Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення

Бакалаврська дипломна робота

на тему: «Розробка методу та програмного засобу для оптимізації мережі громадського транспорту міста»

Виконав: студент IV курсу
групи ЗПІ-20б
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Якубенко О. В.
(прізвище та ініціали)
Керівник: к.т.н., доц. каф. ПЗ Ракитянська Г.Б.
(прізвище та ініціали)
Рецензент: к.т.н., доц. каф ЗІ Баришев Ю.В.
(прізвище та ініціали)

Допущено до захисту

Зав. кафедри ПЗ Романюк О.Н.
«10» 06 2024 р.

ВНТУ – 2024

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти перший (бакалаврський)
Галузь знань 12 «Інформаційні технології»
Спеціальність 121 «Інженерія програмного забезпечення»
Освітньо-професійна програма «Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
О. Н. Романюк
«12» березня 2024 р.

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Якубенку Олександру Владиславовичу

1. Тема роботи – розробка методу та програмного засобу для оптимізації мережі громадського транспорту міста.

Керівник роботи: Ракитянська Ганна Борисівна, к. т. н., доцент кафедри ПЗ, затверджені наказом вищого навчального закладу від 11 березня 2024 р. № 80.

2. Строк подання студентом роботи 3 червня 2024.

3. Вихідні дані до роботи: засоби та методи оптимізації мережі громадського транспорту міста, середовище розробки – IntelliJ IDEA Ultimate, мова розробки – Java, засіб адміністрування бази даних – Database Tools and SQL та MongoDB Compass.

4. Зміст текстової частини: вступ; аналіз стану розвитку систем для оптимізації мережі громадського транспорту та постановка задач дослідження; розробка методу, моделі та алгоритмів програмного продукту; розробка програмних модулів реалізації системи оптимізації мережі громадського транспорту; тестування програми; висновки; список використаних джерел; додатки; ілюстративна частина.

5. Перелік ілюстративного матеріалу: титульний слайд; актуальність розробки; мета, предмет та об'єкт дослідження; постановка задачі; новизна; практична цінність отриманих результатів; існуючі аналоги ч.1-4; аналіз аналогів; схема роботи адаптивного генетичного алгоритму; модель системи на прикладі діаграми діяльності; алгоритми системи ч.1-2; розробка графічного

інтерфейсу; використані технології; демонстрація роботи ч.1-8; апробація та публікація результатів роботи; висновки; фінальний слайд.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1-4	Ракитянська Г. Б., к.т.н., доцент кафедри ПЗ	13.03.24	03.06.24

7. Дата видачі завдання 13 березня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

Ч.ч.	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз розвитку систем для оптимізації мережі громадського транспорту та постановка задач дослідження	16.03.24 – 27.03.24	вик.
2	Розробка методу, моделі та алгоритмів програмного продукту	26.03.24 – 19.04.24	вик.
3	Розробка програмних модулів реалізації системи з оптимізації мережі громадського транспорту	20.04.24 – 10.04.24	вик.
4	Тестування програми	11.05.24 – 16.05.24	вик.
5	Оформлення матеріалів до захисту БДР	17.05.24 – 30.05.24	вик.

Студент _____ **О. В. Якубенко**
(підпис) (ініціали та прізвище)

Керівник бакалаврської дипломної роботи _____ **Г. Б. Ракитянська**
(підпис) (ініціали та прізвище)

АНОТАЦІЯ

Бакалаврська дипломна робота містить 70 сторінок формату А4, на яких є 39 рисунків, 1 таблиця, список використаних джерел містить 22 найменування.

У бакалаврській дипломній роботі проведено детальний аналіз методів і засобів оптимізації мережі громадського транспорту міста, існуючих аналогів. Сформульовано мету досліджень – підвищення автоматизації формування розкладів для зупинок і спрощення процесу моніторингу громадського транспорту за рахунок розробленої системи, а також підвищення якості мережі маршрутів завдяки оптимізації графіку розробленим генетичним алгоритмом.

Розроблено адаптивний генетичний алгоритм оптимізації графіку мережі громадського транспорту, який враховує потребу у наявності мінімального інтервалу між різними транспортними засобами в межах однієї зупинки. Розроблено програмні засоби реалізації системи вебзастосунку з урахуванням цільового призначення та використанням сучасних інструментів. Розроблений вебзастосунок призначений для оптимізації роботи департаменту громадського транспорту використовуючи зручну для кінцевого користувача операційну систему з доступом до мережі інтернет.

Вебзастосунок розроблено з використанням мови Java та середовища IntelliJ IDEA Ultimate. Для розгортання вебзастосунку використовувався Spring Framework, локальний хостинг та база даних MongoDB. Також було використано Git для контролю версій та Apache POI для реалізації функціоналу з Excel/Word.

У результаті виконання бакалаврської дипломної роботи отримано систему, що підвищить якість графіку маршрутів, а також полегшить моніторинг громадського транспорту та автоматизує формування розкладів для зупинок.

Отримані в бакалаврській дипломній роботі результати можна використати для оптимізації роботи мережі громадського транспорту.

Ключові слова: оптимізація, автоматизація, громадський транспорт, Java, Spring Framework.

ABSTRACT

The bachelor thesis contains 70 pages of A4 format, on which there are 39 figures, 1 table, the list of used sources contains 22 names.

A detailed analysis of the methods and means of optimization of the city's public transport network, existing analogues, was carried out in the bachelor thesis. The goal of the research is formulated - to improve the automation of the formation of stop schedules and to simplify the process of public transport monitoring due to the developed system, as well as to improve the quality of network routes using the optimization schedule by the developed genetic algorithm.

An adaptive genetic algorithm for public transport network schedule optimization has been developed, which takes into account the need for a minimum interval between free vehicles within one stop. The software tools for the implementation of the web application system have been developed, taking into account the intended purpose and the use of modern tools. The developed web application is designed to optimize the work of the public transport department using an end-user-friendly operating system with Internet access.

The web application is developed using the Java language and the IntelliJ IDEA Ultimate environment. Spring Framework, local hosting and MongoDB database were used to deploy the web application. Also used was Git for version control and Apache POI to implement Excel/Word functionality.

Based on the results of the bachelor thesis, a system was obtained that increases the quality of the route schedule, as well as facilitates the monitoring of public transport and automates the formation of schedules for stops.

The results obtained in the bachelor thesis can be used to optimize the operation of the public transport network.

Keywords: optimization, automation, public transport, Java, Spring Framework.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. Громадська мережа транспорту є однією з найбільш використовуваних речей у повсякденному житті. Важко уявити скільки користі приносить транспорт й скільки сил та часу зберігається. Крім того, значно збільшується мобільність великої частини населення міста, що в свою чергу поліпшує продуктивність та економічне становище. Люди мають більше можливостей, бо стають менш обмеженими відстанню від домівок до можливого майбутнього місця роботи.

Як і на кожен процес, що має високий попит та значимість, на систему громадського транспорту витрачається чимало людських ресурсів для забезпечення безперебійної роботи, оптимізації та автоматизації.

Технології різних сфер діяльності майже кожного місяця значно покращуються, саме тому багато налаштованих та перевірених роками процесів стають застарілими та потребують змін. Звісно досягнути ідеального рішення неможливо, але люди завжди цього прагнуть. Загалом, поняття ідеалу є занадто розпливчастим і, фактично, може означати найкращий варіант серед можливих у поточний час. В той ж час кожен науковий винахід чи дослідження певною мірою впливає майже на всі навколишні процеси та безпосередньо на висновки щодо їх досконалості у порівнянні з можливостями. Ці зміни можуть сприяти не тільки до оптимізації певного процесу, а й іноді до повної його реорганізації.

Одними із таких впливових інновацій є впровадження машинної праці, використання різних обчислювальних алгоритмів та штучного інтелекту. Так, розглядаючи саме сферу громадського транспорту, можна спостерігати наступні покращення:

1) використання інтерактивного розкладу на зупинках;
2) оплата квитка через QR-код, банківською безконтактною карткою або смартфоном з NFC-модулем;
3) повторюваний звуковий супровід щодо поточної та наступної зупинки;

4) система контролю за оплатою проїзду;
5) трекінг маршрутів в реальному часі.

Запевняти, що машина, запрограмована людьми на певну дію, зможе зробити щось неймовірне та перевершити її розробників не слід. Проте варто підкреслити, що машини мають значно більшу обчислювальну спроможність, що і потрібно використовувати. Майже будь-яку операцію, яку людина буде робити довго згідно різного ряду факторів, машина може зробити достатньо швидко. Головною проблемою є саме спосіб навчити машину виконувати поставлені задачі. Обраний спосіб та програмне рішення будуть напряму впливати на якість кінцевої роботи [1].

Складні на перший погляд задачі з оптимізації мережі громадського транспорту легко вирішуються розробкою відповідних програмних засобів. Використання комп'ютерних алгоритмів сприяє автоматизації процесів, які раніше вимагали значних зусиль людини. Наприклад, алгоритми маршрутизації можуть розраховувати оптимальні маршрути для транспорту з урахуванням різних факторів, таких як час руху, відстань, пасажиропотік, періодичність та інші. Це дозволяє ефективно використовувати час та ресурси, що раніше витрачалися на повторювану роботу. Використання різних алгоритмів також дозволяє швидко знаходити оптимальні рішення для складних проблем. Наприклад, алгоритм генетичного програмування можна використати для пошуку оптимальних рішень серед великої вибірки можливих варіантів, що дозволяє ефективно вирішувати задачі, які були б надто складними для розв'язання вручну або за допомогою традиційних методів.

Вивчивши питання, можна виділити декілька лідерів серед систем, що полегшують користування громадським транспортом. Серед них: Rozklad.in.ua, Google Maps, Zeo, Speedy Route, Optimoroute, MapQuest [2]. Більшість з них є платними або ж з обмеженими можливостями у налаштуванні. Саме тому є актуальною розробка власного рішення для системи з реалізацією обраного алгоритму оптимізації та необхідних кінцевому користувачеві функцій.

Зв’язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення.

Мета та завдання дослідження. Мета дипломної роботи полягає у підвищенні автоматизації формування розкладів для зупинок, спрощенні процесу моніторингу громадського транспорту та оптимізації графіку маршрутів завдяки розробленому алгоритму та його програмній реалізації, що дозволить підвищити якість мережі транспорту.

Робота передбачає розробку наступних модулів: модуль інтерфейсу вебзастосунку, модуль авторизації, модуль завантаження файлів, модуль парсингу даних з Excel файлу, модуль генетичного алгоритму оптимізації графіку маршрутів, модуль створення розкладів для зупинок у форматі Doc, модуль пакування файлів розкладу в архів та модуль вивантаження архіву.

Задачі дослідження:

- провести аналіз стану систем оптимізації мережі громадського транспорту;
- запропонувати новий метод оптимізації та автоматизації мережі транспорту, систему-реалізацію;
- розробити модель системи для оптимізації мережі громадського транспорту;
- розробити програмні модулі запропонованої системи;
- провести тестування розроблених алгоритмів та системи.

Об’єкт дослідження – процес розробки системи для оптимізації мережі громадського транспорту міста.

Предмет дослідження – методи та засоби реалізації системи для оптимізації мережі громадського транспорту міста.

Методи дослідження. У процесі досліджень використовувались:

- методи та засоби реалізації вебзастосунків за патерном MVC (Model-View-Controller) мовою Java з використанням Spring Framework для спрощення розробки та зв’язку між модулями системи;
- методи генетичної оптимізації для оптимізації мережі громадського транспорту міста;
- методи створення NoSQL бази даних та зберігання даних авторизації;
- методи для захисту даних системи та користувачів за допомогою Spring Framework Security;
- методи побудови зв’язку між бекендом та фронтендом вебзастосунку за патерном REST (Representational State Transfer);
- методи тестування реалізованої системи та алгоритмів.

Новизна отриманих результатів:

1. Запропоновано вебзастосунок з реалізацією адаптивного генетичного методу оптимізації графіку маршрутів, який на відміну від існуючих реалізації є доступним в мережі, зберігає подібність до початкового розкладу та є налаштованим на роботу з маршрутами, що дозволяє отримати очікуваний розклад згідно побажань потенційного користувача.
2. Подальшого розвитку набув метод формування звітів-розкладів для зупинок за допомогою Apache POI на стороні серверу та їх завантаження у вебзастосунку, що на відміну від стандартних методів формування звітів не потребує особливих навичок у налаштуванні формату, взаємодії з власником системи та дозволяє не тільки автоматизувати процеси планування руху транспорту з можливістю адаптації до умов експлуатації, а й виконувати оптимізацію з будь-якого пристрою.

Практична цінність отриманих результатів. Практична цінність отриманих результатів полягає в розробці вебзастосунку, який здатний надавати користувачам можливість оптимізувати графік (за бажанням) та отримати звіти відповідно до завантаженого розкладу маршрутів використовуючи браузер. Програмні модулі, розроблені в роботі, можуть бути використані розробниками

для створення нових подібних систем, для розширення функціоналу вебзастосунку або алгоритму оптимізації в цілому та користувачами для звітування власних мереж громадського транспорту.

Особистий внесок здобувача. Усі наукові результати, що викладені у бакалаврській дипломній роботі, отримані особисто автором.

Апробація матеріалів бакалаврської дипломної роботи. Результати бакалаврської дипломної роботи доповідалися на LIII Науково-технічній конференції факультету інформаційних технологій та комп’ютерної інженерії (2024).

Публікації. За тематикою дослідження опубліковано 1 наукову працю у збірнику матеріалів LIII Науково-технічної конференції факультету інформаційних технологій та комп'ютерної інженерії (2024) [1].

Структура та обсяг роботи. Бакалаврська дипломна робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел, що містить 22 найменування, 4 додатків. Робота містить 39 ілюстрацій та 1 таблицю.

1 АНАЛІЗ РОЗВИТКУ СИСТЕМ ДЛЯ ОПТИМІЗАЦІЇ МЕРЕЖІ ГРОМАДСЬКОГО ТРАНСПОРТУ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

1.1 Аналіз стану подібних систем

Сучасні системи оптимізації мереж громадського транспорту використовують різні алгоритми та підходи для забезпечення максимально ефективного розподілу ресурсів і підвищення зручності для пасажирів. Впровадження таких систем дозволяє зменшити не тільки час очікування для пасажирів, а й кількість транспортних засобів та витрати на їх обслуговування, підвищуючи рівень сервісу. Наприклад, автоматизована система контролю оплати проїзду, безконтактна оплата квитка, та трекінг транспортних засобів у реальному часі стають невід'ємною частиною сучасних транспортних мереж.

Інтеграція з штучним інтелектом для покращення ефективності оптимізації та управління громадським транспортом є ще одним важливим аспектом. Алгоритми машинного навчання можуть допомогти в виявленні та прогнозуванні закономірностей в русі транспорту та попиті пасажирів, що сприятиме визначенню оптимальних маршрутів та розкладів. Наприклад, використання нейронних мереж для аналізу історичних даних може дозволити системі передбачити пікові години навантаження та відповідно коригувати графік руху транспорту.

Ще одним важливим аспектом є оптимізація процесу оновлення даних. Системи, що забезпечують реальний час оновлення інформації про рух транспорту, дозволяють пасажирам отримувати найактуальнішу інформацію, що зменшує кількість скарг та підвищує задоволеність користувачів. Часткова чи повна автоматизація процесу збору та оновлення інформації про розклади, маршрути та інші параметри зменшує навантаження на операторів системи та забезпечує актуальність даних.

Якщо розглядати системи, які націлені саме на оптимізацію розкладу користувачами з можливістю автоматизації генерування звітів, то аналогів у відкритому доступі немає. Проте є сучасні системи, які заслуговують на увагу та пов’язані з певною оптимізацією транспортної мережі для користувачів чи пасажирів:

1) Google Maps – система, що надає детальні маршрути громадського транспорту, враховуючи реальний час руху та затримки. Інтеграція з іншими сервісами, такими як навігаційні програми, дозволяє пасажирам легко планувати свої поїздки.

2) Rozklad.in.ua – український сервіс, що надає розклади руху транспорту та можливість планування маршрутів. Він також дозволяє відстежувати рух транспорту в реальному часі.

3) OptimoRoute – комерційне програмне забезпечення для оптимізації маршрутів, що використовується багатьма компаніями для підвищення ефективності своїх транспортних мереж.

Варто зазначити, що розвиток цих систем вимагає не лише впровадження новітніх технологій, а й розробки нових алгоритмів та моделей для аналізу даних. Використання великих даних (Big Data) та машинного навчання дозволяє значно підвищити ефективність роботи транспортних систем, а також забезпечити більш гнучке та адаптивне управління. Хоча більшість використовуваних методів не афішуються, але активно використовуються.

Можна підкреслити, що сучасні системи оптимізації мереж громадського транспорту є складними інтегрованими рішеннями з різноманітними технологіями та підходами для досягнення кращої ефективності та зручності для пасажирів.

1.2 Порівняльний аналіз аналогів

Є декілька лідерів серед систем, що пов’язані з оптимізацією громадського транспорту або користування ним. Серед них: Rozklad.in.ua, Google Maps, Zeo,

Speedy Route, Optimoroute. Більшість з них є платними або ж з обмеженими можливостями у налаштуванні.

Система Rozklad.in.ua надає користувачам доступ до розкладів громадського транспорту декількох українських міст. Команда розробників працює над тим, щоб зробити громадський транспорт дружнім для пасажирів, допомагаючи їм краще орієнтуватися у місті та більш ефективно планувати свій час (рис. 1.1).

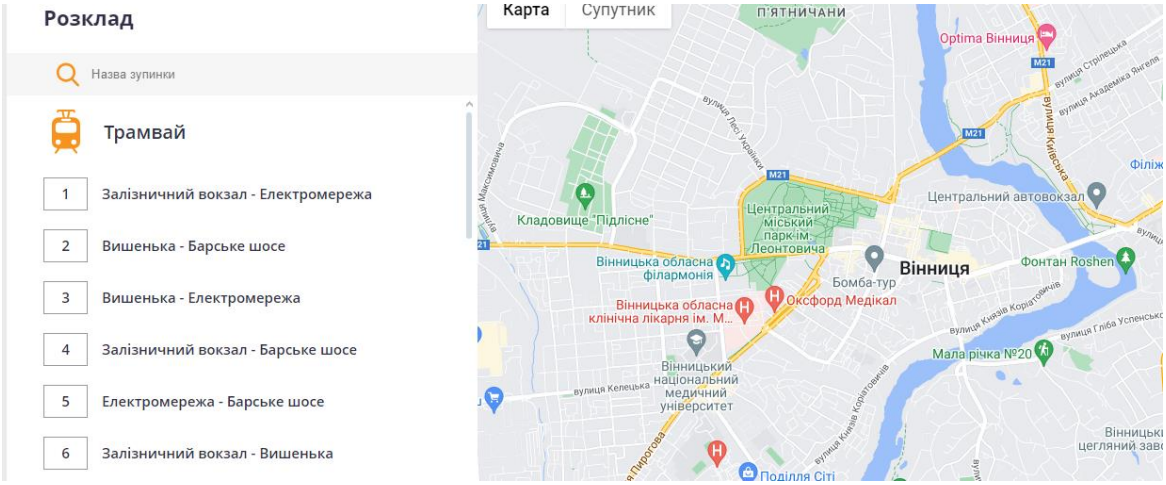

Рисунок 1.1 – Приклад роботи системи Rozklad.in.ua

Розклад сервісу відповідає європейським практикам транспортних компаній у наданні інформації пасажирам та створені з урахуванням принципу «універсального дизайну», тобто такого, що інтуїтивно зрозумілий для будь-яких користувачів [3]. Це дозволяє пасажирам будь-якого віку та іншим подорожуючим, у тому числі іноземним туристам, зручно користуватись розкладами громадського транспорту [4].

Система надає можливість користувачеві отримувати інформацію обраного маршруту громадського транспорту на будь-якому пристрої завдяки браузеру (рис. 1.2).

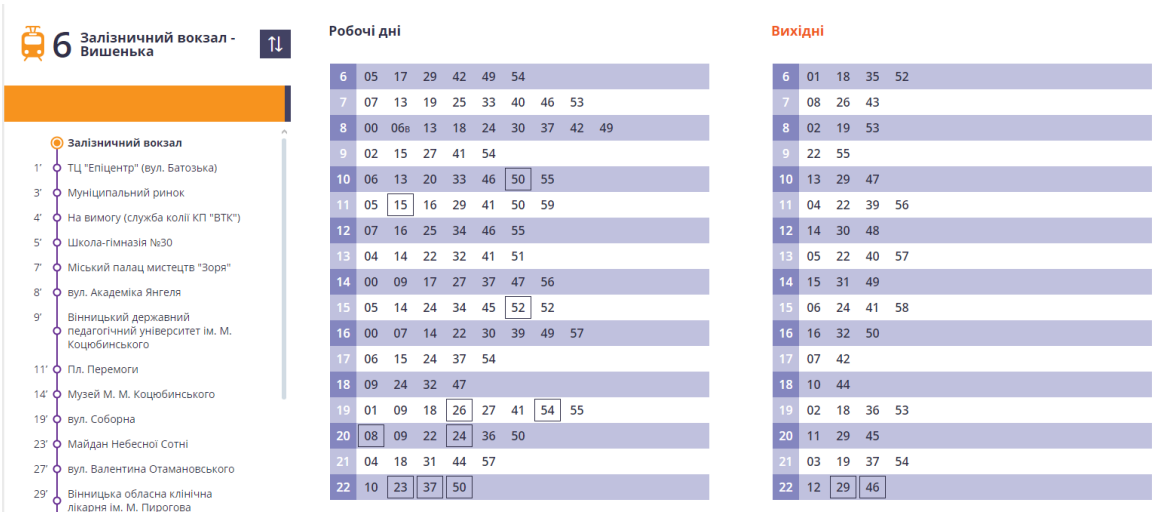

Рисунок 1.2 – Інформація про маршрут у Rozklad.in.ua

Наразі розклади даного сервісу наявні у таких містах: Ужгород, Кременчук, Полтава, Коростень, Коломия, Рівне , Звягель, Житомир, Бровари, Кропивницький, Львів, Вінниця, Івано-Франківськ, Бердичів та Хмельницький (рис. 1.3).

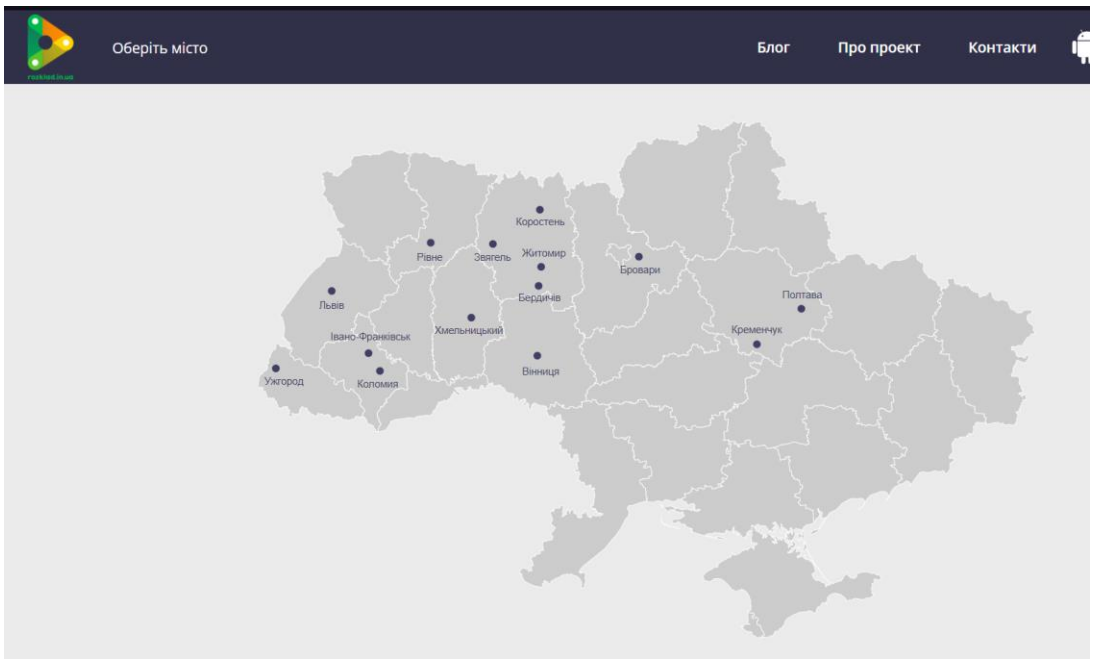

Рисунок 1.3 – Мапа розкладів у Rozklad.in.ua

Google Maps є однією з найпопулярніших в світі картографічних сервісів, що має інформацію про маршрути громадського транспорту, автомобільні та пішохідні маршрути. Маршрути також мають інформацію про приблизний час дороги, час прибуття транспорту та відстань побудованого маршруту.

Крім того, наявна функція оптимізації маршрутів для автомобільних поїздок з урахуванням заторів, функція перегляду зупинок громадського транспорту на карті [5].

Цей сервіс дозволяє користувачам швидко знаходити оптимальний маршрут з точки А в точку В. В питанні громадського транспорту, Google Maps надає інформацію про доступні маршрути, розклади руху та приблизний час подорожі (рис. 1.4).

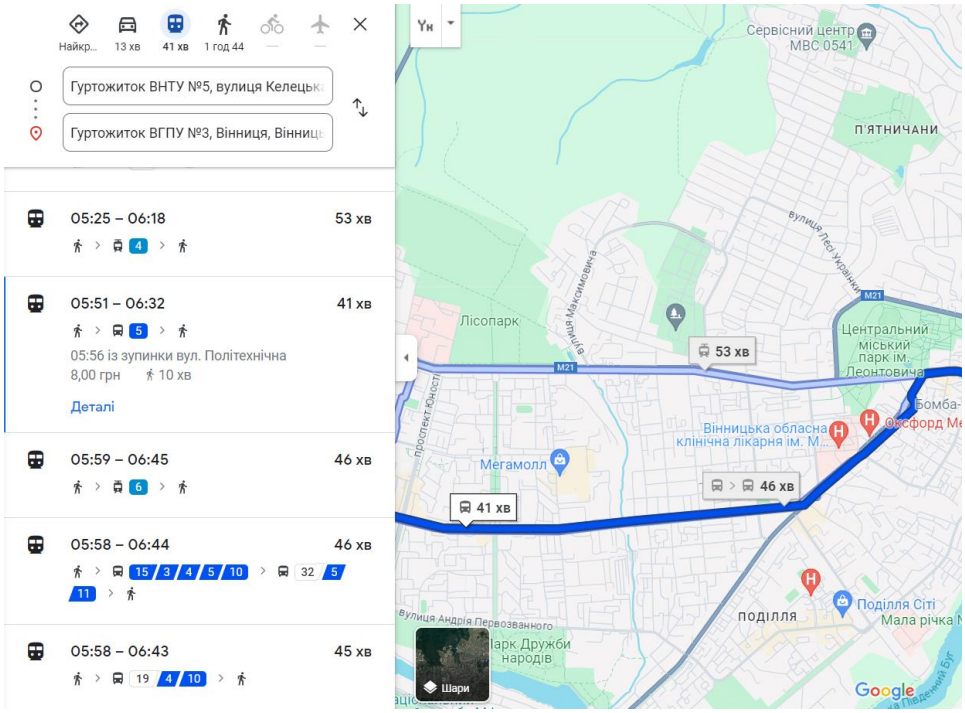

Рисунок 1.4 – Приклад роботи додатку Google Maps

Google Maps доступний як для комп’ютерів, так і для мобільних пристроїв, що робить його дуже популярним серед користувачів, які шукають зручний і надійний спосіб планування своїх маршрутів.

Система Zeo використовується для відстеження місцезнаходження транспортних засобів та підтримки комунікації між водіями та диспетчерами. Ця система допомагає в управлінні маршрутами та вирішенні питань з логістикою, що робить її незамінною для багатьох компаній. Система також має цілодобову живу підтримку, яка допомагає вирішувати всі питання та розуміє ваші вимоги.

Основні функціональні можливості Zeo включають в себе можливість відстежувати місцезнаходження транспортних засобів у реальному часі, отримувати повідомлення про затримки або проблеми з маршрутами (рис. 1.5), а також здійснювати ефективне планування маршрутів для оптимізації часу та витрат пального [6].

