп'ютер або сервер з достатньою оперативною пам'яттю та процесором для обробки даних, а також наявність необхідного програмного забезпечення, зокрема веббраузера для користувачів та серверного програмного забезпечення для запуску системи.

Для функціонування програми потрібно наступне:

- операційна система;
- веббраузер або мобільний пристрій, який підтримує виконання JavaScript;
- наявність встановленого пакетного менеджера, такого як npm;
- встановлений Node.js;
- наявність необхідних пакетів та бібліотек, які використовуються в програмі.

## **Б.2 Функціональне призначення**

Вимоги до функціональних характеристик:

- повинно бути реалізовано перегляд та фільтрування даних у вигляді таблиць;

- повинно бути реалізовано ведення вхідних даних;
- повинно бути реалізовано редагування даних;
- повинно бути реалізовано видалення даних;
- повинно бути реалізовано налаштування параметрів алгоритму;
- повинно бути реалізовано виведення графіку роботи алгоритму;

### Б.3 Опис логічної структури

На рис. Б.1 – Б.2 представлена діаграма класів програмного продукту.

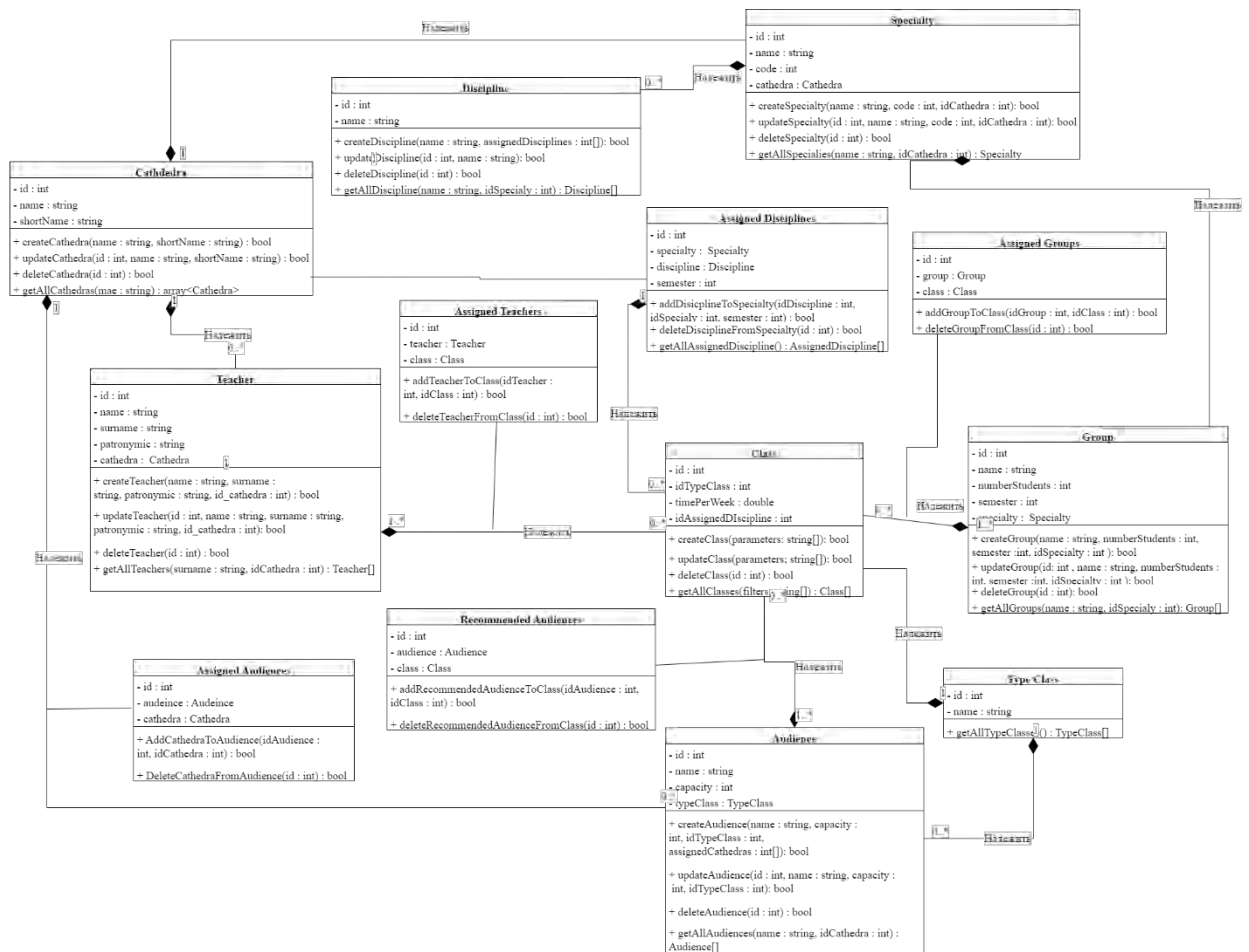


Рисунок Б.1 – Діаграма класів вебзастосунку для складання та керуванням розкладом занять (а)



- Рекомендований розклад;
- Групи;
- Закріплені групи: це групи, які відвідують конкретне заняття;
- Дисципліна;
- Закріплені дисципліни: це дисципліна, яку вивчають на занятті;
- Кафедра;
- Викладач;
- Закріплені викладачі: це викладачі, які введуть певне заняття;
- Аудиторія;
- Закріплені аудиторії за певною кафедрою;
- Рекомендовані для проведення заняття аудиторії;
- Тип заняття;
- Сервер;
- База даних.

#### **Б.4 Вибір інструментарію розробки програмного продукту**

Для розробки програмного продукту було обрано мову програмування JavaScript та вирішено розробити у вигляді вебзастосунку. Розробка вебзастосунків мають ряд переваг:

- вебзастосунки можна запускати на будь-якому пристрої з веббраузером і підключенням до Інтернету, це дозволить забезпечити широку доступність;
- вони не потребують встановлення на пристрої користувача, що значно спрощує їх розгортання та оновлення;
- можна легко масштабувати, додаючи нові сервери або збільшуючи потужність наявних серверів;
- вебзастосунки можна оновлювати на стороні сервера, що забезпечує швидку виправлення помилок і оновлення функціонала;

— можна легко оновлювати й підтримувати без необхідності оновлення кожного клієнта окремо.

Було вирішено розробити клієнтську частину використовуючи React через його високу гнучкість та простоту.

Було обрано протокол GraphQL API через те, що GraphQL дозволяє отримувати точно ті дані, які потрібні клієнту, та надає можливість кешування даних на стороні клієнта, що дозволить зменшити кількість запитів до сервера та покращити швидкість роботи додатка.

## **Б.5 Виклик і завантаження**

Для того, щоб запустити вебзастосунок, треба у терміналі директорії NuzpProjectSchedule/server запустити команду `npm run dev`.

Для того, щоб почати роботу з вебзастосунком, треба перейти за адресою <http://www.nuzp.pp.ua/>.

