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

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему

РОЗРОБКА МЕТОДІВ І ПРОГРАМНОГО ЗАСОБУ ДЛЯ ВЕДЕННЯ РОЗКЛАДУ ПОТЯГІВ ЗАЛІЗНИЦІ ІЗ ВИКОРИСТАННЯМ ДАТЧИКІВ ТА ТРЕКЕРІВ

Виконав: студент II курсу
групи 3ПІ–22м спеціальності
121 – Інженерія програмного забезпечення
Кошелєв Андрій Олександрович

Керівник: к.т.н., доц. каф. ПЗ
Черноволик Г. О.
«13» грудня 2023р.

Опонент: к.т.н., доц. каф. ОТ
Тарновський М. Г.
«18» грудня 2023р.

Допущено до захисту
Завідувач кафедри ПЗ
д.т.н., проф. Романюк О. Н.
(прізвище та ініціали)
«18» грудня 2023р.

ВНТУ – 2023

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Рівень вищої освіти ІІ-й (магістерський)

Галузь знань 12 – Інформаційні технології

Спеціальність 121 – Інженерія програмного забезпечення

Освітньо-професійна програма – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ

Романюк О. Н.

«__» вересня 2023р.

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Кошелєву Андрію Олександровичу

1. Тема роботи – Розробка методів і програмного засобу для ведення розкладу потягів залізниці із використанням датчиків та трекерів.

Керівник роботи: к.т.н., доц. кафедри ПЗ Черноволик Г. О. затверджені наказом вищого навчального закладу від « 18 » вересня 2023 р. № 247.

2. Строк подання студентом роботи

5 грудня 2023 р.

3. Вихідні дані до роботи: модель розробки – ієрархічна; алгоритми реалізації – алгоритм прогнозування затримок потягів; вхідні дані – дані про розташування транспортного засобу; вихідні дані – скільки зупинок проїхав транспортний засіб та приблизний час очікування; середовище розробки – Microsoft Visual Studio Code; мова розробки – JavaScript; операційна система – Windows 10; середовище реалізації додатку – браузер.

4.Зміст текстової частини: обґрунтування вибору методу розробки та постановка задачі дослідження; розробка структури, методу та алгоритмів роботи програмного продукту; розробка програмних компонент для програмного засобу; тестування програми.

5. Перелік ілюстративного матеріалу: порівняльний аналіз аналогів; структура графічного інтерфейсу головного вікна додатку; структура графічного інтерфейсу вікна інформації про програмний засіб; блок-схема алгоритму

знаходження потягу з використанням датчиків в межах однієї гілки; блок-схе
алгоритму знаходження потягу з використанням датчиків по всім гілкам; бло
схема алгоритму обчислення приблизного часу прибуття потягу; тестуван
програми; економічне обґрунтування доцільності розробки; апробація
публікації результатів роботи; фінальний слайд.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада Консультанта	Підпис, дата	
		завдання видав	завдан прийня
1–4	Черноволик Г. О., к.т.н., доц. каф. ПЗ	19.09.2023	01.12.20
5	Причепа І. В. к.е.н., доц. каф. ЕПВМ	11.11.2023	24.11.20

7. Дата видачі завдання 19 вересня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Приміт
1	Аналіз завдання і вибір методу вирішення поставленої задачі дослідження	20.09.2023 – 25.09.2023	вик
2	Розробка методу та алгоритму обчислення приблизного часу прибуття транспортного засобу	25.09.2023 – 29.09.2023	вик
3	Вибір середовища та мови розробки	30.09.2023 – 13.10.2023	вик
4	Розробка модулів програми	14.10.2023 – 27.10.2023	вик
5	Тестування програми	28.10.2023 – 10.11.2023	вик
6	Оцінка економічної ефективності результатів виконаної роботи	11.11.2023 – 24.11.2023	вик
7	Оформлення матеріалів до захисту БДР	25.11.2023 – 01.12.2023	вик

Студент _____ **Кошелєв А.**
(підпис) (прізвище та ініціа

Керівник магістерської кваліфікаційної роботи _____ **Черноволик Г.**
(підпис) (прізвище та ініціа

АНОТАЦІЯ

Кошелєв А. О. Розробка методів і програмного засобу для ведення розкладу потягів залізниці: магістерська кваліфікаційна робота зі спеціальності 121 – інженерія програмного забезпечення, освітня програма – інженерія програмного забезпечення. Вінниця: ВНТУ, 2023. 99 с.

На укр. мові. Бібліогр. : 21 назв ; рис. : 22; табл. 16.

У магістерській кваліфікаційній роботі проведено детальний аналіз стану програмних засобів маршрутизації та розкладу потягів з використанням датчиків та трекерів. Розглянуто особливості методу оцінки затримок потягів. Встановлено об’єкт, предмет, завдання та методи дослідження. Сформульовано мету дослідження – підвищення зручності ведення розкладу для пасажирів за допомогою єдиного місця знаходження розкладу потягів та з прогнозуванням приблизного часу прибуття. Також буде додано можливість сповіщення пасажирів про запізнення транспорту. Для реалізації мети було розроблено спеціалізований програмний засіб.

Запропоновано метод та алгоритм визначення приблизного часу до прибуття у потрібне місце призначення. Розроблено метод визначення скільки і які саме зупинки проїхав транспортний засіб.

Створений програмний засіб розроблено з використанням мови програмування JavaScript, бібліотеки маршрутизації для роботи із програмними засобами та середовища розробки Microsoft Visual Studio Code. В результаті виконання магістерської кваліфікаційної роботи розроблено програмний засіб, працездатність і правильність роботи якого перевірено, підготовлена інструкція користувача.

Ключові слова: програмний засіб, розклад, датчик, трекер, затримка.

ANNOTATION

Koshelev A.O. Development of methods and software for railway trains timetable management: master's thesis on specialty 121 - software engineering, educational program - software engineering. Vinnytsia: VNTU, 2023. 99 p.

In Ukrainian speech Bibliogr. : 21 titles; Fig. : 22; table 16.

In the master's thesis, a detailed analysis of the state of web applications for routing and train schedules using sensors and trackers was carried out. The peculiarities of the method of estimating train delays are considered. The object, subject, tasks and research methods are established. The purpose of the research is formulated - to improve the convenience of keeping a schedule for passengers using a single location for finding the train schedule and predicting the approximate time of arrival. The possibility of notifying passengers about transport delays will also be added. A specialized web application was developed to achieve this goal.

A method and algorithm for determining the approximate time until arrival at the desired destination is proposed. A method of determining how many and exactly which stops a vehicle has passed has been developed.

The created web application is developed using the JavaScript programming language, the routing library for working with software tools, and the Microsoft Visual Studio Code development environment. As a result of the completion of the master's thesis, a software tool was developed, the functionality and correctness of which was checked, and a user manual was prepared.

Keywords: web application, schedule, sensor, tracker, delay.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. Транспорт відіграє ключову роль у сучасному суспільстві і є одним із основних стовпів економіки. Історично дорожній рух і залізничний транспорт були двома головними засобами забезпечення ефективності економіки та сприяли розвитку багатьох країн, включаючи США, Францію, Німеччину, та Японію. Залізниці є надійним і швидким способом переміщення великих об'ємів вантажів та пасажирів на довгі відстані. Вони сприяють підвищенню якості та доступності споживчих товарів, а також сприяють розвитку торгівлі та економіки країни.

Залізничний транспорт є необхідним для мобільності населення та забезпечує доступ до робочих місць, освіти, медичних послуг і розваг. Швидкісні та надзвукові потяги зменшують кількість автомобілів на дорогах, що сприяє зменшенню забруднення повітря та споживанню пального.

У цьому контексті, розробка програмного засобу для ведення розкладу залізниці з використанням датчиків та трекерів є актуальним завданням. Впровадження сучасних технологій дозволить забезпечити точний та актуальний розклад руху поїздів, враховуючи різні фактори, такі як затримки, погодні умови та інші негативні впливи. Використання датчиків та трекерів дозволить отримувати реальний часовий статус поїздів та надавати пасажирам найбільш точну інформацію про їх подорож.

Отже, розробка програмного засобу для ведення розкладу залізниці із використанням датчиків та трекерів має стратегічне значення для підвищення ефективності та надійності залізничного транспорту, покращення якості послуг для пасажирів і сприяння сталому розвитку транспортної галузі та економіки в цілому.

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

Мета і завдання дослідження. Метою даної роботи є підвищення ефективності оповіщення пасажирів за допомогою об’єднання усієї інформації в одному місці, а також максимально автоматизовано повідомляти про можливі зміни в розкладі та за необхідності в ручному режимі доповнити інформацію про затримку.

Задачі дослідження:

– проаналізувати сучасний стан програмних засобів маршрутизації та ведення розкладу із використанням датчиків;

– проаналізувати існуючі алгоритми та їх реалізацію для вирішення питання знаходження місця розташування потягів;

– розробити метод підвищення точності ведення динамічно змінюваного розкладу відповідно до затримок потягів;

– розробити алгоритми для реалізації власної програми для системи пошуку потягів;

– розробити програмний засіб для зберігання та відстеження змін в розкладі;

– провести тестування розробленого програмного програмного засобу.

Об’єкт дослідження – процес збереження та аналізу інформації про маршрути потягів та зміни в розкладі.

Предмет дослідження – методи та засоби розробки програмного засобу програмного засобу для маршрутизації та розкладу з використанням датчиків.

Методи дослідження. У магістерській кваліфікаційній роботі використовувалися: алгоритм обробки даних отриманих датчиками для визначення розташування наземного об'єкта, комп’ютерне моделювання для перевірки та аналізу теоретичних положень, математичне моделювання для оцінки точності розрахування прогнозованих змін в розкладі, методи тестування для перевірки роботи створеного програмного додатку.

Наукова новизна результатів, отриманих у роботі:

– подальшого розвитку отримав метод використання технології автоматичної блок-системи який відрізняється отриманням даних про потяг із вбудованих в неї датчиків і дозволяє виконання обрахунків затримки потягу відносно затвердженого розкладу;

– удосконалено метод внесення змін в розклад, який відрізняється врахуванням автоматично визначеної затримки в розкладі, що дозволяє вносити зміни в реальному часі та підвищує ефективність ведення розкладу потягів.

Практична цінність одержаних результатів. Практична цінність одержаних результатів полягає в тому, що на основі отриманих в магістерській кваліфікаційній роботі теоретичних положень запропоновано алгоритми та розроблено програмні засоби для підвищення ефективності ведення розкладу.

Особистий внесок. Усі наукові результати, викладені у магістерській кваліфікаційній роботі, отримані автором особисто. У наукових працях, опублікованих у співавторстві, автору належать такі результати: розробка алгоритму пошуку потягу на лінії з використанням датчиків та трекерів [2].

Апробація роботи. Результати роботи доповідалися на Всеукраїнській науково-практичній інтернет-конференції молодих вчених та студентів «Сучасні інформаційні системи та технології» (2023 р., м. Херсон).

Публікації. Основні результати досліджень опубліковано в науковій праці [2] у збірниках матеріалів конференцій.

1 ОБҐРУНТУВАННЯ ВИБОРУ МЕТОДУ РОЗРОБКИ ТА ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

1.1 Аналіз стану систем ведення розкладу потягів

Людство завжди намагалося оптимізувати свій рух та транспорт, зокрема на протязі свого існування. Це було спричинене потребою у полегшенні переміщення, розвитку економіки, забезпеченні безпеки та багатьох інших чинників.

Спочатку було винайдено колесо. Винахід колеса був однією з перших важливих кроків у полегшенні переміщення важких предметів та розвитку транспорту. Колесо дозволило створити транспортні засоби, які могли б ідеально прокотитися по рівній поверхні. Від давніх часів люди використовували річковий та морський транспорт для переміщення вантажів та пасажирів. Розвиток суднобудування та навігаційних методів дозволив зробити подорожі по воді більш швидкими та безпечними. Винайдена у 18 столітті парова машина змінила обличчя транспорту. Перші парові локомотиви та судна стали використовуватися для переміщення на залізницях і водних шляхах. Ця технологія дозволила рухатися швидше та подолувати великі відстані. Винахід автомобіля у 19 столітті змінив підхід до особистого та вантажного транспорту. З часом автомобільна промисловість розвинулася, а шляхи та дороги покращилися. Повітряний транспорт зародився у 20 столітті з винаходом літаків та дирижаблів. Ця форма транспорту дозволяє швидко подолати великі відстані, що зробило світ більш доступним. Великі міста розвивають масовий громадський транспорт, такий як метро, автобуси та трамваї, для полегшення переміщення мільйонів людей щоденно. Розвиток сучасних технологій, таких як GPS, сенсори, інтернет речей та штучний інтелект, значно покращив можливості відстеження та оптимізації руху, забезпечуючи більш точний та ефективний транспорт. Виникнення послуг спільного користування, таких як Uber і Lyft, дозволив власникам автомобілів ефективно використовувати свої ресурси та зменшити залежність від власного авто.

За всю свою історію людство постійно шукало способи полегшення та оптимізації руху та транспорту, щоб зробити переміщення більш зручними, швидкими, ефективними та безпечними. Розвиток технологій продовжує відкривати нові можливості для подальшої оптимізації транспортних систем.

Протягом своєї історії, людство завжди старалося оптимізувати свій рух, щоб підвищити продуктивність та скоротити час переміщення до необхідних місць. З розвитком різноманітних транспортних засобів зросла потреба в оптимізації графіків руху транспорту в певних напрямках. Ця оптимізація сприяє ефективному переміщенню пасажирів та спрощує роботу диспетчерів, які контролюють рух громадського транспорту.

Водії та інший персонал бажають оптимального робочого графіку та надійності, а також можливості розпочати маршрут з того самого місця, де він завершується. Операторам важливо відповідати попиту та мати достатню кількість постійних пасажирів на маршруті. Менеджерам інфраструктури також потрібно відводити час для технічного обслуговування. Планування розкладу впроваджується в загальний контекст кількома способами. У відомостях про громадський транспорт можна розглянути процес планування як наступні етапи: розробка мережі маршрутів, створення розкладу, планування рухомого складу та розподілу екіпажів.

Також, для ефективного ведення розкладу, важливо враховувати визначену кількість ресурсів, зокрема, транспортних засобів, необхідних для операцій сервісної лінії. Час відправлення призначених транспортних засобів встановлюється на наступному етапі, який відомий як планування пікових навантажень. Цей етап спрямований на встановлення частоти руху автономних автобусних маршрутів в умовах невизначеного пасажирського попиту та враховує можливість введення додаткових рейсів для зменшення перевантажень на маршруті.

Метод максимальної точки навантаження широко використовується для складних сценаріїв попиту, і його частота визначається на основі простого

виразу замкнутої форми. Він враховує частоту руху досліджуваної автобусної лінії протягом планувального періоду, середню кількість пасажирів, що сідають на борт при відправленні з зупинки протягом цього періоду, вмістимість транспортного засобу і бажаний рівень завантаження протягом планування. Незважаючи на те, що метод максимальної точки навантаження забезпечує високу обслуговуваність для найвищого пасажиропотоку на всіх зупинках протягом планування, такий грубий підхід може призвести до надмірних експлуатаційних витрат і низької продуктивності. Особливо це стосується ситуацій, коли середній пасажирський потік на автобусній зупинці з найбільшим навантаженням у декілька разів перевищує спостережувані автобусні навантаження на всіх інших зупинках, наприклад, де запланована частота повинна забезпечити перевезення майже 100 пасажирів на максимальній точці навантаження однієї зупинки, в той час як на інших зупинках пасажирський потік становить менше 20 осіб.

Та насамперед є ще досить важливий аспект здатний покращити логістику. Розповсюдження розкладу в маси є важливою складовою процесу забезпечення громадського транспорту інформацією про графік руху для пасажирів. Існує кілька способів, якими розклад може бути поширений серед широкої аудиторії.

Системи ведення розкладу потягів становлять невід'ємну частину транспортної інфраструктури та мають стратегічне значення для забезпечення ефективності, точності та безпеки перевезень у сучасному світі. Вони виконують важливу функцію у керуванні рухом поїздів на залізничній мережі, ураховуючи різні аспекти від пасажирського потоку до технічних обмежень і непередбачуваних обставин.

Сучасні системи ведення розкладу потягів користуються передовими технологіями, такими як машинне навчання та аналітика даних для оптимізації та управління рухом поїздів. Це дозволяє автоматизувати процеси планування маршрутів, враховуючи різноманітні фактори, від обсягу пасажирів до стану

інфраструктури та погодних умов. Такий підхід сприяє підвищенню точності, пунктуальності та оптимізації руху потягів.