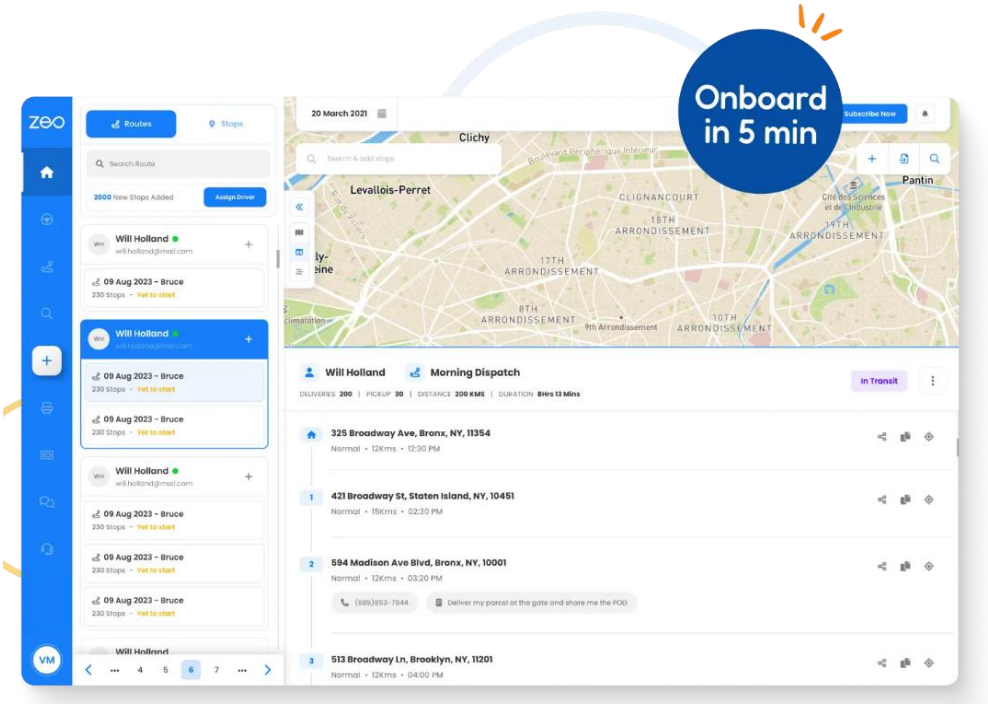

Рисунок 1.5 – Приклад роботи додатку Zeo

Система має здатність працювати в реальному часі, що дозволяє відстежувати порушення у графіках маршрутів, їх пройденому шляху та умовах на дорозі. Крім того, є звіти про продуктивність та ефективність використання транспортних засобів, що дозволяє підвищити рівень обслуговування та зменшити витрати на експлуатацію маршрутів.

Speedy Route обчислює найкращий маршрут для відвідування кількох місць і повернення назад до початкової точки. Це ідеальний інструмент для водіїв доставки, торгових представників на дорозі або будь-кого, хто повинен здійснити кілька зупинок.

Введені місця представляють в найоптимальніший порядок, так що кожне місце відвідується один раз перед поверненням до вашої початкової точки найкоротшим і найшвидшим способом. Speedy Route (рис. 1.7) доступний у Великій Британії, в Сполучених Штатах, інших країнах Європи, а також у Канаді, Австралії та Новій Зеландії, на Далекому Сході, у Південній Америці [7].

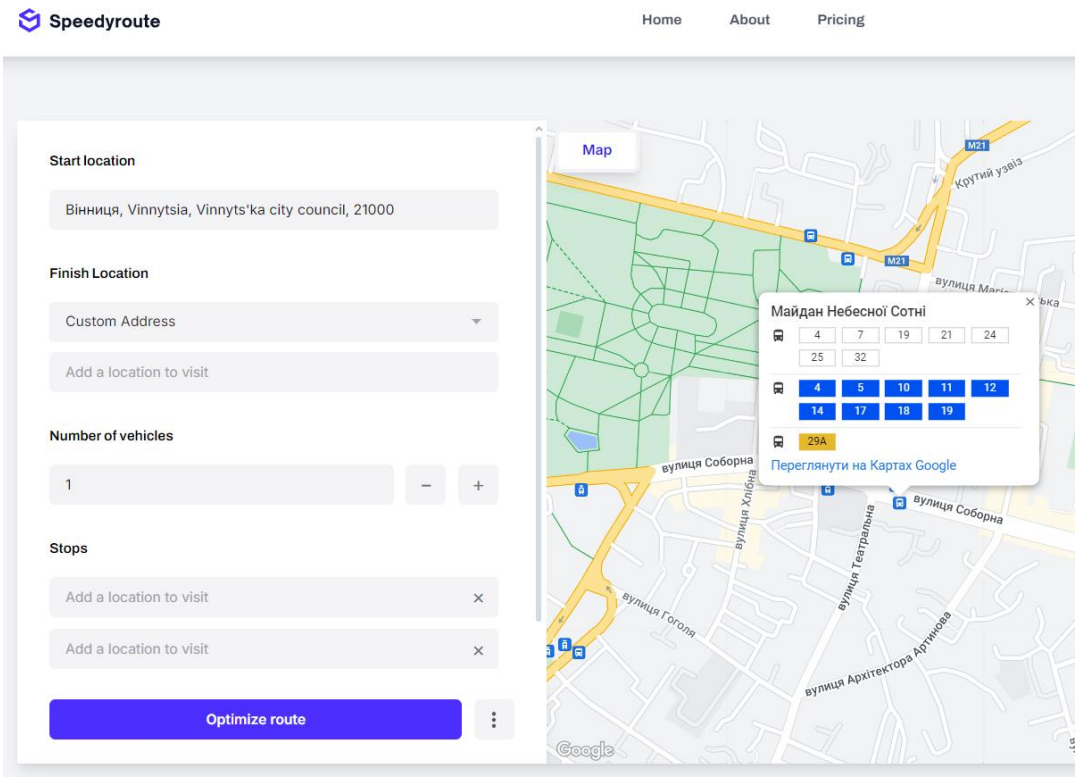

Рисунок 1.7 – Приклад роботи системи Speedy Route

Якщо встановити кількість доступних автомобілів для доставки більше одного, розрахований маршрут може використовувати будь-яку кількість автомобілів до максимально доступної.

Розрахований маршрут може не використовувати всі доступні автомобілі, якщо оптимальний маршрут можна здійснити за допомогою меншої кількості автомобілів.

OptimoRoute – це інструмент для оптимізації маршрутів доставки та обслуговування клієнтів, який допомагає вирішувати складні задачі маршрутизації для компаній з різноманітними потребами.

Система OptimoRoute автоматично розраховує оптимальний маршрут для кожного транспортного засобу, враховуючи різні обмеження, такі як обмеження на час, обмеження по обсягу вантажу та пропускну спроможність маршруту.

Програма дозволяє планувати маршрути для одного або кількох транспортних засобів (рис. 1.8). Крім того, система може автоматично оновлювати маршрути в реальному часі на основі змін у замовленнях або умовах дороги, що дозволяє підтримувати максимальну ефективність.

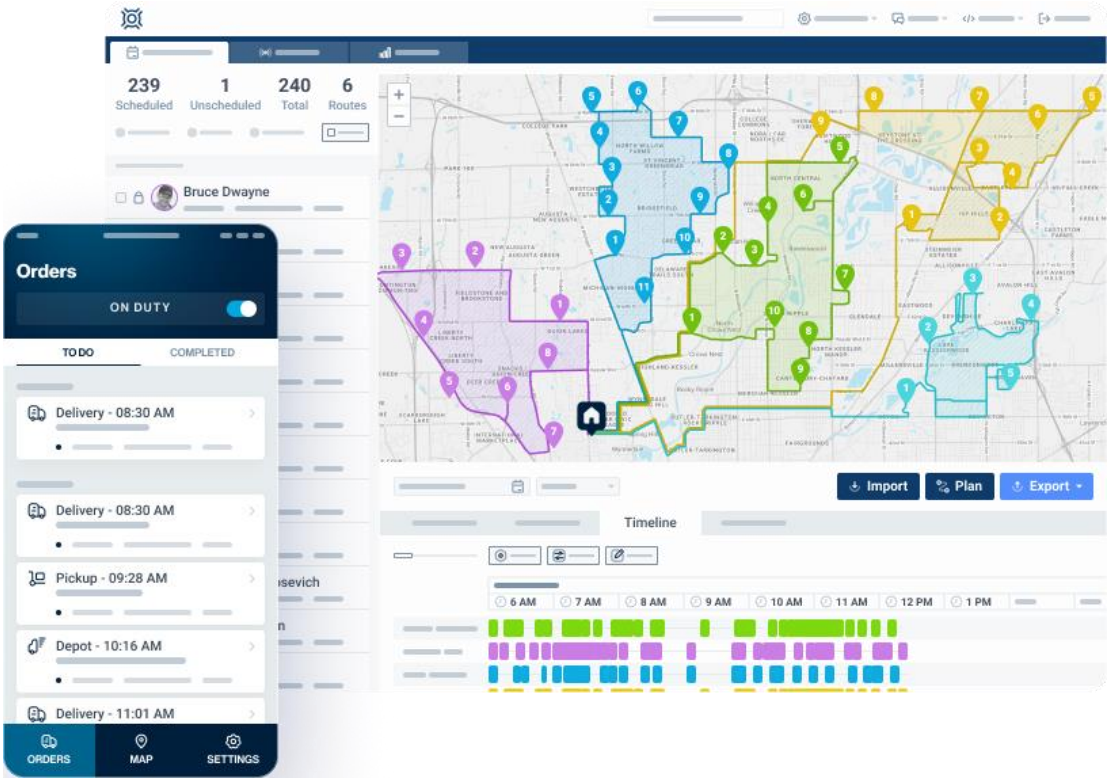

Рисунок 1.8 – Приклад роботи системи OptimoRoute

Користувачам надається широкий спектр інструментів для моніторингу та аналізу ефективності маршрутів, включаючи звіти про пройдений шлях, час доставки та витрати на паливо.

OptimoRoute дозволяє підприємствам зменшити витрати на доставку, підвищити продуктивність та покращити обслуговування клієнтів завдяки оптимізації маршрутів доставки.

Для зручного порівняння було зібрано всі аналоги в таблицю 1.1.

Таблиця 1.1 – Порівняльна таблиця систем-аналогів

Назва системи	Доступність	Ціна	Необхідність втручання власника системи	Оптимізація розкладу	Автоматизація звітів для зупинок
CityTransOpt	Онлайн	Безкоштовно	Відсутня	Так	Наявна
Rozklad.in.ua	Онлайн	Платно	Присутня	Так	Наявна
Google Maps	Онлайн	Безкоштовно	Відсутня	Ні	Відсутня
Zeo	Онлайн	1,389.36 грн	Відсутня	Так	Відсутня
OptimoRoute	Онлайн	30 днів безкоштовно	Відсутня	Так	Відсутня
Speedy Route	Онлайн	2,739.02 грн	Відсутня	Так	Відсутня

За результатами порівняльної таблиці, TransportOptimizer-App видається найбільш привабливим варіантом серед розглянутих аналогів. Він доступний онлайн безкоштовно, має високу гнучкість і оптимізує розклади. Також, система надає автоматизацію створення розкладів для зупинок. Враховуючи ці фактори, можна зробити висновок, що TransportOptimizer-App є найкращим вибором серед розглянутих аналогів.

1.3 Аналіз методів розв’язання задачі

Існує кілька методів реалізації системи для оптимізації громадського транспорту, включаючи мобільний додаток, десктопний додаток і вебзастосунок. Кожен з цих методів має недоліки та переваги, що треба враховувати при пошуку найкращого рішення:

1. Мобільний додаток.

Переваги: зручний для користувачів, оскільки дозволяє отримати доступ до системи з будь-якого місця і у будь-який час через смартфон або планшет; може використовувати функціональні можливості пристрою, наприклад GPS.

Недоліки: вимагає встановлення на пристрої користувача; розробка та підтримка мобільного додатку є витратною та складною.

2. Десктопний додаток.

Переваги: більш функціональний та масштабований у порівнянні з мобільним додатком; зручний для великих екранів і робочих станцій.

Недоліки: вимагає встановлення на комп’ютері; є складнішим у використанні для менш досвідчених користувачів пк.

3. Вебзастосунок.

Переваги: доступний через браузер на будь-якому пристрої з Інтернет-підключенням; не вимагає встановлення на пристрої.

Недоліки: менш зручний для використання на мобільних пристроях порівняно з мобільним додатком.

Створення десктопного або мобільного додатку для оптимізації громадського транспорту може бути виправданим у окремих випадках, але зазвичай вебзастосунок має декілька переваг порівняно з ними:

1. Доступність. Не потрібно завантажувати.
2. Крос-платформенність. Вебзастосунок може працювати на будь-якій операційній системі.
3. Вартість розробки та підтримки. Розробка лише одної реалізації для всіх платформ, а не декілька адаптацій під різні пристрої.

Отже, у випадку оптимізації громадського транспорту вебзастосунок може бути кращим вибором через свою доступність, оновлення, крос-платформенність та вартість розробки та підтримки.

1.4 Постановка задач для розробки системи оптимізації мережі громадського транспорту міста

Проаналізувавши існуючі системи, що пов’язані з оптимізацією мережі громадського транспорту було визначено наступні завдання, які необхідно виконати для розробки вебзастосунку:

1) розробити модуль реєстрації;
2) розробити модуль авторизації;
3) розробити модуль оптимізації графіку;
4) розробити модуль графічного інтерфейсу вебзастосунку;
5) розробити модуль завантаження файлу;
6) розробити модуль парсингу даних;
7) розробити модуль створення розкладів зупинок;
8) розробити модуль пакування в архів файлів розкладів;
9) розробити модуль вивантаження архіву;
10) об’єднати окремі модулі в систему вебзастосунку.

1.5 Висновки

У першому розділі було проведено аналіз поточного стану існуючих систем для оптимізації мережі громадського транспорту. Було проведено порівняння кількох систем-лідерів, що пов’язані з громадським транспортом. Серед них: Rozklad.in.ua, Google Maps, Zeo, Speedy Route, Optimoroute.

У результаті аналізу виявлено недоцільність використання готових систем для вирішення поставленої задачі. Саме тому вирішено розробити власну систему (вебзастосунок) враховуючи недоліки аналогів. Було сформовано перелік задач, які необхідно виконати для його розробки.

2 РОЗРОБКА МЕТОДУ, МОДЕЛІ ТА АЛГОРИТМІВ СИСТЕМИ

2.1 Аналіз вхідних та вихідних даних системи

Вхідні дані для розробки програмного продукту вебзастосунку оптимізації громадського транспорту представлені у формі електронної таблиці у форматах xlsx/xls. Таблиці мають бути створені за певним шаблоном та включати наступну інформацію:

1. Розклад маршрутів (час відбуття від кожної зупинки), необхідний для подальшого звітування по кожній зупинці.
2. Назви зупинок, необхідні для визначення послідовності зупинок у маршруті.
3. Назви маршрутів, необхідні для зв’язності з розкладом та зупинками.

Після парсингу дані з Excel-файлу будуть зберігатись у спеціальному класі, що називається StationData. Клас містить інформацію про зупинку громадського транспорту, таку як назва зупинки, інформація про маршрут. Крім того, наявна колекція об'єктів типу Date, яка представляє розклад відправлення автобусів чи тролейбусів з цієї зупинки. Кожен об’єкт StationData відповідає одній зупинці громадського транспорту і містить всі необхідні дані для подальшої обробки та відображення вебзастосунку.

Вихідні дані для аналізу представлені у вигляді розкладів-звітів у форматі docx, кожен з яких відповідає одній зупинці громадського транспорту. Також до вихідних даних входить файл ексель з оптимізованим графіком маршрутів або з оригінальним, якщо оптимізація не викликалась.

Розклад-звіт називається відповідно до зупинки та містить назву зупинки та таблицю з маршрутами та розкладом по ній. Таблиця матиме наступні колонки: «№ маршруту», «Назва маршруту», «Дні роботи» та «Розклад руху».

Після оптимізації розкладу та створення звітів для кожної зупинки, всі вихідні згенеровані файли об’єднуються в один zip-архів.

Це робить процес вивантаження та доступу до вихідних даних більш зручним для користувача, оскільки весь необхідний обсяг інформації можна отримати одним файлом.

2.2 Розробка методу оптимізації

Предметом оптимізації мережі громадського транспорту може бути оптимізація шляху, графіку прибуття, кількості маршрутів та розміщення зупинок. Кожен має свої складнощі оптимізації, методи оптимізації та результати.

У бакалаврській роботі в якості оптимізації було вирішено оптимізувати саме графік прибуття. Оптимізований графік допоможе зменшити час очікування пасажирів завдяки тому, що маршрути приходитимуть з інтервалом, а не одночасно. Також це частково вирішує проблему, що якщо на зупинку прибувають схожі за напрямком маршрути, то фактично відбивають пасажирів у один одного. Якщо враховувати фактор, що зазвичай маршрути, які мають схожий час прибуття, це маршрути, які курсуються різними компаніями. Тобто це позитивно вплине і на прибуток компанії на окремих проміжках маршруту.

Метою оптимізації є побудова оптимального розкладу без повторів у межах зупинки. Важливим фактором є також оптимізація зі збереженням подібності до оригінального графіку, щоб людям було простіше адаптуватись до змін та керівництву міських рад з питань транспорту не приходилось кардинально змінювати технічні документи маршрутів. Під подібність мається на увазі збереження кількості маршрутів та їх варіація з метою заміни повторюваних значень у чітко прописаних допустимих межах.

Для реалізації описаної оптимізації було обрано цільову функцію та адаптивний генетичний алгоритм через його ефективність та простоту у розв’язанні задач комбінаторної оптимізації. Зазвичай результатом роботи генетичного алгоритму є найкращі або ж близькі до них рішення.

Розклад прибуття маршрутів на зупинки буде відкоригований за допомогою алгоритму та потім оцінений цільовою функцією. Якщо необхідно, то буде повторно згенерований та оцінений. Процес повторюватиметься до тих пір, поки або не буде знайдено найкращого варіанту, або певна кількість генерацій покоління не видаватиме кращого результату.

Для моделювання варто визначити такі поняття:

- ген (години прибуття маршруту) – одиниця успадкування під час схрещування;
- хромосома (одиничний маршрут з годинами прибуття) – носій генів;
- популяція (графік прибуття всіх маршрутів) – набір хромосом, що схрещуються.

Генетичний алгоритм реалізується у декілька етапів (рис. 2.1).

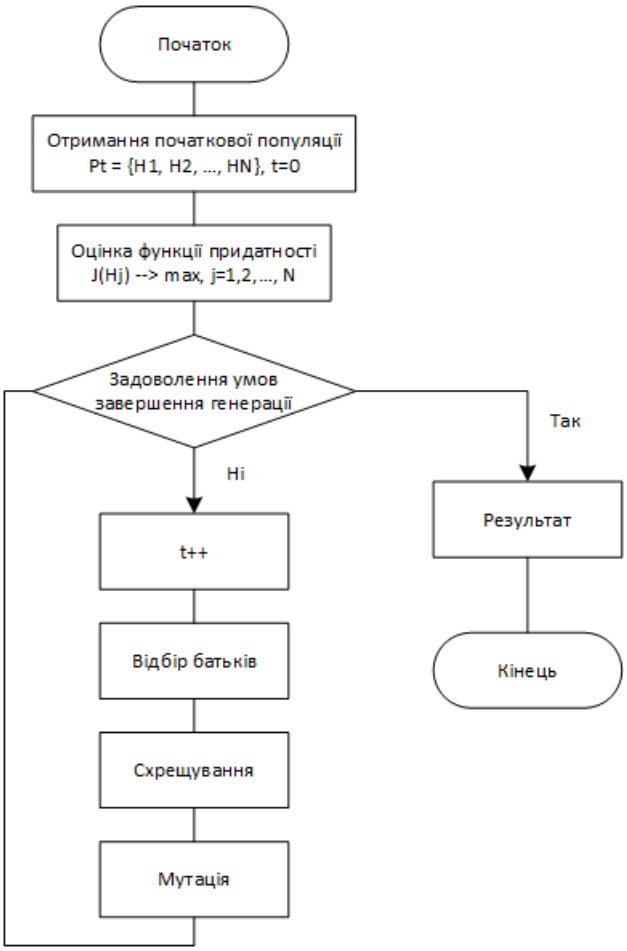

Рисунок 2.1 – Схема роботи генетичного алгоритму

Хромосома буде представляти розклад прибуття транспортних засобів. Вона буде включати двовимірну структуру даних, де вузли будуть відповідати дням тижня, номерам рейсів та зупинкам (див. рис. 2.2). Ця структура буде характеризувати розклад за допомогою інформації про доступні рейси та їх години прибуття на зупинки.

Для створення компонентів розкладу буде використана наступна вихідна інформація:

1. Рейси (R): { r1, r2…rNr }, де ri - номер рейсу, а Nr - загальна кількість рейсів.

2. Зупинки (S): { s1, s2…sNs }, де si - номер зупинки, а Ns - загальна кількість зупинок.

3. Години прибуття (T): { t1, t2…tNt }, де ti - час прибуття рейсу на зупинку, а Nt - загальна кількість часів прибуття.

4. Дні тижня (W): { будні, вихідні}.

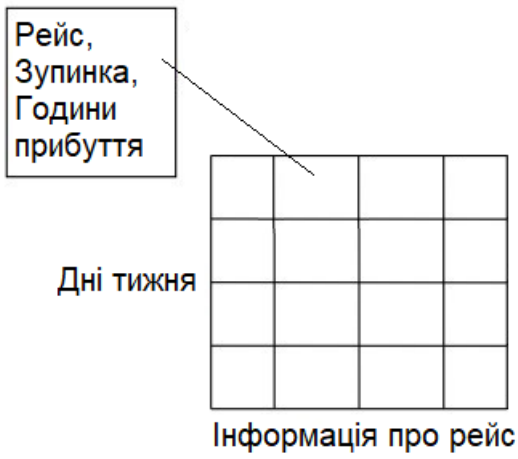

Рисунок 2.2 – Графічне представлення розкладу

Між рейсом, зупинкою та годинами прибуття, існує тісний взаємозв’язок, оскільки кожен рейс має свій графік руху, який включає різні години прибуття на різні зупинки. Тому їх було об’єднано одну структуру даних.

2.3 Розробка цільової функції

Формуючи цільову функцію, треба врахувати фактори ефективності, валідності та оптимальності розкладу з точки зору мережі громадського транспорту.

Жорсткі обмеження:

HL1: має зберігатись подібність до завантаженого розкладу (однакова кількість маршрутів, їх параметри, окрім часу прибуття);

HL2: час прибуття не може перевищувати параметр відхилення від початкового;

HL3: маршрутний засіб може бути одночасно тільки на одній зупинці.

М’які обмеження:

SL1: для одного маршруту час прибуття не має сходитись;

SL2: для однієї зупинки години прибуття маршрутів не мають накладатись;

Порушення жорстких обмежень призведе до створення невалідного розкладу. Тому формування розкладу та проведення всіх генетичних операцій над хромосомами проводиться з урахуванням саме жорстких обмежень. Для оцінки якості хромосоми розроблено функцію пристосованості, що враховує лише м’які обмеження.

Виходячи з цього функція пристосованості матиме наступний вигляд (формула 2.1):

$$F(i) = \frac{\Sigma_{k=1}^{N}\Sigma_{j=1}^{2} w_j \cdot R_j(k)}{N}, \quad (2.1)$$

де N – кількість ген в хромосомі (годин прибуття), R_j – значення j-го обмеження, ω_j – вага j-го обмеження.

Максимальне значення функції пристосованості дорівнює 1, а мінімальне - 0. Мінімальне значення фітнес-функції досягається, коли для кожної години прибуття порушено всі обмеження. Максимальне значення досягається, коли всі обмеження враховано.

2.4 Формалізація задачі оптимізації мережі громадського транспорту

Задача формування розкладу є комбінаторною NP-повною задачею, для вирішення якої необхідно розподілити години прибуття маршрутів по множині зупинок враховуючи різні дні тижня. До того ж існують нелінійні критерії та вимоги, які часто суперечать один одному, що робить задачу визначення цільової функції складною і важко формалізованою у вигляді зваженої суми критеріїв.

Порушення жорстких обмежень призводить до невалідності рішення. Як приклад можна привести розклад, в якому одночасно маршрут повинен стояти відразу на декількох станціях або ж повинен прибути за занадто короткий чи занадто тривалий час. Можливість отримання таких рішень створює некоректні області в просторі пошуку, які необхідно повністю виключити з розгляду.

Для вирішення проблеми невалідності розв’язку, пов’язаної з неможливістю транспорту бути одночасно в декількох місцях, необхідно сконструювати або задати початкове валідне рішення та розробити таку його модифікацію, щоб уникнути прийняття неприпустимих рішень. Недоліком є складність побудови початкового рішення та реалізації операторів, які б приводили тільки до допустимих результатів і при цьому залишали досяжним весь коректний простір пошуку. Тому було вирішено отримувати початкове валідне рішення у форматі таблиці ексель від користувача та на його основі відштовхуватись при перевірці границь модифікації. Тим паче, що в не автоматизованих установах розклад маршрутів зберігається саме в такому форматі.

Крім виконання жорстких варто врахувати і м’які обмеження. Так, рішення може повністю задовольнити всі основні вимоги, але мати погану оцінку від штрафних функцій. Наявність таких рішень буде позитивно впливати на

варіативність, хоча у деяких випадках може збільшити час пошуку кращого рішення.

Для формалізації задачі слід використати такі позначення:

– H – множина годин прибуття;

– F – множина графіків маршруту,

де $F = \{ f | f \subset \mathrm{T} \}$, тобто кожен графік може складатися з однієї або більше годин прибуття, що належать множині T. Залежить від зупинки, що розглядається $s \in S$;

– S – множина зупинок;

– R – множина маршрутів;

– W – множина днів тижня.

У наведених позначеннях загальний план маршрутів P, для якого має бути складено розклад, представлятиме множину елементів $p = (f, s, r)$ з простору $F \times \mathrm{S} \times \mathrm{R}$, а множина просторово-часових слотів T для розміщення занять – елементами $t \in W$.

Таким чином, необхідно знайти таке відображення S: $P \rightarrow T$, яке задовольняло б ряд накладених обмежень. А саме, розподіл тільки тимчасових позицій (W) з урахуванням обмежень по кількості маршрутів у поточний момент часу для зупинки, що дозволить скоротити простір пошуку.

Підсумкова структура розкладу представлена двовимірною бінарною матрицею (формула 2.2):

$$T^1 = (t_{i,j}), \tag{2.2}$$

де $t_{i,j} \in \{0,1\}$ - що описує,

$m = |\mathrm{S}|$ - кількість існуючих зупинок,

n – часові проміжки тижня для маршруту на зупинці,

$i = \overline{1, m}$- ідентифікатор зупинки,

$j = \overline{1, n}$- номер дня в часовому просторі розкладу,

Оцінка якості рішення враховує відхилення від початкового розкладу та конфлікти годин прибуття у межах маршруту. Загальна формула оцінки якості має наступний вигляд (формула 2.3):

$$\min\{f_1(p), f_2(p) \dots, f_n(p)\}, p \in P \quad (2.3)$$

де p - розклад,

f_i - штрафна функція і-го критерію,

n - кількість критеріїв,

P - множина розкладів.

Часто вимоги до розкладу є суперечливими, що унеможливлює отримання рішення в якому б досягався мінімум для всіх критеріїв одночасно. Внаслідок цього в задачах складання розкладів часто застосовують поняття парето-оптимальності, де розподіл ресурсів є таким, що будь-яка зміна, спрямована на покращення одного критерію, неминуче призведе до погіршення іншого. Слід зауважити, що парето-оптимальні розклади не підлягають порівнянню між собою, а їх кількість у теорії є необмеженою.

Задачу створення розкладу можна вирішити як глобально, так і локально. Глобальний підхід передбачає пошук оптимального рішення для всієї задачі, враховуючи всі можливі варіанти, але не підходить через високу обчислювальну складність та вимагає значних ресурсів і часу, а наявність конфліктних критеріїв ускладнює пошук рішення. Велика кількість парето-оптимальних рішень також ускладнює вибір оптимального варіанту. Локальний підхід є більш ефективними, оскільки дозволяє зосередитися на окремих критеріях чи обмеженнях та досягти оптимальних рішень у конкретних умовах.

Було обрано вирішення задачі саме в локальному сенсі з зведенням багатокритеріальної задачі до однокритеріальної за допомогою допоміжної функції. Цільова функція являє собою зважену суму штрафів і має наступний вигляд (формула 2.4):

$$f(p) = \sum_{i=1}^{n} w_i \cdot f_i(p), w_i > 0, p \in P \quad (2.4),$$

де p - розклад,

f_i - штрафна функція i-го критерію,

ω_i - вага штрафу i-го критерію,

n - кількість критеріїв,

P - множина розкладів.

2.5 Розробка інтерфейсу системи

Основним критерієм при розробці інтерфейсу була його простота та дружелюбність до користувача. Тобто максимум компактності та підказок до користування. Також варто вдало підібрати стилі та палітру. Було вирішено використати спокійні та легкі тони синього та білого. Щодо стилів, більш вдалим рішенням виявились заокруглені елементи для кнопок, іконок, рамок. Стиль тексту для логотипу також підібрано відповідно іншим елементам.

Інтерфейс вебдодатку складатиметься з 3 основних сторінок: вхід, реєстрація та основне вікно вебзастосунку.

Сторінка входу використана стандартна для Spring Security, щоб прискорити розробку системи в цілому, та містить форму для вводу логіну та паролю користувача (рис. 2.3).