**ДОДАТОК В**  
**Текст програми**

## В.1 Файл index.js

```
import React from "react";
import ReactDOM from "react-dom";
import App from "./App";
import reportWebVitals from "./reportWebVitals";
import { BrowserRouter } from "react-router-dom";
import { ApolloClient, InMemoryCache, ApolloProvider } from
"@apollo/client";
import "bootstrap/dist/css/bootstrap.min.css";
import AuthProvider from "./Components/Authorization/AuthProvider";
import "react-notifications/lib/notifications.css";
import "./App.css";
import config from "./config.js";

const client = new ApolloClient({
  uri: `http://${config.HOST}:${config.PORT}/graphql`,
  cache: new InMemoryCache(),
});

ReactDOM.render(
  <React.StrictMode>
    <ApolloProvider client={client}>
      <AuthProvider>
        <BrowserRouter>
          <App></App>
        </BrowserRouter>
      </AuthProvider>
    </ApolloProvider>
  </React.StrictMode>,
  document.getElementById("root")
);

reportWebVitals();
```

## В.2 Файл config.js

```
const config = {
  //HOST: "91.219.54.59",
  HOST: "localhost",
  PORT: "3001",
};

export default config;
```

## В.3 Фрагмент файла App.js

```
class App extends React.Component {
  state = {
    openLogin: false,
    isLoggin: false,
    userID: "",
    userEmail: "",
  };
};
```

```

loginSuccess = (data, token) => {
  this.setState({
    ...data,
  });
  this.context.setUser(data.userId, data.userEmail);
  this.context.setToken(token);
};

alert = CreateNotification;

componentDidMount() {
  const res = this.context.loadData();

  res.then((data) => {
    if (data) {
      if (data.isAuth.successful) {
        this.setState({
          isLoggin: true,
          userId: data.id,
          userEmail: data.email,
        });
        this.context.setUser(data.id, data.email);
      }
    }
  });
}

handleOpenModalLogin = () => {
  if (this.state.isLoggin) {
    this.setState({ isLoggin: false, userID: null, userEmail: null });
    this.context.logout();
    CreateNotification({ successful: true, message: "Успішний вихід"
  });
  } else {
    this.setState({
      openLogin: true,
    });
  }
};

handleCloseModalLogin = () => {
  this.setState({
    openLogin: false,
  });
};

render() {
  return (
    <Fragment>
      <ModalAuthorization
        open={this.state.openLogin}
        handleCloseModal={this.handleCloseModalLogin}
        login={this.loginSuccess}
      />
      <NaviBar
        open={this.handleOpenModalLogin}
        isLoggin={this.state.isLoggin}
        email={this.state.userEmail}
      />
      <Routes>
        <Route exact path="/" element={<Schedule></Schedule>} />
        <Route path="/schedules" element={<Schedule></Schedule>} />
      </Routes>
    </Fragment>
  );
}

```

```

        {this.state.isLoggin && (
            <Fragment>
                <Route path="/cathedras" element={<Cathedra></Cathedra>}
            />
                <Route path="/audiences" element={<Audience></Audience>}
            />
                <Route path="/specialties"
            element={<Specialty></Specialty>} />
                <Route path="/disciplines"
            element={<Discipline></Discipline>} />
                <Route path="/classes" element={<Class></Class>} />
                <Route path="/groups" element={<Group></Group>} />
                <Route path="/teachers" element={<Teacher></Teacher>} />
                <Route
                    path="/admin_general_values"
                    element={<GeneralValues></GeneralValues>}
                />
                <Route
                    path="/admin_management_data"
                    element={<ManagementData></ManagementData>}
                />
                <Route
                    path="/admin_submit_data_sheet"
                    element={<SubmitDataSheet></SubmitDataSheet>}
                />
                <Route
                    path="/admin_schedule_data"
                    element={<ScheduleData></ScheduleData>}
                />
                <Route
                    path="/admin_algorithms"
                    element={<Algorithms></Algorithms>}
                />
            </Fragment>
        )}
        <Route path="*" element={<Error></Error>} />
    </Routes>
    <NotificationContainer />
</Fragment>
);
}
}
App.contextType = AuthContext;
export default App;

```

## В.4 Файл App.css

```

li {
    list-style-type: none;
}

```

## В.5 Файл index.html

```

<!DOCTYPE html>
<html lang="ua">
    <head>

```

```

    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1"
  />
    <meta
      name="description"
      content="Web site created using create-react-app"
    />
    <title>Schedule</title>
  </head>
  <body>
    <div id="root"></div>
  </body>
</html>

```

## B.6 Фрагмент файла ValidatedMessage.js

```

export default function ValidatedMessage({ message }) {
  return <div className="text-danger">{message}</div>;
}

```

## B.7 Фрагмент файла Alert.js

```

import { NotificationManager } from "react-notifications";

export const CreateNotification = (message) => {
  if (message.successful) {
    return NotificationManager.success(message.message, "Успіх", 3000);
  } else return NotificationManager.error(message.message, "Помилка",
3000);
};

```

## B.8 Фрагмент файла Teacher/Teacher.js

```

class Teacher extends React.Component {
  state = {
    filters: {
      surname: "",
      id_cathedra: null,
    },
  };
  handleChangeFilters = (name, value) => {
    this.setState((PrevState) => ({
      filters: Object.assign({ ...PrevState.filters }, { [name]: value
    })),
    {}));
  };
  render() {
    const { filters } = this.state;
    return (
      <>
        <TeacherSearch
          handleChangeFilters={this.handleChangeFilters}
        ></TeacherSearch>
      </>
    );
  }
}

```

```

        <TeacherTable filters={filters}></TeacherTable>
      </>
    );
  }
}

export default Teacher;

```

## B.9 Фрагмент файла Specialty/Specialty.js

```

class Specialty extends React.Component {
  state = {
    filters: {
      name: "",
      id_cathedra: null,
    },
  };

  handleChangeFilters = (name, value) => {
    this.setState((PrevState) => ({
      filters: Object.assign({ ...PrevState.filters }, { [name]: value
    })),
  }));
};

render() {
  const { filters } = this.state;
  return (
    <>
      <SpecialtySearch
        handleChangeFilters={this.handleChangeFilters}
      ></SpecialtySearch>
      <SpecialtyTable filters={filters}></SpecialtyTable>
    </>
  );
}
}

export default Specialty;

```

## B.10 Фрагмент файла Schedule/Schedule.js

```

function ScheduleTables({ filters }) {
  const { loading, error, data } = useQuery(GET_INFO,);
  if (loading) return null;
  if (error) return `Error! ${error}`;
  const info = data.GetInfo;

  function SwitchTable({ filters }) {
    switch (filters.scheduleType) {
      case "teacher":
        return <ScheduleTableTeacher filters={filters}
info={info}></ScheduleTableTeacher>
      case "audience":
        return <ScheduleTableAudience filters={filters}
info={info}></ScheduleTableAudience>
    }
  }
}