Проте існують виклики, що стоять перед цими системами. Один із них - інтеграція різних систем ведення розкладу, яка може бути складною через розбіжності в стандартах та протоколах обміну даними між різними операторами залізничного транспорту. Це може призводити до затримок у відображенні актуальної інформації про рух поїздів для пасажирів. Крім того, доступ до інформації для кінцевих користувачів через мобільні додатки або веб-платформи може бути нестабільним або недостатньо повним, що ускладнює планування подорожей.

Ще одним значущим викликом є необхідність підвищення кібербезпеки систем ведення розкладу. Оскільки вони підключені до мережі та використовують технології Інтернету речей, вони стають потенційним об'єктом кібератак та вразливостей, що може вплинути на безпеку та нормальне функціонування систем.

Розвиток та модернізація систем ведення розкладу потягів вимагає комплексного підходу. Інтеграція стандартів, покращення програмного забезпечення, створення єдиних платформ для обміну даними та розширення доступу до актуальної інформації для пасажирів можуть сприяти покращенню якості та надійності цих систем.

Крім того, нові технології, такі як блокчейн або розширена реальність, можуть відкривати нові можливості для покращення систем ведення розкладів. Використання блокчейну може забезпечити більшу безпеку та недоторканість даних про рух потягів, а технологія AR може забезпечити пасажирів більш деталізованою та зручною інформацією під час подорожей.

Усе більше компаній та організацій у цьому секторі звертають увагу на впровадження новітніх технологій, щоб забезпечити кращий сервіс та задоволення потреб пасажирів. Розвиток систем ведення розкладів потягів

відбувається у напрямку більшої автоматизації, точності та гнучкості управління рухом потягів.

Загалом, аналіз стану систем ведення розкладу потягів свідчить про значний потенціал для покращення їхньої ефективності та надійності. Розвиток та вдосконалення цих систем потребує комплексного підходу, включаючи технологічні інновації, стандартизацію, підвищення кібербезпеки та забезпечення зручності для пасажирів.

Одним із найпоширеніших способів розповсюдження розкладу є розміщення його на веб-сайтах громадського транспорту та створення мобільних додатків, які дозволяють пасажирам зручно переглядати і оновлювати інформацію про розклади на своїх смартфонах та планшетах. Приклад веб-сайту наведено на рисунку 1.1.

Рисунок 1.1 – Приклад веб-сайту

Багато міст встановлюють інформаційні табло на громадських транспортних зупинках, які відображають актуальну інформацію про розклади руху. Це дає можливість пасажирам отримувати актуальну інформацію на місці. Приклад такого типу розкладу наведено на рисунку 1.2.

Рисунок 1.2 – Приклад інформаційного табло яке знаходиться безпосередньо на зупинці

Оператори громадського транспорту можуть запропонувати пасажирам підписатися на отримання SMS-повідомлень або електронних листів із інформацією про розклади і зміни у них. Приклад повідомлення по електронній пошті наведено на рисунку 1.3.

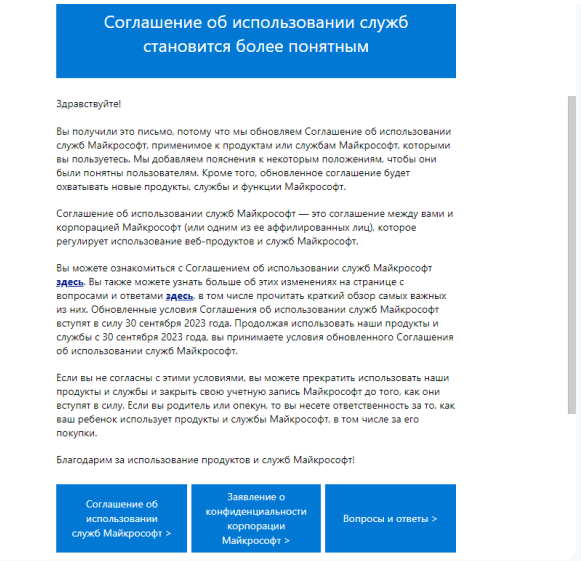


Рисунок 1.3 – Приклад повідомлення по електронній пошті

Спеціалізовані системи транспортного обслуговування: Деякі міста використовують спеціалізовані системи транспортного обслуговування, які дозволяють пасажирам переглядати розклади та отримувати оновлену інформацію через різні пристрої, включаючи кіоски на зупинках. Приклад такого кіоску наведено на рисунку 1.4.

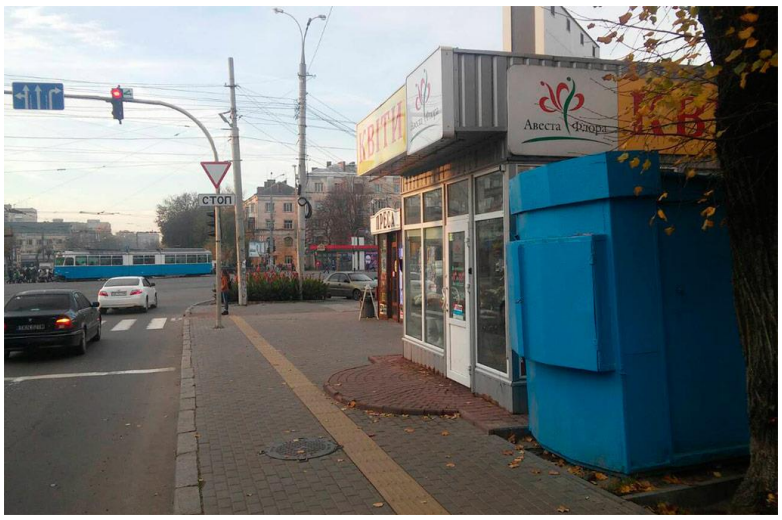


Рисунок 1.3 – Приклад кіоску

Багато операторів громадського транспорту активно використовують соціальні мережі та онлайн-платформи для розповсюдження інформації про розклади та зміни у них. Це дає можливість користувачам легко знаходити інформацію та обмінюватися нею. Приклад онлайн-платформи наведено на рисунку 1.5.

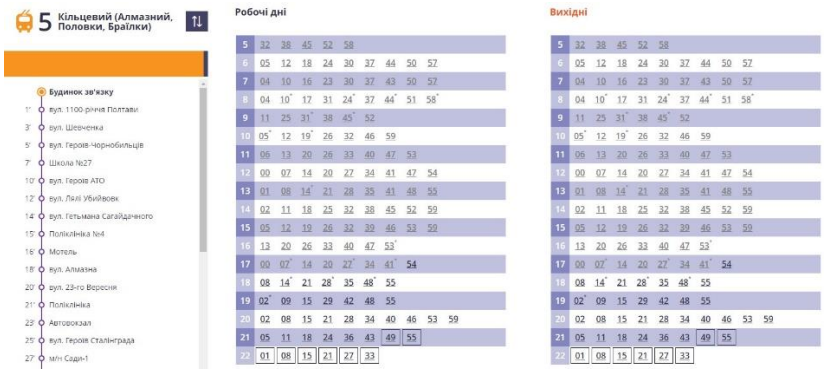


Рисунок 1.3 – Приклад онлайн-платформи

Завдяки цим способам розповсюдження розкладу в маси користувачі можуть бути завжди інформовані про графік руху транспорту, що сприяє більш комфортному та продуктивному користуванню транспортом.

На підставі розгляду способів розповсюдження розкладу громадського транспорту в маси можна зробити наступний висновок. Сучасні технології, включаючи програмні засоби та мобільні платформи, створюють нові можливості для надання інформації пасажирам. Інтернет-додатки надають користувачам зручність перегляду розкладів з будь-якого місця та надійність актуальних даних. У контексті широкого поширення доступу до Інтернету, роздрукований розклад, який розміщується на зупинці, залишається важливим джерелом інформації, але його обмежена актуальність та зручність в порівнянні з програмними засобами. Таким чином, сучасні технології дозволяють пасажирам бути завжди інформованими про розклад руху громадського транспорту, забезпечуючи комфорт та продуктивність у використанні громадського транспорту.

1.2 Порівняльний аналіз аналогів

На сьогоднішній час існує вже досить багато інтернет сторінок з різноманітними розкладами усіх типів сполучення, таких як автобуси, електропотяги, маршрутні таксі і їм подібні. Серед існуючих реалізацій варто виділити інтернет сторінки такі як:

– Мій Гнівань;

– Busfor.ua;

– Poïzdato;

– rozklad.in.ua;

– Вінницький трамвай.

Мій Гнівань – інтернет сторінка яка створена для ведення розкладу приміських автобусів та електропотягів маршрути яких проходять через населений пункт м.Гнівань. Даний веб-ресурс є досить корисним для пасажирів

які мають намір подорожувати як у даний населений пункт так і з нього. Досить зручно коли в одному місці знаходиться розклад усіх типів громадського транспорту які курсують маршрутом що перетинає цей населений пункт. Також доцільним на даному сайті я вважаю розташування календаря, бо є багато рейсів які курсують лише по визначеним наперед днем тижня і місяця. Але також існують і серйозні недоліки цього сайту. Через те що даний сайт створено людьми які не зв'язані з перевізником вони не несуть відповідальність за точність інформації, виставляють рекламу, а також інформація на даному сайті охоплює лише околиці міста Гнівань. Також наявний сайт охоплює лише невелику кількість маршрутів що робить його занадто локальним і обмеженим в плані надання якісної і детальної інформації. На рисунку 1.3 зображено зовнішній вигляд інтернет сторінки Мій Гнівань.

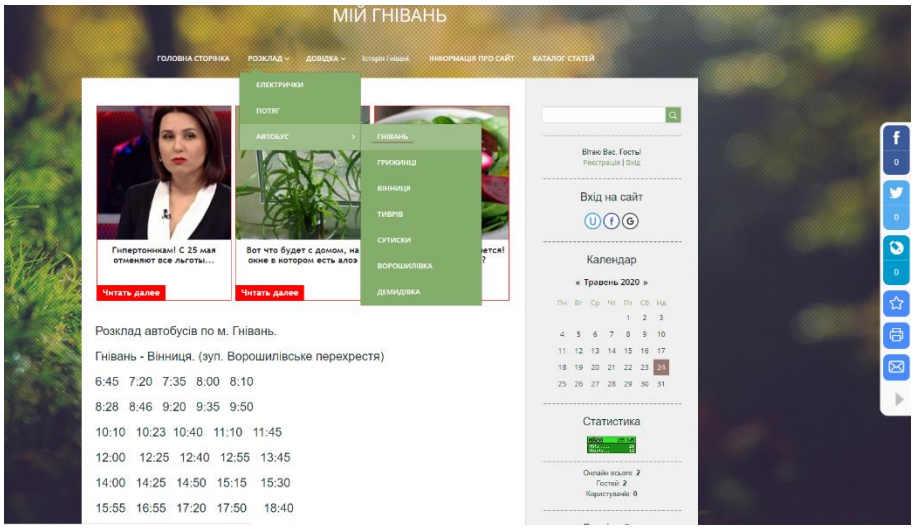


Рисунок 1.3 – Головна сторінка сайту « МІЙ ГНІВАНЬ»

BUSFOR.UA – інтернет сторінка створена для замовлення квитків на автобуси міжміських сполучень, зображення його головної сторінки наведене на рисунку 1.4. На данному сайті є багато корисної інформації, як от доступ до інформації про наявний транспорт, ціна послуги та місце, час відправлення з точки А та час прибуття на кінцеву станцію, можна зв'язатися зі службою підтримки. На сайті також можна зареєструватися для того щоб викупити

заброньований квиток, переглянути інформацію про рейс, завантажити куплений квиток, повернути квитки, залишити відгук про поїздку. На рисунку 1.5 показано ефективні ходи маркетингової служби компанії, як от: якнайшвидше бронювання, потужність, проплата та безпека. Але даний сайт більше сконцентрований на сполучення між далекими містами, а це означає що тут не можна розізнати як потрапити в населені пункти крім обласних чи можливо районних центрів. Також на даному сайті відсутня можливість заздалегідь подивитись усі можливі напрямки руху транспортних сполучень.

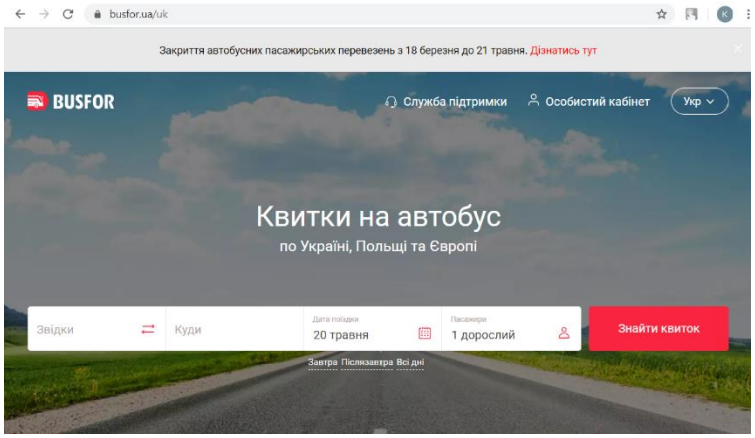


Рисунок 1.4 – Головна сторінка сайту «BUSFOR.UA»

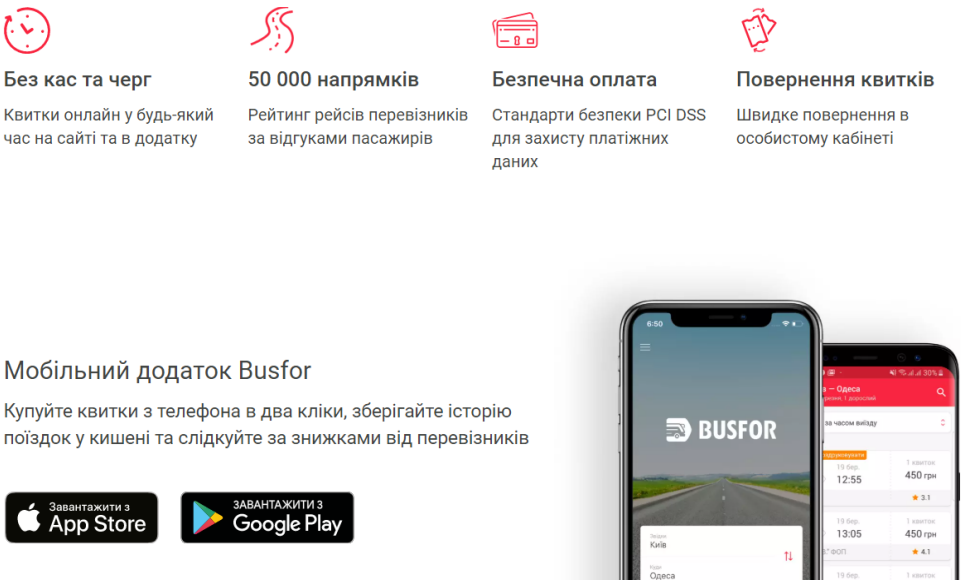


Рисунок 1.5 – Продовження сайту «BUSFOR.UA»

На ряду з цим наявна база даних цієї організації не надає детальної інформації про маршрут автобусів, а саме не можна взнати транзитних зупинок та часу перебування транспортного засобу на цих зупинках.

Poïzdato – інтернет сторінка призначена для ведення розкладу електропоїздів приміського значення. У наявному сайті присутні такі переваги як наявність інформації про час відправлення з станції відправлення та час прибуття в кінцеву. Є контакти з компанією перевізником, а саме форма зворотного зв'язку що дозволяє залишити повідомлення по таким питанням як уточнення в розкладі, консультація з електронного придбання квитків, а також співпраці чи рекламної інтеграції. На рисунку 1.6 зображено зовнішній вигляд сайту Poïzdato.