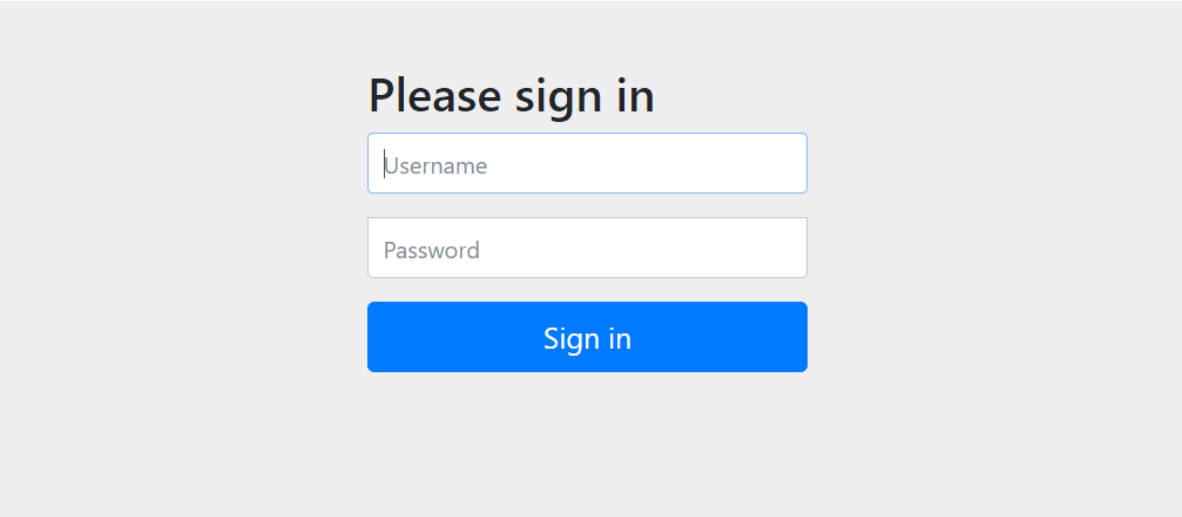

Рисунок 2.3 – Сторінка входу зареєстрованого користувача

Сторінка реєстрації містить форму для вводу пошти, логіну та паролю користувача (рис. 2.4).

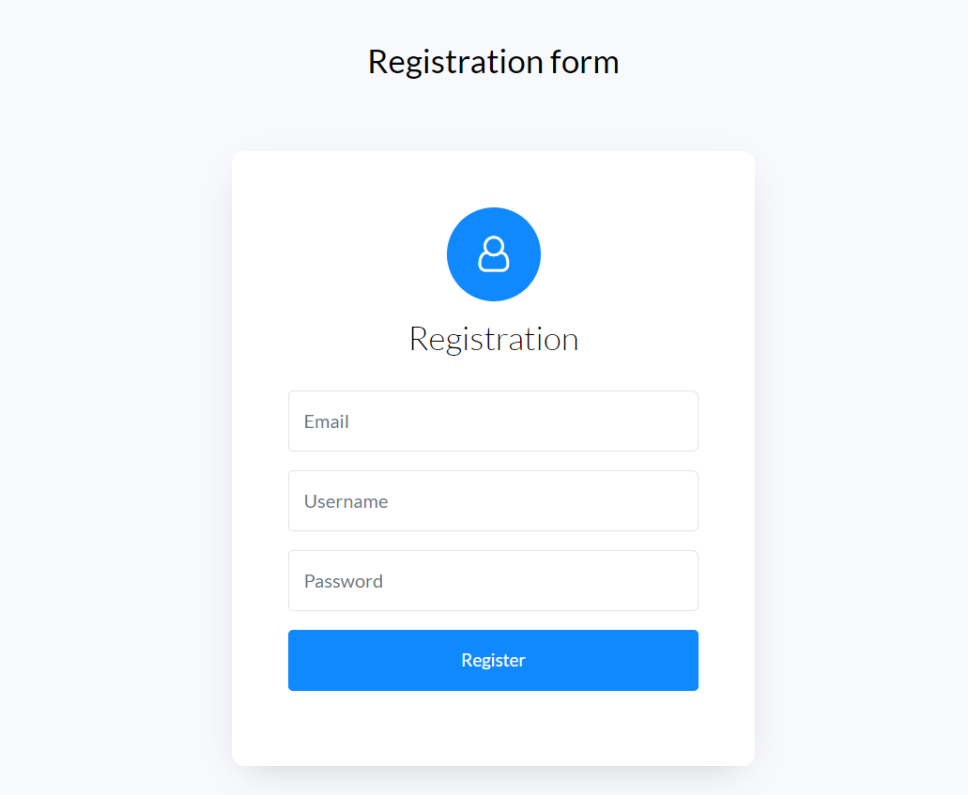

Рисунок 2.4 – Сторінка реєстрації користувача

Основне вікно застосунку міститиме: текстовий логотип вебзастосунку, навігацію у вигляді меню, блок для завантаження вхідного файлу з підказками, меню з кнопками: «Optimize», «Process», «Download», «User Name», «Logout». При натисканні кнопки, де потрібно почекати, буде висвітлено анімацію завантаження, поки операція не закінчиться. При скролінгу сторінки, меню залишається видимим, змінює колір фону з прозорого на світлий та залишається прикріпленим до верхньої частини вікна. У якості фонового зображення всієї сторінки обрано зациклене відео з хмарами. Поки вхідний файл не буде завантажено, деякі кнопки будуть неактивні (рис. 2.5).

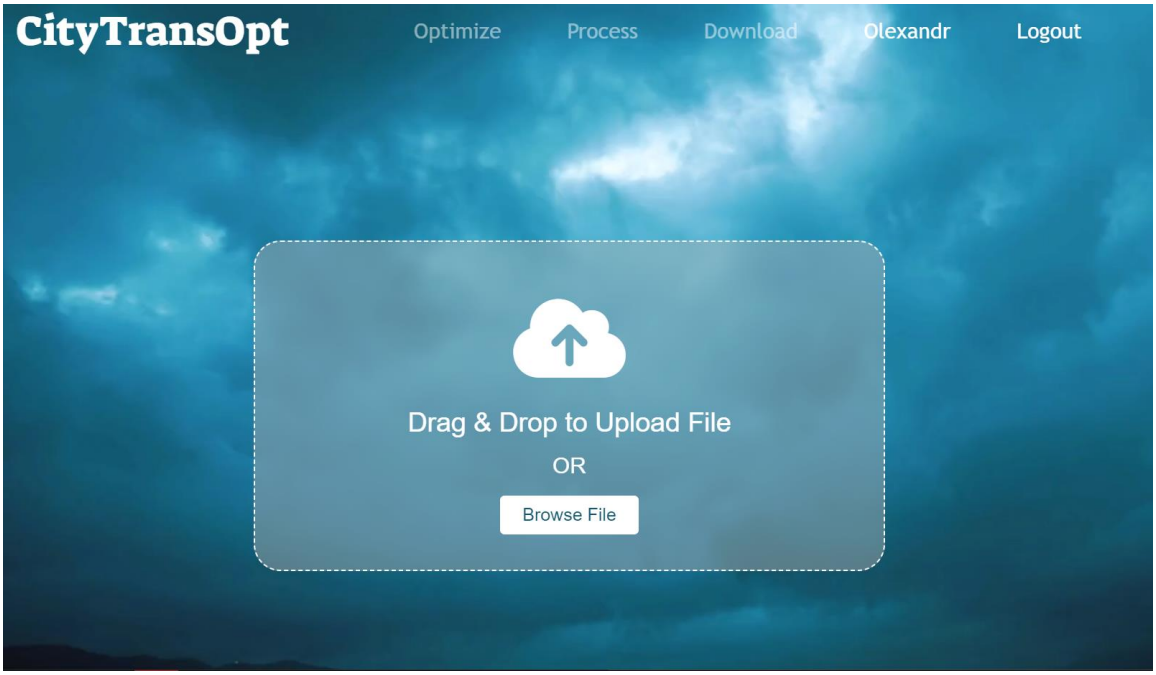

Рисунок 2.5 – Вигляд головної сторінки вебзастосунку «CityTransOpt»

Також необхідно було надати користувачу відповідь на спроби завантажити файл у систему. Реалізовано це зміною опису та кольору блоку завантаження файлу. Наприклад, при завантаженні валідного файлу Excel він буде мати наступний вигляд (рис. 2.6).

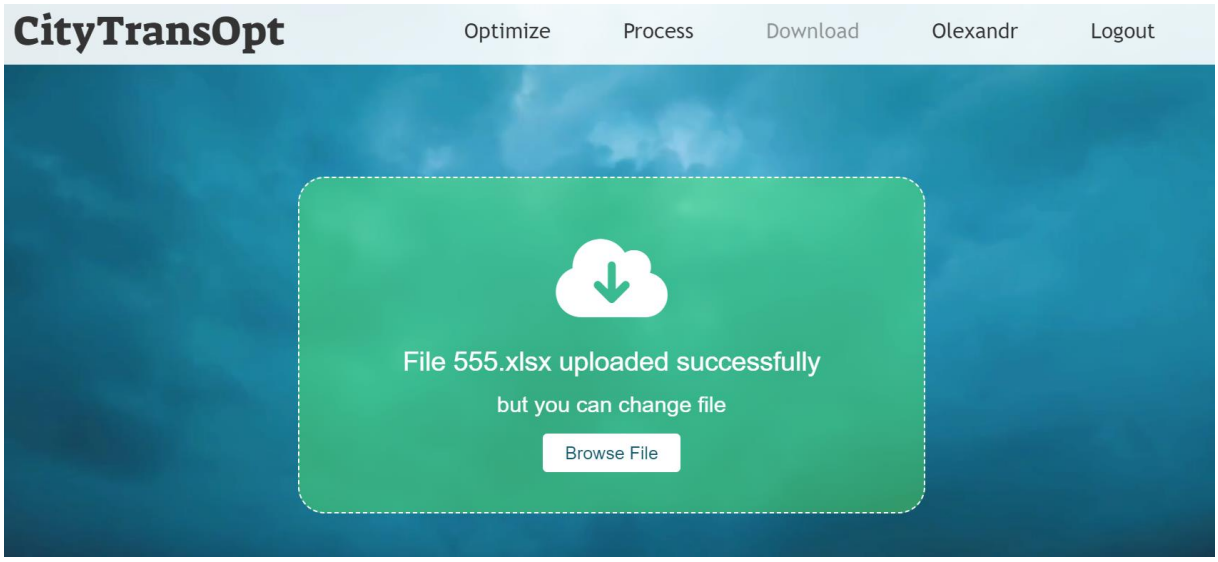

Рисунок 2.6 – Відображення блоку завантаження при валідному файлі

Колір блоку стає зеленуватим та виводиться інформація про завантажений файл. При невалідному файлі, колір буде червоним та матиме відповідно повідомлення з помилкою (рис. 2.7).

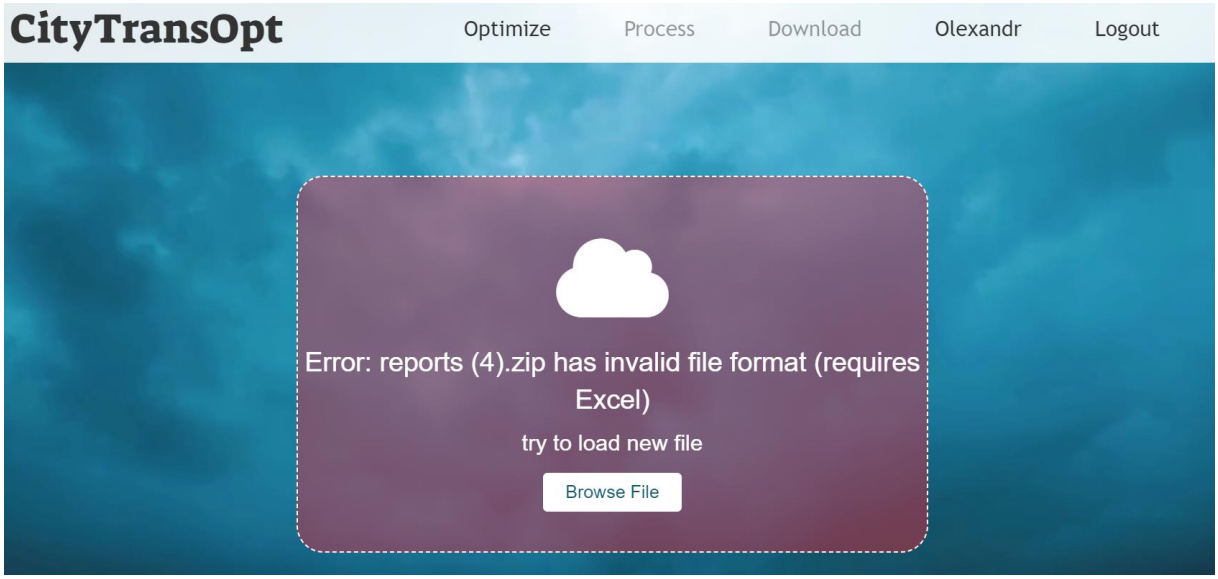

Рисунок 2.7 – Відображення блоку завантаження при невірному файлі

Також для зручності за допомогою JavaScript реалізовано можливість перетягування файлу у блок завантаження (рис. 2.8).

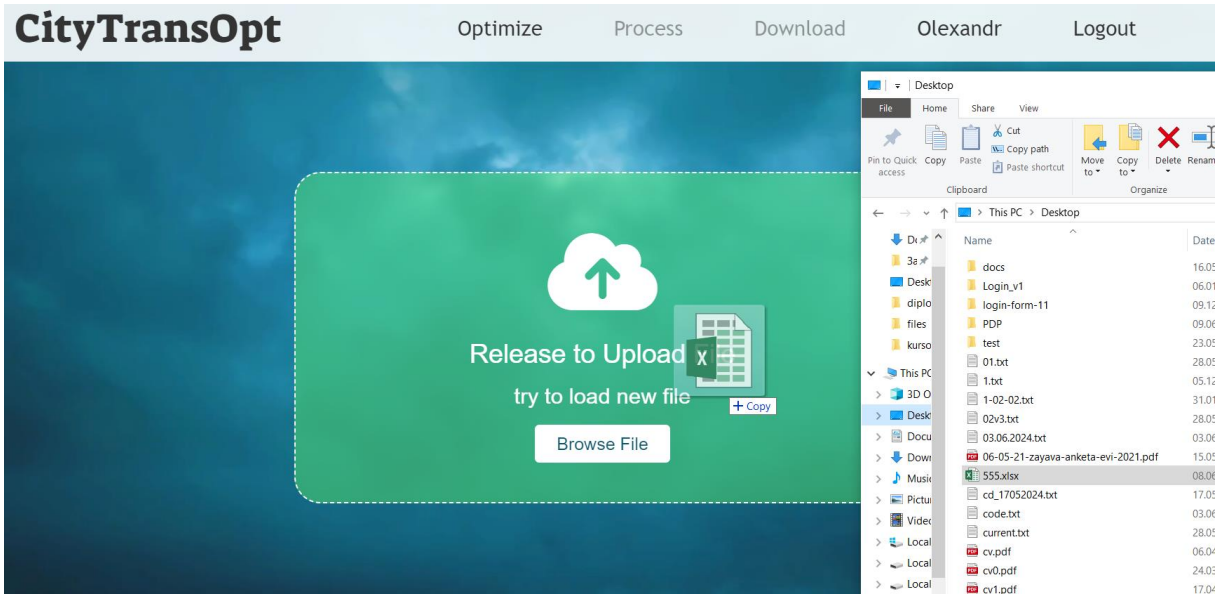

Рисунок 2.8 – Перетягування файлу у блок завантаження

Або ж за допомогою кнопки «Browse File» та викликом провідника Windows. В такому випадку основна сторінка буде затемнена (рис. 2.9).

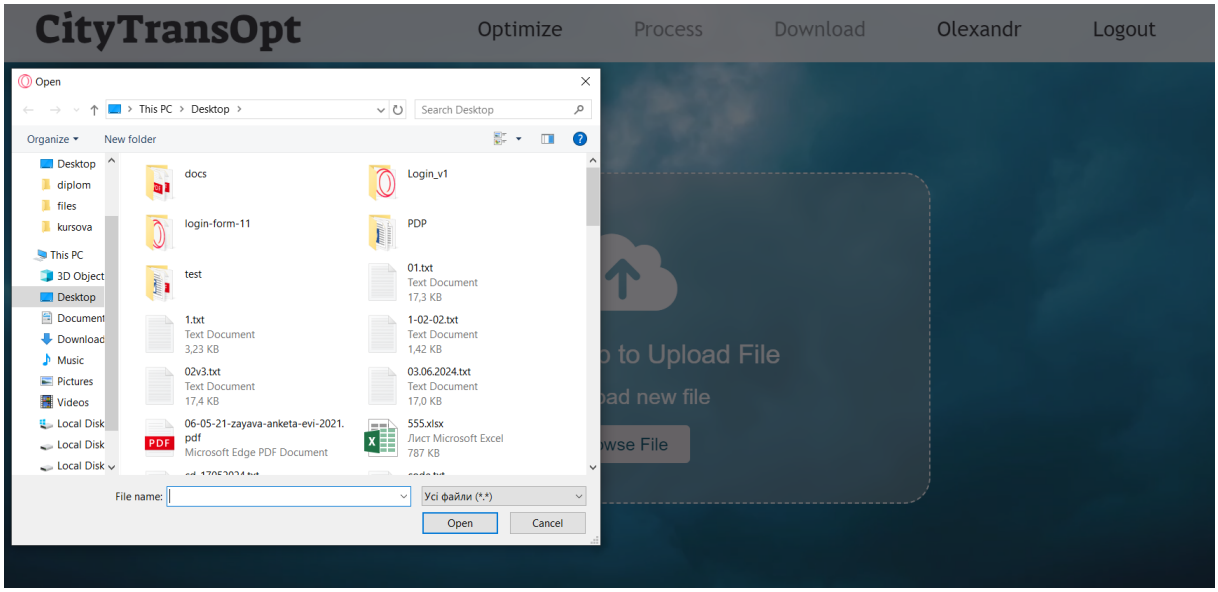

Рисунок 2.9 – Виклик провідника за допомогою кнопки «Browse File»

Як можна побачити, розібратись з розробленим інтерфейсом можливо навіть інтуїтивно.

Крім блоку завантаження вхідного файлу, вебзастосунок містить блок меню з елементами «Optimize», «Process», «Download», «User Name», «Logout».

Елементи меню «Optimize», «Process», «Download» це відповідають методам для оптимізації, опрацювання вхідного файлу та вивантаження вихідного архіву.

Елемент меню «User Name» відображає ім’я користувача.

Елемент меню «Logout» відповідає за вихід з системи для авторизованого користувача.

2.6 Розробка моделі системи

Для відображення роботи вебзастосунку для оптимізації мережі громадського транспорту була розроблена діаграма діяльності.

Діаграма діяльності (Activity Diagram) ілюструє послідовність дій та потік повідомлень у системі. Ця діаграма візуалізує виконання роботи системи та її функціональність, що робить її більш зрозумілою як і для розробників, так і для користувачів [8].

Діаграма діяльності системи показана на рис. 2.10.

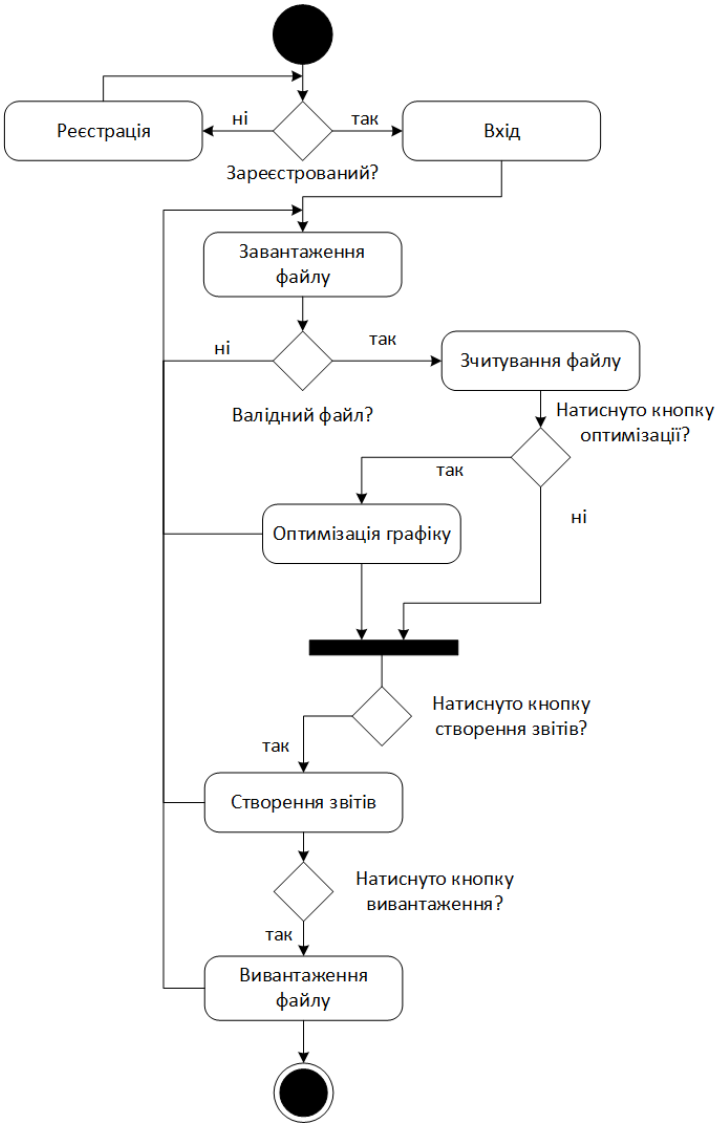

Рисунок 2.10 – Діаграма діяльності системи

На початковому етапі користувач проходить реєстрацію та вхід до системи. Після цього він може завантажити файл у форматі Excel з інформацією про розклад маршрутів.

Якщо файл є валідним, користувач може або викликати метод оптимізації, або відразу викликати метод для генерування розкладів для зупинок.

Якщо генерування розкладів закінчилось, користувачу стане доступна кнопка вивантаження архіву з результатами

Тобто, логіка вебзастосуноку реалізована так, що без входу в профіль не можна завантажити файл. Зроблено це насамперед для захисту від спроб перевантажити сервіс та задля уникнення лишніх викликів модулів.

2.7 Розробка алгоритмів роботи системи

Для основних алгоритмів було створено діаграмами. Серед них: реєстрація користувача, вхід користувача, оптимізація графіку, завантаження файлу, парсинг даних, створення розкладів у Word, пакування розкладів в архів, вивантаження архіву.

Алгоритм реєстрації користувача було спрощено, щоб не перенавантажувати розробку системи (рис. 2.11).

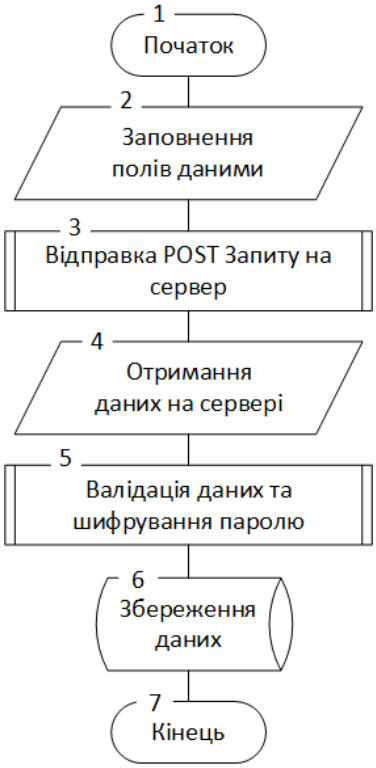

Рисунок 2.11 – Блок-схема спрощеного алгоритму реєстрації користувача

Користувач на сторінці реєстрації заповнює необхідні поля у формі. Далі натискає кнопку «Register», відправляється «POST» запит на сервер з даними користувача для реєстрації. Сервер обробляє запит, перевіряє дані, шифрує пароль та створює запис у базі даних для подальшого входу при відсутності.

Алгоритм авторизації користувача представлено на рис. 2.12.

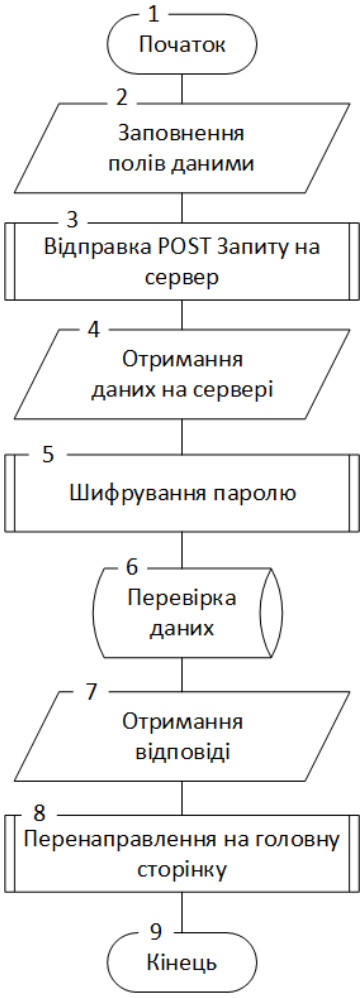

Рисунок 2.12 – Блок-схема спрощеного алгоритму авторизації користувача

Користувач на сторінці входу заповнює необхідні поля у формі. Далі натискає кнопку «Login», відправляється «POST» запит на сервер з даними користувача для входу. Сервер обробляє запит, перевіряє дані, шифрує пароль, відправляє запит у базу даних та на основі відповіді здійснює вхід в акаунт.

Діаграма алгоритму оптимізації графіку представлена на рис. 2.13.

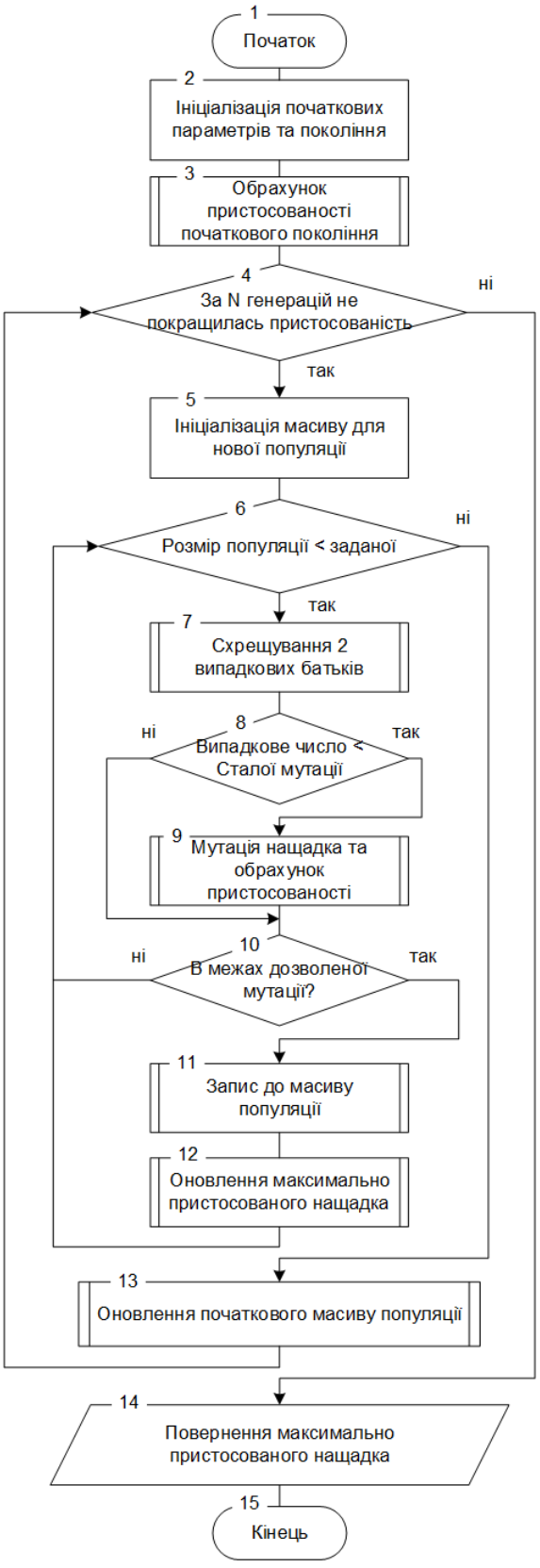

Рисунок 2.13 – Блок-схема алгоритму оптимізації розкладу

Для налагодження роботи алгоритму оптимізації використано такі параметри: кількість генерацій, стала мутації, максимальний інкремент та декремент, розмір популяції. Алгоритм працює поки не пройде заплановану поділку кількості генерацій пристосованість найкращого нащадка не буде максимальною. Спочатку генерується нащадок з випадково обраних батьків у яких беруться найкращі риси. Далі з певним шансом відбувається мутація нащадка. Якщо після мутації нащадок не перевищив граничні значення мутації відносно початкового покоління, то він додається до поточних популяції. Для наступної генерації ці нащадки будуть у якості батьків. Результатом виконання алгоритму є максимально пристосований нащадок з вибірки ряду популяцій.

Процес завантаження файлу у пам’ять серверу представлено на рис. 2.14.

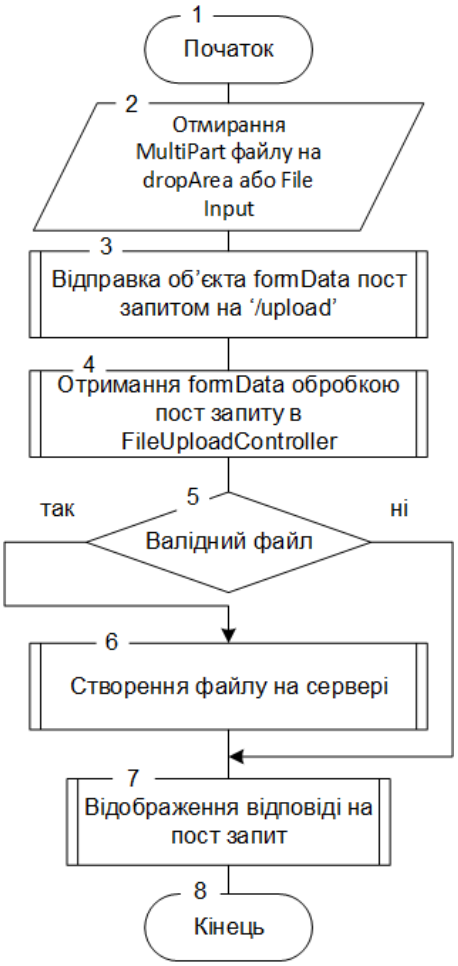

Рисунок 2.14 – Блок-схема алгоритму завантаження файлу пам’ять серверу

Дані на сервері зберігаються не у вигляді локального файлу, а збереженням MultiPart файлу у мапі з іменем користувача у якості ключа. Таким чином для кожного користувача виділяється своє місце для завантаження файлу.