```

```

        case "group":
            return <ScheduleTableGroup filters={filters}>
info={info}></ScheduleTableGroup>;
        default:
            return <ScheduleTableGroup filters={filters}>
info={info}></ScheduleTableGroup>;
    }
}

return (
    <div className="container-fluid w-100">
        <SwitchTable filters={filters}></SwitchTable>
    </div>
);
}

class Schedule extends React.Component {
    state = {
        filters: {
            scheduleType: "group",
            id_specialty: null,
            id_teacher: null,
            id_group: null,
            id_audience: null,
            id_cathedral: null,
            semester: null,
        },
    };

    handleChangeFilters = (name, value) => {
        this.setState((PrevState) => ({
            filters: update(PrevState.filters, { $merge: { [name]: value } }),
        }));
    };

    render() {
        const { filters } = this.state;
        return (
            <>
                <ScheduleSearch
                    filters={filters}
                    handleChangeFilters={this.handleChangeFilters}
                ></ScheduleSearch>
                <ScheduleTables filters={filters}>
                </ScheduleTables>
            </>
        );
    }
}

export default Schedule;

```

## B.11 Фрагмент файла Navibar/Navibar.js

```

class Navibar extends React.Component {
    render() {
        const { open, isLoggin, email } = this.props;
        return (
            <Navbar variant="dark" bg="dark" expand="lg">

```

```

2)??.at(1)}}`}

<Container fluid>
  <Navbar.Brand>NUZP-Розклад</Navbar.Brand>
  <Navbar.Toggle aria-controls="basic-navbar-nav" />
  <Navbar.Collapse className="justify-content-between ">
    <Nav
      variant="tabs"
      activeKey={`/${window.location.pathname.split("/",
      className="mx-0 align-items-center"
    >
      <Nav.Link
        className="text-white-50 bg-dark"
        to="/schedules"
        href="/schedules"
      >
        Розклад
      </Nav.Link>
      {isLoggedIn && (
        <Fragment>
          <Nav.Link
            className=" text-white-50 bg-dark"
            to="/cathedras"
            href="/cathedras"
          >
            Кафедри
          </Nav.Link>
          <Nav.Link
            className=" text-white-50 bg-dark"
            to="/teachers"
            href="/teachers"
          >
            Викладачі
          </Nav.Link>
          <Nav.Link
            className=" text-white-50 bg-dark"
            to="/specialties"
            href="/specialties"
          >
            Спеціальності
          </Nav.Link>
          <Nav.Link
            className="text-white-50 bg-dark"
            to="/groups"
            href="/groups"
          >
            Групи
          </Nav.Link>
          <Nav.Link
            as={Link}
            className="text-white-50 bg-dark"
            to="/audiences"
          >
            Аудиторії
          </Nav.Link>
          <Nav.Link
            className="text-white-50 bg-dark"
            to="/disciplines"
            href="/disciplines"
          >
            Дисципліни
          </Nav.Link>
          <Nav.Link
            className=" text-white-50 bg-dark"

```

```

        to="/classes"
        href="/classes"
    >
        Заняття
    </Nav.Link>

    <Dropdown className="text-white-50 bg-dark ">
        <Dropdown.Toggle as={Nav.Link}>
            Адміністрування
        </Dropdown.Toggle>
        <Dropdown.Menu className="text-white-50 bg-dark">
            <Dropdown.Item
                as={Nav.Link}
                className="my-1 text-white-50 bg-dark"
                href="/admin_algorithms"
                to="/admin_algorithms"
            >
                Запуск алгоритмів
            </Dropdown.Item>
            <Dropdown.Item
                as={Nav.Link}
                href="/admin_schedule_data"
                to="/admin_schedule_data"
                className=" my-1 text-white-50 bg-dark"
            >
                Оцінка поточного розкладу
            </Dropdown.Item>
            <Dropdown.Item
                as={Nav.Link}
                className="my-1 text-white-50 bg-dark"
                href="/admin_general_values"
                to="/admin_general_values"
            >
                Загальні дані
            </Dropdown.Item>

            <Dropdown.Item
                as={Nav.Link}
                className="my-1 text-white-50 bg-dark"
                href="/admin_management_data"
                to="/admin_management_data"
            >
                Видалення даних
            </Dropdown.Item>
            <Dropdown.Item
                as={Nav.Link}
                className="my-1 text-white-50 bg-dark"
                href="/admin_submit_data_sheet"
                to="/admin_submit_data_sheet"
            >
                Завантаження відомостей
            </Dropdown.Item>
        </Dropdown.Menu>
    </Dropdown>
</Fragment>
    )}
</Nav>
<Nav className="mx-2 ">
    <Navbar.Brand className="align-self-center">
        {isLoggedIn ? email : ""}
    </Navbar.Brand>
    <Button
        className="mx-1 my-1 align-self-center"

```

```

        variant="success"
        onClick={open}
      >
        {isLoggedIn ? "Вийти" : "Увійти"}
      </Button>
    </Nav>
  </Navbar.Collapse>
</Container>
</Navbar>
);
}
}
export default NavBar;

```

## В.12 Фрагмент файлу Group/Group.js

```

class Group extends React.Component {
  state = {
    filters: {
      name: "",
      id_specialty: null,
    },
  };

  handleChangeFilters = (name, value) => {
    this.setState((PrevState) => ({
      filters: Object.assign({ ...PrevState.filters }, { [name]: value
    })),
  ));
  };

  render() {
    const { filters } = this.state;
    return (
      <>
        <GroupSearch
          handleChangeFilters={this.handleChangeFilters}
        ></GroupSearch>
        <GroupTable
          filters={filters}
        ></GroupTable>
      </>
    );
  }
}

export default Group;

```

## В.13 Фрагмент файлу Error/Error.js

```

class Error extends React.Component {
  render() {
    return (
      <div>
        <Alert className="mx-2 my-2 text-center" variant="danger">
          Сторінку не знайдено
        </Alert>
      </div>
    );
  }
}

```

```

        </div>
    );
}
}
export default Error;

```

## B.14 Фрагмент файла Discipline/Discipline.js

```

class Discipline extends React.Component {
  state = {
    filters: {
      name: "",
      id_discipline: null,
    },
  };
  handleChangeFilters = (name, value) => {
    this.setState((PrevState) => ({
      filters: Object.assign({ ...PrevState.filters }, { [name]: value
    })),
    {}));
  };
  render() {
    const { filters } = this.state;
    return (
      <>
        <DisciplineSearch
          handleChangeFilters={this.handleChangeFilters}
        ></DisciplineSearch>
        <DisciplineTable filters={filters}></DisciplineTable>
      </>
    );
  }
}

export default Discipline;

```

## B.15 Фрагмент файла Class/Class.js

```

class Class extends React.Component {
  state = {
    filters: {
      id_discipline: null,
      id_teacher: null,
      id_group: null,
      id_specialty: null,
      semester: null,
    },
  };
  handleChangeFilters = (name, value) => {
    this.setState((PrevState) => ({
      filters: Object.assign({ ...PrevState.filters }, { [name]: value
    })),
    {}));
  };
}