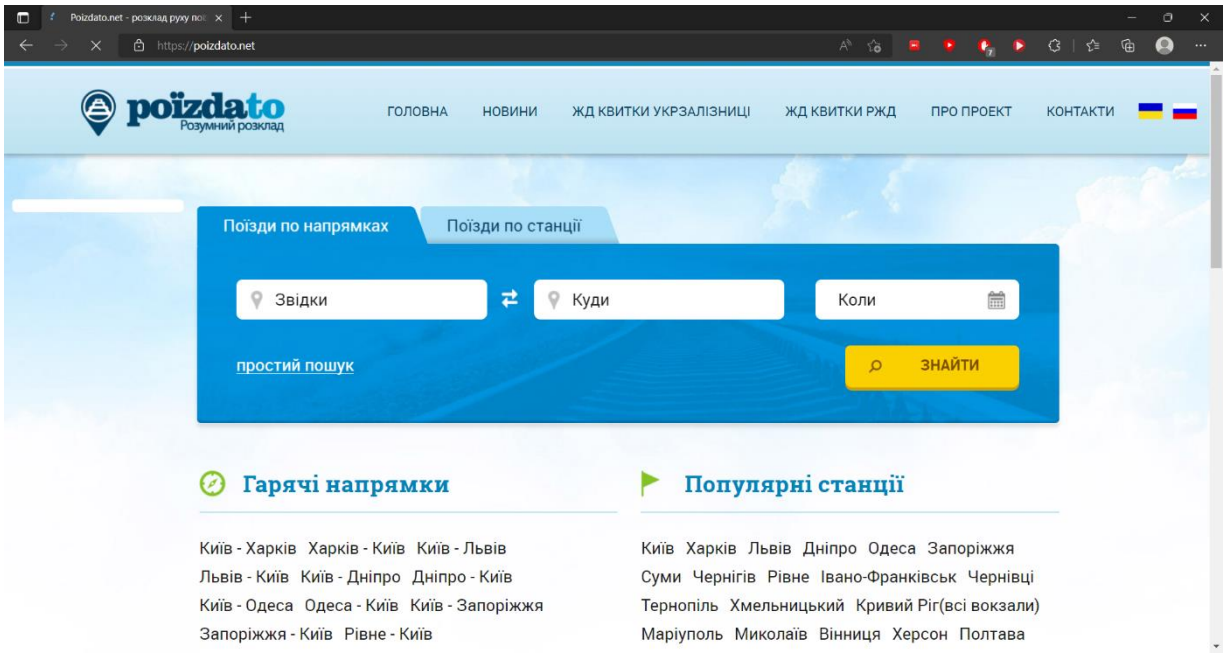


Рисунок 1.6 – Користувацький інтерфейс інтернет сайту Poïzdato

Rozklad.in.ua – інтернет сайт розкладу транспортних засобів міста Вінниця. Наявний сайт надає доступ переглянути усі маршрути які курсують по Вінниці одразу на головній сторінці що дає можливість пасажиру вибрати свій оптимальний маршрут самостійно. Також на сайті інтегровано Карти Google що

дає можливість одразу оглянути обраний маршрут. Але цей сайт надає можливість переглянути маршрути лише ті які не виходять за межі міста, а тому не буде корисним для пасажирів які подорожують поза межами міста. На рисунку 1.7 зображено вигляд сайту Rozklad.in.ua.

Вінницький трамвай – аналогічний інтернет сайт до Rozklad.in.ua. Також спрямований на пасажирів які переміщаються по місту Вінниця, наявна детальна інформація про маршрути, хоч і без карти як в попередньому варіанті. Але тут присутній так званий вказівник який допомагає швидше знайти час наступного транспортного засобу, але дещо незручно перемикатись між різними типами транспорту навіть якщо їхні маршрути співпадають. Також на сайті присутня колонка новини на якій адміністратори висвітлюють актуальні питання по змінам в розкладі На рисунку 1.8 зображено зовнішній вигляд сайту Вінницький трамвай.

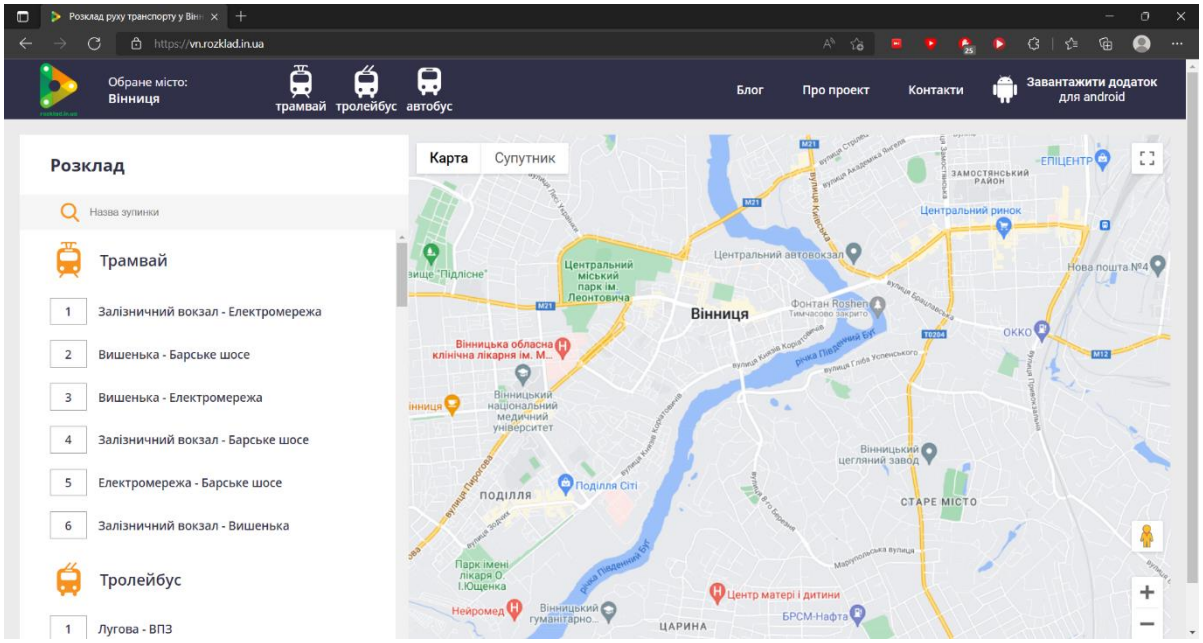


Рисунок 1.7 – зображено вигляд сайту Rozklad.in.ua

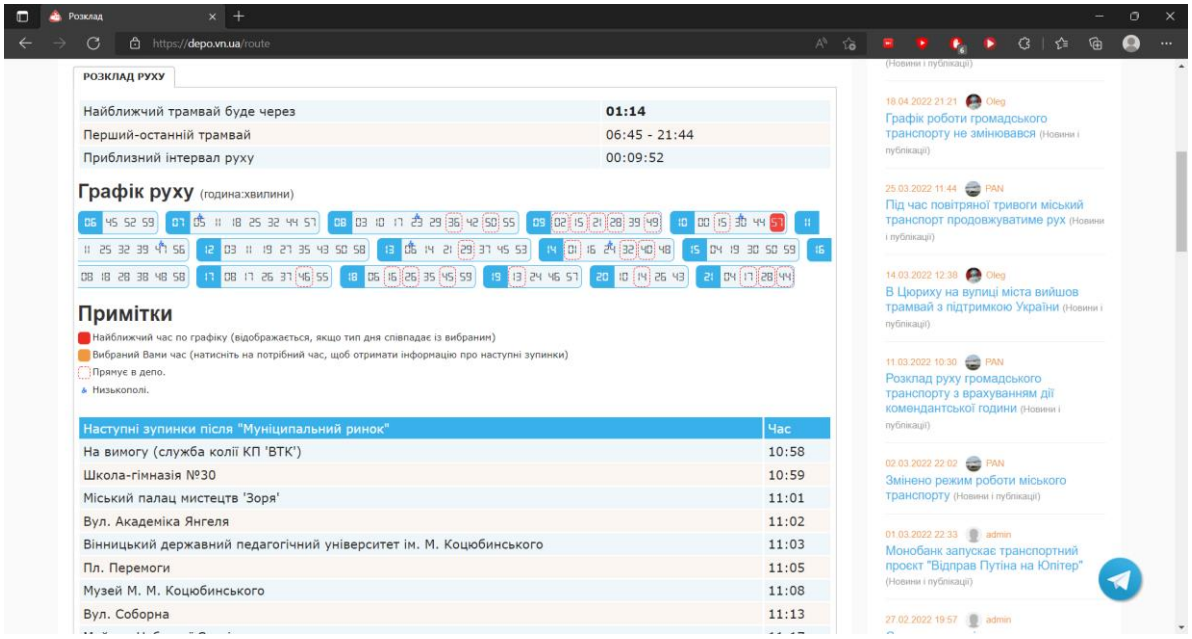


Рисунок 1.8 – зображено вигляд сайту Вінницький трамвай

Порівняльний аналіз аналогів - це ключовий етап прийняття рішень у багатьох областях, від наукових досліджень до бізнесу. Цей процес дозволяє оцінити і порівняти різні варіанти, продукти або концепції для визначення їх переваг, недоліків та відмінностей.

Важливість порівняльного аналізу аналогів полягає в тому, що він допомагає уникнути однобокості при прийнятті рішень, дозволяє зробити об'єктивний вибір, особливо коли є багато варіантів або альтернативних шляхів.

Після аналізу усіх аналогів визначено їхні переваги та недоліки та проведено порівняння із розроблюваним програмним засобом під назвою «Залізниця». Результат порівняння зведено в таблицю 1.1.

В результаті порівняння існуючих аналогів було зроблено висновок, що розробка власного програмного засобу для ведення розкладу потягів є доцільною. В результаті розробки отримаємо продукт, який покриває недоліки існуючих аналогів та забезпечує порівняно вищу ефективність та більший обсяг функціоналу.

Таблиця 1.1 – Порівняльні характеристики інтернет сторінок

Критерій	Мій Гнівань	BUSFOR.UA	Poïzdato	Rozklad.in.ua	Вінницький трамвай	Залізниця
Розрахунок часу до прибуття наступного транспортного засобу	0	0	0	0	1	1
Реєстрування на сайті	1	1	0	1	1	1
Можливість переглянути маршрут на карті	0	0	0	1	0	0
Визначення можливої затримки транспорту	0	0	0	0	0	1
Зворотній звязок	0	1	1	0	1	1
Підсумковий результат	1	2	1	2	3	4

1.3 Аналіз методів розв’язання поставленої задачі

Програмний засіб - це інтерактивна програма, яка працює на сервері і доступна через браузер. Програмний засіб побудований таким чином, щоб інтерфейс користувача надавав дані команді розробників, яка його розробила [4]. Ці дані дають уявлення про інтереси клієнтів, використання та переваги, які можуть виявитися безцінними для продуктових та маркетингових стратегій. Дані також можуть інформувати оптимізацію та інші клієнтоорієнтовані аспекти мобільного додатка або настільних додатків.

Велика різниця між програмними засобами та веб-сторінками полягає в тому, що користувальницький досвід диктує дизайн програмних засобів. Традиційний веб-дизайн заснований на серверних програмістах, які приймають рішення про те, що може підвищити зручність використання. На відміну від цього, програмний засіб має інтерфейс прикладної програми (API), який приймає великомасштабні дані з боку користувача, а потім спрямовує цю інформацію в автоматизацію [4].

Більшість мобільних додатків, які з'являються на смартфонах, є програмними засобами. Ось короткий список поширених програмних засобів:

Facebook, Instagram, Twitter та інші платформи соціальних мереж

Gmail, Yahoo, AOL та будь-які веб-програми електронної пошти

Будь-який портал клієнтів самообслуговування

Сайти запитів, такі як Quora і Google

Важливо відзначити відмінності, які існують між рідними програмними засобами та мобільними додатками. Мобільні додатки живуть на пристрої і призначені для роботи на певній платформі (наприклад, iOS або Android). Приклади включають Facebook Messenger і Карти Google. Пошук Google, який перенесе вас безпосередньо в інтернет-браузер, буде прикладом програмного засобу, тоді як Карти Google – це мобільний додаток.

Веб-сайт побудований на платформі, яку може змінити тільки творець або розробник. З іншого боку, інтерактивний і побудований на платформі, яка дозволяє користувачам інформувати ітерації програми [5]. Оскільки вони є колекцією HTML-документів, програмні засоби можуть бути частиною встановленого веб-сайту або створюватися як самостійні програми.

Застосунок повинен бути доставлений через мережу і підключений до бази даних. Якщо традиційні веб-сайти служать основною метою доставки інформації (наприклад, тексту або відео) користувачеві, програмний засіб дозволяє його взаємодії з запитом користувача для отримання різних можливих результатів.

На залізничних дорогах країн СНГ для визначення місцезнаходження потягів використовуються різноманітні технології та системи. Основні методи включають:

1. Автоматична блок-система (АБС): Це один з основних методів контролю за рухом потягів на залізницях. Використовується система датчиків, розташованих уздовж колії, які взаємодіють зі спеціальним обладнанням на локомотивах. Вони дозволяють визначити місцезнаходження потягу та виключати можливість зіткнення між поїздами за допомогою сигналів та систем аварійного гальмування.

2. Системи GPS та ГЛОНАСС: Деякі залізниці використовують системи навігації, такі як GPS або ГЛОНАСС, для визначення місцезнаходження потягів у реальному часі. Ці системи можуть бути встановлені на локомотивах або вбудовані безпосередньо в інфраструктуру залізниці.

3. Радіочастотна ідентифікація (RFID): Деякі системи використовують технологію RFID для визначення місцезнаходження потягів. Це передбачає використання радіочастотних технологій для ідентифікації та відстеження локомотивів та вагонів під час руху по мережі.

4. Візуальні системи відстеження: Деякі системи можуть використовувати візуальні камери та обробку зображень для відстеження руху потягів та визначення їхнього місцезнаходження.

Ці технології часто використовуються окремо або у поєднанні одна з одною для забезпечення точного визначення розташування потягів на залізниці. Кожна з цих систем має свої переваги та обмеження і використовується залежно від конкретних умов та потреб залізниці.

Автоматична блок-система (АБС) є ключовою технологією для контролю руху потягів на залізницях. Вона має свої переваги та обмеження.

Переваги АБС:

1. Безпека: Однією з основних переваг є забезпечення безпеки руху потягів. АБС дозволяє автоматично контролювати рух поїздів, моніторити ділянки колії та уникати зіткнень.
2. Пунктуальність: Вона сприяє покращенню пунктуальності шляхом оптимізації руху потягів та планування маршрутів.
3. Автоматизація: Зменшення людського втручання в управлінні потягами, оскільки багато процесів автоматизовані.
4. Моніторинг стану шляху: Система дозволяє відстежувати стан залізничної колії та інфраструктури, виявляти несправності та потенційні проблеми.

Обмеження АБС:

1. Високі витрати на впровадження: Встановлення та підтримка АБС може бути дуже витратним процесом для залізничних компаній.

2. Потреба в постійному оновленні: Швидкі зміни технологій можуть потребувати постійного оновлення системи для забезпечення високої ефективності та сумісності з новими рішеннями.

3. Залежність від інфраструктури: Ефективність АБС повністю залежить від якості та наявності інфраструктури, такої як датчики, сигнальні системи тощо.

4. Можливість виникнення помилок: Хоча АБС забезпечує безпеку, вона все ж може бути схильна до помилок чи випадкових перешкод, що може вплинути на її ефективність.

5. Обмеження масштабування: Деякі системи АБС можуть мати обмеження у масштабуванні або інтеграції з іншими технологіями.

Враховуючи ці переваги та обмеження, важливо ретельно вивчити та урахувати контекст та потреби конкретної залізниці перед впровадженням системи АБС.

Системи GPS (Global Positioning System) та ГЛОНАСС (Глобальная навигационная спутниковая система) є глобальними навігаційними системами,

які використовуються для визначення місцезнаходження об'єктів на поверхні Землі. Кожна з цих систем має свої переваги та обмеження.

Переваги систем GPS та ГЛОНАСС:

1. Глобальне покриття: Обидві системи надають глобальне покриття, що охоплює практично всю поверхню Землі, що дозволяє визначати місцезнаходження де завгодно на планеті.

2. Точність: У відкритих просторах обидві системи можуть забезпечити високу точність визначення місцезнаходження.

3. Надійність: Ці системи стали важливим інструментом для навігації та визначення місцеположення завдяки своїй високій надійності.

4. Широкий спектр застосувань: Вони застосовуються в різних галузях, включаючи транспорт, геодезію, картографію, туризм, авіацію та інші.

5. Обмеження систем GPS та ГЛОНАСС:

6. Обмеження в зонах згущення: В густонаселених місцях або областях з високими спорудами може бути втрачений сигнал або знижена точність через перешкоди, такі як будівлі або гірські масиви.