Алгоритм парсингу вхідних даних зображено на рис. 2.15.

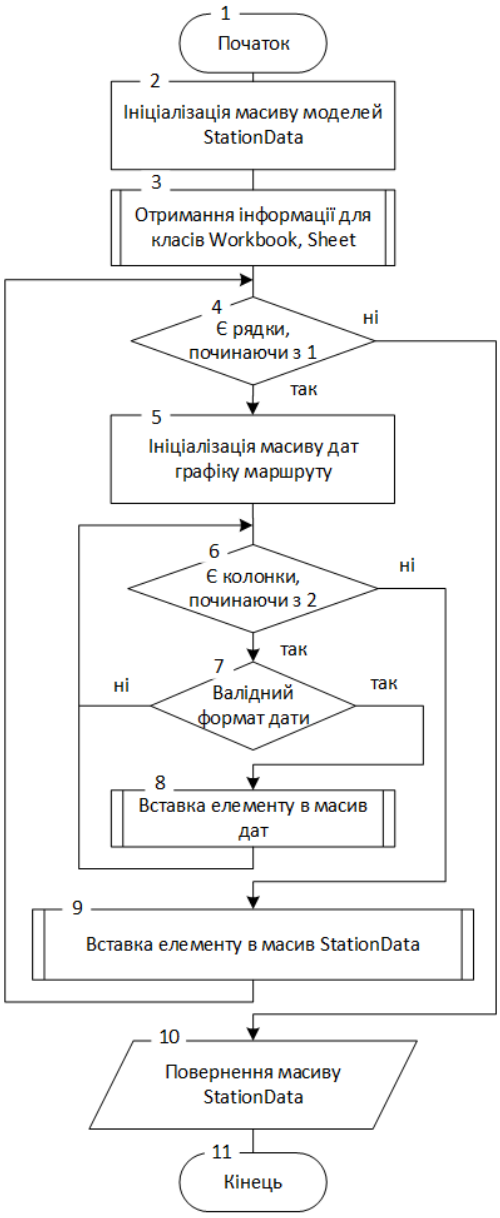

Рисунок 2.15 – Блок-схема алгоритму парсингу файлу

Алгоритм парсингу файлу починається з ініціалізації масиву моделей StationData, класів Workbook і Sheet. Далі алгоритм перевіряє, чи є рядки в електронній таблиці, починаючи з першого рядка. Якщо рядки є, ініціалізується

масив дат графіка маршруту і перевіряється, чи є колонки таблиці, починаючи з другої. Якщо колонки є, алгоритм перевіряє, чи є дані в поточній колонці дійсною датою. Якщо все вірно, дата додається до масиву дат графіка маршруту і елемент до масиву моделей StationData. Потім алгоритм переходить до наступного рядка. Якщо більше рядків немає, алгоритм завершується.

Алгоритм створення розкладу-звіту для зупинки зображено на рис. 2.16.

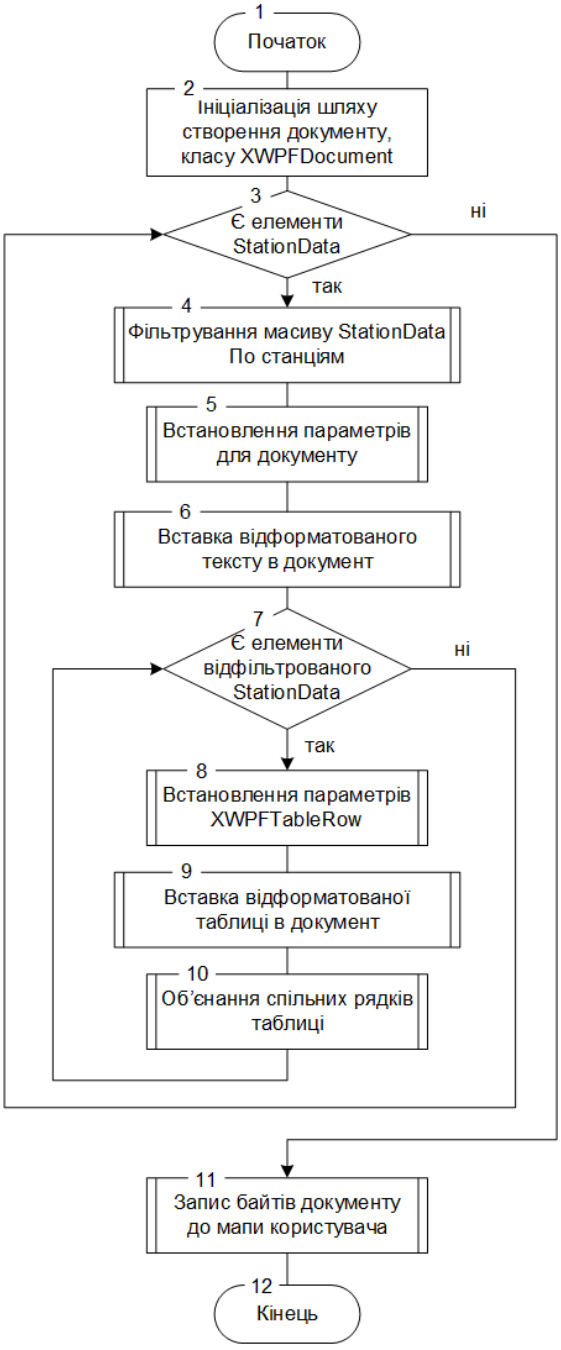

Рисунок 2.16 – Блок-схема алгоритму створення розкладу-звіту для зупинки

Алгоритм створення звіту починається з ініціалізації шляху до документа. Потім створюється новий документ за допомогою класу XWPFDocument. Далі масив Station Data фільтрується по станціях. Для кожної станції встановлюються параметри для документу, вставляється відформатований текст і таблиця. Потім об’єднуються спільні рядки таблиці, і байти документу зберігаються у мапі файлів відповідного користувача.

Алгоритм архівування файлів записує у OutputStream файли типу ZipEntry з масиву файлів, який передано у функцію (рис.2.17).

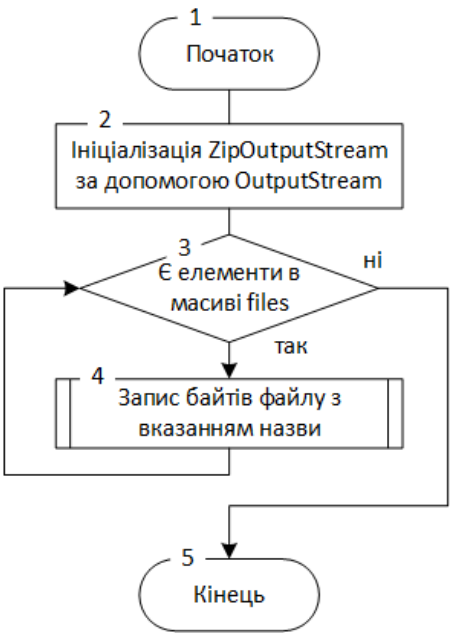

Рисунок 2.17 – Блок-схема алгоритму архівації файлів

В кінці роботи алгоритму всі відкриті потоки автоматично закриваються завдяки використанню try-with-resources конструкції.

Алгоритм вивантаження архіву користувачеві зображено на рис. 2.18.

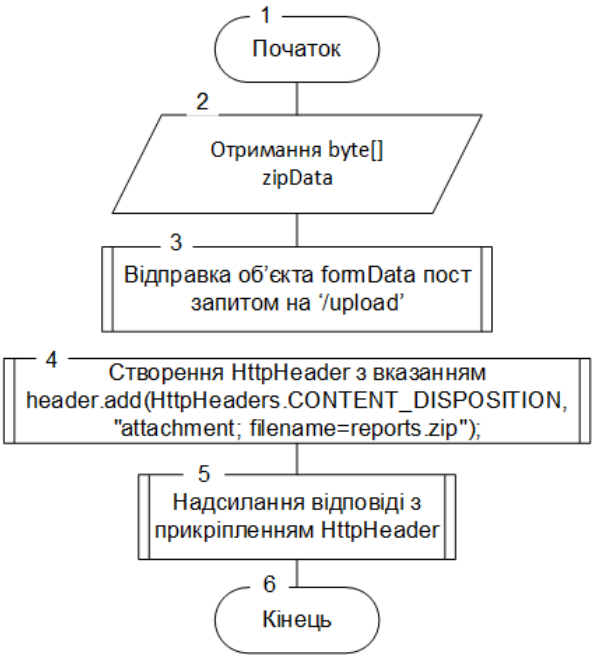

Рисунок 2.18 – Блок-схема алгоритму вивантаження архіву

Алгоритм викликається при отриманні «GET» запиту на адресу «/download». Відповіддю є файл reports.zip, в який записано байти сформованого масиву. Тобто, жодного локального файлу на сервері не створюється, все зберігається в масивах байтів та видаляється після вивантаження.

2.8 Висновки

У другому розділі було проаналізовано вихідні та вхідні дані вебзастосунку. Було розроблено основні алгоритми системи, такі як реєстрація користувача, вхід користувача, оптимізація графіку, завантаження файлу, парсинг даних, створення розкладів у Word, пакування розкладів в архів, вивантаження архіву. Для цього було використано бібліотеку Apache POI для роботи з документами формату docx, xlsx та архівування результатів. Також була розроблена модель роботи програмного продукту за допомогою діаграми діяльності.

3. РОЗРОБКА ПРОГРАМНИХ МОДУЛІВ РЕАЛІЗАЦІЇ СИСТЕМИ З ОПТИМІЗАЦІЇ МЕРЕЖІ ГРОМАДСЬКОГО ТРАНСПОРТУ

3.1 Обґрунтування вибору засобів для реалізації системи

Розробка програмних модулів для створення вебдодатку, який оптимізує громадський транспорт, потребує уважного вивчення найкращих підходів до цієї задачі. Для досягнення цієї мети було обрано наступні інструменти та технології:

1. Spring Boot. Spring Boot надає просту структуру для створення вебзастосунків, що дозволяє швидко розгортати проекти. Його гнучкість полягає в можливості легко налаштовувати та розширювати функціонал за допомогою вбудованих інструментів та бібліотек.

Spring Boot – частина фреймворку Spring Framework і дозволяє зручно розробляти та тестувати Spring додатки. Завдяки Spring можна автоматично збирати проекти Maven і Gradle, упаковувати додаток в jar-файл зі вбудованим контейнером Tomcat. Spring Boot дозволяє мінімізувати використання конфігураційних xml, що зазвичай потрібні для розробки Spring додатків.

Важливим елементом у будь-якому додатку є валідація даних (Validation). Spring Boot підтримує Jakarta Validation, раніше відому як Bean Validation, яка є стандартом для валідації об’єктів у Java. Використовуючи анотації, такі як @NotNull, @NotBlank, @Size, @Email та інші, можна легко додавати правила валідації до класів моделі. Завдяки такому механізму можна запевнити, що дані, введені користувачами або отримані з інших джерел, будуть коректними та придатними для подальшої обробки.

Spring Framework має широкий спектр можливостей для розробки java enterprise додатків. Він складається з множини менших фреймворків (наприклад Spring Security, Thymeleaf, Spring Data MongoDB), які працюють незалежно, але мають кращу функціональність при одночасному використанні [9].

Spring Security – це java фреймворк, розроблений для вирішення проблем безпеки додатків, включаючи авторизацію, автентифікацію, захист від

різноманітних атак та інших загроз. Він дозволяє легко інтегрувати різні механізми аутентифікації та керувати доступом до ресурсів. Крім того, Spring Security має захист від веб-загроз, таких як CSRF та XSS, і підтримує гнучку конфігурацію через Java або XML.

Thymeleaf – це сучасний шаблонізатор Java на стороні сервера, що акцентує увагу на використанні природних HTML-шаблонів, які можна переглядати в браузері без запуску самого сервера. Це дуже зручно при розробці сторінок вебзастосунку. На сьогодні Thymeleaf пропонує один із найбільших наборів функцій, для полегшення заміни JSP-сторінок, і активно розвивається та підтримується. Під JSP розуміються динамічні сторінки, у яких динамічна частина генерується завдяки Java, а статична мовою розмітки, найчастіше HTML.

Spring Data MongoDB — це частина Spring Data, яка спрощує роботу з MongoDB та забезпечує зручний API для виконання CRUD-операцій (create-read-update-delete) підтримує складні запити, індексацію та агрегації, автоматично перетворює Java об’єкти у документи MongoDB і навпаки. А інтеграція з Spring Boot дозволяє швидко налаштувати підключення до MongoDB і створювати репозиторії. Наприклад можна унаслідуватись від інтерфейсу MongoRepository, який надає базові методи для роботи з базою даних MongoDB без необхідності написання власних запитів, що дозволяє виконувати основні операції з базою даних, такі як збереження, видалення, оновлення та пошук об’єктів.

Наприклад, якщо є клас User з полями id, name та email, то можна створити інтерфейс UserRepository, який розширить MongoRepository<User, String> (де User – клас моделі, а String - ідентифікатор), і викликати методи save(), findById(), findAll(), deleteById(), тощо. MongoRepository також підтримує створення власних методів запитів, які базуються на іменованій конвенції. Наприклад, findByEmail(String email) або deleteByName(String name).

2. Maven. Maven є потужним інструментом для управління залежностями у Java додатках, що дозволяє зручно підключати сторонні бібліотеки, управляти

їх версіями та робить процес розробки простішим і ефективнішим. Крім того, у IntelliJ IDEA є плагін «Edit Starters», який спрощує підключення часто використовуваних фреймворків. Для налаштування maven треба в файл pom.xml додати потрібні рядки коду з офіційного сайту Maven Repository. При роботі з будь-яким фреймворком використання maven є необхідним саме завдяки вирішенню проблеми з завантаження сторонніх бібліотек [10].

3. MongoDB. MongoDB це NoSQL база даних, що надає таку структуру документів, яка може зберігати дані з різними полями у колекції. MongoDB – це документо-орієнтована база даних, яка використовує BSON (Binary JSON) для зберігання даних. Основні поняття MongoDB включають в себе базу даних, колекції, документи, поля та _id.

База даних в MongoDB є загальним сховищем для колекцій документів. Це важлива особливість MongoDB, що вона фізично не існує, поки є пустою.

Колекція в MongoDB – це набір документів, який аналогічний таблиці в реляційних базах даних. Одна колекція може містити документи з різними розмірами, зв’язками та структурами.

Документ в MongoDB – це файл у форматі BSON, який зберігає дані у вигляді ключів та відповідних значень. Документ може містити інформацію зі складною структурою, що робить його більш гнучким порівняно з рядком у реляційній базі даних.

Поле в MongoDB – це запис, який містить набір даних та є аналогом стовпця у таблиці реляційної бази даних.

_id в MongoDB – це ідентифікатор, який є обов’язковим для кожного документа та є аналогом первинного ключа. Він допомагає однозначно ідентифікувати документ та служить для відмінності записів один від одного та автоматично генерується, якщо не заданий явно при створенні нового документа [11].

4. Git. Git – це система керування версіями з розподіленою архітектурою, яка призначена для відстеження змін у коді під час його розробки.Git дозволяє

спільно працювати, вносячи зміни у код, відстежувати історію змін, розвивати різні гілки для роботи над різними функціями, а також об’єднувати їх у єдину версію, робити бекапи старих версій. Git є дуже популярним інструментом серед розробників через його швидкість, ефективність та можливості. Також його можна використовувати для поширення коду в інтернеті та для аналізу сайтом чи рев’ю іншими користувачами (рис. 3.1). Ігнорування файлів та папок для завантаження налаштовується завдяки файлу .gitignore.

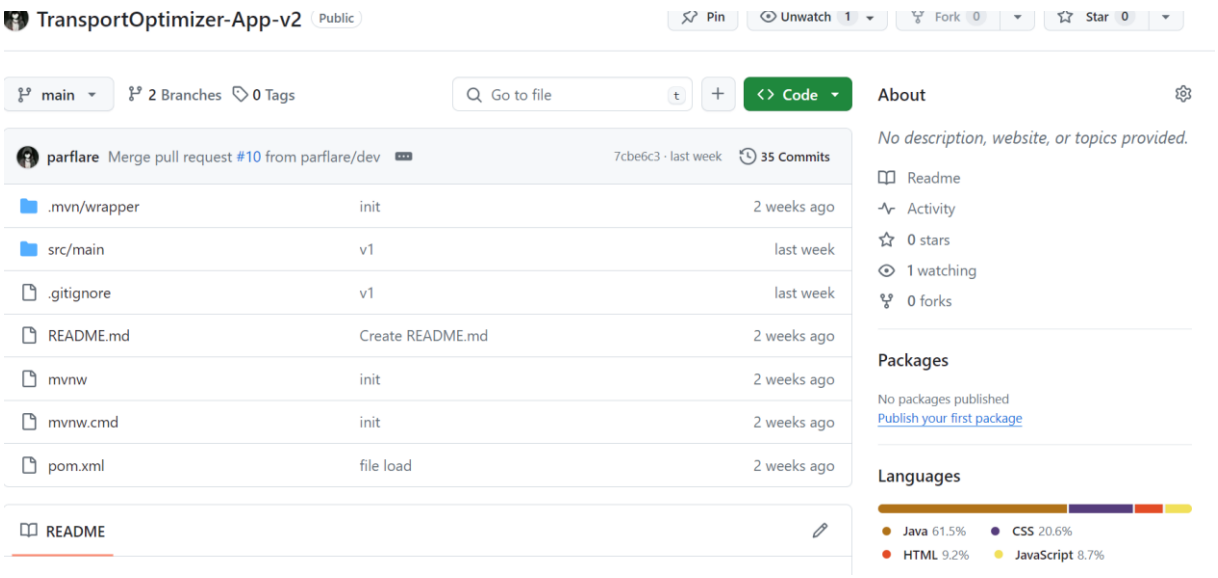

Рисунок 3.1 – Завантажений проект на Github

5. Apache POI. Apache POI надає інтерфейс для роботи з файлами Excel та Word для Java додатків. Використання вбудованих класів Apache POI спрощує парсинг даних та створення звітів. Для роботи з Excel є класи ExcelExtractor і XSSFExcelExtractor, для роботи з Word – WordExtractor і XWPFWordExtractor, для PowerPoint – SlideShowExtractor і для Publisher – PublisherExtractor. Для кожного формату файлу є відповідний клас, який дозволяє отримати текст з документа [12].

За допомогою методів бібліотеки Apache POI можна створити звіт у ворді наступного вигляду (рис. 3.2).

Зупинка «Автовокзал»

№ маршруту	Назва маршруту	Дні роботи	Розклад руху
1	Ринок-Райлікарня	будні дні	07:14, 07:51, 08:56, 09:28, 10:05, 10:58, 12:14, 12:58, 13:26, 14:35, 15:26, 16:28, 17:28, 18:19
		субота	07:14, 08:52, 10:05, 12:14, 13:32, 15:22, 17:32
		неділя	07:14, 07:52, 08:52, 09:32, 10:05, 10:52, 12:14, 12:52, 13:32, 14:32, 15:22, 16:22, 17:32, 18:22
		будні дні	07:23, 07:52, 08:24, 08:53, 09:24, 09:54, 10:23, 10:54, 11:24, 11:53, 12:24, 12:54, 13:23, 13:53, 14:24, 14:54, 15:23, 15:54, 16:24, 16:53

Рисунок 3.2 – Вигляд створеного системою звіту

6. Postman. Postman є інструментом для тестування і автоматизації API (Application Programming Interface). Він дозволяє розробникам створювати, надсилати та аналізувати різні HTTP-запити (GET, POST, PUT, DELETE тощо) до API. Це допомагає виявляти та виправляти помилки, навіть без розробленого інтерфейсу тестованої майбутньої системи [13].

В якості структури проекту було дотримано легко масштабованого шаблону MVC (Model-View-Controller). Його суть в тому, що система поділяється на три складові, які можуть змінюватись окремо один від одного (рис. 3.3).

У патерні MVC, Controller і View залежать від Model, а Model від них – ні. У контролері знаходиться логіка отримання даних з моделі та передавання їх до представлення. Контролер передає запити, сформовані у View Model, де зберігаються всі дані. Модель змінюється відповідно до отриманого запиту. Представлення отримує необхідні зміни від моделі і відображає оновлені дані [14].

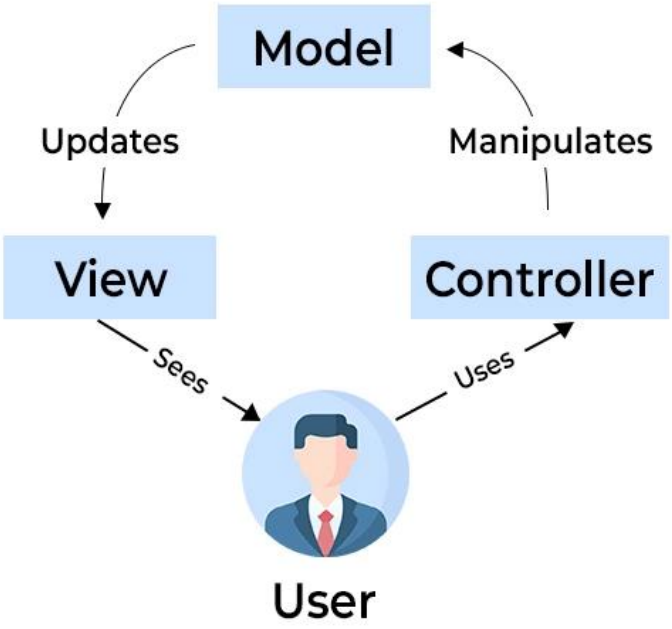

Рисунок 3.3 – Діаграма шаблону MVC

В якості підходу до архітектури мережевих протоколів було використано REST. Вебзастосунок складається з двох основних частин: frontend (клієнтська частина) і backend (серверна частина), що більш зрозуміло пояснено на рис. 3.4. REST виступає посередником між цими частинами, дозволяючи frontend надсилати запити до backend і отримувати дані у відповідь [15].

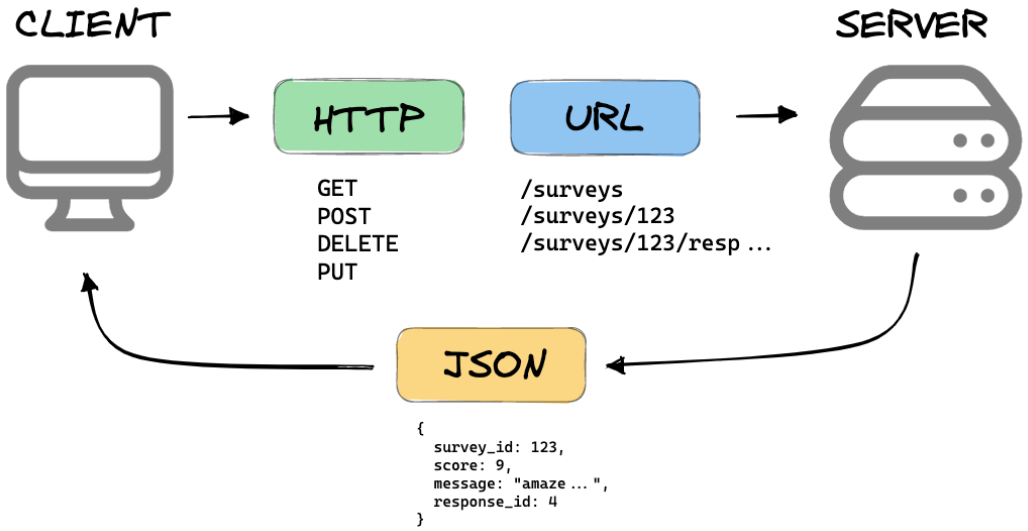

Рисунок 3.4 – Діаграма патерну REST

Отже, вибір технологій для розробки вебдодатку, що оптимізує громадський транспорт, виявився ключовим етапом. Використання шаблону MVC, Spring Boot для швидкого розгортання проектів, Maven для керування залежностями, MongoDB для зберігання даних авторизації, Apache POI для роботи з файлами Excel і Word, REST та Postman для тестування API створить зручне для розробника середовище, що забезпечить кращу якість системи та ефективність для користувачів відповідно.

3.2 Розробка модуля інтерфейсу системи

Створення простого та дружнього до користувача інтерфейсу є ключовим аспектом у розробці програмного забезпечення. Інтерфейс визначає, як користувачі взаємодіють з програмою, забезпечуючи їхній комфорт та задоволення від використання продукту.

Простий інтерфейс спрощує навігацію та розуміння функцій програми для користувачів будь-якого рівня обізнаності, що дозволяє їм ефективно використовувати програму.

Дружній інтерфейс враховує потреби та очікування користувачів, зробивши взаємодію з програмою приємною та зрозумілою. Це включає в себе зручне розташування елементів керування, зрозумілі підказки та інструкції, а також інтуїтивно зрозумілі дії. Створення такого інтерфейсу допомагає залучити більше користувачів до використання програми, підвищує їхню задоволеність від роботи з нею і сприяє позитивному сприйняттю продукту на ринку.

Фронтенд будь-якого сайту або програми можна розділити на три основні частини:

HTML – мова розмітки документів, за допомогою якої створюється структура сторінки, включаючи заголовки, абзаци та інші елементи.

CSS – мова, яка відповідає за зовнішній вигляд документа. За допомогою CSS-коду визначаються стилізація та оформлення елементів, що дозволяє браузеру правильно відображати сторінку.

JavaScript – мова програмування, яка надає можливість "оживити" вебсторінку. Вона відповідає за обробку подій, таких як кліки мишки, переміщення курсора, а також інших дій користувача, що дозволяє зробити вебсторінку більш інтерактивною і зручною у використанні [16].

Тому, інтерфейс вебзастосунку створено за допомогою HTML, CSS та JavaScript. Основну структуру вебзастосунку визначає файл index.html, який містить основний вміст сторінки, такий як заголовки, тексти, кнопки та інші елементи. Цей файл відповідає за відображення вебзастосунку і є основою для взаємодії користувача з ним. Згідно запланованого функціоналу засобами html, css та js було створено два основні блоки сторінки – меню та поле завантаження файлу. Лістинг представлено у додатку В, модулі index.html.

Усі елементи меню добре організовані та мають своє практичне призначення. У меню наявні кнопки «Optimize» (оптимізація графіку), «Process» (генерація звітів-розкладів), «Download» (завантаження файлів), «UserName» (ім’я користувача) та «Logout» (вихід з системи). Для окремих кнопок меню було створено клас «disabled», який відповідає за блокування елементів, поки не буде виконано певних умов. Крім того створено клас «loader», що відповідає за відображення анімації процесу виконання викликаної функції.

За завантаження файлу користувачем при перетягуванні відповідає блок div на html з тегом класу «upload-file-area». Крім того створено елемент input з ідентифікатором «fileInput», який є допоміжним методом, щоб викликати діалогове вікно для завантаження через системний провідник файлів.

До елементів застосовуються різні стилі завдяки styles.css. Деяку логіку з інтерактивністю реалізовано за допомогою scripts.js.

Наприклад зміна кольору хедеру при прогортанні вниз реалізована методом slowScroll, який приймає ідентифікатор позиції заголовку та прокручує до нього сторінку з фіксованою швидкістю та уповільненням біля необхідного місця. Лістинг представлено у додатку В, модулі scripts.js.

Блок з перетягуванням файлу виконує схожу логіку, та має 3 стани: active, failed та standard. Кожен з яких додається відповідно до певного обробника подій. Active це успішне завантаження файлу, failed – помилка завантаження чи формату файлу та standard – початковий стан поля, який реалізовано не класом. Лістинг представлено у додатку В, модулі scripts.js.

Крім перетягування файлу користувачем було реалізовано стандартний виклик вікна вибору файлу через кнопку «Browse File».

Також було забрано перетягування його елементів меню за допомогою preventDefault методу для події перетягування, що фактично його відміняє при виклику.

У якості фону було використано зациклений відео-фрагмент руху хмар на синьому фоні, що по кольоровому поєднанні та анімації буде привабливим для користувача.

Розробка інтерфейсу є важливим етапом для вебзастосунку, тому було враховано всі потреби та можливі побажання користувачів, як з точки зору функціоналу, так і візуальної частини та юзабіліті.

3.3 Розробка модуля завантаження файлу в систему

Розробка модуля завантаження файлу в систему є важливою складовою згідно основної логіки запланованого вебзастосунку, що дозволить користувачам передавати файли на сервер для подальшої обробки або зберігання.

Модуль складається з методу loadFile на JavaScript, лістинг якого представлено у додатку В, модуль scripts.js.

Метод loadFile створює об’єкт FormData, додає до нього файл й ім’я користувача. Потім виконує «POST» запит на сервер за url «/upload». Якщо відповідь від сервера успішна, змінюються CSS класи елемента iconElement, оновлюється текст у dragText та dragSpan, а кнопки з ідентифікаторами processFileButton та optimizeFileButton стають активними. Якщо відповідь

неуспішна, відображається повідомлення про помилку, processFileButton стає неактивною. Отриманий запит на «/upload» «ловить» клас FileController та відповідно обробляє згідно бізнес-логіки у методі uploadFile з тегом @PostMapping(«/upload»).