```

```

render() {
  const { filters } = this.state;
  return (
    <>
      <ClassSearch
        handleChangeFilters={this.handleChangeFilters}
      ></ClassSearch>
      <ClassTable filters={filters}></ClassTable>
    </>
  );
}
}

export default Class;

```

## В.16 Фрагмент файла Cathedra/Cathedra.js

```

class Cathedra extends React.Component {
  state = {
    filters: {
      name: "",
    },
  };

  handleChangeFilters = (name, value) => {
    this.setState((PrevState) => ({
      filters: Object.assign({ ...PrevState.filters }, { [name]: value
    })),
  }));
  };

  render() {
    const { filters } = this.state;
    return (
      <>
        <CathedraSearch
          handleChangeFilters={this.handleChangeFilters}
        ></CathedraSearch>
        <CathedraTable filters={filters}></CathedraTable>
      </>
    );
  }
}

export default Cathedra;

```

## В.17 Фрагмент файла Audience/Audience.js

```

class Audience extends React.Component {
  state = {
    filters: {
      name: "",
      id_cathedra: null,
    },
  };
};

```

```

    handleChangeFilters = (name, value) => {
      this.setState((PrevState) => ({
        filters: Object.assign({ ...PrevState.filters }, { [name]: value
    })),
    }));
  };

  render() {
    const { filters } = this.state;
    return (
      <>
        <AudienceSearch
          handleChangeFilters={this.handleChangeFilters}
        ></AudienceSearch>
        <AudienceTable filters={filters}></AudienceTable>
      </>
    );
  }
}

export default Audience;

```

## B.18 Фрагмент файлу Admin/Algorithms/Algorithms.js

```

function GetLabelForResults(params, params_value) {
  let label = "";
  params = JSON.parse(params);
  params_value = JSON.parse(params_value);

  for (let i = 0; i < params.length - 1; i++) {
    label += `${params[i].short}: ${params_value[params[i].name]}, `;
  }
  if (params.length)
    label += `${params[params.length - 1].short}: ${
      params_value[params[params.length - 1].name]
    }, `;
  else label = "Без параметрів";
  return label;
}

function GetBestResults(algorithms) {
  let best_results = [];
  algorithms.forEach((obj) => {
    if (obj.results_algorithms.length) {
      let best_res = JSON.parse(obj.results_algorithms[0].results);
      for (let i = 0; i < obj.results_algorithms.length; i++) {
        let temp = JSON.parse(obj.results_algorithms[i].results);
        if (best_res[best_res.length - 1][1] > temp[temp.length - 1][1])
          best_res = temp;
      }
      best_results.push({
        name: obj.name,
        label: obj.label,
        results: JSON.stringify(best_res),
      });
    }
  });
  return best_results;
}

```

```

function ChooseAlgorithm({ state, handleChangeState }) {
  const { loading, error, data, refetch } = useQuery(GET_ALL_ALGORITHM);
  if (loading) return null;
  if (error) return `Error! ${error}`;
  let arrAlgorithm = data.GetAllAlgorithm;
  let best_results = GetBestResults(arrAlgorithm);
  let results_algorithms = [];
  if (state.nameAlgorithm) {
    arrAlgorithm.forEach((obj) => {
      if (obj.name === state.nameAlgorithm) {
        obj.results_algorithms.forEach((res) => {
          results_algorithms.push({
            name: res.params_value,
            label: GetLabelForResults(obj.params, res.params_value),
            results: res.results,
          });
        });
      }
    });
  }
  return (
    <>
      <div className="col-md-6 offset-md-3">
        <Card className="my-2">
          <Card.Header      className="text-center">Вибір
алгоритму</Card.Header>
          <Card.Footer>
            <Form.Group as={Row} className="my-2 mx-2">
              <SelectAlgoritm
                handleChangeState={handleChangeState}
                arrAlgorithm={arrAlgorithm}
              ></SelectAlgoritm>
            </Form.Group>
            <Form.Group as={Row} className="my-2 mx-2">
              <ButtonRunAlgorithm
                name={state.nameAlgorithm}
                id_cathedra={state.id_cathedra}
                refetch={refetch}
              ></ButtonRunAlgorithm>
            </Form.Group>
          </Card.Footer>
        </Card>
      </div>
      <AlgorithmForm
        state={state}
        handleChangeState={handleChangeState}
        refetch={refetch}
      ></AlgorithmForm>
      {results_algorithms?.length ? (
        <MultiCharts
          results={results_algorithms}
          nameAlgorithm={state.nameAlgorithm}
        ></MultiCharts>
      ) : null}
      <MultiCharts
        results={best_results}
        nameAlgorithm={"allResults"}
      ></MultiCharts>
    </>
  );
}

```

```

function AlgorithmForm({ state, handleChangeState, refetch }) {
  let { nameAlgorithm, params } = { ...state };
  return (
    <>
      {nameAlgorithm ? (
        <>
          <div className="col-md-6 offset-md-3">
            <Card className="my-2">
              <Card.Header className="text-center">
                {"Складання розклад за допомогою " + state.label}
              </Card.Header>
              <Card.Body>
                {params.map((param) => (
                  <Form.Group as={Row} className="my-2 mx-2">
                    <Form.Label className="col-md-
4">{param.label}</Form.Label>
                    <Col className="col-md-8">
                      {param.type == "select" ? (
                        <Select
                          value={{
                            value: param.value,
                            label: param.options.find(
                              (opt) => opt.value === param.value
                            ).label,
                          }}
                          options={param.options}
                          onChange={(e) => {
                            debugger;
                            param.value = e.value;
                            handleChangeState("params", params);
                          }}
                        ></Select>
                      ) : (
                        <Form.Control
                          value={param.value}
                          type={param.type}
                          min={param.min}
                          max={param.max}
                          step={param.step}
                          onChange={(e) => {
                            param.value = +e.target.value;
                            handleChangeState("params", params);
                          }}
                        />
                      )}
                    </Col>
                  </Form.Group>
                )]}
              </Card.Body>
              <Card.Footer>
                {params.length ? (
                  <ButtonUpdateAlgorithm
                    name={nameAlgorithm}
                    params={JSON.stringify(params)}
                    refetch={refetch}
                  />
                ) : null}
                <ButtonDeleteResults
                  name_algorithm={nameAlgorithm}
                  refetch={refetch}
                ></ButtonDeleteResults>
              </Card.Footer>
            </div>
          </>
        ) : null}
      </>
    )
  );
}

```

```

        </Card>
      </div>
    </>
  ) : null}
</>
);
}

export default class Algorithms extends React.Component {
  constructor(args) {
    super(args);
    this.state = {
      id_cathedral: null,
      nameAlgorithm: null,
      label: null,
      params: null,
      results_algorithms: null,
    };
  }

  handleChangeState = (name, item) => {
    this.setState({ [name]: item });
  };

  render() {
    return (
      <>
        <div className="col-md-6 offset-md-3">
          <Card className="my-2">
            <Card.Header className="text-center">Ви́бір
кафедри</Card.Header>
            <Card.Footer>
              <Form.Group as={Row} className="my-2 mx-2">
                <SelectCathedral
                  handleChangeState={this.handleChangeState}
                  id_cathedral={this.state.id_cathedral}
                ></SelectCathedral>
              </Form.Group>
            </Card.Footer>
          </Card>
        </div>
        <ChooseAlgorithm
          state={this.state}
          handleChangeState={this.handleChangeState}
        />
      </>
    );
  }
}