7. Затримки сигналу: У деяких випадках сигнал може бути затриманий або втрачений через атмосферні умови або інші фактори.

8. Залежність від зовнішніх чинників: Вони можуть бути вразливі до впливу зовнішніх чинників, таких як інтерференція, військові аспекти (наприклад, блокування сигналу в певних зонах).

9. Необхідність відкритого неба: Вони потребують прямого "видимого" з'єднання з супутниками, тому в низькозависимих областях або в замкнутих приміщеннях можуть виникати проблеми з отриманням сигналу.

Ці системи використовуються широко та є невід'ємною частиною багатьох сучасних технологій, але важливо усвідомлювати їхні можливості та обмеження при їх використанні для різних застосувань.

Радіочастотна ідентифікація (RFID) - це технологія, що дозволяє бездротово ідентифікувати та відстежувати об'єкти за допомогою радіочастот. Вона має свої переваги та обмеження.

Переваги технології RFID:

1. Автоматизація і швидкість: RFID дозволяє швидко і автоматично ідентифікувати об'єкти без прямого контакту. Це дозволяє прискорити процеси ідентифікації та відстеження товарів, товарно-матеріальних цінностей, пасажирів та інших об'єктів.

2. Дальність читання: Залежно від типу RFID, вона може дозволяти читати інформацію на відстані від кількох метрів до кількох десятків метрів, що дозволяє відстежувати об'єкти навіть на великих відстанях.

3. Широкий спектр застосувань: RFID застосовується у багатьох сферах, таких як логістика, транспорт, служби безпеки, медицина, а також в промисловості та управлінні запасами.

4. Мініатюризація технології: Чіпи RFID можуть бути дуже малими, навіть мікроскопічними, що дозволяє легко вбудовувати їх у предмети різних розмірів та форм.

Обмеження технології RFID:

1. Вартість: Деякі варіанти RFID можуть бути витратними для масового впровадження, особливо у великих кількостях, що може стати обмеженням для бізнесів.

2. Інтерференція та читання в густині: В окремих умовах можуть виникати проблеми з читанням через інтерференцію, особливо у місцях з великою кількістю RFID сигналів.

3. Проблеми з приватністю та безпекою даних: Зберігається ризик зловживання чи несанкціонованого використання даних, тому що чіпи можуть бути читані без наочного контакту.

4. Обмежена місткість інформації: Деякі типи RFID мають обмежену місткість для зберігання інформації, що може стати проблемою для деяких застосувань.

RFID - потужний інструмент для відстеження та ідентифікації об'єктів, але при його використанні важливо усвідомлювати ці переваги та обмеження для оптимального використання технології в різних галузях.

Візуальні системи відстеження використовуються для моніторингу та відстеження об'єктів за допомогою відеокамер і обробки зображень. Вони мають свої переваги та обмеження.

Переваги візуальних систем відстеження:

1. Висока точність: Деякі візуальні системи можуть мати високу точність відстеження об'єктів, особливо при використанні високоякісних відеокамер та програмного забезпечення для обробки зображень.

2. Широкий спектр застосувань: Вони використовуються в різних галузях, таких як виробництво, транспорт, безпека, медицина, наукові дослідження тощо.

3. Реальний час: Деякі системи пропонують відстеження в реальному часі, що дозволяє операторам швидко реагувати на події чи відхилення.

4. Можливості аналізу: Обробка зображень може дати можливість для аналізу даних, виявлення паттернів чи аномалій.

Обмеження візуальних систем відстеження:

1. Потреба в освітленні та умовах: Візуальні системи можуть бути чутливі до умов освітлення. Погане освітлення може вплинути на якість зображення та точність відстеження.

2. Потреба в обробці даних: Обробка великої кількості даних з камер може вимагати потужних комп'ютерних ресурсів та програмного забезпечення.

3. Обмежена точність на великих відстанях: На великих відстанях візуальні системи можуть мати обмежену точність відстеження об'єктів.

4. Проблеми з приватністю: Використання візуальних систем може породжувати питання про приватність, оскільки вони можуть відстежувати людей без їхньої згоди.

5. Чутливість до перешкод: Препятствія, такі як перешкоди або сторонні об'єкти, можуть вплинути на якість зображення та відстеження.

Візуальні системи відстеження є ефективними у багатьох сферах, проте важливо враховувати їхні обмеження при виборі та використанні для конкретних завдань чи застосувань.

Для розробки програмного продукту було використано саме АБС. Переваги АБС полягають у зниженні ризику аварій, підвищенні безпеки пасажирів та ефективності руху поїздів. Вони дозволяють автоматизувати та контролювати процеси на залізничній дорозі, що робить їх більш надійними.

Однак, вартість впровадження, потреба у системному оновленні та можливість помилок в роботі системи - це деякі з обмежень, які варто враховувати. Але в даному випадку ця система вже працює на дорожньому полотні, що дозволяє знехтувати вартістю системи при розгляданні витрат. Все ж, АБС залишає однією з ключових технологій для забезпечення безпеки та ефективності руху поїздів на залізничних дорогах.

Для ефективної обробки інформації яка надходить за допомогою АБС було вирішено створити окремий програмний модуль який на основі отриманих даних буде оцінювати затримку, якщо вона є, і відповідно змінювати час прибуття.

Реалізація даного програмного додатку у вигляді програмного засобу є неймовірно вигідною по тій простій причині, що доступ до інтернету і завантажений веб-переглядач зараз є майже у кожної людини в світі. Для використання цього програмного продукту не потрібно встановлювати ніяких сторонніх програм, що в наш час може бути досить небезпечно. Достатньо перейти за посиланням, або знайти по ключовим словам в браузері.

API (інтерфейс програмування застосунків) - це набір правил і протоколів, які дозволяють різним програмам взаємодіяти одна з одною. Це як набір будівельних блоків, за допомогою яких різні програми можуть обмінюватися інформацією або виконувати певні функції.

API визначає, які конкретні запити можна робити до програми або сервісу, які параметри потрібно передавати разом із запитом і які дані будуть отримані у відповідь. Це може бути все, починаючи з доступу до бази даних до отримання доступу до функціоналу веб-сервісів чи програмних бібліотек.

API використовується в різних сферах: веб-розробці, мобільних додатках, соціальних мережах, хмарних сервісах та багатьох інших областях. Наприклад, веб-сайт може використовувати API соціальних мереж для відображення вмісту з соціального профілю користувача на своєму власному веб-сайті. API дозволяє різним сервісам взаємодіяти та обмінюватися даними, розширюючи можливості програм та сприяючи їх інтеграції.

Для ефективної взаємодії програмного засобу з сервером потрібно організувати передачу повідомлень. Сервер обміну повідомленнями – це програма, яка обробляє повідомлення між двома або більше програмами. Ці повідомлення передаються до програми проміжного програмного забезпечення за допомогою API обміну повідомленнями (MAPI). Сервери обміну повідомленнями можуть зберігати повідомлення в черзі, доки їх не можна буде доставити до застосунків-одержувачів.

У наявності досить багато різноманітних бібліотек, які використовують Visual Viewport API. Є дуже багато як переваг так і недоліків у кожній із цих бібліотек. Lodash - це бібліотека, яка містить багато зручних методів маніпулювання даними. Приклади включають методи масиву, такі як squash вкладеного масиву в звичайний масив. Існує також зведення всіх рівнів вкладеного масиву в плоский масив.

1.4 Постановка задач дослідження

Проаналізувавши питання розробки програмного засобу для ведення розкладу потягів, було визначено наступні завдання, які необхідно виконати для розробки програмного продукту:

– проаналізувати сучасний стан програмних засобів ведення розкладу;

– проаналізувати існуючі алгоритми та їх реалізацію для вирішення питання знаходження місця розташування транспортних засобів;

– розробити метод підвищення точності ведення динамічно змінюваного розкладу відповідно до затримок транспортних засобів;

– розробити алгоритми для реалізації власної програми для системи пошуку транспортних засобів;

– розробити програмний засіб для зберігання та відстеження змін в розкладі;

– провести тестування розробленого програмного програмного засобу.

Технічне завдання до роботи подано у Додатку А.

1.5 Висновки до розділу

У цьому розділі було проведено аналіз стану систем ведення розкладу які є актуальними на даний момент часу. Було проведено аналіз аналогів які на сьогодні є в наявності та проведено порівняння їх між собою та програмним засобом ведення розкладу «Залізниця». У результаті чого було доведено доцільність розробки магістерської кваліфікаційної роботи. Також проведено аналіз існуючих підходів до вирішення проставленої задачі, в результаті чого вирішено створити окремий програмний модуль для обробки інформації яка буде надходити від системи датчиків. Також було вирішено використовувати бібліотеку Lodash, яка допоможе краще маніпулювати даними і підвищить ефективність роботі усього програмного продукту. Було встановлено основні завдання, які необхідно виконати для розробки програмного засобу ведення розкладу руху потягів.

2 РОЗРОБКА СТРУКТУРИ, МЕТОДУ ТА АЛГОРИТМІВ РОБОТИ ПРОГРАМНОГО ПРОДУКТУ

2.1 Аналіз даних

Користувач розпочинає роботу із кожним програмним засобом за допомогою звичайного браузера, далі він у своєму веб-переглядачі запускає сайт через посилання, або знаходить його через пошуковик по ключовим словам. І від цього часу починається тісна взаємодія користувача і програмного засобу. Основна проблема цього зв’язку полягає в тому що він проходить через відкриту мережу інтернет, а оскільки у функціоналі програмного засобу є можливість реєстрування клієнта потрібно забезпечити збереження даних які будуть передаватись по мережі [7].

Для того щоб зберегти інформацію в процесі передачі її потрібно зашифрувати. Більшість фахівців з інтернет безпеки розбивають шифрування на три різні методи: симетричний, асиметричний і хешування. Ці методи також поділяються на різні типи.

Хешування створює унікальний підпис фіксованої довжини для набору даних або повідомлення. Кожне конкретне повідомлення має свій унікальний хеш, що робить незначні зміни в інформації легко відстежується. Дані, зашифровані хешуванням, не можуть бути розшифровані або повернуті назад у вихідну форму. Саме тому хешування використовується тільки як метод перевірки даних. Багато експертів з інтернет-безпеки навіть не розглядають хешування фактичним методом шифрування, але суть полягає в тому, що це ефективний спосіб показати, що ніхто не втручався в інформацію [8].

Симетричний метод шифрування, відомий також як криптографія закритого ключа або алгоритмом секретного ключа, цей метод вимагає, щоб відправник і одержувач мали доступ до одного ключа [9]. Отже, одержувач повинен мати ключ перед розшифровкою повідомлення. Цей метод найкраще підходить для закритих систем, які мають менший ризик вторгнення третьої

сторони. З позитивного боку симетричне шифрування відбувається швидше, ніж асиметричне шифрування. Однак з негативного боку обидві сторони повинні переконатися, що ключ надійно зберігається і доступний тільки для програмного забезпечення, яке йому потрібно використовувати. А оскільки інформація буде передаватись у відкритій то такий спосіб не буде безпечним у цій ситуації.

Асиметричний метод шифрування, або криптографія з відкритим ключем, цей метод використовує два ключі для процесу шифрування, публічний і закритий ключ, які математично пов'язані [10]. Користувач використовує один ключ для шифрування, а інший для розшифровки, хоча не має значення, який ви виберете в першу чергу. Як випливає з назви, відкритий ключ є у вільному доступі для будь-кого, тоді як закритий ключ залишається лише у призначених одержувачів, яким він потрібен для розшифровки повідомлень. Обидва ключі - це просто великі числа, які не ідентичні, але в парі один з одним, де надходить "асиметрична" частина.

Алгоритм RSA - найбільш широко використовуваний асиметричний алгоритм - вбудований в SSL / TLS, який використовується для забезпечення безпечного зв'язку через комп'ютерну мережу [11]. RSA отримує свою безпеку від обчислювальної складності факторингу великих цілих чисел, які є добутком двох великих простих чисел. Множення двох великих простих чисел відбувається легко, але складність визначення початкових чисел з коду, факторинг, становить основу криптографії відкритого ключа. Час, необхідний для факторингування продукту двох досить великих простих чисел, виходить за рамки можливостей більшості нападників. Що означає що цей метод є досить захищеним для відкритих мереж, а з цього випливає що саме цей метод шифрування потрібен для програмного засобу який розробляється.

Для ефективної взаємодії між користувачем і програмним засобом в основному достатньо наперед створених команд у вигляді кнопок, це досить зручно враховуючи що додаток запускається за допомогою браузера. Але у

програмі присутні такі функції як зворотній зв'язок та реєстрування за допомогою пошти, що змушує використовувати текстовий спосіб передачі інформації, який буде найбільш оптимальним для даних функцій.

Для програмних засобів існує купа різних бібліотек, наприклад jQuery UI. jQuery UI - це розширення JavaScript бібліотеки jQuery, яка надає набір інструментів для створення веб-інтерфейсів та програмних засобів з додатковими можливостями. Ця бібліотека включає в себе різноманітні компоненти, які можна легко використовувати для створення елементів інтерфейсу, таких як кнопки, вікна, меню, перетягування елементів та інші.

Основні компоненти jQuery UI:

1. Віджети (Widgets): Набір готових до використання елементів інтерфейсу, таких як кнопки, діалогові вікна, меню, календарі, слайдери тощо.
2. Ефекти (Effects): Анімаційні ефекти, такі як зникаючі/з'являючі елементи, анімація зміни розміру, зміна прозорості тощо.
3. Взаємодія користувача (Interactions): Можливості взаємодії з користувачем, наприклад, перетягування елементів, переміщення, масштабування, зміна порядку елементів тощо.

jQuery UI дозволяє використовувати ці компоненти та функціонал для створення користувацького інтерфейсу без необхідності писати велику кількість JavaScript-коду з нуля. Вона забезпечує простоту використання та швидкість розробки.

Ще однією перевагою є те, що jQuery UI працює на багатьох сучасних браузерах, що робить її популярним вибором для розробників, які шукають швидке та зручне створення інтерфейсу користувача. Однак, через зростання альтернативних технологій та фреймворків, jQuery UI може бути менш актуальною для розробки сучасних програмних засобів, оскільки вона пропонує обмежений функціонал порівняно з більш сучасними та потужними фреймворками.

Bootstrap - це потужний фреймворк фронтенду, що базується на HTML, CSS і JavaScript, розроблений для швидкого та простого створення веб-інтерфейсів та респонсивних веб-сайтів. Він надає набір зручних інструментів та компонентів, які дозволяють розробникам ефективно організовувати та стилізувати структуру веб-сторінок без необхідності писати багато власного CSS або JavaScript коду.

Основні характеристики Bootstrap:

1. Готові компоненти: Bootstrap надає широкий набір готових до використання компонентів, таких як кнопки, форми, навігаційні панелі, каруселі, модальні вікна тощо. Ці компоненти дозволяють швидко створювати структуру веб-сторінок та додавати функціонал безпосередньо на сторінку.

2. Респонсивний дизайн: Bootstrap вбудовує респонсивність у свої компоненти, що робить сторінки сумісними з різними пристроями та розмірами екранів, пристосовуючи їх відображення до розмірів пристрою користувача.

3. Сітка (Grid system): Фреймворк має сітку, яка дозволяє організовувати контент на сторінці за допомогою рядків і стовпців, що спрощує структурування та розміщення елементів на сторінці.

4. Споживачі CSS та JavaScript: Bootstrap має вбудовані стилі та скрипти, які можна використовувати безпосередньо з коробки. Це робить розробку швидшою та менш складною.

5. Підтримка браузерів: Bootstrap підтримує багато популярних браузерів, що робить його універсальним для використання на різних платформах.

За допомогою Bootstrap розробники можуть ефективно створювати сучасні веб-сайти з адаптивним дизайном, використовуючи готові компоненти та зручні інструменти для швидкої розробки. Однак, у деяких випадках, використання стандартних компонентів Bootstrap може призвести до схожості веб-сайтів та вимагати додаткових налаштувань, щоб зробити дизайн більш унікальним.

Pure CSS - це набір CSS-фреймворків, які дозволяють розробникам створювати стилізовані та естетичні веб-сторінки, використовуючи лише CSS (без JavaScript або інших зовнішніх бібліотек). Він спрощує розробку, надаючи базові стилі та компоненти для створення привабливого дизайну без необхідності великої кількості коду.