Сам ж метод обробляє запит на шлях «/upload», приймаючи параметрами ім’я користувача та Multipart файл. Потім викликається метод saveFile з fileService, передаючи туди ім’я користувача та файл, повертається відповідь з результатом у випадку успішного виконання. Якщо виникає помилка, то відповідь міститиме код статусу HTTP 500 (Internal Server Error) і відповідним повідомленням.

Метод saveFile приймає ім’я користувача та файл, зчитує ім’я файлу та перевіряє його формат. Якщо файл має формат Excel-файлу, то зчитуються байти та передаються до методу processExcelFile класу excelProcessor, який оброблює файл та повертає дані у вигляді списку StationData. Отримані дані зберігаються у мапі userFileDataStorage з ключем імені користувача. В кінці метод повертає рядок з інформацією про успішне завантаження файлу.

3.4 Розробка модуля парсингу файлу

Основна мета модуля парсингу файлу – це автоматизувати процес обробки даних, що містяться у файлі. Так, як і попередній модуль, модуль парсингу даних викликається зі сторони клієнта «POST» запитом через JavaScript на url «/process», а контролер опрацьовує запит та виконує бізнес логіку.

Модуль складається з методу processExcelFile на Java, лістинг якого представлено у додатку В, модуль ExcelProcessor.java.

Хоча файл має деякі обмеження по шаблону, проте вони є мінімальними. В іншу чергу, метод обробки файлу не вимагає якихось параметрів для налаштування, а відразу зчитає весь файл по вказаному листу.

Цей модуль може бути частиною більшої системи або самостійним інструментом.

Метод для обробки Excel файлу (processExcelFile) приймає масив байтів файлу як вхідний параметр і повертає список об'єктів StationData. Сам метод починається з оголошення списку data, який буде заповнений даними з файлу. За допомогою спроби відкрити робочу книгу з вхідного масиву байтів, створюється об’єкт Workbook. З книги обирається лист «Combined Data», після чого визначається кількість рядків та створюється новий список для збереження даних з файлу.

Далі, за допомогою циклу, кожен рядок листа обробляється окремо, починаючи з другого рядка (оскільки перший містить заголовки). З кожного рядка зчитуються значення першої та другої клітинок, які містять назву станції та інформацію про маршрут відповідно. Після цього створюється ще один список для зберігання дат з інших клітинок цього рядка. Всі наступні клітинки перевіряються на те, чи містять вони дату, і якщо так, додаються до списку.

Після зчитування всіх даних з поточного рядка, створюється новий об’єкт StationData, з назвою станції, інформацією про маршрут та список дат. Цей об’єкт додається до списку data. Після завершення обробки всіх рядків, робоча книга закривається, і метод повертає заповнений список data.

Якщо не вдаватись у деталі коду, то основні кроки роботи методу включають:

1. Створення робочої книги: Функція відкриває файл Excel і отримує посилання на вибраний лист у цьому файлі.
2. Парсинг даних: По кожному рядку у вибраному листі функція отримує необхідні значення (назва станції, інформація про маршрут, час прибуття).
3. Створення об’єктів StationData: Для кожного рядка у листі функція створює об'єкт StationData, який містить отримані дані.
4. Повернення результату: Функція повертає список об’єктів StationData, який містить дані, що були зчитані з файлу Excel.

У випадку помилки під час парсингу функція виводить відповідне повідомлення у консоль і повертає null.

3.5 Розробка модуля створення звітів

Основна мета модуля створення звітів – це автоматизувати процес формування звітів для користувачів, зменшити час, необхідний для їх підготовки, і забезпечити правильність створених звітів.

Модуль складається з методу createDocx на Java, лістинг якого представлено у додатку В, модуль FileService.java.

Метод createDocx створює Docx-документ для певної зупинки на основі наданих даних та зберігає його у форматі байтів. Метод формує ім’я файлу, створює XWPFDocument, встановлює його властивості за допомогою допоміжних методів класу docxGenerator (метод setDocumentProperties), додає відформатований текст із назвою зупинки (метод createText) та таблицю з даними (метод createTable). Далі документ записується у ByteArrayOutputStream, а байти отримані з нього зберігаються у мапі userReportStorage з ключем імені користувача. При успішному створенні документа, метод повертає true, інакше – false, виводячи помилку у консоль.

Метод setDocumentProperties класу docxGenerator встановлює властивості документа Word. На вході отримує об’єкт XWPFDocument та встановлює розміри сторінки документа на формат А4, 21х29.7 см, лістинг коду представлено у додатку В, модуль DocxGenerator.

Метод createText класу docxGenerator створює текст у документі Word з вирівнюванням по центру, інтервалами перед і після абзацу та міжрядковим інтервалом, лістинг коду представлено у додатку В, модуль DocxGenerator.

Отримуючи об’єкт XWPFDocument і рядок тексту, який потрібно вставити у документ, метод створює новий абзац для тексту з встановленим вирівнюванням по центру, інтервалами та іншими властивостями форматування. Після цього метод створює об’єкт XWPFRun, який представляє текстовий рядок у документі, і до нього додає вказаний текст з шрифтом Times New Roman розміром 14 пунктів.

Метод createTable класу docxGenerator створює таблицю у документі Word, вставляє в неї текст, форматує його та сполучає спільні комірки, лістинг коду представлено у додатку В, модуль DocxGenerator.

3.6 Розробка модуля оптимізації графіків

Модуль представлено методом optimizeSchedule, що оптимізує розклад автобусів за допомогою генетичного алгоритму, лістинг коду представлено у додатку В, модуль optimizeSchedule.java.

Оптимізація полягає у коригуванні даних про прибуття маршруту на зупинки таким чином, щоб не було повторень в межах однієї зупинки. Задано кількість генерацій популяції та розмір популяції. Налаштовано також границі зміни відносно початкового часу, щоб на виході алгоритму отримати схожий розклад, а не зовсім новий.

Спочатку відбувається ініціалізація даних з початкової популяції (початковою популяцією є зчитаний графік з файлу), а семе original і maxScoreData.

Після цього метод обраховує початкову пристосованість нащадка методом evaluateFitness розраховує загальну пристосованість для набору даних про станції. На початку метод ініціалізує змінну totalFitness зі значенням 0.0, яка зберігатиме загальну пристосованість, та змінну maxFitness зі значенням 1.0, яка визначає максимальну можливу пристосованість.

Для кожного об’єкта StationData в переданому списку даних метод evaluateFitness виконує наступне:

1. Ініціалізує змінну stationFitness зі значенням 0.0, яка буде зберігати пристосованість для поточної станції.
2. Отримує список часів прибуття для маршруту цієї станції, використовуючи метод getRouteTime об'єкта StationData.
3. Виконує перевірку на порушення обмежень:

- Перевіряє порушення обмеження SL1 за допомогою методу countConflicts, який рахує конфлікти для цієї станції. Якщо кількість конфліктів більше нуля, обмеження SL1 порушене.
- Перевіряє порушення обмеження SL2 за допомогою методу countMiniConflicts, який рахує конфлікти у часах прибуття для цього маршруту. Якщо кількість конфліктів більше нуля, обмеження SL2 порушене.

4. Розраховує пристосованість для станції:

- Якщо обидва обмеження не порушені, пристосованість встановлюється на максимальне значення maxFitness.
- Якщо порушене тільки одне з обмежень, пристосованість встановлюється на половину від максимального значення maxFitness * 0.5.

5. Додає значення пристосованості для поточної станції до загальної пристосованості totalFitness.

Після проходження по всіх об`єктах StationData метод нормалізує загальну пристосованість шляхом ділення суми на кількість станцій, щоб отримати середнє значення пристосованості в діапазоні від 0 до 1. Отримане значення повертається як результат роботи методу.

Загальна кількість балів пристосованості ділиться на кількість записів. Далі запускається цикл, який триває, поки за кілька генерацій не буде досягнуто кращої пристосованості (додано, щоб у разі проблемності створення оптимального графіку для вхідних даних не відбувалось безкінечне зациклення).

В кожному поколінні створюється нова популяція вказаного розміру шляхом випадкового вибору батьків, їх схрещування та мутації нащадків (метод mutate), які оцінюються та додаються до нової популяції, якщо вони відповідають допустимим відхиленням.

Метод mutate виконує мутацію наданої структури даних DataStructure, змінюючи години прибуття для маршрутів на станціях. Метод mutate виконує наступне:

1. Отримує список об'єктів StationData з структури даних за допомогою методу getData.
2. Ініціалізує об’єкт Random для генерації випадкових чисел.
3. Завантажує години прибуття для кожної станції і відповідного робочого дня в карту times за допомогою методу loadTimes.

Далі йде обробка кожної станції і кожного робочого дня:

1. Перебираються всі станції у карті times.
2. Для кожної станції перебираються всі робочі дні.
3. Для кожного робочого дня зберігається список часів прибуття, який сортується за допомогою Collections.sort.
4. Для кожного часу прибуття в списку виконується наступне:
   - Якщо є наступний час прибуття в списку, обчислюється інтервал часу між поточним часом і наступним у хвилинах.
   - Якщо інтервал менший за одну хвилину, випадково вирішується, який з часів змінити та відповідно або віднімається, або додається декілька хвилин до поточного часу прибуття.

Таким чином, метод проходить по всім годинам прибуття для кожної станції і робочого дня, і вносить випадкові зміни графік, якщо інтервали між ними занадто малі.

Допустимі відхилення регулюються методом isWithinAllowedDeviation, який перевіряє кожне значення з розкладу на перевищення встановлених меж і вже від результату додає створеного нащадка до поточної популяції або ж генерує нового.

Метод isWithinAllowedDeviation перевіряє, чи знаходяться зміни в годинах прибуття в допустимих межах відхилення. Він починається з отримання списків даних для початкової та нової структур даних за допомогою методу getData. Потім метод проходить по всіх станціях у списку.

Для кожної станції метод отримує списки годин прибуття для початкових даних та нових даних. Далі він перебирає всі години прибуття для поточної станції. Для кожної години прибуття обчислюється різниця у хвилинах між початковою годиною та новою годиною. Якщо ця різниця перевищує максимальне дозволене збільшення або зменшення часу, метод повертає false, що означає, що відхилення не знаходиться в допустимих межах.

Якщо жодна з перевірок не виявить перевищення допустимих меж відхилення, метод повертає true, що означає, що всі зміни в годинах прибуття знаходяться в межах дозволеного відхилення.

При мутації сортуються дані різних маршрутів відповідно до зупинки та дня курсування маршруту (будні, вихідні). Мутації та схрещування відбуваються лише для графіків, тобто назви маршрутів та зупинок хоч і фігурують у класі StationData, але не змінюються. Потім оновлюється максимальна оцінка і нова популяція замінює стару, що зроблено для економії ресурсів. Тобто кожного покоління створюється нова вибірка об’єктів DataStructure, що складається з масиву StationData (масив зупинок, що включає: назву зупинки, маршрут та години прибуття цього маршруту), у якому кожен має свої мутації у розкладі.

В кінці метод повертає нащадка з найкращою оцінкою для графіку маршрута.

3.7 Розробка модуля пакування розкладів в архів

Основна мета модулю пакування розкладів в архів – групування згенерованих звітів та оптимізованого графіку в зручний для завантаження формат. Так, як і попередні модулі, модуль пакування розкладів в архів викликається зі сторони клієнта «POST» запитом через js на url «/download», а контролер опрацьовує запит та виконує бізнес логіку. Методи з бізнес-логіки (processAndGenerateReport, createZipWithReports) представлено представлено у додатку В, модуль FileService.

Метод processAndGenerateReport обробляє дані користувача, створює звіти і повертає повідомлення про успішне завершення. Метод приймає параметр username.

Спочатку метод отримує дані користувача з пам’яті за допомогою userFileDataStorage.get(username). Якщо дані не знайдено, викидається виключення IOException з повідомленням про відсутність файлу в пам’яті для даного користувача.

Далі, метод використовує computeIfAbsent для збереження нових даних у userReportStorage для даного користувача. За допомогою методу excelProcessor.createExcelFile(data) створюється файл Excel, який зберігається під ім’ям !new_data.xlsx.

Метод об’єднує дані станцій у список stationData за допомогою combineStation(data). Потім отримує унікальні назви станцій у вигляді списку stationNames за допомогою getUniqueStationNames(stationData) і сортує їх.

Метод ініціалізує лічильники faultCounter і successCounter для відстеження успішних і невдалих операцій створення звітів. У циклі for, що проходить по всім станціям, метод викликає createDocx для кожної станції. Якщо звіт успішно створено, збільшується лічильник successCounter, інакше збільшується faultCounter. У консоль виводиться прогрес обробки у форматі «Оброблено [поточний_індекс/загальна_кількість] - успішні|невдалі».

В кінці метод повертає повідомлення «Reports generated successfully!», що вказує на успішне завершення процесу створення звітів.

Метод createZipWithReports створює ZIP-архів з звітами для вказаного користувача і повертає його у вигляді масиву байтів. Метод приймає параметр username.

Спочатку метод отримує звіти користувача з пам’яті за допомогою іншої мапи userReportStorage.get(username). Якщо звіти не знайдено або вони порожні, метод викидає виключення IOException з повідомленням про відсутність звітів для користувача.

Потім метод створює об’єкт ByteArrayOutputStream, який буде використовуватися для збереження ZIP-архіву, й викликається метод zipUtil.createZipArchive, який створює ZIP-архів з звітами користувача і записує його у ByteArrayOutputStream.

Після створення архіву метод очищає дані користувача з пам’яті, використовуючи clear для мапи звітів і файлів поточного користувача.

В кінці метод повертає масив байтів, що представляє ZIP-архів.

3.8 Розробка модуля вивантаження архіву

Модуль вивантаження архіву реалізований у контролері й ловить «POST» запит на адресу «/download». Метод викликає createZipWithReports з сервісу fileService, який створює архів із звітами для вказаного користувача.

Лістинг коду представлено у додатку В, модуль FileController.java.

Метод createZipWithReports, що обробляє HTTP-запити GET на адресі /download, створює ZIP-архів з файлами звітів для вказаного користувача і повертає його у відповіді.

Метод приймає параметр запиту userName для ідентифікації користувача. Далі викликає метод createZipWithReports сервісу fileService, який генерує ZIP-архів у вигляді масиву байтів. Потім встановлюються HTTP-заголовки для відповіді, зокрема Content-Disposition, який вказує, що відповідь повинна бути оброблена як файл для скачування з ім’ям reports.zip, та Content-Length, який задає розмір файлу.

Метод повертає відповідь ResponseEntity, що містить потік даних з ZIP-архівом, використовуючи InputStreamResource, яка створюється на основі ByteArrayInputStream з масивом байтів ZIP-архіву. Якщо під час виконання виникає виключення IOException, метод повертає відповідь зі статусом HTTP 500 (Internal Server Error).

3.9 Висновки

У третьому розділі було проведено аналіз варіантів і обрано необхідні засоби для розробки вебзастосунку системи оптимізації мережі громадського транспорту. Описано розробку модулів інтерфейсу, авторизації, завантаження файлів, парсингу даних з Excel файлу, генетичного алгоритму оптимізації графіку маршрутів, створення розкладів для зупинок у форматі Doc, пакування файлів розкладу в архів та вивантаження архіву. В результаті цього дослідження були визначені ключові складові програмного забезпечення, які дозволять оптимізувати роботу громадського транспорту та забезпечити ефективне використання ресурсів.

4 ТЕСТУВАННЯ ПРОГРАМИ

4.1 Тестування системи

Для тестування інтерфейсу користувача було використано методи візуального тестування. Кожна сторінка вебзастосунку була протестована на відповідність дизайну та коректну роботу функціоналу. Було перевірено відображення елементів інтерфейсу, їх правильне розташування та функціональні можливості, щоб забезпечити зручність та ефективність використання користувачем [17].

Перевірено відображення вебзастосунку для різних розширень екрану.

При розширенні монітору звичайного ноутбуку відображення буде наступним (рис. 4.1).

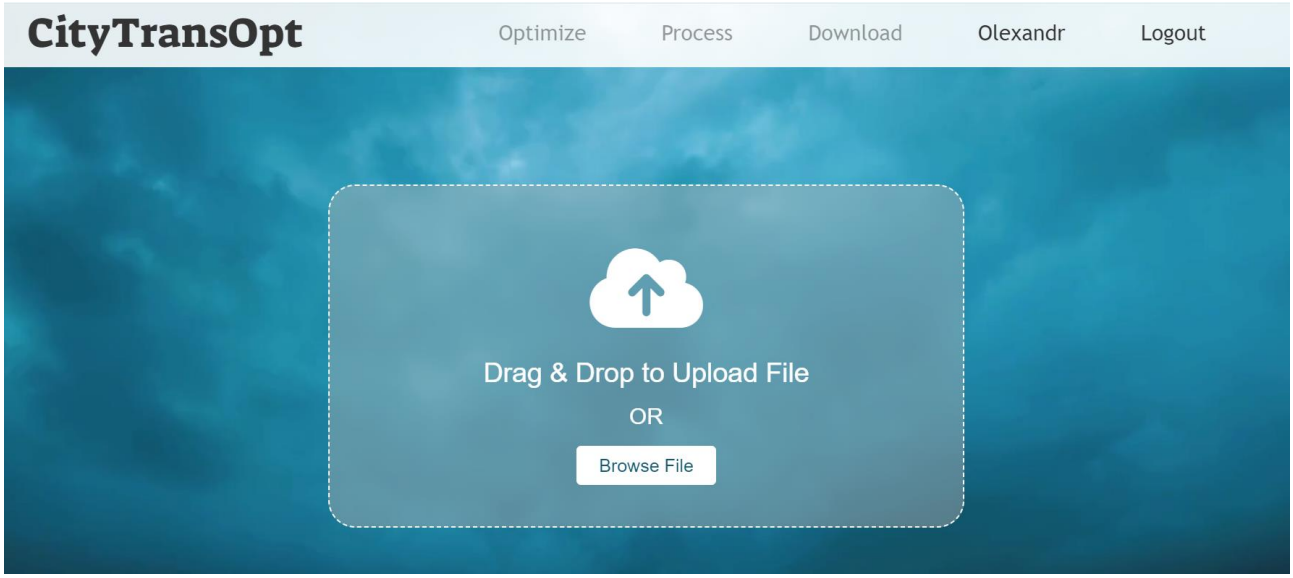

Рисунок 4.1 – Вигляд інтерфейсу при звичайному розширенні

Інтерфейс адаптується під різні розширення зменшуючи та зміщуючи елементи сторінки (рис. 4.2).

Рисунок 4.2 – Вигляд інтерфейсу при зменшеному розширенні

Для мобільної версії вебзастосунок матиме наступний вигляд (рис. 4.3).

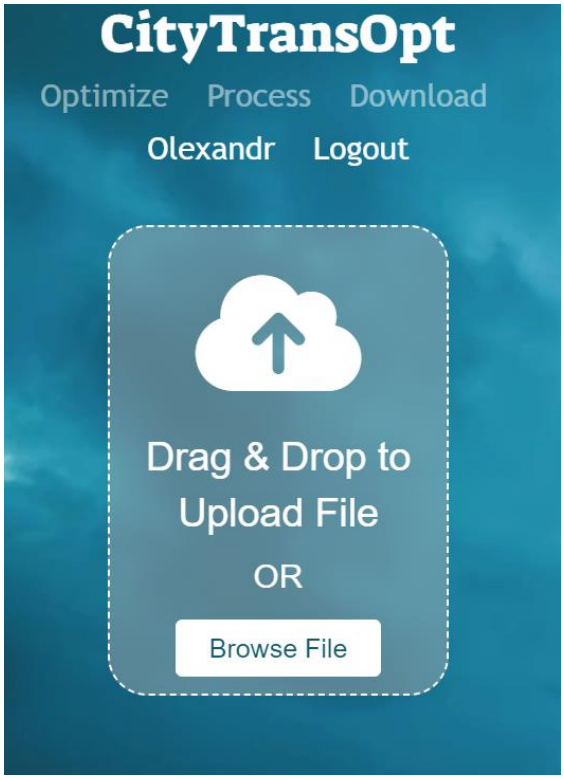

Рисунок 4.3 – Вигляд інтерфейсу на мобільному пристрої

Для тестування модулю завантаження та створення звітів було використано реальну базу транспортної мережі у Excel-файлі (194 записів зупинок) [18].

Файл був завантажений у систему для подальшого оброблення (рис. 4.4).

```
Документ Word успішно створено: src/main/resources/docs/вул.Успенська (Кріплення).docx
Документ Word успішно створено: src/main/resources/docs/вул.Успенська (МлинI).docx
Документ Word успішно створено: src/main/resources/docs/вул.Успенська (МлинII).docx
Документ Word успішно створено: src/main/resources/docs/вул.Центральна (Кладовище) с. Полянецьке.docx
Документ Word успішно створено: src/main/resources/docs/вул.Центральна (Кладовище) с. ПолянецькеI.docx
Документ Word успішно створено: src/main/resources/docs/вул.Шевченка.docx
Документ Word успішно створено: src/main/resources/docs/вул.Ярослава Мудрого.docx
Документ Word успішно створено: src/main/resources/docs/парк Черняховського.docx
Документ Word успішно створено: src/main/resources/docs/ст.Швидкої допомоги.docx
Документ Word успішно створено: src/main/resources/docs/школа №14.docx
Оброблено [194/194] - 194👍|0👎
```

Рисунок 4.4 – Результат стресового тестування

Після завантаження файлу було проведено перевірку правильності оброблення та парсингу даних, а також коректності створення звітів (рис. 4.5).

Зупинка «вул.Європейська (Класика)»

№ маршруту	Назва маршруту	Дні роботи	Розклад руху
1	Ринок-Райлікарня	будні дні	05:56, 06:49, 07:40, 08:19, 09:19, 09:55, 10:40, 11:37, 12:40, 13:19, 13:57, 15:07, 15:57, 16:56, 17:55
		субота	05:56, 07:40, 09:19, 10:40, 12:40, 14:00, 16:00, 17:55
		неділя	05:56, 06:51, 07:40, 08:21, 09:19, 09:54, 10:40, 11:41, 12:40, 13:19, 14:00, 15:04, 16:00, 17:01, 17:55
1-А	Вул.Виговського-Райлікарня	будні дні	06:52, 07:21, 07:53, 08:22, 08:53, 09:23, 09:52, 10:23, 10:53, 11:22, 11:53, 12:23, 12:52, 13:22, 13:53, 14:23, 14:52, 15:23, 15:53, 16:22, 16:53, 17:23, 17:52, 18:23, 18:55, 19:31, 20:01
		вихідні	06:56, 07:26, 07:56, 08:26, 08:56, 09:26, 09:56, 10:26, 10:56, 11:26, 11:56, 12:26, 12:56, 13:26, 13:56,

Рисунок 4.5 – Вигляд відформатованого звіту

Крім того було проаналізовано неоптимізований та оптимізований графік на кількість повторів (рис. 4.6).

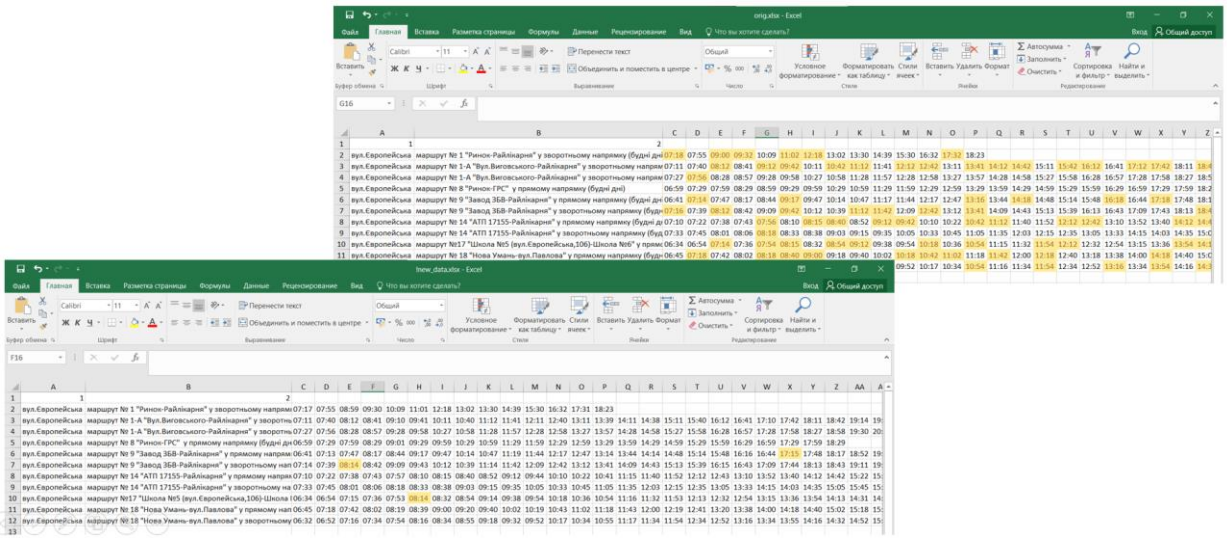

Рисунок 4.6 – Неоптимізований та оптимізований графік

Результати тестування підтвердили правильну роботу модулю та його здатність ефективно опрацьовувати великі обсяги даних.

Інформація «Profile the process» у IntelliJ IDEA надає можливість виконати профілювання додатка для аналізу його роботи та виявлення проблем з продуктивністю (рис. 4.7). Ця функція дозволяє вам визначити, які частини вашого програмного забезпечення виконуються повільно або використовують багато ресурсів [19].

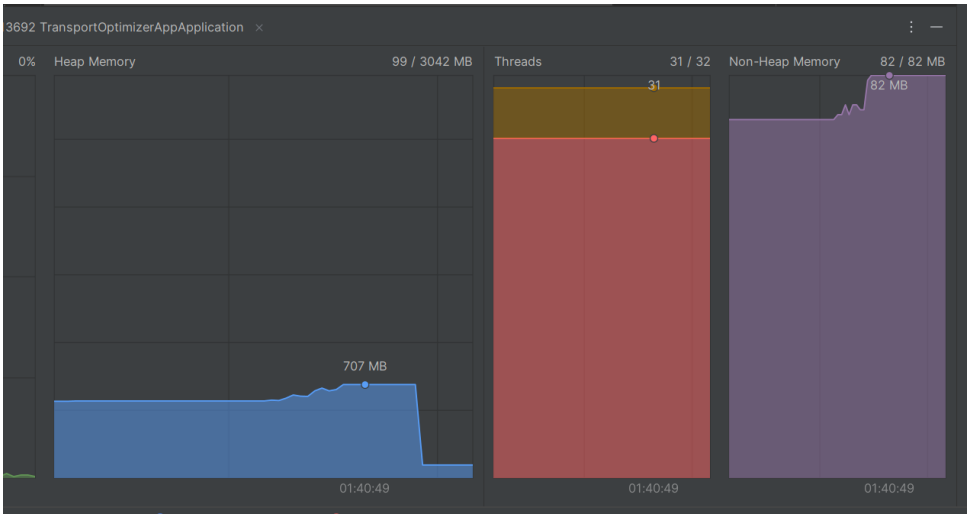

Рисунок 4.7 – Результат використання пам’яті у «Profile the process»

Частина «Packages» у профілюванні може показати, які пакети (або модулі) вашого програмного забезпечення використовуються найбільше ресурсів часу або пам'яті. Використання пам'яті варіюється на різних стадіях роботи застосунку, але для поставленої задачі є нормальним показником (для показників малих вебзастосунків). При простої у Heap Memory займає 36 Mb, при роботі алгоритму оптимізації 253-600 Mb, при процесі генерування звітів 269-500 Mb, при створенні та завантаженні архіву 283 Mb. Non-Heap Memory тримається на 77-82 Mb.

Для перевірки функціональності вебзастосунку було використано Postman, який дозволяє тестувати API-запити та аналізувати відповіді сервера для реєстрації користувача (POST на «/api/users/register») (рис. 4.8), виводу всіх користувачів (GET на «/api/users/find/**»), пошуку користувача (GET на «/api/users/all») та інших REST запитів.