```

### В.19 Фрагмент файла Admin/GeneralValues/GeneralValues.js

```

function DataForm({ handleChangeState, info }) {
  const { loading, error, data, refetch } = useQuery(GET_INFO);
  if (loading) return null;
  if (error) return `Error! ${error}`;
  let general_values =
    info.general_values !== null
      ? info.general_values
      : JSON.parse(data.GetInfo.general_values);
}

```

```

        let max_day = info.max_day !== null ? info.max_day :
data.GetInfo.max_day;
        let max_pair = info.max_pair !== null ? info.max_pair :
data.GetInfo.max_pair;
        if (!general_values) {
            general_values = {};
        }
        return (
            <>
            <Form.Group as={Row} className="my-2 mx-2">
                <Form.Label className="col-md-3">Вага: вікна групи</Form.Label>
                <Col className="col-md-9">
                    <Form.Control
                        value={general_values?.penaltyGrWin}
                        type="number"
                        min={0}
                        onChange={(e) => {
                            general_values.penaltyGrWin = Number(e.target.value);
                            handleChangeState("general_values", general_values);
                        }}
                    />
                </Col>
            </Form.Group>
            <Form.Group as={Row} className="my-2 mx-2">
                <Form.Label className="col-md-3">Вага: вікна
викладачів</Form.Label>
                <Col className="col-md-9">
                    <Form.Control
                        value={general_values?.penaltyTeachWin}
                        type="number"
                        min={0}
                        onChange={(e) => {
                            general_values.penaltyTeachWin = Number(e.target.value);
                            handleChangeState("general_values", general_values);
                        }}
                    />
                </Col>
            </Form.Group>
            <Form.Group as={Row} className="my-2 mx-2">
                <Form.Label className="col-md-3">Вага: пізні
заняття</Form.Label>
                <Col className="col-md-9">
                    <Form.Control
                        value={general_values?.penaltyLateSc}
                        type="number"
                        min={0}
                        onChange={(e) => {
                            general_values.penaltyLateSc = Number(e.target.value);
                            handleChangeState("general_values", general_values);
                        }}
                    />
                </Col>
            </Form.Group>
            <Form.Group as={Row} className="my-2 mx-2">
                <Form.Label className="col-md-3">Вага: рівномірний
розклад</Form.Label>
                <Col className="col-md-9">
                    <Form.Control
                        value={general_values?.penaltyEqSc}
                        type="number"
                        min={0}
                        onChange={(e) => {
                            general_values.penaltyEqSc = Number(e.target.value);

```

```

        handleChangeState("general_values", general_values);
    }}
    />
</Col>
</Form.Group>
<Form.Group as={Row} className="my-2 mx-2">
    <Form.Label className="col-md-3">Вага:    накладання
занять</Form.Label>
    <Col className="col-md-9">
        <Form.Control
            value={general_values?.penaltySameTimesSc}
            type="number"
            min={0}
            onChange={(e) => {
                general_values.penaltySameTimesSc = Number(e.target.value);
                handleChangeState("general_values", general_values);
            }}
        />
    </Col>
</Form.Group>
<Form.Group as={Row} className="my-2 mx-2">
    <Form.Label className="col-md-3">Вага:    співпадіння
занять</Form.Label>
    <Col className="col-md-9">
        <Form.Control
            value={general_values?.penaltySameRecSc}
            type="number"
            min={0}
            onChange={(e) => {
                general_values.penaltySameRecSc = Number(e.target.value);
                handleChangeState("general_values", general_values);
            }}
        />
    </Col>
</Form.Group>
<Form.Group as={Row} className="my-2 mx-2">
    <Form.Label className="col-md-3">Кількість днів</Form.Label>
    <Col className="col-md-9">
        <Form.Control
            value={max_day}
            type="number"
            min={1}
            onChange={(e) => {
                handleChangeState("max_day", e.target.value);
            }}
        />
    </Col>
</Form.Group>
<Form.Group as={Row} className="my-2 mx-2">
    <Form.Label className="col-md-3">Кількість занять</Form.Label>
    <Col className="col-md-9">
        <Form.Control
            value={max_pair}
            type="number"
            min={1}
            onChange={(e) => {
                handleChangeState("max_pair", e.target.value);
            }}
        />
    </Col>
</Form.Group>
<Form.Group as={Row} className="my-2 mx-2">

```

```

refetch={refetch}></ButtonUpdateInfo>
    </Form.Group>
  </>
);
}

export default class GeneralValues extends React.Component {
  constructor(args) {
    super(args);
    this.state = {
      general_values: null,
      max_pair: null,
      max_day: null,
    };
  }

  handleChangeState = (name, item) => {
    this.setState({ [name]: item });
  };

  render() {
    return (
      <>
        <div className="col-md-6 offset-md-3">
          <Card className="my-2">
            <Card.Header className="text-center">
              Загальні дані для алгоритмів
            </Card.Header>
            <Card.Body>
              <DataForm
                handleChangeState={this.handleChangeState}
                info={this.state}
              ></DataForm>
            </Card.Body>
          </Card>
        </div>
      </>
    );
  }
}

```

## **В.20 Фрагмент файлу Admin/ManegementData/DeleteData.js**

```

export default class DeleteData extends React.Component {
  constructor(args) {
    super(args);
    this.state = {
      id_cathedral: null,
    };
  }

  handleChangeState = (name, item) => {
    this.setState({ [name]: item });
  };

  render() {
    return (
      <Card>
        <Card.Header className="d-flex justify-content-center m-2">

```

```

        Видалення даних
    </Card.Header>
    <Card.Body>
        <Form.Group as={Row} className=" mx-2">
            <SelectCathedra
                id_cathedra={this.state.id_cathedra}
                handleChangeState={this.handleChangeState}
            ></SelectCathedra>
        </Form.Group>
    </Card.Body>
    <Card.Footer>
        <Form.Group as={Row} className="my-2 mx-4">
            <ButtonDeleteAllData
                id_cathedra={this.state.id_cathedra}
            ></ButtonDeleteAllData>
        </Form.Group>

        <Form.Group as={Row} className="my-2 mx-4">
            <GetSchedule></GetSchedule>
        </Form.Group>
    </Card.Footer>
</Card>
    );
}
}

```

### **В.21 Фрагмент файлу Admin/ManegementData/ManegementData.js**

```

export default class ManagementData extends React.Component {
    render() {
        return (
            <>
                <div className="col-md-6 offset-md-3">
                    <DeleteData></DeleteData>
                </div>
            </>
        );
    }
}