Основні характеристики Pure CSS:

1. Легкість використання: Pure CSS пропонує прості та зрозумілі класи, які можна додавати до HTML-елементів для стилізації. Це робить його простим у використанні для початківців та швидким для досвідчених розробників.

2. Респонсивний дизайн: Багато компонентів Pure CSS розроблені з урахуванням респонсивності, що дозволяє створювати веб-сайти, які працюють і виглядають добре на різних пристроях та розмірах екранів.

3. Модульність і кастомізація: Фреймворк пропонує набір модульних компонентів, які можна поєднувати для створення власних унікальних дизайнів. Розробники можуть легко налаштовувати стилі відповідно до своїх потреб.

4. Швидкість завантаження сторінки: Без використання JavaScript чи інших скриптів, Pure CSS може сприяти швидкому завантаженню сторінок, оскільки не вимагає додаткових завантажень бібліотек.

5. Сумісність із браузерами: Більшість функцій Pure CSS підтримуються більшістю сучасних браузерів, що дозволяє йому бути досить універсальним у використанні.

Pure CSS - це корисний інструмент для швидкого створення привабливого дизайну без необхідності використання JavaScript або інших бібліотек. Однак, він може бути менш функціональним у порівнянні з іншими CSS-фреймворками, що пропонують більше готових компонентів та можливостей для розробки складніших інтерфейсів.

Але одна з кращих основа якої якісна візуалізація додатку це API Visual Viewport[12]. Ця бібліотека надає явний механізм запиту та зміни властивостей

візуального viewport вікна. Візуальний режим перегляду – це візуальна частина екрана, за винятком екранних клавіатур, областей за межами області зуму щіпки або будь-якого іншого екранного артефакту, який не масштабується з розмірами сторінки.

Веб-сайт містить два viewports, макет viewport і візуальний viewport. Порт перегляду макета охоплює всі елементи на сторінці, а візуальний вид - це те, що насправді видно на екрані. Коли користувач стискає масштабування сторінки, візуальний viewport стискається, але порт перегляду макета залишається незмінним. Функції інтерфейсу користувача, такі як екранна клавіатура (OSK), можуть стискати візуальний вигляд, не впливаючи на режим перегляду макета. Візуальний viewport дозволяє веб-розробникам вирішити це, позиціонуючи елементи щодо того, що показано на екрані.

Щоб отримати доступ до візуального вигляду вікна, можна отримати об'єкт VisualViewport з властивості window.visualViewport. Об'єкт включає в себе набір властивостей, що описують viewport. Він також додає дві події, і , що вогонь кожного разу, коли візуальний вид змінюється. Ці події дають змогу розташувати елементи відносно візуального поля зору, які зазвичай прив'язуються до режиму перегляду макета .onresize onscroll. Посилання лише для читання на об'єкт VisualViewport вікна. Якщо ця властивість не існує, API не підтримується.

Для розробки було обрано саме VisualViewport. VisualViewport Представляє візуальний вигляд для заданого вікна. Об'єкт вікна надає інформацію про розташування та розмір viewport, а також отримує події зміни розміру та прокрутки, які можна відстежувати, щоб дізнатися, коли відбуваються зміни в viewport вікна.VisualViewport.

2.2 Розробка структури графічного інтерфейсу програмного засобу

Користувацький інтерфейс веб додатку буде частково залежати від браузера який обере користувач, але в більшості випадків різниця між

браузерами не досить велика і займає лише верхню полоску, хоч іноді і зустрічаються випадки де окрім верхньої полоски також є ще бокова. Але проблем із цим не виникне оскільки програмний засіб є кросплатформерним і чудово буде відображатись навіть на смартфоні.

У шапці програмного засобу зліва зображено його назву, а зправа чотири кнопки переходу: реєстрування, вхід, контакти та зворотній зв’язок. Кнопки відправляють на відповідні паралельні вікна програмного засобу.

У лівій частині знаходиться колонка розкладу на якій користувач може обрати тип його відображення, а саме за маршрутом, зупинкою, типом транспорту та дня.

У правій частині знаходиться колонка з новинами сервісу та можливими змінами розкладу.

У центральній частині знаходиться поле для розкладу, а саме назва маршруту, або якщо було обрано іншу вкладку з лівої колонки то її відповідник, знизу сам розклад і з правого боку від розкладу панель індикаторів затримок, для того щоб показати чи затримується транспорт чи йде по своєму розкладі.

Під час розробки інтерфейсу програмного засобу маршрутизації та розкладу з використанням датчиків були реалізовані способи керування за допомогою кнопок на відповідній панелі для переходу на потрібне вікно, а також текстові повідомлення які використовуються для реалізації функцій авторизації, реєстрування та зворотного зв’язку.

На рисунках 2.1 та 2.2 зображені структурні схеми інтерфейсів вікон програмного засобу.

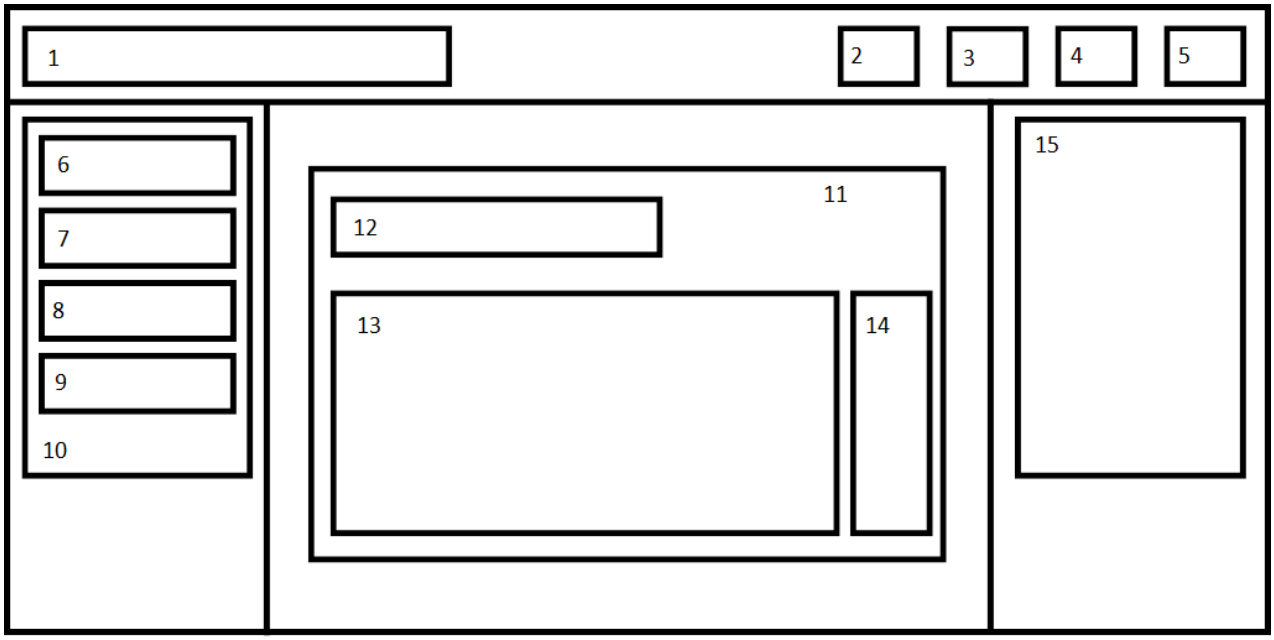


Рисунок 2.1 – Графічна схема вікна маршрути програмного засобу «Залізниця»

Основні елементи інтерфейсу вікна маршрути:

1. Назва програмного засобу.
2. Пункт «Зворотній зв’язок».
3. Пункт «Контакти».
4. Пункт «Вхід».
5. Пункт «Реєстрування».
6. Кнопка «Маршрут».
7. Кнопка «Зупинка».
8. Кнопка «Тип транспорт».
9. Кнопка «Тип дня».
10. Область вибору типу відображення розкладу.
11. Область відображення розкладу.
12. Назва області «Маршрут».
13. Список розкладу.
14. Панель перевірки затримки.
15. Область відображення новин.

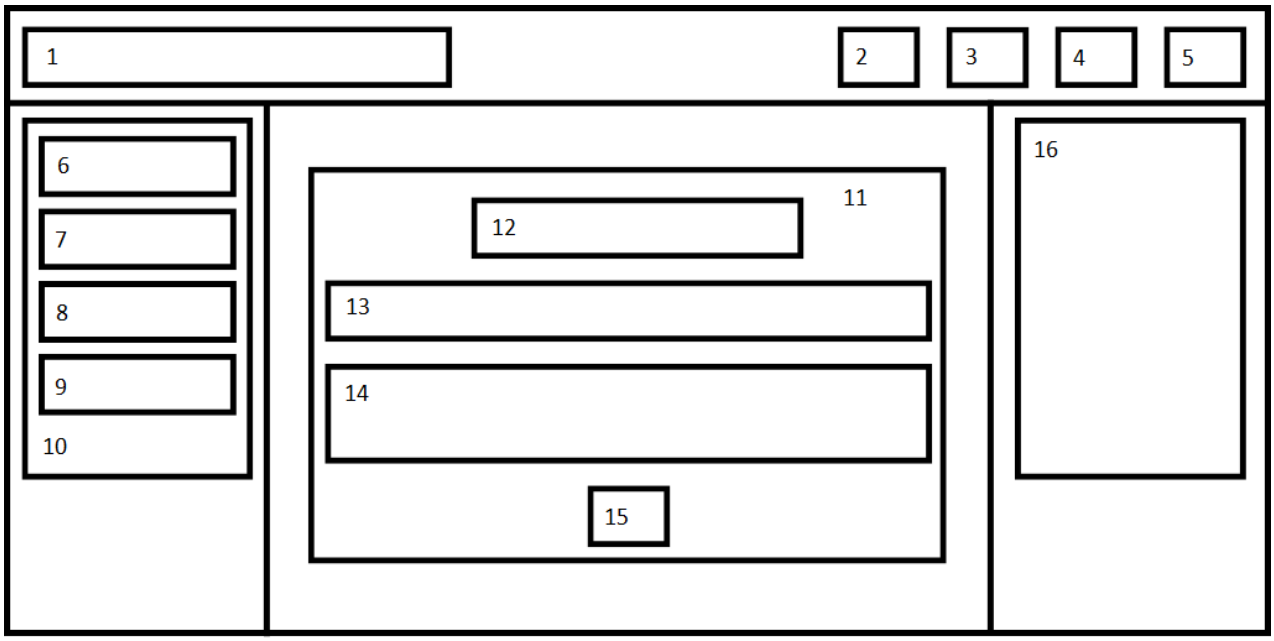


Рисунок 2.2 – Графічна схема вікна зворотній зв’язок програмного засобу «Залізниця»

Основні елементи інтерфейсу вікна маршрути:

1. Назва програмного засобу.
2. Пункт «Зворотній зв’язок».
3. Пункт «Контакти».
4. Пункт «Вхід».
5. Пункт «Реєстрування».
6. Кнопка «Маршрут».
7. Кнопка «Зупинка».
8. Кнопка «Тип транспорт».
9. Кнопка «Тип дня».
10. Область вибору типу відображення розкладу.
11. Область «Зворотній зв’язок».
12. Назва області «Зворотній зв’язок».
13. Текстове поле вводу теми звернення.
14. Текстове поле вводу повідомлення.

15. Кнопка «Надіслати».
16. Область відображення новин.

Тому що велика кількість користувачів зазвичай оцінює зовнішній вигляд перед тим як почати використовувати програмний засіб інтерфейс повинен бути приємним та зручним у використанні. Він повинен відображати всі важливі функції додатку, але й не навантажувати користувача понад міру великою кількістю інформації.

2.3 Розробка основних методів та алгоритмів роботи програмного засобу

Одним із основних методів програмного засобу маршрутизації та розкладу з використанням геолокації є обрахування затримки транспортного засобу. Для його реалізації було створено формулу (формула 2.4). Формула повинна знаходити час прибуття транспорту. А для цього потрібно знайти суму часу на даний момент і часу прогнозованого руху:

$$arrivalt = realt + delayt, \tag{2.1}$$

де $realt$ – час серверу, $delayt$ – прогнозований час руху транспорту до наступної зупинки, який розраховується за формулою:

$$delayt = \frac{k_1 * k_2 * k_3 * |S|}{V}; \tag{2.2}$$

k_1 – ваговий коефіцієнт кривизни дороги; k_2 – ваговий коефіцієнт якості дорожнього покриття; k_3 – ваговий коефіцієнт завантаженості ділянки дороги іншими транспортними засобами; V – коефіцієнт середньої швидкості руху транспорту відповідно до швидкісного режиму ділянки дороги та технічних характеристик транспорту; S – відстань між координатами транспорту і зупинки що розраховується за формулою:

$$S = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}; \tag{2.3}$$

x_1 – широта зупинки; y_1 – довгота зупинки; x_2 – широта транспорту; y_2— довгота транспорту;

Ця формула на основі даних отриманих із системи датчиків про місцезнаходження транспорту та бази даних розкладу про кількість зупинок які вже залишено позаду, час прибуття на наступну зупинку та коефіцієнти кривизни, якості дорожнього покриття та завантаженості ділянки дороги. Формула знаходження затримки набула ось такого вигляду:

$$arrivalt = realt + \frac{k_1 * k_2 * k_3 * \left|\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}\right|}{V}. \quad (2.3)$$

Ця формула на основі даних отриманих із системи датчиків про місцезнаходження транспорту та бази даних розкладу про час серверу, місцезнаходження наступної зупинки та коефіцієнти кривизни, якості дорожнього покриття, завантаженості ділянки дороги та середньої швидкості руху транспорту розраховує прогнозований час прибуття на наступну зупинку.

Формула була програмно реалізована у алгоритмі обрахування ймовірної затримки транспортного засобу (рисунок 2.3) і складається з наступних кроків:

Крок 1. Початок.

Крок 2. Перевірка лічильника, який рахує кількість зупинок які вже проїхав транспортний засіб за заданим маршрутом.

Крок 3. Отримання даних місцезнаходження навігатора водія, а саме відстань по маршруту до потягу яка є сумою відстані до датчика і від датчика до потяга (можлива від’ємна величина якщо потяг не перетнув сам датчик) транспортного засобу.

Крок 4. Отримання даних про місцезнаходження зупинки із бази даних розкладу.

Крок 5. Визначення відстані від транспортного засобу до зупинки використовуючи відстань по маршруту.

Крок 6. Обрахувати прогнозований час прибуття транспорту з врахуванням усіх вагових коефіцієнтів.

Крок 7. Отримати час прибуття згідно з розкладом в базі даних та створити прапорець запізнення.

Крок 8. Порівняти час прибуття і час прибуття за розкладом, якщо час прибуття за розкладом менше прогнозованого часу то перейти до кроку 9 в іншому випадку до кроку 10.

Крок 9. Час прибуття за розкладом замінити значенням прогнозованого часу прибуття, а також змінити значення прапорця запізнення в активне положення.

Крок 10. Вивести значення час прибуття за розкладом разом з прапорцем запізнення.

Крок 11. Кінець.

Рисунок 2.3 – Алгоритм обрахування затримки транспортного засобу

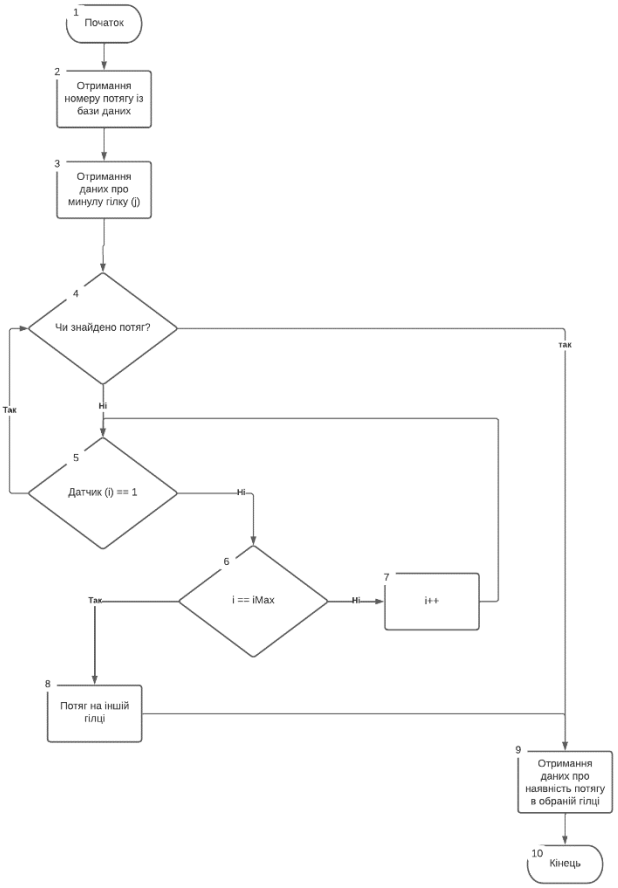


Рисунок 2.4 – Алгоритм знаходження потягу в межах одної гілки за допомогою датчиків

Алгоритм знаходження потягу в межах одної гілки за допомогою датчиків складається з наступних кроків:

Крок 1. Початок.