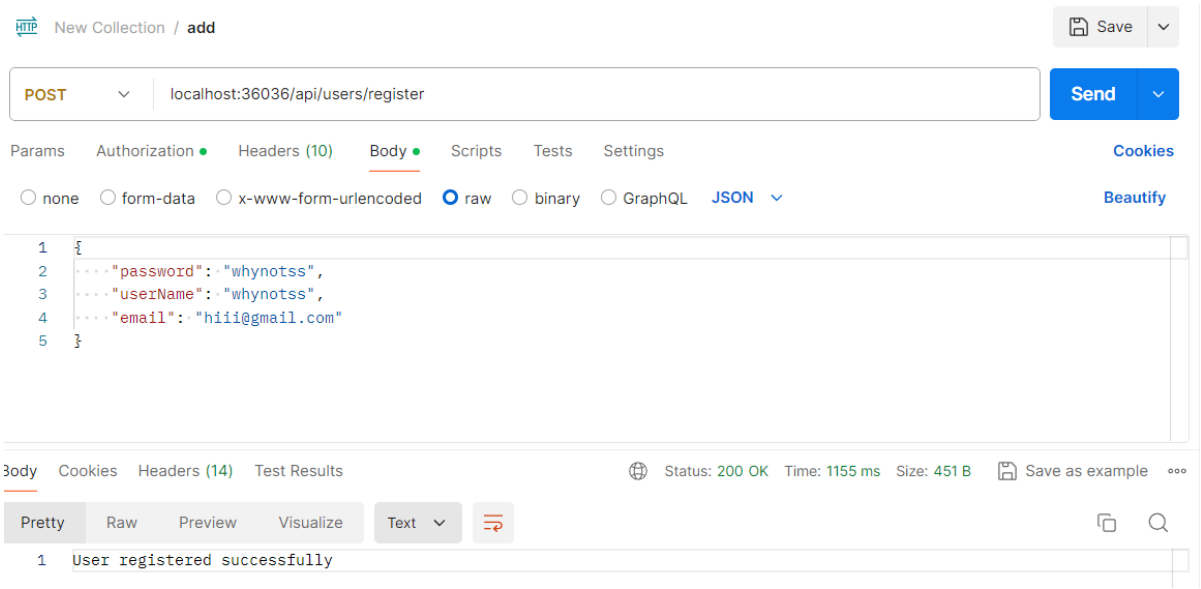

Рисунок 4.8 – Результат тестування реєстрації через Postman

Тестування API дозволяє передбачити реакцію системи на дії реальних користувачів, повторювати ті самі тести з різними наборами вхідних даних, виконувати допоміжні операції зі створення тестових даних і ситуацій, що значно прискорює та підвищує якість тестування [20].

4.2 Розробка інструкції користувача

Посібник користувача – це документ, який надає інформацію користувачам про використання певного продукту чи сервісу. Його створення передбачає аналіз аудиторії, структурування інформації таким чином, щоб вона була зрозумілою та доступною для користувачів, і пояснення ключових понять і процесів у простій формі. Посібник призначений для того, щоб допомогти користувачам ефективно використовувати продукт чи сервіс, розв'язувати проблеми та забезпечити позитивний досвід використання [21].

Структура Excel-файлу:

1. Перший рядок містить назви стовпців.
2. Перший стовпець містить назви зупинок.
3. Другий стовпець містить назви маршрутів.
4. Від третього стовпця і далі йдуть дані про графік відправок від станції для кожного маршруту. Кожен стовпець представляє час відправки для певної станції та маршруту.

Для кращого розуміння розміщення кнопок на головні сторінці вебзастосунку наведено рис. 4.9.

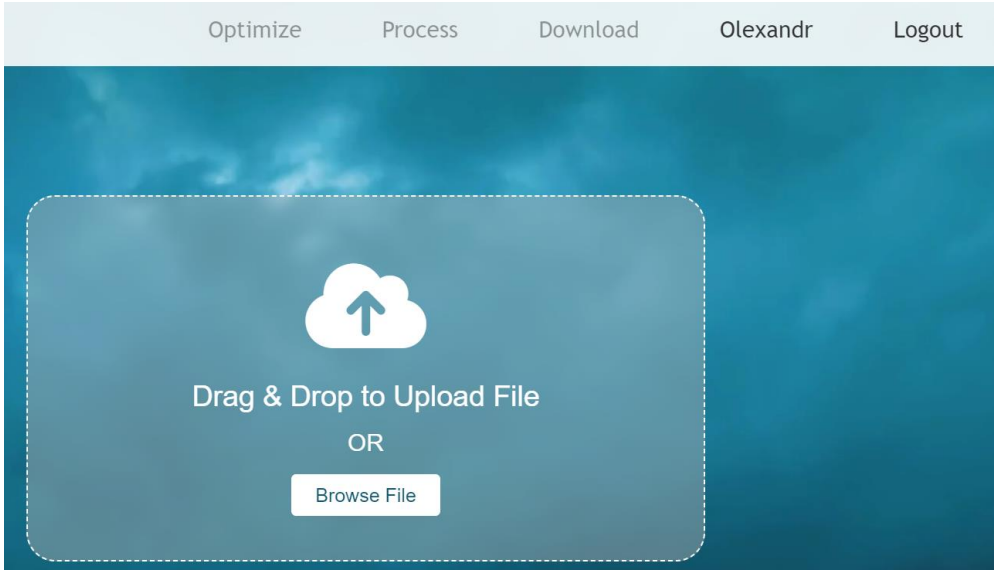

Рисунок 4.9 – Елементи головної сторінки вебзастосунку

Завантаження файлу:

1. Натисніть кнопку «Browse File» або оберіть цю опцію з меню.
2. Оберіть файл Excel зі свого комп’ютера.
3. Натисніть кнопку «Load» або «Open».

Оптимізація графіку файлу:

1. Завантажте файл.
2. Натисніть кнопку «Optimize».

Створення звітів:

1. Завантажте файл.
2. Дочекайтеся завершення «Optimize» (опціонально).
3. Натисніть кнопку «Process».

Завантаження архіву звітів та оптимізованого розкладу:

1. Завантажте файл.
2. Дочекайтеся завершення «Optimize» (опціонально).
3. Дочекайтеся завершення «Process».
4. Натисніть кнопку «Download».

Вихід з системи:

1. Натисніть кнопку «Logout».

4.3 Висновки

У четвертому розділі було проведено тестування програмних модулів вебзастосунку для оптимізації громадської мережі транспорту. Було розроблено інструкцію користувача по завантаженню файлу до системи та ознайомлено зі структурою файлу для завантаження. Протестовано адаптивність інтерфейсу до різних розширень екрану. роботу модулю з парсингу та створення звітів у документах та працездатність контролерів вебзастосунку.

ВИСНОВКИ

У бакалаврській дипломній роботі було розроблено систему оптимізації мережі громадського транспорту «CityTransOpt» для покращення автоматизації формування розкладів для зупинок та оптимізації графіку маршрутів.

Проведено аналіз стану систем оптимізації мережі громадського транспорту, розглянуто існуючі аналоги, такі як: Rozklad.in.ua, Google Maps, Zeo, Speedy Route та Optimoroute. Визначено переваги та недоліки в порівнянні з розробленою системою та встановлено перевагу у функціональності.

Відповідно до результатів аналізу та порівняння, було поставлено основні задачі бакалаврської дипломної роботи та обрано методи розробки системи. Проведено варіантний аналіз та обґрунтування вибору засобів реалізації програмного продукту. В якості мови програмування було обрано мову програмування Java. В якості середовища програмної розробки – IntelliJ IDEA Ultimate.

Відповідно до поставлених задач було:

- проведено аналіз стану систем оптимізації мережі громадського транспорту;
- запропоновано новий метод оптимізації та автоматизації мережі транспорту, систему-реалізацію;
- розроблено модель системи для оптимізації мережі громадського транспорту;
- розроблено програмні модулі запропонованої системи;
- проведено тестування розроблених алгоритмів та системи.

Крім того, було розроблено адаптивний генетичний метод, який дозволяє користувачеві оптимізувати графік руху маршрутів.

Також було розроблено інструкцію користувача.

Бакалаврську дипломну роботу було оформлено відповідно до методичних вказівок [22].

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Використання трансферних генетичних алгоритмів для оптимізації мереж громадського транспорту / О.В. Якубенко, Г.Б. Ракитянська // Матеріали LIII Всеукраїнської науково-технічної конференції факультету інформаційних технологій та комп'ютерної інженерії (2024), Вінниця, 2024. [Електронний ресурс] – Режим доступу до ресурсу: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2024/paper/view/20378/16886.

2. Безкоштовні програми для планування маршрутів [Електронний ресурс] – Режим доступу до ресурсу: https://zeorouteplanner.com/uk/free-route-planner-apps/ .

3. Dr. Ralf Borndorfer. Mathematical Optimization and Public Transportation – 2010. – 174 с.

4. Офіційний сайт проєкту Rozklad.in.ua [Електронний ресурс] – Режим доступу до ресурсу: https://rozklad.in.ua.

5. Просування на Google Maps: Ключові аспекти та оптимізація точки на карті [Електронний ресурс] – Режим доступу до ресурсу: https://it-rating.ua/news-3759.

6. Офіційний сайт проєкту Zeo [Електронний ресурс] – Режим доступу до ресурсу:https://zeorouteplanner.com/uk/планувальник-маршрутів-для-автопарків/.

7. Speedy Route official page [Електронний ресурс] – Режим доступу до ресурсу: https://www.speedyroute.com.

8. UML-діаграми [Електронний ресурс] – Режим доступу до ресурсу: https://dou.ua/forums/topic/40575/.

9. K. Siva Prasad Reddy, Sai Upadhyayula. Beginning Spring Boot 3. Build Dynamic Cloud-Native Java Applications and Microservices – 2022. – 450 с.

10. Основи Maven [Електронний ресурс] – Режим доступу до ресурсу: https://javarush.com/ua/groups/posts/uk.2523.chastina-4-osnovi-maven.

11. Загальна інформація про MongoDB [Електронний ресурс] – Режим доступу до ресурсу: https://www.ukraine.com.ua/uk/wiki/mongodb/overview/

12. Apache POI - Text Extraction [Електронний ресурс] – Режим доступу до ресурсу: https://poi.apache.org/text-extraction.html.

13. Postman – що це за інструмент і які його основні функції? [Електронний ресурс] – Режим доступу до ресурсу: https://foxminded.ua/postman/.

14. Розробка вебсайтів [Електронний ресурс] – Режим доступу до ресурсу: https://brander.ua/technologies/mvc.

15. REST API як спосіб спілкування компонент веб-додатків [Електронний ресурс] – Режим доступу до ресурсу: https://foxminded.ua/shcho-take-rest-api/.

16. Front-end розробка [Електронний ресурс] – Режим доступу до ресурсу: https://brainlab.com.ua/uk/blog-uk/front-end-rozrobka.

17. Тестування програмного забезпечення інтерфейсу користувача [Електронний ресурс] – Режим доступу до ресурсу: https://www.zaptest.com/uk/що-таке-тестування-програмного-забез.

18. Графіки руху автобусів [Електронний ресурс] – Режим доступу до ресурсу: https://uman-rada.gov.ua/index.php/ekonomika/transport/hrafiky-rukhu-avtobusiv.

19. Get Started With Java Profiling in IntelliJ IDEA [Електронний ресурс] – Режим доступу до ресурсу: https://blog.jetbrains.com/idea/2021/05/get-started-with-profiling-in-intellij-idea/.

20. Використання postman в тестуванні [Електронний ресурс] – Режим доступу до ресурсу: https://training.qatestlab.com/blog/technical-articles/use-postman-in-testing/.

21. Посібник користувача [Електронний ресурс] – Режим доступу до ресурсу: https://uk.itpedia.nl/2018/06/25/een-gebruikershandleiding-maken.

22. Методичні вказівки до виконання бакалаврських кваліфікаційних робіт для студентів студентів спеціальності 121 "Інженерія програмного забезпечення" / Уклад. О. Н. Романюк, Г. О. Черноволик – Вінниця: ВНТУ, 2022. – 52с.

ДОДАТКИ

Додаток А – Технічне завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О. Н.
«12» березня 2024 р.

Технічне завдання
на бакалаврську дипломну роботу
«Розробка методу та програмного засобу для оптимізації мережі громадського транспорту міста»
за спеціальністю 121 – Інженерія програмного забезпечення

Керівник бакалаврської дипломної роботи:
_____ к.т.н., доц. Ракитянська Г.Б.
«12» березня 2024 р.

Виконав:
_____ студент гр. ЗПІ-20б Якубенко О.В.
«12» березня 2024 р.

Вінниця – 2024

1. Найменування та галузь застосування

Бакалаврська дипломна робота: «Розробка методу та програмного засобу для оптимізації мережі громадського транспорту міста».

Галузь застосування – міська мережа транспорту.

2. Підстава для розробки.

Завдання на роботу, яке затверджене на засіданні кафедри програмного забезпечення – протокол №17 від 20 березня 2024 р.

3. Мета та призначення розробки.

Мета дипломної роботи полягає у підвищенні автоматизації формування розкладів для зупинок, спрощенні процесу моніторингу громадського транспорту та оптимізації графіку маршрутів завдяки розробленому алгоритму та його програмній реалізації, що дозволить підвищити якість мережі транспорту.

Призначення роботи – розробка програмного продукту для оптимізації мережі громадського транспорту міста.

4. Вихідні дані для проведення НДР

1. K. Siva Prasad Reddy, Sai Upadhyayula. Beginning Spring Boot 3. Build Dynamic Cloud-Native Java Applications and Microservices – 2022. – 450 с.

2. Dr. Ralf Borndorfer. Mathematical Optimization and Public Transportation – 2010. – 174 с.

3. Kishori Sharan, Adam L. Davis. Beginning Java 17 Fundamentals. Object-Oriented Programming in Java 17 – 2021. – 999.

5. Технічні вимоги

Комп’ютер з 64-розрядним процесором та тактовою частотою 2 ГГц. Об’єм оперативної пам’яті 2 ГБ, розмір жорсткого диску 128 ГБ. Операційна система Windows 7/8/10/11.

6. Конструктивні вимоги.

Додаток має відповідати ергономічним та технічним вимогам. Текстова та графічна документація повинна відповідати стандартам України.

7. Перелік технічної документації, що пред'являється по закінченню робіт:

– пояснювальна записка до бакалаврської дипломної роботи;

– технічне завдання;

– лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки

№ з/п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи
1	Аналіз розвитку систем для оптимізації мережі громадського транспорту та постановка задач дослідження	16.03.24 – 27.03.24
2	Розробка методу, моделі та алгоритмів програмного продукту	26.03.24 –19.04.24
3	Розробка програмних модулів реалізації системи з оптимізації мережі громадського транспорту	20.04.24 – 10.04.24
4	Тестування програми	11.05.24 – 16.05.24
5	Оформлення матеріалів до захисту БДР	17.05.24 – 30.05.24

10 Порядок контролю та прийняття.

Всі етапи бакалаврської роботи контролюються науковим керівником згідно плану по виконанню роботи. Прийняття бакалаврської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком захисту. Дозволяється коригування бакалаврської дипломної роботи.

Додаток Б – ПРОТОКОЛ ПЕРЕВІРКИ БАКАЛАВРСЬКОЇ ДИПЛОМНОЇ РОБОТИ НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: «Розробка методу та програмного засобу для оптимізації мережі громадського транспорту міста»

Тип роботи: БДР

Підрозділ : кафедра програмного забезпечення, ФІТКІ

Науковий керівник: Ракитянська Г.В.

Оригінальність	95,9 %
Схожість	4,1%

Аналіз звіту подібності

■ **Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.**

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.

□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку: _____ Черноволик Г. О.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck

Автор роботи: _____ Якубенко О.В.

Керівник роботи: _____ Ракитянська Г.Б.

Додаток В – Лістинг програми

Лістинг модуля SecurityConfig.java

```
package ua.parflare.transportoptimizerapp.config;
import lombok.RequiredArgsConstructor;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.security.web.SecurityFilterChain;

@Configuration
@EnableWebSecurity
@RequiredArgsConstructor
public class SecurityConfig {
   @Bean
   public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
      http
            .csrf(csrf -> csrf.disable())
            .authorizeHttpRequests((requests) -> requests
                  .requestMatchers("/login", "/register","/api/users/all", "/api/users/find/**",
"/error").permitAll()
                  .requestMatchers("/files/**", "/css/**", "/images/**", "/fonts/**", "/js/**",
"/video/**").permitAll()
                  .anyRequest().authenticated()
            )
            .formLogin((form) -> form
                  .defaultSuccessUrl("/home")
                  .permitAll()
            )
            .logout((logout) -> logout.permitAll());
      return http.build();
   }
   @Bean
   public PasswordEncoder passwordEncoder() {
      return new BCryptPasswordEncoder();
   }
}
```

Лістинг модуля AuthorizationController.java

```
package ua.parflare.transportoptimizerapp.controller;
import jakarta.validation.Valid;
import lombok.AllArgsConstructor;
import org.springframework.http.ResponseEntity;
```

```
import org.springframework.security.authentication.AuthenticationProvider;
import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.AuthenticationException;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import ua.parflare.transportoptimizerapp.entity.User;
import ua.parflare.transportoptimizerapp.service.impl.UserServiceImpl;

@Controller
@AllArgsConstructor
public class AuthorizationController {
    private final UserServiceImpl userService;
    private final AuthenticationProvider authenticationProvider;
    @GetMapping("/login")
    public String login() {
        return "login"; // Назва HTML файлу без розширення
    }
    @GetMapping("/register")
    public String register() {
        return "register"; // Назва HTML файлу без розширення
    }
    @GetMapping("/home")
    public String index() {
        return "index"; // Назва HTML файлу без розширення
    }
    @GetMapping("/logout")
    public String logout() {
        System.out.println("logout()");
        return "redirect:/login"; // перенаправляємо користувача на сторінку успішного виходу
    }
    @PostMapping("/register")
    public ResponseEntity<String> registerUser(@Valid @RequestBody User user) {
        try {
            userService.addUser(user);
            return ResponseEntity.ok("User registered successfully");
        } catch (IllegalArgumentException e) {
            return ResponseEntity.badRequest().body(e.getMessage());
        }
    }
    @PostMapping("/login")
    public ResponseEntity<String> loginUser(@Valid @RequestBody User user) {
        try {
            UsernamePasswordAuthenticationToken authToken = new
UsernamePasswordAuthenticationToken(user.getUsername(), user.getPassword());
            Authentication authentication = authenticationProvider.authenticate(authToken);
            if (authentication.isAuthenticated()) {
                return ResponseEntity.ok("User authenticated successfully");
            } else {
```

```
            return ResponseEntity.badRequest().body("Authentication failed");
        }
    } catch (AuthenticationException e) {
        return ResponseEntity.badRequest().body("Invalid username or password");
    }
  }
}
```

Лістинг модуля FileController.java

```
package ua.parflare.transportoptimizerapp.controller;
import lombok.RequiredArgsConstructor;
import org.springframework.core.io.InputStreamResource;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.multipart.MultipartFile;
import ua.parflare.transportoptimizerapp.service.FileService;
import java.io.ByteArrayInputStream;
import java.io.IOException;

@RestController
@RequiredArgsConstructor
public class FileController {
  private final FileService fileService;
  @PostMapping("/upload")
  public ResponseEntity<String> uploadFile(
      @RequestParam("userName") String userName,
      @RequestParam("file") MultipartFile file) {
    try {
      String result = fileService.saveFile(userName, file);
      return ResponseEntity.ok(result);
    } catch (IOException e) {
      return ResponseEntity
          .status(HttpStatus.INTERNAL_SERVER_ERROR)
          .body(e.getMessage());
    }
  }
  @PostMapping("/optimize")
  public ResponseEntity<String> optimizeFile(@RequestParam("userName") String userName) {
    try {
      String result = fileService.optimize(userName);
      return ResponseEntity.ok(result);
    } catch (IOException e) {
      return
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).body(e.getMessage());
```

```
        }
    }
    @PostMapping("/process")
    public ResponseEntity<String> processFile(@RequestParam("userName") String userName) {
        try {
            String result = fileService.processAndGenerateReport(userName);
            return ResponseEntity.ok(result);
        } catch (IOException e) {
            return
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).body(e.getMessage());
        }
    }
    @GetMapping("/download")
    public ResponseEntity<InputStreamResource> createZipWithReports(
            @RequestParam("userName") String userName) {
        try {
            byte[] zipData = fileService.createZipWithReports(userName);
            // Встановлення заголовків для скачування файлу
            HttpHeaders headers = new HttpHeaders();
            headers.add(HttpHeaders.CONTENT_DISPOSITION, "attachment; filename=reports.zip");
            headers.setContentLength(zipData.length); // Додано розмір файлу
            // Повертаємо потік з даними
            return ResponseEntity.ok()
                    .headers(headers)
                    .body(new InputStreamResource(new ByteArrayInputStream(zipData)));
        } catch (IOException e) {
            return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).body(null);
        }
    }
}
```

Лістинг модуля UserController.java

```
package ua.parflare.transportoptimizerapp.controller;
import jakarta.validation.Valid;
import lombok.RequiredArgsConstructor;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import ua.parflare.transportoptimizerapp.entity.User;
import ua.parflare.transportoptimizerapp.service.UserService;
import java.util.List;
import java.util.Optional;

@RestController
@RequestMapping("/api/users")
@RequiredArgsConstructor
public class UserController {
    private final UserService userService;
    @GetMapping("/all")
    public ResponseEntity<List<User>> getAllUsers() {
```

```
        List<User> users = userService.getAllUsers();
        return ResponseEntity.ok(users);
    }
    @GetMapping("/find/{name}")
    public ResponseEntity<Optional<User>> getUserByName(@PathVariable String name) {
        Optional<User> user = userService.getUserByName(name);
        return ResponseEntity.ok(user);
    }
    @PostMapping("/register")
    public ResponseEntity<String> registerUser(@Valid @RequestBody User user) {
        try {
            userService.addUser(user);
            return ResponseEntity.ok("User registered successfully");
        } catch (IllegalArgumentException e) {
            return ResponseEntity.badRequest().body(e.getMessage());
        }
    }
}
```

Лістинг модуля StationData.java

```
package ua.parflare.transportoptimizerapp.entity;
import lombok.Getter;
import lombok.Setter;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Date;
import java.util.Objects;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

@Getter
@Setter
public class StationData {
    private final String stationName;
    private final String routeGeneralInfo;
    private String routeNumber;
    private String routeName;
    private String routeWorkingDays;
    private int miniFitness;
    private ArrayList<Date> routeTime;
    public StationData(String stationName, String routeGeneralInfo, ArrayList<Date> routeTime) {
        stationName = stationName.replaceAll("\\s+", " ");
        if (stationName.endsWith(" ")) {
            stationName = stationName.substring(0, stationName.length() - " ".length());
        }
        if (stationName.startsWith(" ")) {
            stationName = stationName.replaceFirst("^\\s+", "");
        }
```

```
        if (stationName.contains("\"")) {
            int firstQuoteIndex = stationName.indexOf("\"");
            // Находим индекс второго вхождения двойных кавычек
            int secondQuoteIndex = stationName.indexOf("\"", firstQuoteIndex + 1);
            // Формируем результат, заменяя двойные кавычки на французские только в
найденных позициях
            stationName = stationName.substring(0, firstQuoteIndex) + "«" +
                    stationName.substring(firstQuoteIndex + 1, secondQuoteIndex) + "»" +
                    stationName.substring(secondQuoteIndex + 1);
            //stationName = stationName.replace("\"", "«");
            //stationName = stationName.replace("\"", "»");
        }
        this.stationName = stationName;
        this.routeGeneralInfo = routeGeneralInfo;
        this.routeTime = routeTime;
        extractGeneralInfo(routeGeneralInfo);
    }
    public void combineRouteTime(ArrayList<Date> routeTime) {
        this.routeTime.addAll(routeTime);
        Collections.sort(this.routeTime);
    }
    private void extractGeneralInfo(String info) {
        this.routeNumber = getRouteNumber(info);
        this.routeName = getRouteName(info);
        this.routeWorkingDays = getRouteWorkingDays(info);
    }
    private String getRouteNumber(String text) {
        Pattern pattern = Pattern.compile("№\\s*([^\"\\s]+)");
        Matcher matcher = pattern.matcher(text);
        if (matcher.find()) {
            return matcher.group(1);
        }
        return null;
    }
    private String getRouteName(String text) {
        Pattern pattern = Pattern.compile("\"(.*?)\"");
        Matcher matcher = pattern.matcher(text);
        if (matcher.find()) {
            String result = matcher.group(1);
            if (result.endsWith(" ")) {
                result = result.trim();
            }
            return result;
        }
        return null;
    }
    private String getRouteWorkingDays(String text) {
        Pattern pattern = Pattern.compile("\\((.*?)\\)");
        Matcher matcher = pattern.matcher(text);
        String lastMatch = null;
        while (matcher.find()) {
```

```
            lastMatch = matcher.group(1);
        }
        if (lastMatch == null || lastMatch.contains("щоденно")) {
            return "щоденно";
        }
        return lastMatch;
    }
    @Override
    public String toString() {
        final StringBuilder sb = new StringBuilder("StationData{");
        sb.append("stationName='").append(stationName).append('\'');
        sb.append(", routeInfo='").append(routeGeneralInfo).append('\'');
        sb.append(", routeTime=[");
        routeTime.forEach(date -> sb.append(new
SimpleDateFormat("HH:mm").format((date))).append(", "));
        sb.delete(sb.length() - 2, sb.length());
        sb.append("]}");
        return sb.toString();
    }
    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        StationData that = (StationData) o;
        return Objects.equals(stationName, that.stationName)
                && Objects.equals(routeNumber, that.routeNumber)
                && Objects.equals(routeName, that.routeName)
                && Objects.equals(routeWorkingDays, that.routeWorkingDays);
    }
    @Override
    public int hashCode() {
        return Objects.hash(stationName, routeNumber, routeName, routeWorkingDays);
    }
}
```

Лістинг модуля User.java

```
package ua.parflare.transportoptimizerapp.entity;
import jakarta.validation.constraints.Email;
import jakarta.validation.constraints.NotBlank;
import jakarta.validation.constraints.Size;
import lombok.Builder;
import lombok.Data;
import lombok.Generated;
import org.springframework.data.annotation.Id;
import org.springframework.data.mongodb.core.index.Indexed;
import org.springframework.data.mongodb.core.mapping.Document;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;
import ua.parflare.transportoptimizerapp.entity.enums.UserRole;
```

```
import java.util.Collection;
import java.util.Set;
import java.util.UUID;

@Data
@Builder
@Document(collection = "users")
public class User implements UserDetails {
    @Id
    @Generated
    private UUID id;
    @NotBlank(message = "Name is mandatory")
    @Size(min = 2, max = 30, message = "Name must be between 2 and 30 characters")
    @Indexed(unique = true)
    private String userName;
    @NotBlank(message = "Password is mandatory")
    @Size(min = 6, max = 30, message = "Name must be between 6 and 30 characters")
    private String password;
    @Email(message = "Email should be valid")
    private String email;
    private boolean active;
    private Set<UserRole> roles;
    @Override
    public Collection<? extends GrantedAuthority> getAuthorities() {
        return roles;
    }
    @Override
    public String getUsername() {
        return userName;
    }
    @Override
    public boolean isAccountNonExpired() {
        return true;
    }
    @Override
    public boolean isAccountNonLocked() {
        return true;
    }
    @Override
    public boolean isCredentialsNonExpired() {
        return true;
    }
    @Override
    public boolean isEnabled() {
        return active;
    }
}
```

Лістинг модуля CustomAuthenticationProvider.java

```
package ua.parflare.transportoptimizerapp.provider;
import lombok.AllArgsConstructor;
import org.springframework.security.authentication.AuthenticationProvider;
import org.springframework.security.authentication.BadCredentialsException;
import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.AuthenticationException;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.stereotype.Component;
import ua.parflare.transportoptimizerapp.service.impl.UserServiceImpl;

@Component
@AllArgsConstructor
public class CustomAuthenticationProvider implements AuthenticationProvider {
   private final UserServiceImpl userDetailsService;
   private final PasswordEncoder passwordEncoder;
   @Override
   public Authentication authenticate(Authentication authentication) throws
AuthenticationException {
      String username = authentication.getName();
      String password = authentication.getCredentials().toString();
      UserDetails userDetails = userDetailsService.loadUserByUsername(username);
      if (!passwordEncoder.matches(password, userDetails.getPassword())) {
         throw new BadCredentialsException("Incorrect username or password");
      }
      return new UsernamePasswordAuthenticationToken(userDetails, password,
userDetails.getAuthorities());
   }
   @Override
   public boolean supports(Class<?> authentication) {
      return authentication.equals(UsernamePasswordAuthenticationToken.class);
   }
}
```