```

### **В.22 Фрагмент файлу Admin/ScheduleData/ScheduleData.js**

```

function DataForm() {
    const { loading, error, data, refetch } = useQuery(GET_INFO);
    if (loading) return null;
    if (error) return `Error! ${error}`;
    let fitnessValue = JSON.parse(data.GetInfo.fitness_value);
    if (!fitnessValue) {
        fitnessValue = {};
    }
    return (
        <>
            <Form.Group as={Row} className="my-2 mx-2">
                <Form.Label className="col-md-3">Загальний фітнес</Form.Label>
                <Col className="col-md-9">
                    <Form.Control value={fitnessValue?.fitnessValue} readOnly />
                </Col>
            </Form.Group>
        </>
    );
}

```

```

        </Col>
    </Form.Group>
    <Form.Group as={Row} className="my-2 mx-2">
        <Form.Label className="col-md-3">Фітнес рек. час</Form.Label>
        <Col className="col-md-9">
            <Form.Control value={fitnessValue?.fitnessSameRecSc} readOnly
/>

        </Col>
    </Form.Group>
    <Form.Group as={Row} className="my-2 mx-2">
        <Form.Label className="col-md-3">Фітнес груп:</Form.Label>
    </Form.Group>
    <Form.Group as={Row} className="my-2 mx-2">
        <Form.Label className="col-md-3">Загальний</Form.Label>
        <Col className="col-md-9">
            <Form.Control
                value={fitnessValue?.fitnessGr?.fitnessValue}
                readOnly
            />
        </Col>
    </Form.Group>
    <Form.Group as={Row} className="my-2 mx-2">
        <Form.Label className="col-md-3">Вікна</Form.Label>
        <Col className="col-md-9">
            <Form.Control
                value={fitnessValue?.fitnessGr?.fitnessGrWin}
                readOnly
            />
        </Col>
    </Form.Group>
    <Form.Group as={Row} className="my-2 mx-2">
        <Form.Label className="col-md-3">Накладання занять</Form.Label>
        <Col className="col-md-9">
            <Form.Control
                value={fitnessValue?.fitnessGr?.fitnessSameTimesSc}
                readOnly
            />
        </Col>
    </Form.Group>
    <Form.Group as={Row} className="my-2 mx-2">
        <Form.Label className="col-md-3">Фітнес викладачів:</Form.Label>
    </Form.Group>
    <Form.Group as={Row} className="my-2 mx-2">
        <Form.Label className="col-md-3">Загальний</Form.Label>
        <Col className="col-md-9">
            <Form.Control
                value={fitnessValue?.fitnessTeach?.fitnessValue}
                readOnly
            />
        </Col>
    </Form.Group>
    <Form.Group as={Row} className="my-2 mx-2">
        <Form.Label className="col-md-3">Вікна</Form.Label>
        <Col className="col-md-9">
            <Form.Control
                value={fitnessValue?.fitnessTeach?.fitnessTeachWin}
                readOnly
            />
        </Col>
    </Form.Group>
    <Form.Group as={Row} className="my-2 mx-2">
        <Form.Label className="col-md-3">Накладання занять</Form.Label>
        <Col className="col-md-9">

```

```

        <Form.Control
          value={fitnessValue?.fitnessTeach?.fitnessSameTimesSc}
          readOnly
        />
      </Col>
    </Form.Group>
    <Form.Group as={Row} className="my-2 mx-2">
      <Form.Label className="col-md-3">Фітнес аудиторій:</Form.Label>
    </Form.Group>
    <Form.Group as={Row} className="my-2 mx-2">
      <Form.Label className="col-md-3">Накладання занять</Form.Label>
      <Col className="col-md-9">
        <Form.Control
          value={fitnessValue?.fitnessAud?.fitnessSameTimesSc}
          readOnly
        />
      </Col>
    </Form.Group>

    <Form.Group as={Row} className="my-2 mx-2">
      <ButtonCalcFitness refetch={refetch}></ButtonCalcFitness>
    </Form.Group>
  </>
);
}
export default class ScheduleData extends React.Component {
  render() {
    return (
      <>
        <div className="col-md-6 offset-md-3">
          <Card className="my-2">
            <Card.Header className="text-center">
              Оцінка поточного розкладу
            </Card.Header>
            <Card.Body>
              <DataForm
                handleChangeState={this.handleChangeState}></DataForm>
            </Card.Body>
          </Card>
        </div>
      </>
    );
  }
}

```

**ДОДАТОК Г**  
**Слайди презентації**

Національний університет «Запорізька політехніка»

Кафедра програмних засобів

Випускна робота бакалавра на тему:

## **Розробка вебзастосунку для створення та керування розкладом. Frontend частина**

Виконав:  
студент групи КНТ-219

К. В. МЄДВЄДЄВ

Керівник:  
доцент, к.т.н, доцент

О. О. СТЕПАНЕНКО

2023

Рисунок Г.1 – Слайд № 1

### **Мета, об'єкт, предмет і задачі дослідження**

2

Об'єктом дослідження є процес складання розкладу для університету, включаючи процес планування, підготовки та оновлення розкладу.

Предметом дослідження є розробка системи для автоматизованого складання розкладу, яка забезпечує оптимальне використання ресурсів, враховує потреб студентів та викладачів, дозволяє знизити час та уникнути кількості помилок.

Метою роботи є розробка та впровадження системи для автоматизованого складання розкладу для університету з метою забезпечення оптимального та точного складання розкладу занять, зменшення часу на підготовку до навчального семестру та уникнення помилок при створенні розкладу навчального процесу.

На цей час існує багато різних систем, що допомагають у розв'язанні цієї задачі. Однак, деякі з цих систем мають свої недоліки, такі як складність у використанні або невідповідність потребам користувачів. Тому, актуальним завданням є розробка нової системи для автоматизованого складання розкладу занять для ВНЗ, яка б враховувала всі потреби користувачів та дозволяла знизити час та зусилля, необхідні для складання розкладу.

Рисунок Г.2 – Слайд № 2

| Огляд наявних систем                    |                          |                  |                           |
|-----------------------------------------|--------------------------|------------------|---------------------------|
| Характеристика                          | FET                      | UniTime          | aSc TimeTables            |
| Відкритий код                           | Так                      | Так              | Ні                        |
| Операційна система                      | Кросплатформна           | Кросплатформна   | Windows                   |
| Підтримка різних мов                    | Багатомовний             | Багатомовний     | Багатомовний              |
| Наявність ліцензії                      | Ні                       | Ні               | Так                       |
| Адаптивність до різних типів установ    | Так                      | Так              | Так                       |
| Тип програми                            | Десктопна                | Вебдодаток       | Десктопна                 |
| Автоматичне створення розкладу          | Так                      | Так              | Так                       |
| Можливість ручного редагування розкладу | Так                      | Так              | Так                       |
| Можливість імпорту/експорту даних       | Так                      | Так              | Так                       |
| Формат збереження даних                 | Файли з розширенням .fet | База даних MySQL | Формат з розширенням .tsm |
| Редагування даних                       | Так                      | Так              | Так                       |
| Фільтрація та сортування даних          | Так                      | Так              | Так                       |
| Формування звітів                       | Так                      | Так              | Так                       |

Рисунок Г.3 – Слайд № 3

### Постановка завдання до роботи

Метою роботи є дослідження та реалізація автоматизованої системи для складання розкладу занять для університету.