Крок 2. Отримання даних про потяг який потрібно знайти з бази даних застосунку «Залізниця».

Крок 3. Отримання даних про попереднє місце перебування, тобто залізничну гілку, потягу з бази даних.

Крок 4. Цикл перевірки чи дійсно знайдено потяг в результаті роботи алгоритму, базове значення не знайдено. Якщо не знайдено перейти до кроку 5, інакше перейти до кроку 9.

Крок 5. Перевірка інформації з датчику і-го порядку, відлік починається з першого датчика. Якщо потяг знайдено перейти до кроку 4, інакше перейти до кроку 6.

Крок 6. Перевірка чи було досягнуто останнього датчика, тобто максимального значення їх кількості. Якщо було досягнуто максимального значення перейти до кроку 8, інакше до кроку 7

Крок 7. Збільшення лічильника датчиків на 1.

Крок 8.Потяг не було знайдено, вивести інформацію про те що потяг на іншій гілці маршруту.

Крок 9. Отримання даних про місцеперебування потягу, або його відсутність на конкретній гілці.

Крок 10. Завершення алгоритму.

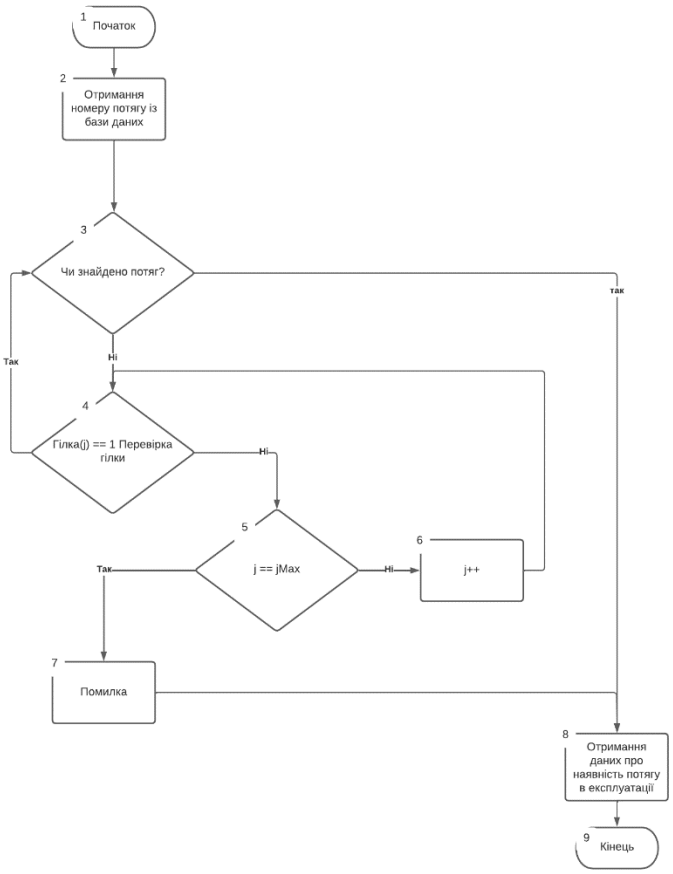


Рисунок 2.5 – Загальний алгоритм знаходження потягу

Загальний алгоритм знаходження потягу за допомогою датчиків з використанням попереднього алгоритму складається з наступних кроків:

Крок 1. Початок.

Крок 2. Отримання даних про потяг який потрібно знайти з бази даних застосунку «Залізниця».

Крок 3. Цикл перевірки чи дійсно знайдено потяг в результаті роботи алгоритму, базове значення не знайдено. Якщо не знайдено перейти до кроку 4, інакше перейти до кроку 8.

Крок 4. Перевірка інформації з гілки j-го порядку використовуючи попередній алгоритм, відлік починається з першої гілки. Якщо потяг знайдено перейти до кроку 3, інакше перейти до кроку 5.

Крок 5. Перевірка чи було досягнуто останньої гілки, тобто максимального значення їх кількості. Якщо було досягнуто максимального значення перейти до кроку 7, інакше до кроку 6

Крок 6. Збільшення лічильника датчиків на 1.

Крок 7.Потяг не було знайдено, вивести інформацію про помилку в результаті пошуку.

Крок 8. Отримання даних про місцеперебування потягу, або помилку в результаті пошуку.

Крок 9. Завершення алгоритму.

2.4 Висновки до розділу

У другому розділі було розглянуто найпоширеніші способи взаємодії можливого пасажира з програмним засобом. Проаналізовано способи безпечного обміну даних та обрано асиметричний метод шифрування RSA. Даний метод найефективніший для збереження конфіденціальної інформації користувача у відкритій мережі інтернет.. На основі цих даних розроблено ескіз графічного інтерфейсу програмного засобу. Також розроблено формулу обрахування можливої затримки транспорту та створено базові алгоритми.

3 РОЗРОБКА ПРОГРАМНИХ КОМПОНЕНТІВ ДЛЯ ПРОГРАМНОГО ЗАСОБУ

3.1 Варіантний аналіз і обґрунтування вибору мови програмування

Для того щоб мати можливість просто та якісно розробити програмний додаток потрібно підібрати мову програмування яка найкраще підходить для поставленої задачі, але також потрібно не забувати про те що мова має бути зручною для використання. Однією із таких мов є Tcl.

Tcl перекладається як командна мова інструментів [14]. Tcl — це насправді мова сценаріїв, а також інтерпретатор для цієї мови, який легко вбудувати у програму. Tcl і пов’язаний з ним набір інструментів X windows, Tk, були розроблені та виготовлені професором Джоном Оустерхаутом з каліфорнійського університету в Берклі. Інтерпретатор було перенесено з UNIX в середовища DOS і Macin tosh. Як мова сценаріїв, Tcl подібний до інших мов оболонки UNIX, таких як Bourne Shell, C Shell, Korn Shell і Perl. Програми оболонки дозволяють виконувати інші програми. Вони забезпечують достатню програмованість, щоб могжна було створювати складні сценарії, які об’єднують існуючі програми в новий інструмент, пристосований до потреб розробника. Оболонки чудово підходять для автоматизації рутинних справ. Це можливість легко додати інтерпретатор Tcl до програми, яка виділяє його серед інших оболонок. Tcl виконує роль мови розширення, яка використовується для налаштування програм. Немає потреби винаходити командну мову для нового додатка або намагатися надати якусь користувальницьку програму для інструменту. Замість цього, додавши інтерпретатор Tcl, рекомендується структурувати програму як набір примітивних операцій, які можуть бути складені за допомогою сценарію, щоб найкращим чином задовольнити потреби користувачів. Це також дозволяє програмно контролювати програму іншими програмами, що призводить до наборів програм, які добре працюють разом.

Також корисною може бути мова Clojure [15]. Це невелика мова, основними цілями якої є простота і коректність. Як функціональна мова, вона підкреслює незмінність і декларативне програмування. Веб-розробка має багато мов на вибір і стільки ж думок щодо того, що робить мову «хорошою». Деякі мови прості, але багатослівні. Хоча насправді в цьому немає суті. Справжнє питання полягає не в тому, чи можна щось висловити в принципі, а те наскільки добре мова відповідає проблемі, що вирішується. Одна мова дозволяє мислити в термінах визначеної проблеми, а інша змушує перекладати проблему на її конструкції. Останнє часто стомлює і рідко приносить задоволення. У підсумку доводиться писати багато шаблонного коду і постійно повторюватися. Досить іронічно писати повторюваний код. З іншого боку, деякі мови є лаконічними, оскільки надають багато різних інструментів для вирішення проблем. На жаль, цей величезний набір інструментів створює різні проблеми. Чим більше функцій має мова, тим більше речей потрібно тримати в голові, щоб ефективно працювати з мовою. З часом виявляється, що постійно витрачаються накладні витрати на роздуми про всі різні функції та про те, як вони взаємодіють одна з одною. Справді важливо, чи можна використовувати мову, не замислюючись про це. Коли мові не вистачає виразності, приходить гостре усвідомлення, що код зовсім неприглядний. З іншого боку, коли мова має занадто багато функцій, вона може здаватися перевантаженою, і легко спантеличувати тою великою кількістю різноманітних способів вирішення завдання. Доцільно провести аналогію з математикою, розуміння кількох фундаментальних теорем та їх наслідків набагато корисніше, ніж запам’ятовування конкретних формул. Ось тут на допомогу приходить Clojure. Він дозволяє легко отримати рішення певної проблеми з невеликого набору загальних шаблонів. Все, що потрібно, щоб стати продуктивним, це вивчити кілька простих понять і трохи синтаксису. Потім ці концепції можна комбінувати безліччю способів для вирішення всіляких проблем.

Clojure може похвалитися десятками тисяч користувачів у сотнях компаній, він використовується в широкому діапазоні умов, включаючи банки та лікарні. Clojure, ймовірно, є найпопулярнішим сьогодні діалектом Lisp для початку нових розробок. Він зарекомендував себе в серйозних виробничих системах, і відгуки були переважно позитивними. Оскільки веб-розробка є одним з основних доменів для використання Clojure, кілька популярних бібліотек і фреймворків виникли й дозріли в цій області. Багато популярних платформ змушують йти на компроміс. Деяким платформам не вистачає продуктивності, іншим потрібно багато шаблонів, а іншим не вистачає інфраструктури, необхідної для реальних додатків. Clojure вирішує питання продуктивності та інфраструктури. Працюючи з Clojure, не доведеться турбуватися про обмеження часу виконання, коли програма буде рости. Крім того, Clojure не обмежується лише сервером. Її можна скомпілювати в Java Script, який відповідає популярним інтерфейсним фреймворкам. Більшість веб-платформ також вимагає вивчення та написання JavaScript. У Clojure є загальний набір інструментів як для клієнта, так і для сервера. Найпоширенішим способом роботи з шаблоном у програмних засобах є використання фреймворку. Приклади включають Ruby on Rails, Django та Spring. Фреймворки надають функціональні можливості, необхідні для створення сучасного сайту. Переваги, які пропонують ці фреймворки, також пов’язані з притаманними витратами. Оскільки багато операцій виконуються неявно, потрібно запам’ятовувати, які наслідки може мати будь-яка дія. Ця непрозорість ускладнює міркування написаного коду. Коли потрібно зробити щось, що суперечить дизайну фреймворка, це може швидко стати незручним і складним. Можливо, доведеться глибоко зануритися у внутрішню частину цього фреймворка та розібратися з очікуваною поведінкою. Замість використання фреймворків, Clojure робить доступними ряд потужних бібліотек, і можливо об’єднати ці бібліотеки так, щоб це було доцільно для конкретного проекту.

Не менш цікавою є мова програмування Go [16]. Вона була задумана у вересні 2007 року Робертом Ґрісом Емером, Робом Пайком та Кеном Томпсом, усі працювали в Google, і було оголошено в листопаді 2009 року. Цілі мови та супутніх інструментів повинні були бути виразними, ефективними як у компіляції, так і при виконанні, а також ефективними у написанні надійних програм. Go має зовнішню подібність до C і, як і C, є інструментом для професійних програмістів, що досягає максимального ефекту за допомогою мінімальних засобів. Але це набагато більше, ніж оновлена версія C. Вона запозичує рядки та адаптує гарну ідею із багатьох інших мов, уникаючи функцій, які призвели до складності та ненадійного коду. Її засоби для паралельного використання є новими та ефективними, а підхід до абстракції та об’єктно-орієнтованого програмування надзвичайно гнучкий. Вона має автоматичне керування пам’яттю. Go особливо добре підходить для побудови інфраструктури, наприклад мережевих серверів, а також інструментів і систем для розробників програм, але це справді загальна мова, яка знайшла застосування в різноманітних областях. Наприклад: графіка, мобільні додатки та машинне навчання. Вона стала популярно як заміна для нетипових скриптових мов програмування, оскільки врівноважує виразність із безпекою: програми Go зазвичай працюють швидше, ніж програми, написані на динамічних мовах, і зазнають набагато менше збоїв через неочікувані типичні помилки. Go — це відкритий проект, тому вихідний код для його компілятора, бібліотек та інструментів є вільно доступним будь-кому. Внесок у проект надходить від активної світової спільноти. Go працює на Unix-подібних системах — Linux, Free eBSD, OpenBSD, Mac OS X — і на Plan9 та Microsoft Windows. Програми, написані в одному з цих середовищ, як правило, працюють без змін на інших. Як і біологічні види, успішні мови народжують потомство, яке включає переваги своїх предків; схрещення інколи призводить до дивовижних сильних сторін; і дуже інколи без прецеденту виникає радикальна нова особливість. Інколи Go описують як "C-подібну мову" або "C для 21-го

століття". Від C, Go успадкувала виразний синтаксис, оператори керування потоком, основні типи даних, передачу параметрів обчислення за значенням, вказівники і, перш за все, C акцентує увагу на програмах, які компілюються до ефективного машинного коду і природно взаємодіють з абстракціями поточних операційних систем. Але в родинному древі Go є й інші предки. Частково взято риси від мов Ніклауса Вірта, починаючи з Паскаля. Modula-2 надихнула концепцію пакета. Oberon усунув відмінність між файлами інтерфейсу модуля та файлами реалізації модуля. Oberon-2 впливає на синтаксис для пакетів, імпорту та оновлень, а ObjectOberon забезпечує синтаксис для оновлень методів. Також у її родоводі присутні предки якиі роблять Go відмінною серед останніх мов програмування, це послідовність невеликих дослідницьких мов, розроблених у Bell Labs, і всі вони натхненні концепцією передачі послідовних процесів (CSP) із основоположної статті Тоні Хоара 1978 року про основи синхронізації. У CSP програма є паралельним складанням процесів, які не мають загального стану, процеси зв’язуються та синхронізуються за допомогою каналів. Але CSP Хоара був офіційною мовою для опису основних концепцій паралельності, а не мовою програмування для написання програм, які можна виконувати. Роб Пайк та інші почали експериментувати з реалізацією CSP як фактичної мови. Перша була названою Squeak («Мова спілкування з мишами»), яка надає мову для обробки подій миші та клавіатури, зі статично створеними каналами. За цим слідував Newsqueak, який запропонував синтаксис операторів і виразів, схожих на C, а також відмову від типу Pascal. Це була суто функціональна мова зі збором сміття, знову ж таки спрямована на керування подіями клавіатури, миші та вікон. Канали стали першими значеннями класу, які динамічно створювалися і зберігалися у змінних. Операційна система Plan9 запропонувала ці ідеї як передові на мові, що називається Alef. Alef намагався зробити Newsqueak життєздатною мовою системного програмування, але його відсутність збирання сміття зробило паралельність надто болючою. Інші її конструкції в Go показують вплив генів, що не є предками, то тут, то там.

Наприклад, частина вільна від APL, а лексична область з вкладеними функціями зі схеми. Тут також можливо знайти нові мутації. Інноваційні фрагменти Go забезпечують динамічні масиви з ефективним випадковим доступом, але також дозволяють розповсюджувати різноманітні елементи, що нагадують зв’язані списки. А оператор відкладення є новим у Go.

Також серед скриптових мов програмування досить сильно виділяється JavaScript. Ця мова більш відома і частіше використовується програмістами для створення програмних засобів. JavaScript спочатку був створений, щоб "зробити веб-сторінки живими" [17]. Програми на цій мові називаються скриптами. Вони можуть бути записані прямо в HTML веб-сторінки і запускатися автоматично, коли сторінка завантажується. Скрипти надаються і виконуються у вигляді звичайного тексту. Вони не потребують спеціальної підготовки або компіляції для запуску. У цьому аспекті JavaScript сильно відрізняється від іншої мови під назвою Java. Коли був створений JavaScript, він спочатку мав іншу назву: «LiveScript». Але Java була дуже популярна в той час, тому було вирішено, що позиціонування нової мови як «молодшого брата» Java допоможе. Але з розвитком JavaScript став повністю незалежною мовою зі своєю власною специфікацією під назвою ECMAScript, і тепер він взагалі не має ніякого відношення до Java. Сьогодні JavaScript може виконуватися не тільки в браузері, але і на сервері, або фактично на будь-якому пристрої, який має спеціальну програму під назвою движок JavaScript. Браузер має вбудований движок, який іноді називають «віртуальною машиною JavaScript».