Лістинг модуля FileService.java

```
package ua.parflare.transportoptimizerapp.service;
import lombok.RequiredArgsConstructor;
import org.apache.poi.xwpf.usermodel.XWPFDocument;
import org.springframework.stereotype.Service;
import org.springframework.web.multipart.MultipartFile;
import ua.parflare.transportoptimizerapp.entity.StationData;
import ua.parflare.transportoptimizerapp.util.DocxGenerator;
import ua.parflare.transportoptimizerapp.util.ExcelProcessor;
import ua.parflare.transportoptimizerapp.util.TransportOptimizerUtil;
import ua.parflare.transportoptimizerapp.util.ZipUtil;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.util.ArrayList;
```

```
import java.util.Collections;
import java.util.HashSet;
import java.util.Map;
import java.util.concurrent.ConcurrentHashMap;

@Service
@RequiredArgsConstructor
public class FileService {
   private final ExcelProcessor excelProcessor;
   private final DocxGenerator docxGenerator;
   private final ZipUtil zipUtil;
   private final TransportOptimizerUtil optimizerUtil;
   // Map для збереження файлів у пам'яті по імені користувача
   private final Map<String, ArrayList<StationData>> userFileDataStorage = new
ConcurrentHashMap<>();
   private final Map<String, Map<String, byte[]>> userReportStorage = new
ConcurrentHashMap<>();
   public String saveFile(String username, MultipartFile file) throws IOException {
      String fileName = file.getOriginalFilename();
      assert fileName != null;
      if (!isExcelFile(fileName)) {
         throw new IOException(fileName + " has invalid file format (requires Excel)");
      }
      byte[] fileBytes = file.getBytes(); // Зчитуємо байти з MultipartFile
      ArrayList<StationData> data = excelProcessor.processExcelFile(fileBytes);
      userFileDataStorage.put(username, data);  // Зберігаємо байти у мапі
      return "File " + fileName + " uploaded successfully";
   }
   public String optimize(String username) throws IOException {
      var data = userFileDataStorage.get(username);
      if (data == null) {
         throw new IOException("File not found in memory for user " + username);
      }
      ArrayList<StationData> optimizedData = optimizerUtil.optimizeSchedule(data);
      userFileDataStorage.put(username, optimizedData);
      return "File optimized successfully";
   }
   public String processAndGenerateReport(String username) throws IOException {
      var data = userFileDataStorage.get(username);
      if (data == null) {
         throw new IOException("File not found in memory for user " + username);
      }
      userReportStorage
            .computeIfAbsent(username, k -> new ConcurrentHashMap<>())
            .put("!new_data.xlsx", excelProcessor.createExcelFile(data));
      ArrayList<StationData> stationData = combineStation(data);
      ArrayList<String> stationNames = getUniqueStationNames(stationData);
      Collections.sort(stationNames);
      int totalIteration = stationNames.size();
      System.out.println();
      int faultCounter = 0, successCounter = 0;
```

```
        for (int i = 0; i < totalIteration; i++) {
            if (createDocx(stationNames.get(i), stationData, username)) {
                successCounter++;
            } else {
                faultCounter++;
            }
            System.out.printf("\rОброблено [%d/%d] - %d\uD83D\uDC4D|%d\uD83D\uDC4E",
                    i + 1, totalIteration, successCounter, faultCounter);
        }
        return "Reports generated successfully!";
    }
    private boolean createDocx(String search,
                               ArrayList<StationData> data,
                               String username) {
        String fileName = search + ".docx";
        try (XWPFDocument document = new XWPFDocument()) {
            docxGenerator.setDocumentProperties(document);
            docxGenerator.createText(document, String.format("Зупинка «%s»", search));
            docxGenerator.createTable(document, getStationDataByName(search, data));
            try (ByteArrayOutputStream out = new ByteArrayOutputStream()) {
                document.write(out);
                // Отримуємо байти з ByteArrayOutputStream
                byte[] docxBytes = out.toByteArray();
                // Зберігаємо байти документа у ConcurrentHashMap
                userReportStorage
                        .computeIfAbsent(username, k -> new ConcurrentHashMap<>())
                        .put(fileName, docxBytes);
            }
            return true;
        } catch (Exception e) {
            System.err.println("\rПомилка у " + fileName);
            return false;
        }
    }
    public ArrayList<StationData> getStationDataByName(String name, ArrayList<StationData>
stationData) {
        ArrayList<StationData> result = new ArrayList<>();
        for (StationData data : stationData) {
            if (data.getStationName().equals(name)) {
                result.add(data);
            }
        }
        //System.out.println(result);
        return result;
    }
    public ArrayList<StationData> combineStation(ArrayList<StationData> stationData) {
        ArrayList<StationData> uniqueStationData = new ArrayList<>();
        for (StationData data : stationData) {
            if (uniqueStationData.contains(data)) {
                int index = uniqueStationData.indexOf(data);
                uniqueStationData.get(index).combineRouteTime(data.getRouteTime());
```

```
            } else {
                uniqueStationData.add(data);
            }
        }
        return uniqueStationData;
    }
    private ArrayList<String> getUniqueStationNames(ArrayList<StationData> stationData) {
        HashSet<String> stationNames = new HashSet<>();
        for (StationData data : stationData) {
            stationNames.add(data.getStationName());
        }
        return new ArrayList<>(stationNames);
    }
    public byte[] createZipWithReports(String username) throws IOException {
        Map<String, byte[]> userReports = userReportStorage.get(username);
        if (userReports == null || userReports.isEmpty()) {
            throw new IOException("No reports found for user " + username);
        }
        try (ByteArrayOutputStream baos = new ByteArrayOutputStream()) {
            zipUtil.createZipArchive(userReports, baos);
            // Отримання мапи для користувача і видалення її з пам'яті
            userReportStorage.get(username).clear();
            userFileDataStorage.get(username).clear();
            return baos.toByteArray();
        }
    }
    private boolean isExcelFile(String fileName) {
        String extension = fileName.substring(fileName.lastIndexOf("."));
        return extension.equalsIgnoreCase(".xls") || extension.equalsIgnoreCase(".xlsx");
    }
}
```

Лістинг модуля DocxGenerator.java

```
package ua.parflare.transportoptimizerapp.util;
import org.apache.poi.xwpf.usermodel.*;
import org.openxmlformats.schemas.wordprocessingml.x2006.main.*;
import org.springframework.stereotype.Component;
import ua.parflare.transportoptimizerapp.entity.StationData;
import java.math.BigInteger;
import java.text.SimpleDateFormat;
import java.util.*;

@Component
public class DocxGenerator {
    private static String convertDataListToString(StationData data) {
        StringBuilder sb = new StringBuilder();
        for (Date date : data.getRouteTime()) {
            sb.append(new SimpleDateFormat("HH:mm").format((date))).append(", ");
        }
```

```
    sb.delete(sb.length() - 2, sb.length());
    return sb.toString();
}
public void setDocumentProperties(XWPFDocument document) {
    CTDocument1 doc = document.getDocument();
    CTBody body = doc.getBody();
    if (!body.isSetSectPr()) {
        body.addNewSectPr();
    }
    CTSectPr section = body.getSectPr();
    if (!section.isSetPgSz()) {
        section.addNewPgSz();
    }
    CTPageSz pageSize = section.getPgSz();
    pageSize.setW(BigInteger.valueOf(11906)); // 11906 твіпів = 21 см = A4
    pageSize.setH(BigInteger.valueOf(16838)); // 16838 твіпів = 29.7 см = A4
}
public void createText(XWPFDocument document, String text) {
    XWPFParagraph para = document.createParagraph();
    para.setAlignment(ParagraphAlignment.CENTER); // Вирівнювання посередині
    para.setSpacingBeforeLines(0); // Інтервал перед абзацем
    para.setSpacingAfter(0); // Інтервал після абзацу
    para.setSpacingBetween(1.5); // Міжстроковий інтервал
    XWPFRun run = para.createRun();
    run.setText(text);
    run.addBreak();
    run.setFontFamily("Times New Roman");
    run.setFontSize(14);
}
public void createTable(XWPFDocument document, ArrayList<StationData> stationData) {
    XWPFTable table = document.createTable(stationData.size() + 1, 4);
    table.setTableAlignment(TableRowAlign.CENTER);
    XWPFTableRow firstRow = table.getRow(0);
    firstRow.getCell(0).setText("№ маршруту");
    firstRow.getCell(1).setText("Назва маршруту");
    firstRow.getCell(2).setText("Дні роботи");
    firstRow.getCell(3).setText("Розклад руху");
    for (int j = 0; j < stationData.size(); j++) {
        StationData data = stationData.get(j);
        XWPFTableRow secondRow = table.getRow(j + 1);
        secondRow.getCell(0).setText(data.getRouteNumber());
        secondRow.getCell(1).setText(data.getRouteName());
        secondRow.getCell(2).setText(data.getRouteWorkingDays());
        secondRow.getCell(3).setText(convertDataListToString(data));
    }
    mergeCellsInColumns(table);
    table.getRows().forEach(row -> row.getTableCells().forEach(cell -> {
        cell.getParagraphs().forEach(paragraph -> {
            paragraph.setAlignment(ParagraphAlignment.CENTER);
            paragraph.setSpacingBetween(1);
            paragraph.setSpacingAfter(0);
```

```
            paragraph.setSpacingBefore(0);
            paragraph.getRuns().forEach(run -> {
                run.setFontFamily("Times New Roman");
                run.setFontSize(14);
            });
        });
        cell.setWidthType(TableWidthType.AUTO);
        cell.setVerticalAlignment(XWPFTableCell.XWPFVertAlign.CENTER);
    }));
  }
  public void mergeCellsInColumns(XWPFTable table) {
    mergeCellsInColumn(table, 0);
    mergeCellsInColumn(table, 1);
  }
  private void mergeCellsInColumn(XWPFTable table, int columnIndex) {
    Map<String, List<XWPFTableCell>> cellMap = new HashMap<>();
    // Заповнюємо мапу, де ключ - текст комірки першого стовпчика, значення - список
комірок з таким текстом
    for (int i = 0; i < table.getNumberOfRows(); i++) {
      XWPFTableRow row = table.getRow(i);
      XWPFTableCell cell = row.getCell(columnIndex);
      String cellText = cell.getText();
      if (!cellMap.containsKey(cellText)) {
        cellMap.put(cellText, new ArrayList<>());
      }
      cellMap.get(cellText).add(cell);
    }
    // Об'єднуємо комірки з однаковим текстом в першому стовпчику
    for (Map.Entry<String, List<XWPFTableCell>> entry : cellMap.entrySet()) {
      List<XWPFTableCell> cells = entry.getValue();
      if (cells.size() > 1) { // Якщо є більше однієї комірки з таким текстом, об'єднуємо їх
        XWPFTableCell firstCell = cells.get(0);
        for (int i = 1; i < cells.size(); i++) {
          XWPFTableCell cell = cells.get(i);
          // Об'єднання комірок
          firstCell.getCTTc().addNewTcPr().addNewVMerge().setVal(STMerge.RESTART);
          cell.getCTTc().addNewTcPr().addNewVMerge().setVal(STMerge.CONTINUE);
        }
      }
    }
  }
}
```

Лістинг модуля ExcelProcessor.java

```
package ua.parflare.transportoptimizerapp.util;
import org.apache.poi.ss.usermodel.*;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;
import org.springframework.stereotype.Component;
```

```
import ua.parflare.transportoptimizerapp.entity.StationData;
import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;

@Component
public class ExcelProcessor {
   public ArrayList<StationData> processExcelFile(byte[] fileBytes) throws IOException {
      ArrayList<StationData> data;
      try (Workbook readWorkbook = WorkbookFactory.create(new
ByteArrayInputStream(fileBytes))) {
         Sheet readSheet = readWorkbook.getSheet("Combined Data");
         data = new ArrayList<>(readSheet.getLastRowNum());
         for (int j = 1; j <= readSheet.getLastRowNum(); j++) {
            Row row = readSheet.getRow(j);
            String stationName = row.getCell(0).getStringCellValue();
            String routeInfo = row.getCell(1).getStringCellValue();
            ArrayList<Date> dates = new ArrayList<>(row.getLastCellNum() - 2);
            for (int i = 2; i < row.getLastCellNum(); i++) {
               Cell cell = row.getCell(i);
               if (cell != null
                  && cell.getCellType().equals(CellType.NUMERIC)
                  && DateUtil.isCellDateFormatted(cell)) {
                  Date time = cell.getDateCellValue();
                  dates.add(time);
               }
            }
            data.add(new StationData(stationName, routeInfo, dates));
         }
      } catch (IOException e) {
         throw new IOException("Помилка при зчитуванні Excel файлу: " + e.getMessage(), e);
      }
      return data;
   }
   public byte[] createExcelFile(ArrayList<StationData> stationDataList) throws IOException {
      try (Workbook writeWorkbook = new XSSFWorkbook()) {
         Sheet writeSheet = writeWorkbook.createSheet("Combined Data");
         for (int i = 0; i < stationDataList.size(); i++) {
            Row row = writeSheet.createRow(i + 1);
            StationData stationData = stationDataList.get(i);
            row.createCell(0).setCellValue(stationData.getStationName());
            row.createCell(1).setCellValue(stationData.getRouteGeneralInfo());
            ArrayList<Date> dates = stationData.getRouteTime();
            for (int j = 0; j < dates.size(); j++) {
               Cell cell = row.createCell(j + 2);
               cell.setCellValue(new SimpleDateFormat("HH:mm").format((dates.get(j))));
               CellStyle cellStyle = writeWorkbook.createCellStyle();
               CreationHelper createHelper = writeWorkbook.getCreationHelper();
```

```
                cellStyle.setDataFormat(createHelper.createDataFormat().getFormat("HH:mm"));
                cell.setCellStyle(cellStyle);
            }
        }
        ByteArrayOutputStream outputStream = new ByteArrayOutputStream();
        writeWorkbook.write(outputStream);
        return outputStream.toByteArray();
    }
  }
}
```

Лістинг модуля TransportOptimizerUtil.java

```
package ua.parflare.transportoptimizerapp.util;

import lombok.Getter;
import lombok.Setter;
import org.springframework.stereotype.Component;
import ua.parflare.transportoptimizerapp.entity.StationData;

import java.util.*;

@Component
public class TransportOptimizerUtil {

   private static final double MUTATION_RATE = 0.08;
   private static final int MAX_INCREMENT_MINUTES = 5;
   private static final int MAX_DECREMENT_MINUTES = 5;
   private static final int POPULATION_SIZE = 15;
   public static final int ATTEMPTS_WITHOUT_CHANGES = 10;
   private DataStructure original;
   private ArrayList<DataStructure> population = new ArrayList<>();
   private DataStructure maxScoreData;

   public TransportOptimizerUtil() {
      original = null;
      maxScoreData = null;
   }

   private static ArrayList<Date> getDateArrayList(ArrayList<Date> timeList) {
      ArrayList<Date> filteredTimes = new ArrayList<>();

      for (int i = 0; i < timeList.size(); i++) {
         Date currentTime = timeList.get(i);
         if (i + 1 < timeList.size()) {
            Date nextTime = timeList.get(i + 1);
            long interval = (nextTime.getTime() - currentTime.getTime()) / (60 * 1000); // Interval in
minutes
            if (interval < 1) {
```

```
                if (!filteredTimes.contains(currentTime)) {
                    filteredTimes.add(currentTime);
                }
                filteredTimes.add(nextTime);
            }
        }
    }
    return filteredTimes;
}

public void initializeStationData(ArrayList<StationData> initialPopulation) {
    original = new DataStructure(initialPopulation);
    maxScoreData = new DataStructure(original);
    population.add(new DataStructure(original.getData()));
}
public ArrayList<StationData> optimizeSchedule(ArrayList<StationData> initialPopulation) {
    initializeStationData(initialPopulation);

    original = new DataStructure(initialPopulation);
    maxScoreData = new DataStructure(original);
    population.add(new DataStructure(original.getData()));
    System.out.println();
    int currentGeneration = 0;
    int attempt = 0;

    double origFitness = evaluateFitness(original);
    ArrayList<DataStructure> newPopulation;
    double tmpFitness = origFitness;
    int generationsWithoutImprovement = 0; // Додано лічильник поколінь без покращення
    while (generationsWithoutImprovement < ATTEMPTS_WITHOUT_CHANGES) { //
Перероблено умову циклу
        newPopulation = new ArrayList<>(POPULATION_SIZE);
        while (newPopulation.size() < POPULATION_SIZE) {
            attempt++;
            DataStructure parent1 = selectParent();
            DataStructure parent2 = selectParent();
            DataStructure offspring = crossover(parent1, parent2);
            if (Math.random() < MUTATION_RATE) {
                mutate(offspring);
            }
            if (isWithinAllowedDeviation(offspring)) {
                offspring.reevaData();
                newPopulation.add(offspring);
            }
            tmpFitness = offspring.getFitness();
            System.out.printf("\rcurrentFitness: %.03f, change: %.03f->%.03f, iter: %d, popul: %d,
attempt: %d", tmpFitness, origFitness, maxScoreData.getFitness(), currentGeneration,
newPopulation.size(), attempt);
        }
        attempt = 0;
        boolean isChangedMAxValue = updateMaxScore(newPopulation);
```

```
        System.out.printf("\rcurrentFitness: %.03f, change: %.03f->%.03f, iter: %d", tmpFitness,
origFitness, maxScoreData.getFitness(), ++currentGeneration);

        if (isChangedMAxValue) { // Перевірка на покращення результату
            generationsWithoutImprovement = 0; // Скидаємо лічильник
        } else {
            generationsWithoutImprovement++; // Збільшуємо лічильник
        }

        population = new ArrayList<>(newPopulation);
    }
    population.clear();
    return maxScoreData.getData();
  }


  private boolean isWithinAllowedDeviation(DataStructure offspring) {
    ArrayList<StationData> originalData = original.getData();
    ArrayList<StationData> offspringData = offspring.getData();
    for (int i = 0; i < originalData.size(); i++) {
      ArrayList<Date> originalTimes = originalData.get(i).getRouteTime();
      ArrayList<Date> offspringTimes = offspringData.get(i).getRouteTime();
      for (int j = 0; j < originalTimes.size(); j++) {
        long originalTime = originalTimes.get(j).getTime();
        long offspringTime = offspringTimes.get(j).getTime();
        long timeDifference = (offspringTime - originalTime) / (60 * 1000); // Difference in
minutes
        if (timeDifference > MAX_INCREMENT_MINUTES || timeDifference < -
MAX_DECREMENT_MINUTES) {
          return false;
        }
      }
    }
    return true;
  }

  public boolean updateMaxScore(ArrayList<DataStructure> dataStructure) {
    boolean res = false;
    for (DataStructure data : dataStructure) {
      //data.reevaData();
      if (data.getFitness() > maxScoreData.getFitness()) {
        maxScoreData = new DataStructure(data);
        res = true;
      }
    }
    return res;
  }

  private double evaluateFitness(DataStructure data) {
    double totalFitness = 0.0;
    double maxFitness = 1.0;
```

```
        for (StationData stationData : data.getData()) {
            double stationFitness = 0.0;
            ArrayList<Date> routeTimes = stationData.getRouteTime();
            // Перевірка обмежень SL1 та SL2 для кожної зупинки
            boolean sl1Violated = countConflicts(data.getData(), stationData.getStationName()) > 0;
            boolean sl2Violated = countMiniConflicts(routeTimes) > 0;
            // Розрахунок пристосованості для кожної зупинки з врахуванням обмежень
            if (!sl1Violated && !sl2Violated) {
                stationFitness = maxFitness;
            } else if (!sl1Violated || !sl2Violated) {
                stationFitness = maxFitness * 0.5; // Половина максимального балу, якщо порушено
лише одне обмеження
            }
            totalFitness += stationFitness;
        }
        // Нормалізація загальної пристосованості до діапазону [0, 1]
        return totalFitness / data.getData().size();
    }

    private double evaluateFitness(ArrayList<StationData> data) {
        double totalFitness = 0.0;
        double maxFitness = 1.0;

        for (StationData stationData : data) {
            double stationFitness = 0.0;
            ArrayList<Date> routeTimes = stationData.getRouteTime();
            // Перевірка обмежень SL1 та SL2 для кожної зупинки
            boolean sl1Violated = countConflicts(data, stationData.getStationName()) > 0;
            boolean sl2Violated = countMiniConflicts(routeTimes) > 0;
            // Розрахунок пристосованості для кожної зупинки з врахуванням обмежень
            if (!sl1Violated && !sl2Violated) {
                stationFitness = maxFitness;
            } else if (!sl1Violated || !sl2Violated) {
                stationFitness = maxFitness * 0.5; // Половина максимального балу, якщо порушено
лише одне обмеження
            }
            totalFitness += stationFitness;
        }
        // Нормалізація загальної пристосованості до діапазону [0, 1]
        return totalFitness / data.size();
    }


    private Map<String, Map<String, ArrayList<Date>>> loadTimes(ArrayList<StationData>
stationDataList) {
        Map<String, Map<String, ArrayList<Date>>> times = new HashMap<>();
        // Collect times for each station and working days
        for (StationData data : stationDataList) {
            times.computeIfAbsent(data.getStationName(), k -> new HashMap<>())
                    .computeIfAbsent(data.getRouteWorkingDays(), k -> new ArrayList<>())
```

```
                .addAll(data.getRouteTime());
        }
        return times;
    }

    private int countConflicts(ArrayList<StationData> stationDataList, String stationName) {
        int conflicts = 0;
        Map<String, Map<String, ArrayList<Date>>> times = loadTimes(stationDataList);
        Map<String, Set<String>> routesForStations = new HashMap<>();

        // Збираємо маршрути для кожної станції
        for (StationData data : stationDataList) {
            routesForStations.computeIfAbsent(data.getStationName(), k -> new HashSet<>())
                    .add(data.getRouteGeneralInfo());
        }

        // Перевіряємо тільки вказану станцію
        Map<String, ArrayList<Date>> stationTimes = times.get(stationName);
        if (stationTimes != null) {
            for (Map.Entry<String, ArrayList<Date>> daysEntry : stationTimes.entrySet()) {
                ArrayList<Date> timeList = daysEntry.getValue();
                Collections.sort(timeList);
                ArrayList<Date> filteredTimes = getDateArrayList(timeList);
                conflicts += filteredTimes.size();
            }
        }

        return conflicts;
    }

    private int countMiniConflicts(ArrayList<Date> times) {
        int conflicts = 0;

        for (int i = 0; i < times.size(); i++) {
            for (int j = i + 1; j < times.size(); j++) {
                long diff = Math.abs(times.get(i).getTime() - times.get(j).getTime());
                if (diff <= 60 * 1000) { // Якщо різниця в часі менше 1 хвилини
                    conflicts++;
                }
            }
        }
        return conflicts;
    }

    private DataStructure selectParent() {
        Random rand = new Random();
        DataStructure parent1 = population.get(rand.nextInt(population.size()));
        DataStructure parent2 = population.get(rand.nextInt(population.size()));
        return parent1.getFitness() > parent2.getFitness() ? parent1 : parent2;
    }
```

```
    private DataStructure crossover(DataStructure parent1, DataStructure parent2) {
        ArrayList<StationData> newRouteData = new ArrayList<>(parent1.getData().size());
        Random rand = new Random();
        for (int i = 0; i < parent1.getData().size(); i++) {
            StationData tmpStationData = parent1.getData().get(i);
            ArrayList<Date> newDates = new ArrayList<>(tmpStationData.getRouteTime().size());
            for (int j = 0; j < tmpStationData.getRouteTime().size(); j++) {
                Date tmpRouteTime = tmpStationData.getRouteTime().get(j);
                if (rand.nextBoolean()) {
                    newDates.add(tmpRouteTime);
                } else {
                    newDates.add(parent2.getData().get(i).getRouteTime().get(j));
                }
            }
            newRouteData.add(new StationData(tmpStationData.getStationName(),
tmpStationData.getRouteGeneralInfo(), newDates));
        }
        return new DataStructure(newRouteData);
    }

    private void mutate(DataStructure dataStructure) {
        ArrayList<StationData> stationDataList = dataStructure.getData();
        Random rand = new Random();
        Map<String, Map<String, ArrayList<Date>>> times = loadTimes(stationDataList);
        // Sort and print times for each station and working days
        for (Map.Entry<String, Map<String, ArrayList<Date>>> stationEntry : times.entrySet()) {
            for (Map.Entry<String, ArrayList<Date>> daysEntry : stationEntry.getValue().entrySet()) {
                ArrayList<Date> timeList = daysEntry.getValue();
                Collections.sort(timeList);
                for (int i = 0; i < timeList.size(); i++) {
                    Date currentTime = timeList.get(i);
                    if (i + 1 < timeList.size()) {
                        Date nextTime = timeList.get(i + 1);
                        long interval = (nextTime.getTime() - currentTime.getTime()) / (60 * 1000); //
Interval in minutes
                        if (interval < 1) {
                            if (rand.nextBoolean()) {
                                nextTime.setTime(
                                        nextTime.getTime()
                                        + (rand.nextInt(MAX_INCREMENT_MINUTES / 2) + 1) * 60 * 1000);
                            } else {
                                currentTime.setTime(
                                        currentTime.getTime()
                                        - (rand.nextInt(MAX_DECREMENT_MINUTES / 2) + 1) * 60 * 1000);
                            }
                        }
                    }
                }
            }
        }
    }
```

```
    @Getter
    @Setter
    private class DataStructure {
        private double fitness;
        private ArrayList<StationData> data;

        public DataStructure(ArrayList<StationData> data) {
            this.data = deepCopy(data);
            this.fitness = evaluateFitness(this.getData());
        }

        public DataStructure(DataStructure clone) {
            this.data = deepCopy(clone.getData());
            this.fitness = clone.getFitness();
        }

        public void reevaData() {
            this.fitness = evaluateFitness(this.getData());
        }

        private ArrayList<StationData> deepCopy(ArrayList<StationData> original) {
            ArrayList<StationData> copy = new ArrayList<>();
            for (StationData stationData : original) {
                ArrayList<Date> newRouteTimes = new ArrayList<>();
                for (Date date : stationData.getRouteTime()) {
                    newRouteTimes.add(new Date(date.getTime()));
                }
                copy.add(new StationData(stationData.getStationName(),
stationData.getRouteGeneralInfo(), newRouteTimes));
            }
            return copy;
        }
    }
}
```

Лістинг модуля ZipUtil.java

```
package ua.parflare.transportoptimizerapp.util;
import org.springframework.stereotype.Component;
import java.io.ByteArrayInputStream;
import java.io.IOException;
import java.io.OutputStream;
import java.util.Map;
import java.util.zip.ZipEntry;
import java.util.zip.ZipOutputStream;

@Component
public class ZipUtil {
    public void createZipArchive(Map<String, byte[]> files, OutputStream out) throws IOException
{
```

```
        int totalFiles = files.size();
        int processedFiles = 0;
        System.out.println();
        try (ZipOutputStream zos = new ZipOutputStream(out)) {
            for (Map.Entry<String, byte[]> entry : files.entrySet()) {
                processedFiles++;
                ZipEntry zipEntry = new ZipEntry(entry.getKey());
                zos.putNextEntry(zipEntry);
                try (ByteArrayInputStream bis = new ByteArrayInputStream(entry.getValue())) {
                    byte[] buffer = new byte[1024];
                    int len;
                    while ((len = bis.read(buffer)) > 0) {
                        zos.write(buffer, 0, len);
                    }
                }
                // Виводимо прогрес у консоль
                System.out.printf("\rProcessed [%d/%d] files",
                        processedFiles, totalFiles);
            }
        }
    }
}
```

Лістинг модуля scripts.js

```
function slowScroll(id) {
    $('html, body').animate({
        scrollTop: $(id).offset().top - 100
    }, 500);
}
$(document).on("scroll", function () {
    if ($(window).scrollTop() === 0)
        $("header").removeClass("fixed");
    else
        $("header").attr("class", "fixed");
});
var username = null;
$(document).ready(function () {
    document.querySelectorAll('.menu a').forEach(link => {
        link.addEventListener('dragstart', (event) => {
            event.preventDefault();
        });
    });
//selecting all required elements
    const dropArea = document.querySelector(".drag-area"),
        iconElement = document.querySelector('.icon i');
    let dragText = dropArea.querySelector("p"),
        dragSpan = dropArea.querySelector("span"),
        button = dropArea.querySelector("button"),
        input = dropArea.querySelector("input");
```

```
username = document.getElementById('username').innerText.trim()
var file; //this is a global variable and we'll use it inside multiple functions
button.onclick = () => {
    input.click(); //if user click on the button then the input also clicked
}
input.addEventListener("change", function () {
    dropArea.classList.remove("failed");
    dropArea.classList.add("active");
});
dropArea.addEventListener("dragover", (event) => {
    event.preventDefault(); //preventing from default behaviour
    changeClasses(iconElement, 'fa-solid', 'fa-cloud-upload');
    dropArea.classList.remove("failed");
    dropArea.classList.add("active");
    dragText.textContent = "Release to Upload File";
});
dropArea.addEventListener("dragleave", () => {
    dropArea.classList.remove("active");
    dragText.textContent = "Drag & Drop to Upload File";
});
dropArea.addEventListener("drop", (event) => {
    event.preventDefault();
    dropArea.classList.remove("failed");
    dropArea.classList.add("active");
    file = event.dataTransfer.files[0];
    loadFile();
});
document.getElementById('fileInput').addEventListener('change',
        ev => {
    file = document.getElementById('fileInput').files[0];
    loadFile();
});
function changeClasses(element, defaultClass, newClass) {
    for (let i = 0; i < element.classList.length; i++) {
        let currentClass = element.classList.item(i);
        if (currentClass !== defaultClass) {
            element.classList.remove(currentClass);
        }
    }
    element.classList.add(newClass);
}
function loadFile() {
    var formData = new FormData();
    formData.append("file", file);
    formData.append("userName", username);
    fetch('/upload', {
        method: 'POST',
        body: formData
    })
        .then(response => {
            if (response.ok) {
```

```
                response.text().then(result => {
                    changeClasses(iconElement, 'fa-solid', 'fa-cloud-download');
                    dragText.textContent = result;
                    dragSpan.textContent = 'but you can change file';
                    document.getElementById('processFileButton').classList.remove('disabled');
                    document.getElementById('optimizeFileButton').classList.remove('disabled');
                })
            } else {
                response.text().then(errorMessage => {
                    document.getElementById('processFileButton').classList.add('disabled');
                    document.getElementById('optimizeFileButton').classList.remove('disabled');
                    throw new Error(errorMessage);
                })
                    .catch(error => {
                        console.log(error);
                        dropArea.classList.remove("active");
                        dropArea.classList.add("failed");
                        dragText.textContent = error;
                        dragSpan.textContent = 'try to load new file';
                        changeClasses(iconElement, 'fa-solid', 'fa-cloud');
                    });
            }
        })
    }
});
async function optimizeFile() {
    const loader = document.getElementById('optimizeLoader');
    loader.style.display = 'inline-block';
    try {
        const response = await fetch(`/optimize?userName=${username}`, {
            method: 'POST'
        });
        if (response.ok) {
            const result = await response.text();
            alert(result);
        } else {
            console.error('Помилка при оптимізації файлу');
        }
    } catch (error) {
        console.error('Помилка при оптимізації файлу', error);
    } finally {
        loader.style.display = 'none';
    }
}
async function processFile() {
    const loader = document.getElementById('processLoader');
    loader.style.display = 'inline-block';
    try {
        const response = await fetch(`/process?userName=${username}`, {
            method: 'POST'
        });
```

```
        if (response.ok) {
            const result = await response.text();
            alert(result);
            // Показуємо кнопку "Download zip"
            document.getElementById('downloadZipButton').classList.remove('disabled');
        } else {
            document.getElementById('downloadZipButton').classList.add('disabled');
            console.error('Помилка при обробці файлу');
        }
    } catch (error) {
        console.error('Помилка при обробці файлу', error);
    } finally {
        loader.style.display = 'none';
    }
}
function downloadFile() {
    const loader = document.getElementById('downloadLoader');
    loader.style.display = 'inline-block';
    var xhr = new XMLHttpRequest();
    xhr.open('GET', `/download?userName=${username}`, true);
    xhr.responseType = 'blob';
    xhr.onload = function (e) {
        loader.style.display = 'none';
        if (this.status === 200) {
            var blob = new Blob([this.response], {type: 'application/zip'});
            var url = window.URL.createObjectURL(blob);
            var a = document.createElement('a');
            a.href = url;
            a.download = 'reports.zip';
            document.body.appendChild(a);
            a.click();
            window.URL.revokeObjectURL(url);
        } else {
            console.error('Failed to download file:', this.statusText);
        }
    };
    xhr.onerror = function () {
        loader.style.display = 'none';
        console.error('Network error occurred.');
    };
    xhr.send();
}
```