Вхідними даними для системи будуть: дані про аудиторії, дані про кафедри, дані про спеціальності, дані про дисципліни, дані про групи, дані про викладачів, дані про заняття.

Вихідними даними буде оптимальний розклад занять, складений для обраної кафедри, в якому будуть враховуватися рекомендації до проведення та мінімізовано наступні параметри: кількість накладань занять для груп, вчителів та аудиторій, кількість вікон між заняття для груп та вчителів.

До системи висуваються наступні вимоги: перегляд даних, редагування та видалення вхідних даних, ручне та автоматичне ведення вхідних даних, фільтрація даних, вибір та запуск алгоритму, налаштування параметрів алгоритму, порівняння роботи алгоритму з різними параметрами.

Рисунок Г.4 – Слайд № 4

| Вибір мови програмування                       |                                                                    |                                                                   |                                                                        |
|------------------------------------------------|--------------------------------------------------------------------|-------------------------------------------------------------------|------------------------------------------------------------------------|
| Критерії порівняння                            | C++                                                                | JavaScript                                                        | Python                                                                 |
| Складність синтаксису                          | Складний                                                           | Середній                                                          | Простий                                                                |
| Швидкість виконання                            | Швидка                                                             | Помірна                                                           | Повільна                                                               |
| Керованість пам'яті                            | Вручну                                                             | Автоматично                                                       | Автоматично                                                            |
| Ефективність виконання багатопотокових програм | Висока                                                             | Помірна                                                           | Помірна                                                                |
| Наявність фреймворків для веброзробки          | Більш призначений для системного програмування                     | Багато вебфреймворків, таких як Angular, React, Vue.js та Node.js | Доступний великий вибір вебфреймворків, таких як Django, Flask та інші |
| Кросплатформність                              | Компілятори для цієї мови доступні для багатьох операційних систем | Може працювати на будь-якому пристрої з підтримкою браузера       | Інтерпретатор Python доступний для багатьох операційних систем         |

Рисунок Г.5 – Слайд № 5



Рисунок Г.6 – Слайд № 6

## Вибір фреймворку для реалізації клієнтської частини

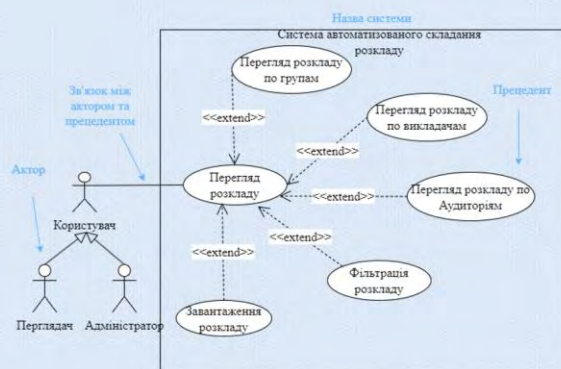
| Характеристика                | React    | Angular  | Vue.js   |
|-------------------------------|----------|----------|----------|
| Розмір                        | Середній | Великий  | Середній |
| Швидкодія                     | Швидкий  | Помірний | Швидкий  |
| Компонентна архітектура       | +        | +        | +        |
| Складність вивчення           | Середня  | Висока   | Середня  |
| Маршрутизація                 | +        | +        | +        |
| Наявність повної документації | +        | +        | +        |

Порівнявши фреймворки, було вирішено розробити клієнтську частину використовуючи React через його високу гнучкість та простоту.

Порівнявши наступні протоколи API: REST, GraphQL, SOAP, JSON-RPC, було обрано GraphQL через те, що GraphQL дозволяє отримувати точно ті дані, які потрібні клієнту, та надає можливість кешування даних на стороні клієнта, що дозволить зменшити кількість запитів до сервера та покращити швидкість роботи додатка

Рисунок Г.7 – Слайд № 7

## Модель взаємодії користувача з системою



Діаграма прецедентів частина 1



Діаграма прецедентів частина 2

Рисунок Г.8 – Слайд № 8

## Модель взаємодії користувача з системою

9



Діаграма прецедентів частина 3

На діаграмі прецедентів присутні наступні актори: переглядач, адміністратор, користувач.

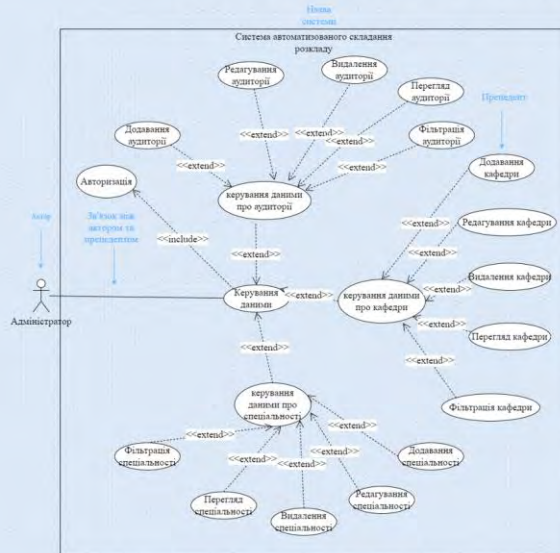
Користувач переглядає розклад. Додатково він може переглядати розклад по групах, викладачам, аудиторіям, фільтрувати розклад, завантажувати розклад.

Адміністратор взаємодіє з наступними прецедентами: завантаження відомості даних, перегляд якості поточного розкладу, редагування параметрів алгоритму, керування алгоритмами, який розширюється запуск алгоритмів та редагування параметрів алгоритмів, керування даними, які вимагають виконання прецеденту Авторизація.

Рисунок Г.9 – Слайд № 9

## Модель взаємодії користувача з системою

10



Діаграма прецедентів частина 4

Виконання прецеденту Керування даними може бути розширене наступними прецедентами: керування даними про дисципліни, заняття, групи, викладача, аудиторії, кафедри, спеціальності. Всі ці прецеденти включають наступні розширення: видалення, додавання, редагування, перегляд, фільтрація даних.

Рисунок Г.10 – Слайд № 10

## Опис застосування програми

10

Вебзастосунок для складання та керування розкладу занять є інструментом для планування розкладу занять в університеті. Ця програма допомагає автоматизувати процес створення розкладу, забезпечуючи оптимальну організацію часу та ресурсів. Програма виконує такі функції: перегляд розкладу занять, авторизація користувача для адміністрування даних, ручне управління даними про групи, аудиторії, дисципліни, кафедри, спеціальності, викладачів, заняття, алгоритми, запуск алгоритмів для складання розкладу, автоматичне завантаження даних з навчальних відомостей, перегляд результатів роботи алгоритмів у вигляді графіків.