В результаті детального аналізу характеристик мов програмування було створено таблицю 3.1.

Опираючись на дані таблиці легко зробити висновок, що мова програмування JavaScript з усіх розглянутих підходить найкраще. Вона задовільняє усі потреби що будуть виникати в процесі розробки програмного засобу для маршрутизації та розкладу з використанням геолокації. JavaScript має обширну і доступну документацію, є досить популярною серед інших

розробників що дає можливість проконсультуватись з іншим розробником який процює в цій сфері, а також можливість ефективно працювати із базами даних.

Таблиця 3.1 – Порівняння мов програмування

Критерій	Tcl	Clojure	Go	JavaScript
Простота синтаксису	0,5	1	0,5	1
Наявність документації	0,5	0,5	1	1
Ефективність роботи із базами даних	0,5	0,5	1	1
Різноманітність наявних фреймворків	1	0	0	1
Підсумковий результат	2,5	2	2,5	4

3.2 Вибір середовища розробки та СУБД

Окрім вибору мови програмування також досить важливим є вибір інтегрованого середовища для розробки програми. Інтегроване середовище розробки (IDE) - це програмне забезпечення для створення додатків, яке об'єднує загальні інструменти розробника в єдиний графічний інтерфейс користувача (GUI) [18]. IDE зазвичай складається з:

1. Редактор вихідного коду: Текстовий редактор, який може допомогти в написанні програмного коду з такими функціями, як підсвічування синтаксису візуальними сигналами, забезпечення автоматичного завершення конкретної мови та перевірка на наявність помилок як коду.

2. Локальна автоматизація побудови: Утиліти, які автоматизують прості, повторювані завдання в рамках створення локальної збірки програмного забезпечення для використання розробником, такі як компіляція вихідного коду комп'ютера в двійковий код, упаковка двійкового коду та запуск автоматизованих тестів.

3. Налагоджувач: програма для тестування інших програм, які можуть графічно відображати розташування помилки в оригінальному коді.

Інтегроване середовище розробки (Integrated Development Environment, IDE) - це програмне забезпечення, яке надає розробникам усі необхідні інструменти для створення програмного коду. IDE зазвичай містить текстовий редактор, компілятор або інтерпретатор, дебагер, інструменти для автоматизації рутинних задач (наприклад, автодоповнення коду, перевірка помилок) і інші корисні функції.

IDE дозволяє розробникам швидко запускати програмування нових програм, оскільки кілька утиліт не потрібно налаштовувати та інтегрувати вручну як частину процесу налаштування. Розробникам також не потрібно витрачати години на індивідуальне навчання використанню різних інструментів, коли кожна утиліта представлена в одному робочому місці. Це може бути особливо корисно для адаптації нових розробників, які можуть покладатися на IDE, щоб прискорити стандартні інструменти та робочі процеси команди. Насправді, більшість функцій ВПО призначені для економії часу, таких як інтелектуальне заповнення коду та автоматичне створення коду, що усуває необхідність введення повних послідовностей символів.

Для розробки програмного засобу було обрано Visual Studio Code. Це редактор коду, створений компанією «Microsoft», багато в чому схожий із середовищем розробки PyCharm [19]. Серед головних можливостей можна виділити відлагодження, автоматичне підсвічування синтаксису, рефакторинг та інтеграцію Git. Має вбудований термінал, регулярно оновлюється, швидкий в роботі та є універсальним. У Visual Studio Code підтримується велика кількість різноманітних мов програмування, серед яких є і Python. Дане середовище розробки є безкоштовним, доступне на операційних системах Windows, Mac OS X та Linux. Приклад інтерфейсу наведено на рисунку 3.1.

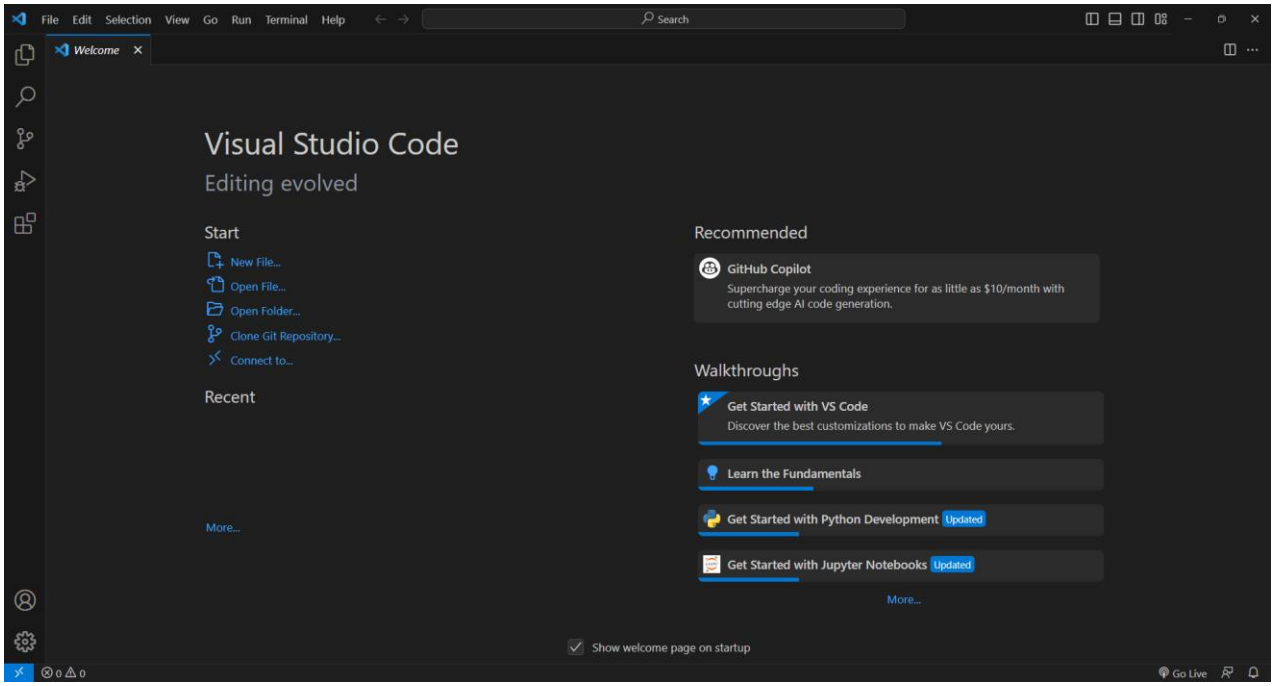


Рисунок 3.1 – Приклад інтерфейсу Visual Studio Code

При виборі СУБД було обрано реляційну модель через її відносну простоту в розумінні принципу роботи. Реляційні системи управління базами даних (RDBMS) - це системи, засновані на теорії множин, реалізовані у вигляді двовимірних таблиць з рядками і стовпцями[20]. Канонічним засобом взаємодії з ОБДБД є написання запитів структурованою мовою запитів (SQL). Значення даних наводяться і можуть бути числовими, рядками, датами, неінтерпретованими краплями або іншими типами. Типи застосовуються системою. Важливо відзначити, що таблиці можуть з'єднуватися і перетворюватися в нові, більш складні таблиці через їх математичну основу в реляційній (заданій) теорії. Є багато реляційних баз даних з відкритим вихідним кодом на вибір, включаючи MySQL, H2, HSQLDB, SQLite, PostgreSQL та багато інших.

Серед наявних баз даних було обрано PostgreSQL. Загартований у битві PostgreSQL на сьогоднішній день є найстарішою та найнадійнішою базою даних. Завдяки своєму дотриманню стандарту SQL, він буде відчувати себе знайомим кожному, хто працював з реляційними базами даних раніше, і він

забезпечує тверду точку порівняння з іншими базами даних, з якими ми будемо працювати.

3.3 Програмна реалізація основних модулів

Для реалізації роботи програмного засобу маршрутизації та розкладу з використанням датчиків та трекерів потрібно розробити модуль який буде знаходити координати транспортного засобу та обробляти дані для подальшого обрахунку затримки.

Початок. Задаємо вхідні дані. Перевірка лічильника. Визначаємо, скільки зупинок вже пройшов транспортний засіб.

```
// Початок.
// Спочатку задамо дані: лічильник зупинок, відстань до потягу, відстань до зупинки, час прибуття за розкладом.
let stopCounter = 5; // Лічильник зупинок, які проїхав транспортний засіб.
let distanceToTrain = 300; // Відстань по маршруту до потягу.
let distanceToStop = 150; // Відстань до зупинки.
let scheduledArrivalTime = '15:00'; // Час прибуття за розкладом.
```

Отримання даних про місцезнаходження зупинки. Здійснюємо запит в базу даних розкладу для отримання інформації про зупинку.

```
// Отримання даних про місцезнаходження зупинки із бази даних розкладу.
let stopLocationData = getStopLocationFromDatabase(); // Функція для отримання даних про місцезнаходження зупинки.
```

Отримання даних місцезнаходження потягу. Визначаємо відстані до потягу та до зупинки на основі вхідних даних.

```
// Визначення відстані від транспортного засобу до зупинки використовуючи відстань по маршруту.
```

```
let distanceToStopFromVehicle = distanceToTrain - distanceToStop;
```

Обрахування прогнозованого часу прибуття. Виконуємо розрахунок часу прибуття, використовуючи вагові коефіцієнти.

```
// Обрахунок прогнозованого часу прибуття транспорту з урахуванням усіх вагових коефіцієнтів.
let estimatedArrivalTime = calculateEstimatedArrivalTime(distanceToStopFromVehicle);
```

Обрахування прогнозованого часу прибуття. Виконуємо розрахунок часу прибуття, використовуючи вагові коефіцієнти.

```
// Отримання часу прибуття згідно з розкладом в базі даних та створення прапорця запізнення.
let delayFlag = false;
if (scheduledArrivalTime < estimatedArrivalTime) {
    scheduledArrivalTime = estimatedArrivalTime;
    delayFlag = true;
}
```

Виведення результатів. Виводимо значення часу прибуття за розкладом та прапорець запізнення.

```
// Виведення значення часу прибуття за розкладом разом з прапорцем запізнення.
console.log("Час прибуття за розкладом:", scheduledArrivalTime, "Прапорець запізнення:", delayFlag);

// Кінець.
```

Було реалізовано також графічний інтерфейс з використанням Використання VisualViewport API під час розробки програмного засобу. Це

надало можливість отримувати актуальні дані про видиму частину веб-сторінки. Це API дозволяє реагувати на зміни розміру вікна браузера та прокрутку, надаючи цінну інформацію про розмір та положення видимої області. В результаті це створило основу для більш адаптивного та відзивчивого дизайну, оскільки дозволило динамічно пристосовувати розміщення елементів на сторінці відповідно до положення користувача.

Наприклад, під час розробки реагуючого інтерфейсу, VisualViewport API дозволяє змінювати розміщення та параметри відображення елементів в залежності від розміру видимої області. Така можливість була особливо корисною для створення адаптивного дизайну, де елементи мали реагувати на різні розміри екранів користувачів. При використанні VisualViewport API відбувалася динамічна адаптація розміщення та стилів, забезпечуючи оптимальне відображення контенту при будь-яких умовах перегляду.

Приклад роботи додатку для ПК наведено на рисунку 3.2.

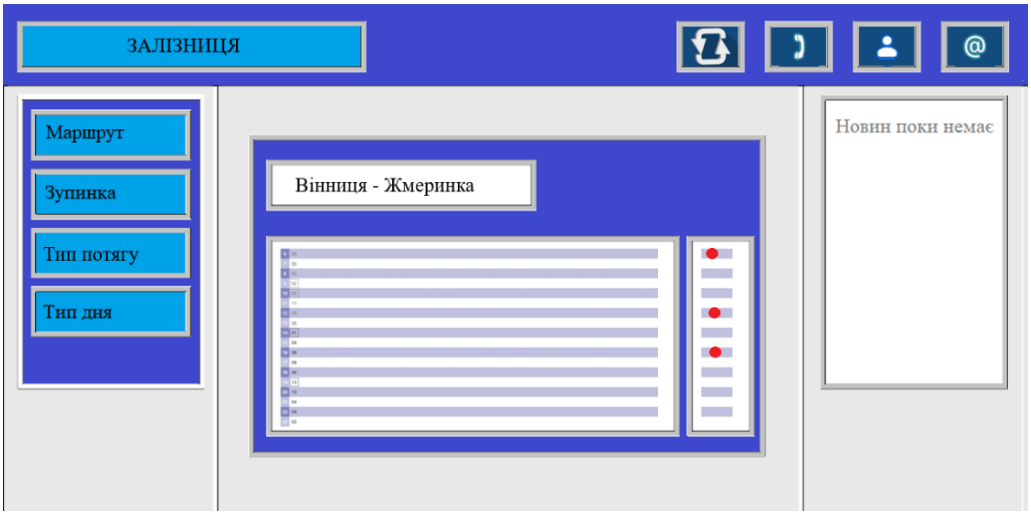


Рисунок 3.2 – Приклад інтерфейсу для ПК

Приклад роботи додатку для телефону наведено на рисунку3.3.

Рисунок 3.3 – Приклад інтерфейсу для телефону

Приклад роботи додатку для планшету наведено на рисунку3.4.

Рисунок 3.4 – Приклад інтерфейсу для планшету

3.4 Висновки до розділу

У розділі було детально розглянуто обрану мову програмування JavaScript. Наявна мова програмування повністю задовільняє усі потреби обраної розробки. Також в результаті даного аналізу було прийнято рішення використовувати Visual Studio Code та СУБД PostgreSQL для розробки програмного засобу маршрутизації та розкладу з використанням датчиків.

4 ТЕСТУВАННЯ ПРОГРАМИ

4.1 Аналіз методів тестування програмного забезпечення

Тестування програмного забезпечення є не лише процесом пошуку помилок, але й стратегічним етапом у розробці програми, що відіграє ключову роль у забезпеченні якості, надійності та відповідності функціональності задокументованим вимогам.

Важливість тестування:

1. Надійність та стабільність: Тестування допомагає виявляти помилки, недоліки і дефекти програми, що можуть призвести до аварійного виконання чи неправильної роботи програми.

2. Відповідність вимогам: Тестування перевіряє, чи відповідає програма функціональним та нефункціональним вимогам, які були визначені на етапі розробки.

3. Покращення якості: Виявлення та усунення дефектів на ранніх етапах дозволяє підвищити якість програми та зменшити ймовірність виходу на ринок з проблемами.

4. Вчасна виправка помилок: Раннє виявлення помилок зменшує витрати на виправлення та поліпшення програми, оскільки виправлення дефектів на пізніших етапах може бути витратним та складним.

5. Взаємодія з користувачем: Правильно проведене тестування допомагає покращити взаємодію з користувачем та забезпечити високий рівень задоволення від використання програми.

Типи тестування:

1. Функціональне тестування: Перевірка функціональності програми відповідно до специфікацій.

2. Навантажувальне тестування: Оцінка продуктивності програми при високому навантаженні.

3. Автоматизоване тестування: Використання програмних скриптів для автоматизації тестів.

4. Тестування на різних платформах та пристроях: Перевірка сумісності програми з різними пристроями та платформами.

Функціональне тестування - це процес перевірки програми на відповідність функціональним вимогам. Його мета полягає у перевірці роботи програми з точки зору функціональності, яка була специфікована на етапі розробки. Під час функціонального тестування визначається, чи виконує програма очікувані функції, чи працює коректно та чи відповідає вимогам користувача.

Тестування функціональності полягає в створенні тестових сценаріїв, щоб перевірити, як програма реагує на певні вхідні дані. Це включає в себе введення різних значень та очікуваних результатів, відповідно до специфікацій та очікуваних результатів роботи програми. Основна мета - переконатися, що програма виконує функції, які були обіцяні у специфікаціях та вимогах користувача, і робить це безпомилково та ефективно.