Лістинг модуля index.html

```
<!DOCTYPE html>
<html lang="uk" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta name="description" content="City Transport Optimization Web App">
    <title>CityTransOpt</title>
    <link rel="stylesheet" type="text/css" href="css/styles.css">
    <link rel="stylesheet" type="text/css" href="css/adapt.css">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.5.1/css/all.min.css">
    <link rel="icon" type="image/svg+xml" href="images/bus-transport.svg">
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
    <script src="js/scripts.js"></script>
</head>
<body>
<header>
    <div id="logo" onclick="slowScroll('#top')">
        <span>CityTransOpt</span>
    </div>
    <div class="menu">
        <a id="optimizeFileButton" class="disabled" href="#" title="Optimize file"
          onclick="optimizeFile()">Optimize<span id="optimizeLoader" class="loader"
style="display: none;"></span></a>
        <a id="processFileButton" class="disabled" href="#" title="Process file"
          onclick="processFile()">Process<span id="processLoader" class="loader" style="display:
none;"></span></a>
        <a id="downloadZipButton" class="disabled" href="#" title="Download zip"
          onclick="downloadFile()">Download<span id="downloadLoader" class="loader"
style="display: none;"></span></a>
        <a href="#" title="Profile" id="username"
th:text="${#authentication.principal.username}">Profile</a>
        <a th:href="@{/logout}" title="Logout">Logout</a>
    </div>
</header>
<video autoplay muted loop>
    <source src="video/clouds.mp4" type="video/mp4">
    Your browser does not support the video tag.
</video>
<div id="top">
    <div class="upload-file-area">
        <div class="drag-area">
            <div class="icon">
                <i class="fa-solid fa-cloud-upload"></i>
            </div>
            <p>Drag & Drop to Upload File</p>
            <span>OR</span>
            <button>Browse File</button>
            <input type="file" id="fileInput" hidden>
        </div>
    </div>
</div>
<script>
</script>
</body>
```

```
</html>
```

Додаток Г – Ілюстративний матеріал

ІЛЮСТРАТИВНА ЧАСТИНА

РОЗРОБКА МЕТОДУ ТА ПРОГРАМНОГО ЗАСОБУ ДЛЯ ОПТИМІЗАЦІЇ МЕРЕЖІ ГРОМАДСЬКОГО ТРАНСПОРТУ МІСТА

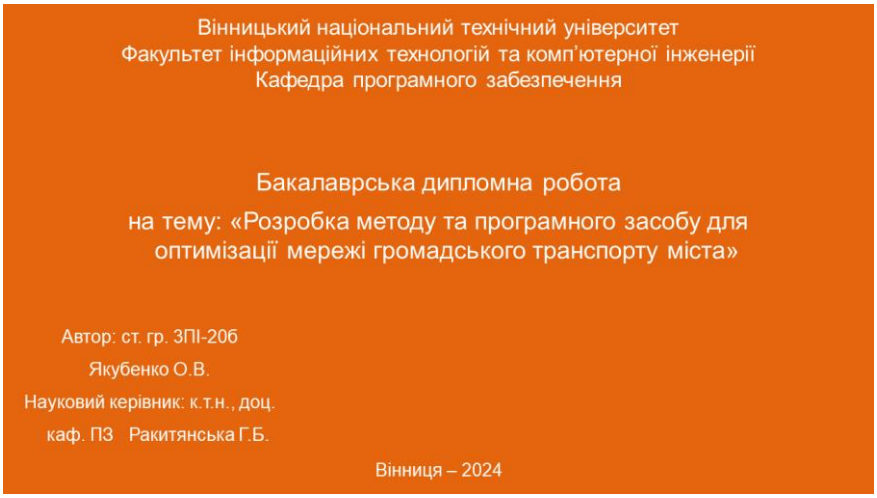

Рисунок Г.1 – Титульний слайд

Рисунок Г.2 – Слайд «Актуальність розробки»

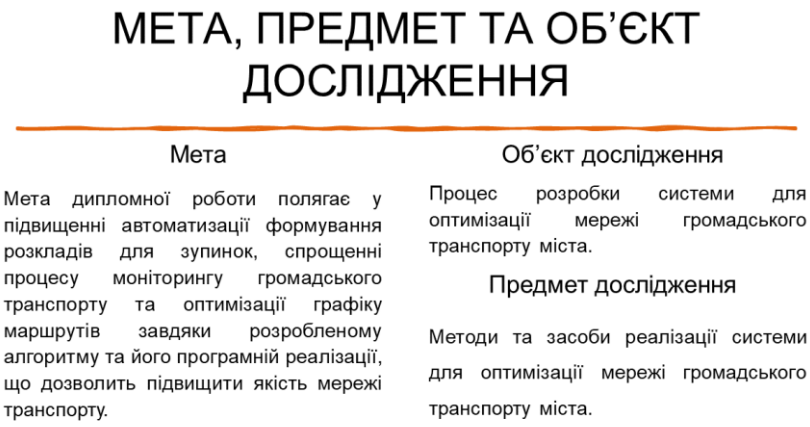

Рисунок Г.3 – Слайд «Мета, предмет та об'єкт дослідження»

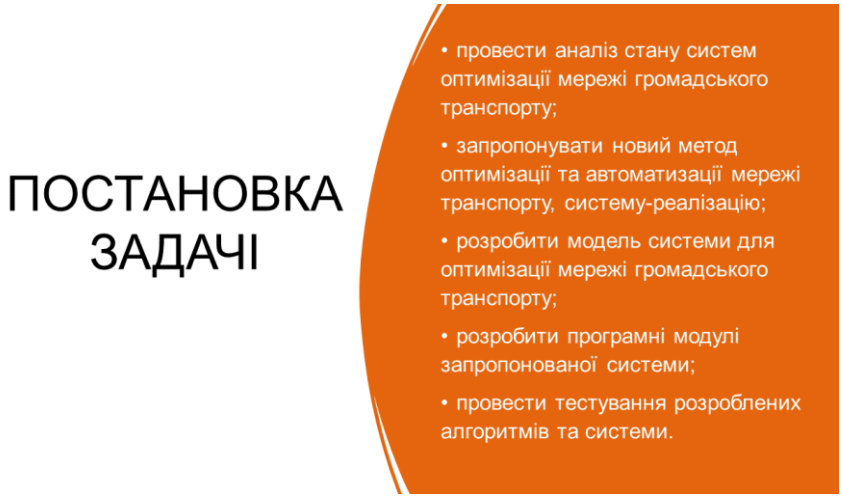

Рисунок Г.4 – Слайд «Постановка задачі»

Рисунок Г.5 – Слайд «Новизна»

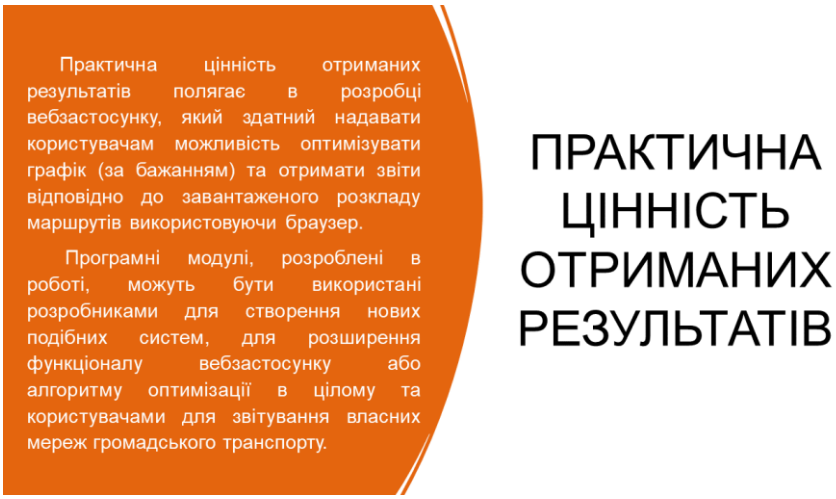

Рисунок Г.6 – Слайд «Практична цінність отриманих результатів»

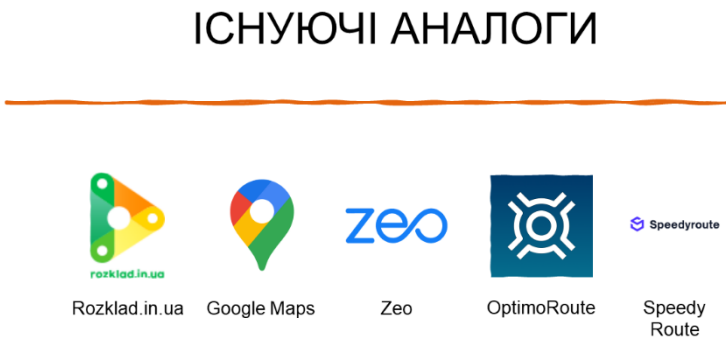

Рисунок Г.7 – Слайд «Існуючі аналоги ч.1»

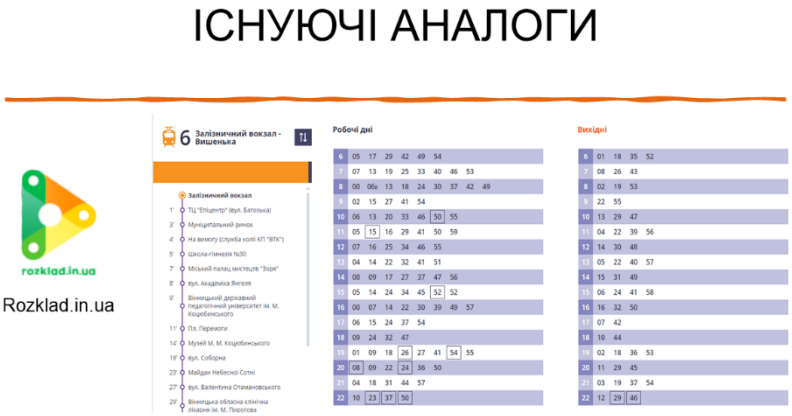

Рисунок Г.8 – Слайд «Існуючі аналоги ч.2»

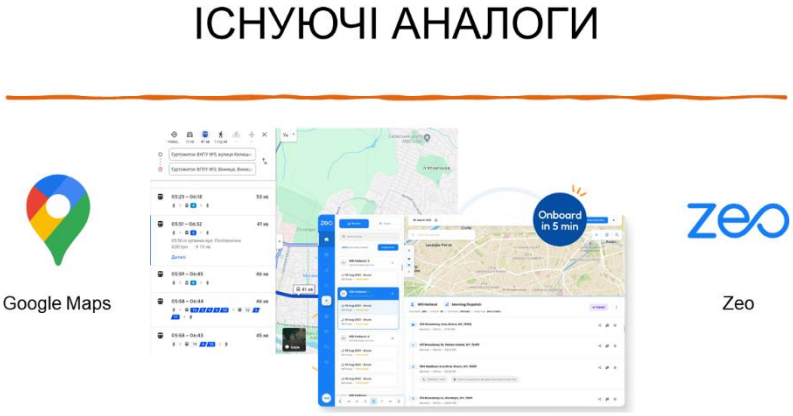

Рисунок Г.9 – Слайд «Існуючі аналоги ч.3»

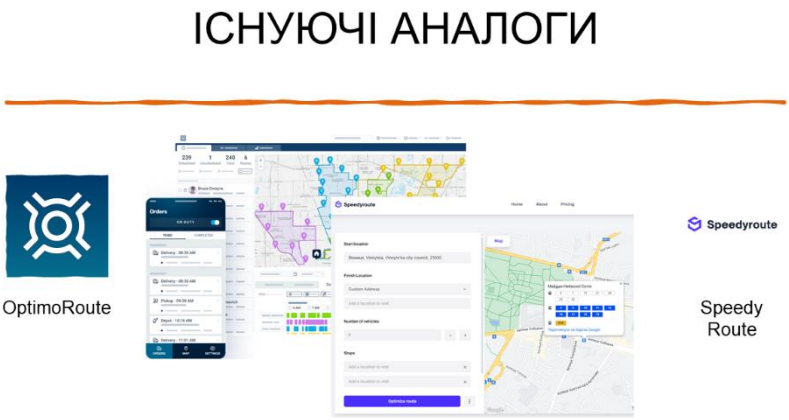

Рисунок Г.11 – Слайд «Існуючі аналоги ч.4»

АНАЛІЗ АНАЛОГІВ

Назва системи	Доступність	Ціна	Гнучкість	Оптимізація розкладу	Автоматизація розкладу
CityTransOpt	Онлайн	Безкоштовно	Висока	Так	Наявна
Rozklad.in.ua	Онлайн	Платно	Висока	Так	Наявна
Google Maps	Онлайн	Безкоштовно	Висока	Ні	Частково
Zeo	Онлайн	1,389.36 грн	Висока	Так	Частково
OptimoRoute	Онлайн	30 днів безкоштовно	Висока	Так	Частково
Speedy Route	Онлайн	2,739.02 грн	Висока	Так	Частково

Рисунок Г.12 – Слайд «Аналіз аналогів»

Рисунок Г.13 – Слайд «Схема роботи адаптивного генетичного алгоритму»

Рисунок Г.14 – Слайд «Модель системи на прикладі діаграми діяльності»

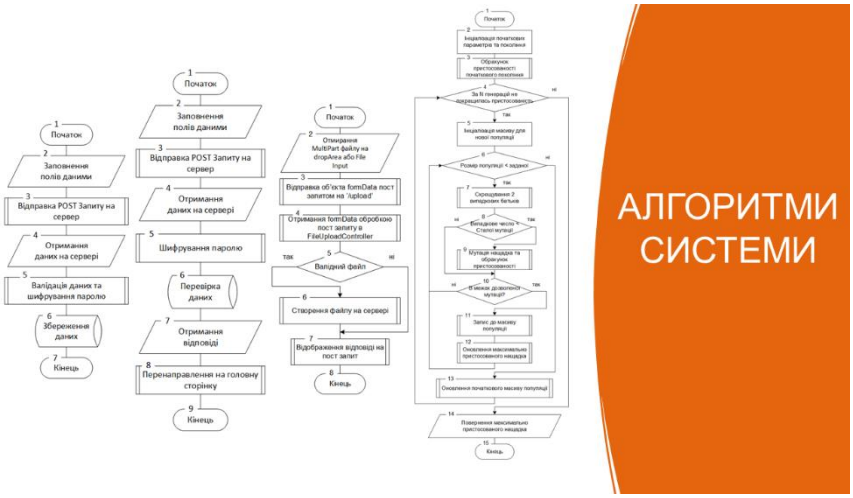

Рисунок Г.15 – Слайд «Алгоритми системи ч.1»

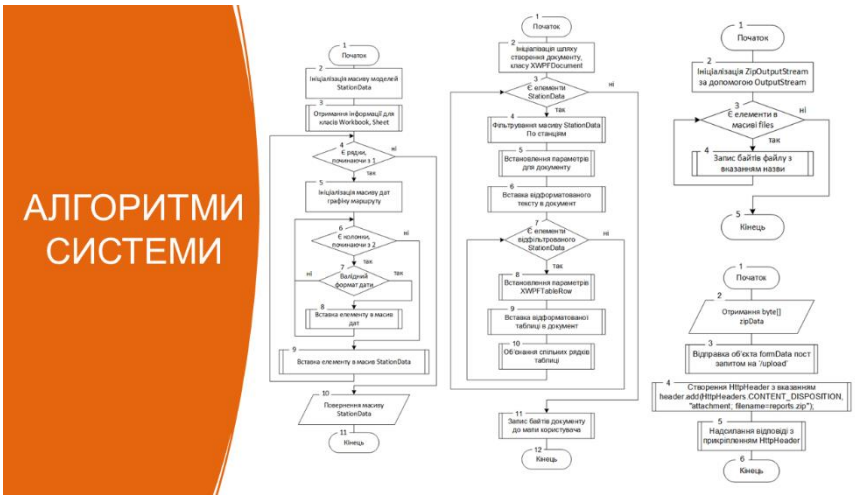

Рисунок Г.16 – Слайд «Алгоритми системи ч.2»

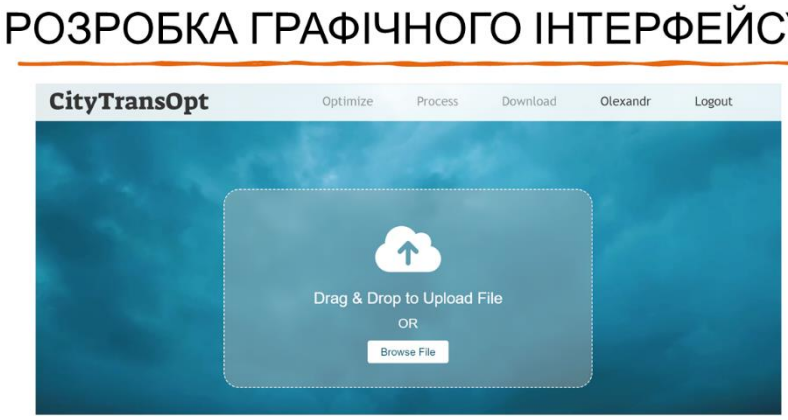

Рисунок Г.17 – Слайд «Розробка графічного інтерфейсу»

Рисунок Г.18 – Слайд «Використані технології»

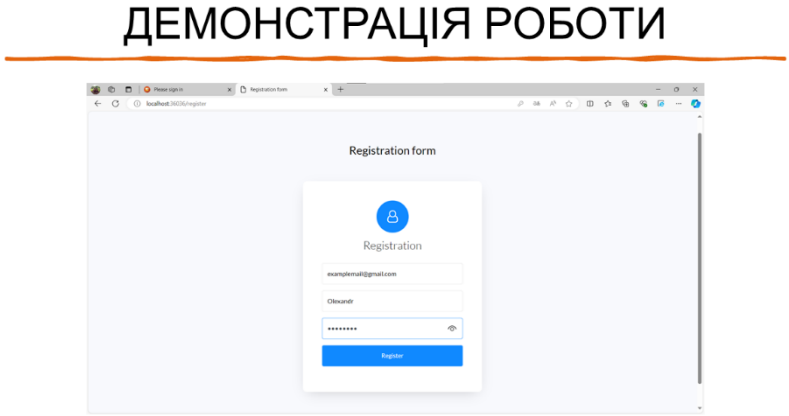

Рисунок Г.19 – Слайд «Демонстрація роботи ч.1»

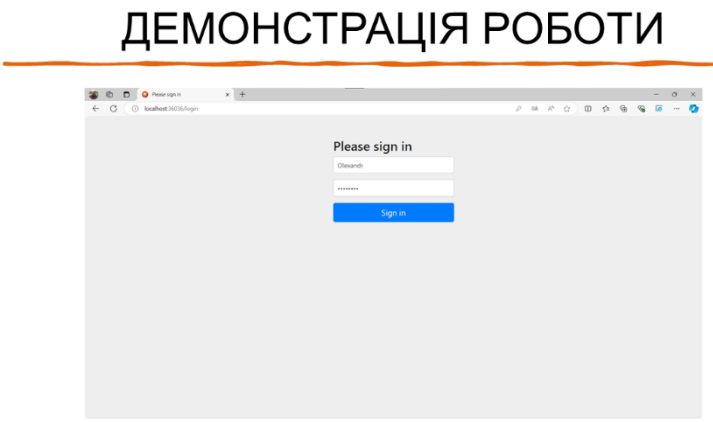

Рисунок Г.20 – Слайд «Демонстрація роботи ч.2»

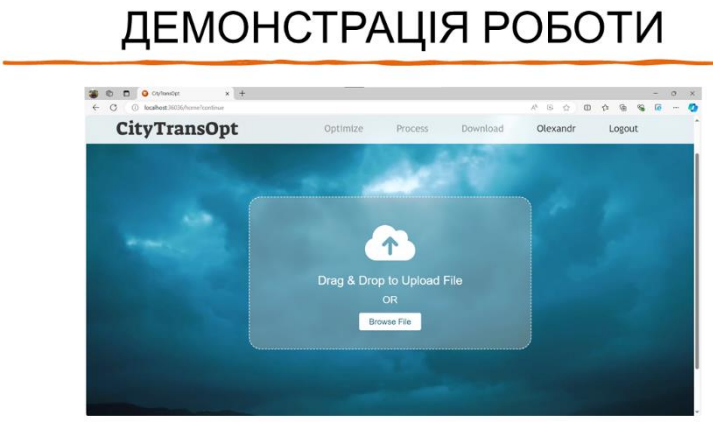

Рисунок Г.21 – Слайд «Демонстрація роботи ч.3»

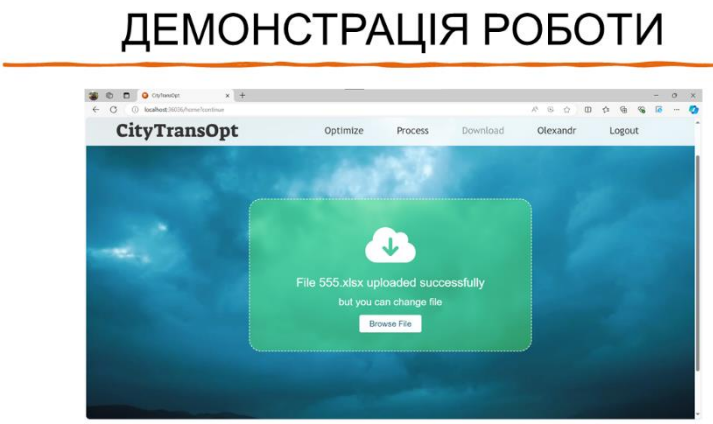

Рисунок Г.22 – Слайд «Демонстрація роботи ч.4»

ДЕМОНСТРАЦІЯ РОБОТИ

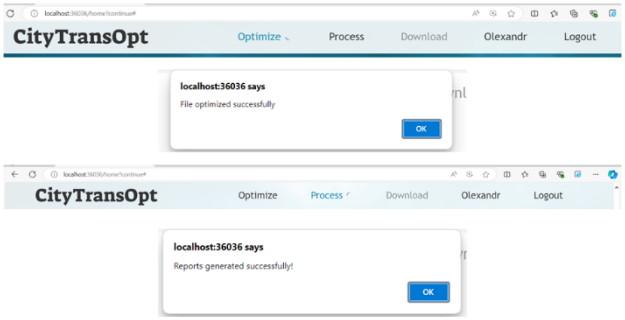

Рисунок Г.23 – Слайд «Демонстрація роботи ч.5»

ДЕМОНСТРАЦІЯ РОБОТИ

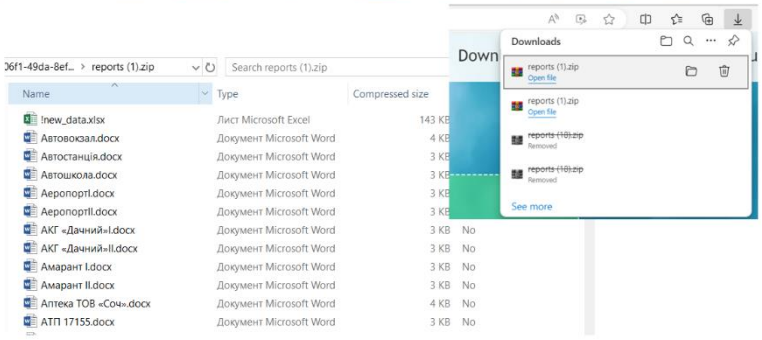

Рисунок Г.24 – Слайд «Демонстрація роботи ч.6»

ДЕМОНСТРАЦІЯ РОБОТИ

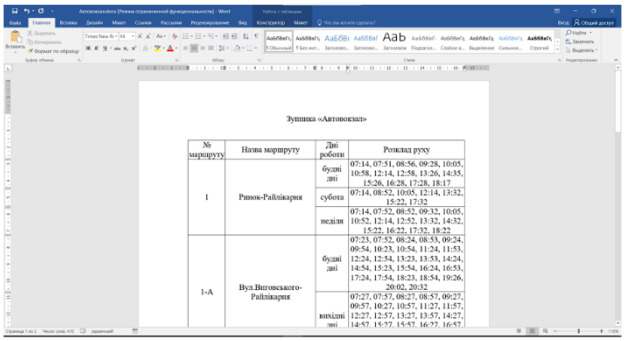

Рисунок Г.25 – Слайд «Демонстрація роботи ч.7»

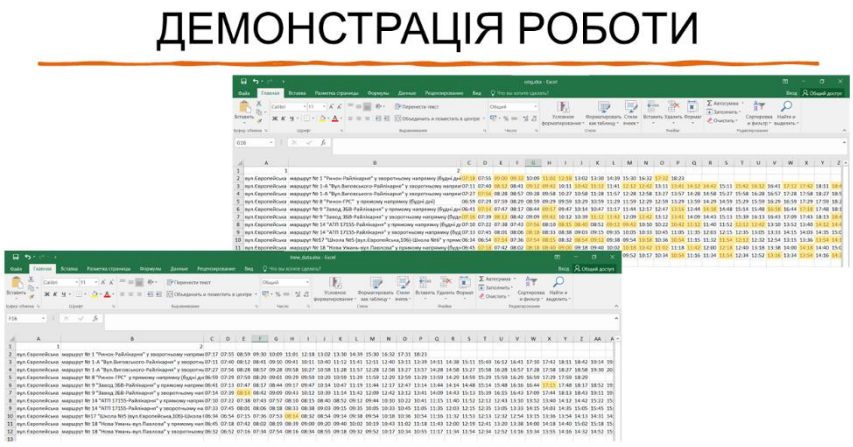

Рисунок Г.26 – Слайд «Демонстрація роботи ч.8»

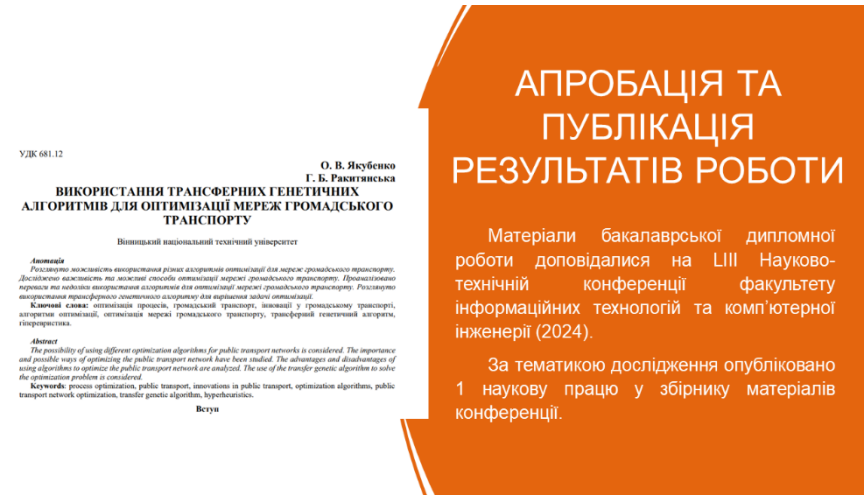

Рисунок Г.27 – Слайд «Апробація та публікація результатів роботи»

Рисунок Г.28 – Слайд «Висновки»

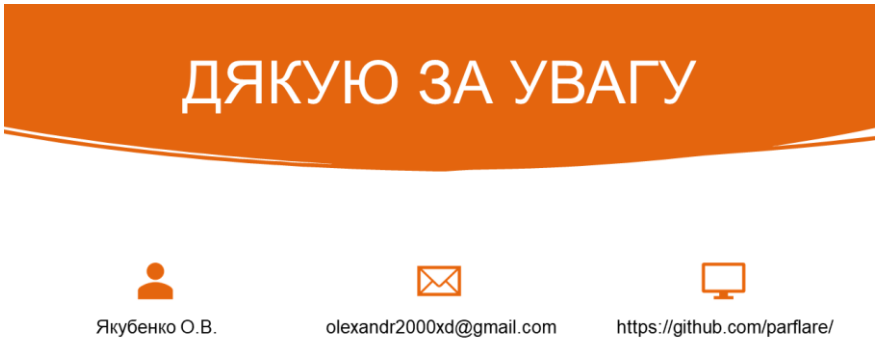

Рисунок Г.29 – Фінальний слайд