Для того, щоб почати роботу з вебзастосунком, треба перейти за адресою <http://www.nuzp.pp.ua/>.

| Група   | Понеділок                                         | Вівторок | Середа | Четверг | П'ятниця                                          | Субота                                            |
|---------|---------------------------------------------------|----------|--------|---------|---------------------------------------------------|---------------------------------------------------|
| 801-110 | Решіть від 57 Основар програмування Дойменг 8.10. |          |        |         |                                                   |                                                   |
| 2       |                                                   |          |        |         | Решіть від 45 Основар програмування Лисенко 8.10. |                                                   |
| 3       |                                                   |          |        |         |                                                   | Решіть від 37 Основар програмування Дойменг 8.10. |

Головна сторінка вебдодатку для перегляду розкладу

Рисунок Г.11 – Слайд № 11

## Приклади застосування програми

11

Авторизація користувача

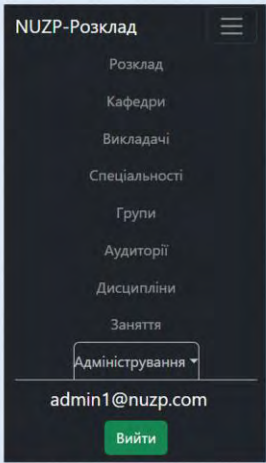
| Дисципліна                                          | Спеціальність - Семестр                                    |     |
|-----------------------------------------------------|------------------------------------------------------------|-----|
| САО/САМС/САЕ системи                                | 801 - 0 - 2<br>801 - 0 - 1<br>801 - 2 - 14<br>801 - 1 - 14 | ☑ ☹ |
| Будів системи                                       | 801 - 1 - 14                                               | ☑ ☹ |
| Алгоритми та структури даних                        | 801 - 1 - 8<br>801 - 1 - 6                                 | ☑ ☹ |
| Алгоритми та програмування                          | 801 - 2 - 6                                                | ☑ ☹ |
| Аналіз витрат до програмування забезпечення         | 801 - 1 - 10                                               | ☑ ☹ |
| Алгоритми та програмування програмного забезпечення | 801 - 1 - 10                                               | ☑ ☹ |

Сторінка з даними про дисципліни

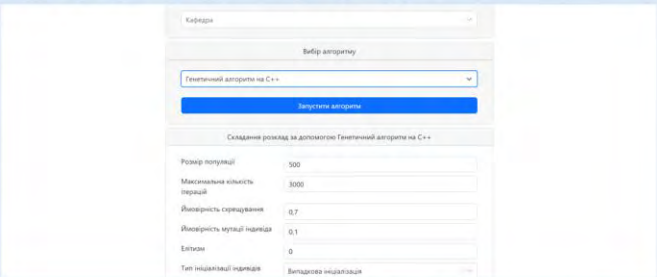
Додавання запису дисципліни

Рисунок Г.12 – Слайд № 12

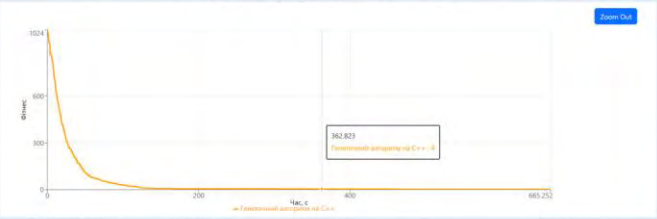
## Приклади застосування програми



Навігація додатку



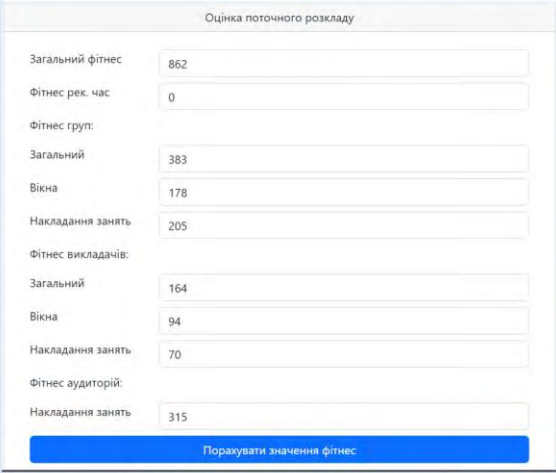
Сторінка запуск алгоритмів



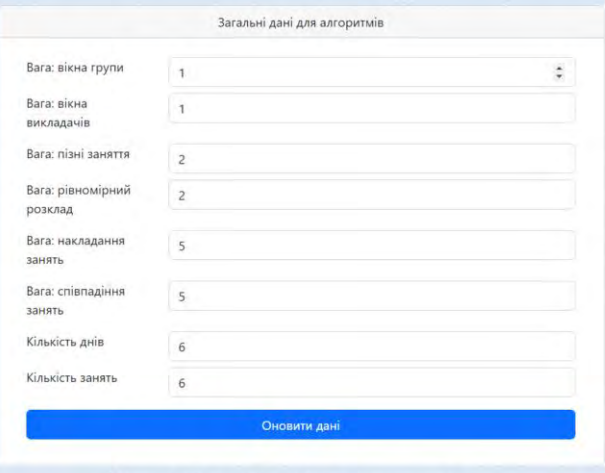
Графік роботи алгоритму

Рисунок Г.13 – Слайд № 13

## Приклади застосування програми



Форма оцінка поточного розкладу



Форма Загальні дані для алгоритмів

Рисунок Г.14 – Слайд № 14

## Приклади застосування програми

15

Відомість доручень

Вибрати файл    Файл не вибрано

Кафедра

Завантажити шаблон відомості

Завантажити дані

Форма для завантаження даних

Видалення даних

Кафедра

Видалити

Завантажити дані розкладу в json для алгоритму

Форма для видалення даних

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет «Запорізька політехніка»

ВІДОМІСТЬ

доручень на \_\_\_\_\_ семістр \_\_\_\_\_ навчального року

КАФЕДРА \_\_\_\_\_

| № з/п | Назва дисципліни | № групи | Вид заняття | Загальна кількість годин за навчання | Кількість годин на тиждень за навчання | Контроль | Прізвище І.П. По безпосередньому викладачів | Примітки до складання розкладу занять |
|-------|------------------|---------|-------------|--------------------------------------|----------------------------------------|----------|---------------------------------------------|---------------------------------------|
| 1     | 2                | 3       | 4           | 5                                    | 6                                      | 7        | 8                                           | 9                                     |

Шаблон відомості доручень

Рисунок Г.15 – Слайд № 15

13

## Висновок

Під час виконання роботи було виконано дослідження та програмна реалізація системи для автоматизованого складання розкладу занять для університету.

Було розроблено вебдодаток для створення та керування розкладом навчальних занять. Розроблене програмне забезпечення може бути використане в навчальних закладах для автоматизації процесу складання розкладу занять. Рекомендується його впровадження для покращення ефективності та зменшення затрат часу на складання розкладу.

Рисунок Г.16 – Слайд № 16