У процесі функціонального тестування експерти здійснюють тестування всіх можливих функцій програми, перевіряючи, чи працюють вони так, як очікується. Основні кроки цього виду тестування включають ініціювання різних вхідних даних, перевірку результатів та порівняння їх із сподіваними значеннями. Такий підхід допомагає виявити невідповідності функцій програми до вимог та уникнути потенційних помилок у функціональності.

Навантажувальне тестування — це процес визначення та оцінки продуктивності програми або системи за умов великого обсягу робочих навантажень. Основна мета полягає у визначенні максимальних обсягів робочих навантажень, при яких програма чи система здатна функціонувати без збоїв.

Під час навантажувального тестування створюються тестові сценарії, які симулюють велику кількість користувачів або транзакцій, щоб перевірити

реакцію системи на таке навантаження. Це може включати симуляцію одночасних запитів, обробку великого обсягу даних чи транзакцій, а також визначення максимальних меж її функціональності.

Мета такого тестування полягає у виявленні потенційних слабких місць програми або системи при великому обсязі робочих навантажень. Це дозволяє виявити можливі проблеми з продуктивністю, швидкістю реакції, ресурсами системи та оптимізацією роботи в умовах великої активності.

Такий вид тестування допомагає визначити обсяги, які система може обробляти без втрати продуктивності чи виникнення помилок, що дозволяє розробникам та інженерам забезпечити оптимальну роботу системи під великими навантаженнями.

Автоматизоване тестування - це процес використання програмних засобів для виконання тестів та перевірки програми без прямого втручання користувача. Це означає, що тестові сценарії та їх виконання контролюються програмними скриптами або інструментами, які автоматизують процес тестування.

Під час автоматизованого тестування розробляються тестові скрипти, які можуть відтворювати дії користувача чи виконувати тестові сценарії. Ці скрипти роблять все те, що й користувач, але автоматично, що дозволяє ефективно виконувати тести без необхідності ручного втручання.

Головна перевага автоматизованого тестування - це здатність швидко виконувати тести та повторювати їх при необхідності. Це допомагає виявляти помилки швидше та ефективніше, зменшуючи витрати часу та ресурсів. Також цей підхід забезпечує більшу точність, оскільки тести виконуються однаково кожен раз.

Автоматизоване тестування дозволяє покращити процес розробки, сприяючи забезпеченню якості програмного забезпечення шляхом швидкого виявлення помилок та покращення ефективності розробки.

Тестування на різних платформах та пристроях - це процес перевірки програми чи веб-сайту на сумісність та коректну роботу на різних операційних системах, пристроях та браузерах безпосередньо відповідно до їх характеристик та вимог користувачів.

Цей вид тестування спрямований на переконання, що програма чи веб-сайт працюють однаково ефективно та коректно на різних пристроях, таких як комп'ютери, смартфони, планшети тощо, і на різних операційних системах, таких як Windows, macOS, Android, iOS та інші.

Це важливо для забезпечення однакової якості користувацького досвіду на будь-якому пристрої чи платформі. Тестування на різних пристроях та платформах допомагає виявляти та усувати можливі проблеми з адаптацією до різних розмірів екранів, роздільної здатності, особливостей операційних систем та характеристик пристроїв. Такий підхід дозволяє забезпечити позитивний користувацький досвід для всіх користувачів, незалежно від їхнього пристрою чи платформи, яку вони використовують.

Тестування ПЗ є фундаментальним етапом у створенні високоякісного програмного забезпечення, оскільки воно не лише забезпечує виявлення помилок, а й сприяє поліпшенню продукту та його взаємодії з користувачем.

Звичайні види тестування мають різні цілі та фокусуються на різних аспектах якості програмного забезпечення.

Функціональне тестування зосереджене на перевірці функціональності програми, відповідності її функцій вимогам та коректності роботи відповідно до специфікацій. Навантажувальне тестування оцінює продуктивність та поведінку системи під великими обсягами даних чи користувачів. Автоматизоване тестування використовує програмні засоби для виконання тестів без прямого втручання користувача. Тестування на різних платформах перевіряє сумісність та працездатність програми на різних пристроях та операційних системах.

Кожен з цих типів тестування має свої унікальні особливості та застосовується для досягнення певних цілей у забезпеченні якості та надійності програмного забезпечення.

Існують різні методи, які можуть бути використані для тестування програмного забезпечення. Тестування програмного забезпечення – процес технологічного дослідження, призначений для визначення інформації про якість програмного продукту [21].

Методика тестування без будь-яких знань про внутрішню роботу програми називається тестуванням чорної скриньки. Тестер не звертає уваги на архітектуру системи і не має доступу до вихідного коду. Як правило, під час виконання тесту чорної скриньки тестер буде взаємодіяти з інтерфейсом користувача системи, надаючи вхідні дані та вивчаючи виходи, не знаючи, як і де обробляються входи.

Переваги тестування «чорної скриньки»:

- добре підходить і ефективна для великих сегментів коду;
- доступ до коду не потрібен;
- чітко відокремлює точку зору користувача від точки зору розробника через чітко визначені ролі;
- велика кількість помірно кваліфікованих тестувальників можуть протестувати додаток без знання реалізації, мови програмування або операційних систем.

Недоліки тестування «чорної скриньки»:

- обмежене охоплення, оскільки насправді виконується лише вибрана кількість тестових сценаріїв;
- неефективне тестування, пов'язане з тим, що тестувальник має лише обмежені знання про додаток;
- сліпе покриття, оскільки тестер не може націлюватися на певні сегменти коду або області помилок;
- тестові кейси важко спроектувати.

Тестування «білої скриньки» - це детальне дослідження внутрішньої логіки і структури коду. Тестування білої скриньки також називається тестуванням скла або тестуванням з відкритими коробками [22]. Для того, щоб виконати тестування білої скриньки на додаток, тестувальник повинен знати внутрішню роботу коду.

Переваги тестування «білої скриньки»:

- оскільки тестер має знання вихідного коду, стає дуже легко з'ясувати, який тип даних може допомогти в ефективному тестуванні програми;
- це допомагає оптимізувати код;
- додаткові рядки коду можуть бути видалені, які можуть призвести до прихованих дефектів;
- завдяки знанням тестувальника про код, максимальне охоплення досягається під час написання тестового сценарію.

Недоліки тестування «білої скриньки»:

- у зв'язку з тим, що для проведення тестування білої скриньки потрібен кваліфікований тестувальник, витрати збільшуються;
- іноді неможливо заглянути в кожен куточок і куточок, щоб з'ясувати приховані помилки, які можуть створити проблеми, так як багато шляхів залишаться неперевіреними;
- важко підтримувати тестування білої коробки, оскільки для цього потрібні спеціалізовані інструменти, такі як аналізатори коду та інструменти налагодження.

Тестування «сірої скриньки» - це метод тестування програми з обмеженими знаннями внутрішньої роботи програми. У тестуванні програмного забезпечення фраза чим більше ви знаєте, тим краще несе велику вагу під час тестування програми.

Освоєння домену системи завжди дає тестувальнику перевагу над кимось з обмеженими знаннями домену. На відміну від тестування чорних ящиків, де тестер тестує лише інтерфейс користувача програми, при тестуванні сірої коробки тестер має доступ до проектної документації та бази даних. Маючи ці знання, тестер може підготувати кращі тестові дані та сценарії тестування під час складання плану тестування.

Переваги тестування «сірої скриньки»:

- пропонує комбіновані переваги тестування чорної і білої скриньки, де це можливо;
- тестувальники не покладаються на вихідний код, замість цього вони покладаються на визначення інтерфейсу та функціональні специфікації;
- ґрунтуючись на обмеженій доступній інформації, тестер сірої коробки може розробляти відмінні сценарії тестування, особливо навколо протоколів зв'язку та обробки типів даних;
- тест проводиться з точки зору користувача, а не дизайнера.

Недоліки тестування «сірої скриньки»:

- оскільки доступ до вихідного коду недоступний, можливість перейти по коду і тестовому покриттю обмежена;
- тести можуть бути зайвими, якщо розробник програмного забезпечення вже запустив тестовий випадок;
- тестування кожного можливого вхідного потоку нереалістичне, оскільки це займе необґрунтовану кількість часу; таким чином, багато шляхів програми залишаться неперевіреними.

У таблиці 4.1 наведено пункти, які відрізняють тестування чорних ящиків, тестування сірого ящика та тестування білих ящиків.

Таблиця 4.1 – Порівняння методів тестування.

Тестування чорної скриньки	Тестування сірої скриньки	Тестування білої скриньки
Внутрішня робота програми не повинна бути відома	Тестувальник має обмежені знання про внутрішню роботу програми	Тестер володіє повними знаннями про внутрішню роботу програми
Також відоме як тестування з закритим ящиком, тестування на основі даних або функціональне тестування	Також відоме як напівпрозоре тестування, оскільки тестер має обмежені знання нутрощів програми	Також відоме як прозоре тестування, структурне тестування або тестування на основі коду
Виконується кінцевими користувачами, а також тестувальниками та розробниками	Виконується кінцевими користувачами, а також тестувальниками та розробниками	Зазвичай це роблять тестувальники та розробники
Тестування базується на зовнішніх очікуваннях - внутрішня поведінка програми невідома	Тестування проводиться на основі діаграм баз даних високого рівня і діаграм потоку даних	Внутрішня робота повністю відома, і тестер може відповідним чином розробляти тестові дані
Він вичерпний і найменш трудомісткий	Частково трудомісткий і вичерпний	Найбільш вичерпний і трудомісткий вид тестування
Не підходить для алгоритму тестування	Не підходить для алгоритму тестування	Підходить для алгоритму тестування

Шляхом аналізу переваг та недоліків методів тестування «чорної скриньки», «білої скриньки» та «сірої скриньки» для тестування програмного засобу «Приміські маршрути Вінниці» обрано метод «чорної скриньки».

4.2 Тестування розробленого програмного продукту

Додаток було протестовано для перевірки його коректної роботи. Для цього створено таблицю тестових випадків (табл. 4.2). Усі випадки пройдено успішно.

Таблиця 4.2 – Тестові випадки

Назва	Кроки	Очікуваний результат
Перевірка відображення елементів головного вікна	1. Завантажити програмний засіб, відкривши файл «Dodatok.html».	1. Відображається назва додатку. 2. Відображається меню вибору типу розкладу та панель швидкого доступу. 3. Відображається розклад. 4. Відображається панель новин.
Перевірка відкриття розкладу за маршрутами	1. Завантажити додаток, відкривши файл «Dodatok.html». 2. Натиснути на пункт меню «Маршрут» в області вибору типу розкладу 3. В області відображення розкладу обрати потрібний маршрут з викидного списку серед наявних та натиснути на нього	1. Обновляється область відображення розкладу. 2. В області відображення розкладу, при натисненні на маршрут зі списку, відкривається розклад, а також наступний список де можна вибрати тип потягу. 3. Після вибору потягу пропонується також вибрати тип дня. 4. При виборі типу дня повністю деталізований розклад відображається правильно.

Продовження таблиці 4.2

Назва	Кроки	Очікуваний результат
Перевірка відкриття розкладу за зупинкою	1. Завантажити додаток, відкривши файл «Dodatok.html». 2. Натиснути на пункт меню «Зупинка» в області вибору типу розкладу 3. В області відображення розкладу обрати потрібний напрямок з викидного списку серед наявних та натиснути на нього	1. Обновляється область відображення розкладу. 2. В області відображення розкладу, при натисненні на напрямок зі списку, відкривається розклад, а також наступний список де можна вибрати тип потягу. 3. Після вибору потягу пропонується також вибрати тип дня. 4. При виборі типу дня повністю деталізований розклад відображається правильно.
Перевірка відкриття області «Контакти»	1. Завантажити додаток, відкривши файл «Dodatok.html». 2. Натиснути на кнопку «Контакти»	1. Відкривається область «Контакти».
Перевірка відкриття області «Зворотній зв’язок»	1. Завантажити додаток, відкривши файл «Dodatok.html». 2. Натиснути на кнопку «Зворотній зв’язок»	1. Відкривається область «Зворотній зв’язок».
Перевірка завершення роботи програми	1. Завантажити додаток, відкривши файл «Dodatok.html». 2. Натиснути на пункт меню «Вихід» розділу «Файл»	1. Програма завершує свою роботу.
Перевірка відображення інформації в панелі новин	1. Завантажити додаток, відкривши файл «Dodatok.html». 2. Натиснути на будь-яку новину зі списку.	1. Додаткова інформація з’являється у вигляді поверхневого вікна.

4.3 Розробка інструкції користувача

Оскільки «Залізниця» є саме програмним засобом, для його запуску потрібен встановлений браузер на персональному комп’ютері чи смартфоні. Потім у пошуковику ввести адресу сайту – «https://train_routes.com». Приклад зображено на рисунку 4.1.

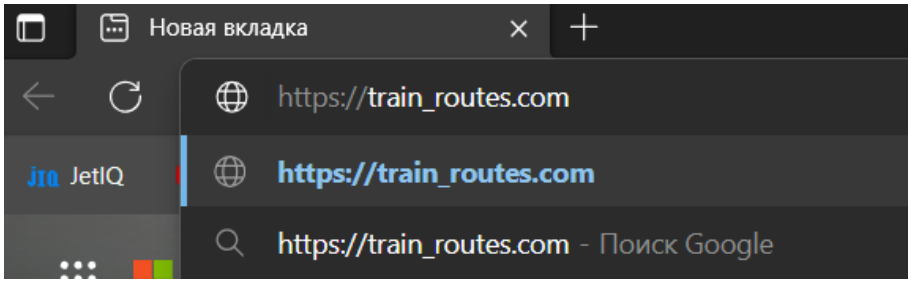


Рисунок 4.1 – Введення адреси сайту «https:// suburban_routes.com»

Після цього відкриється головне вікно сайту де буде запропоновано обрати розклад потягу за наявними критеріями, а саме за маршрутом, зупинкою, типом потягу чи дня. З’явиться викидний список з якого потім можна буде обрати потрібний варіант. Приклади зображено на рисунках 4.2 та 4.3

Рисунок 4.2 – Зображення першої сторінки

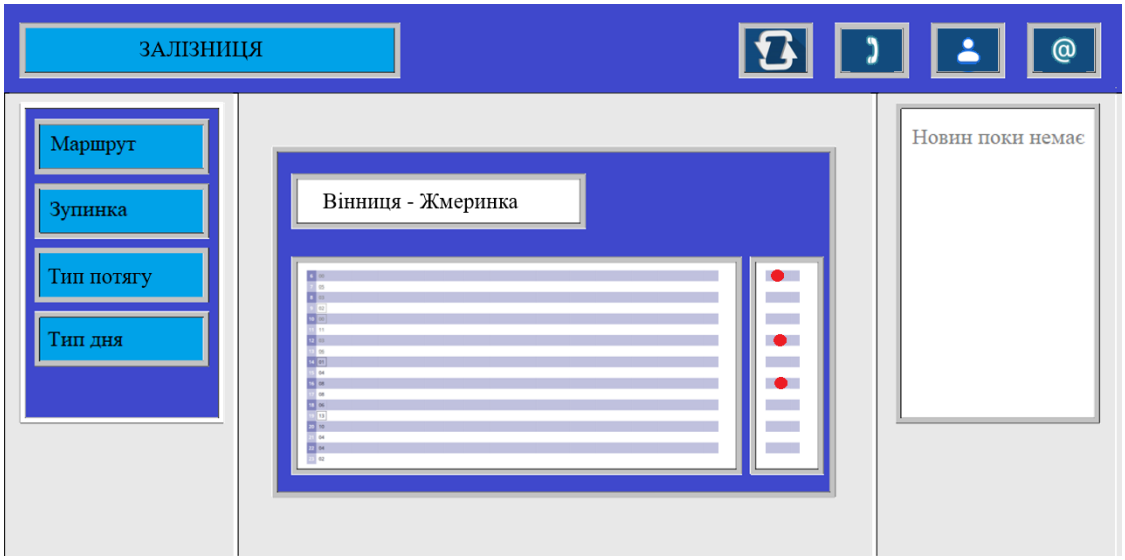


Рисунок 4.3 – Зображення результату вибору

Також реалізовано функцію зворотного зв’язку. Після натиснення відповідної кнопки зверху (рисунок 4.4) буде відкрито вкладку (рисунок 4.5) де потрібно ввести своє повідомлення а також електронну пошту для зв’язку.

Рисунок 4.4 – Зображення кнопки зворотного зв’язку

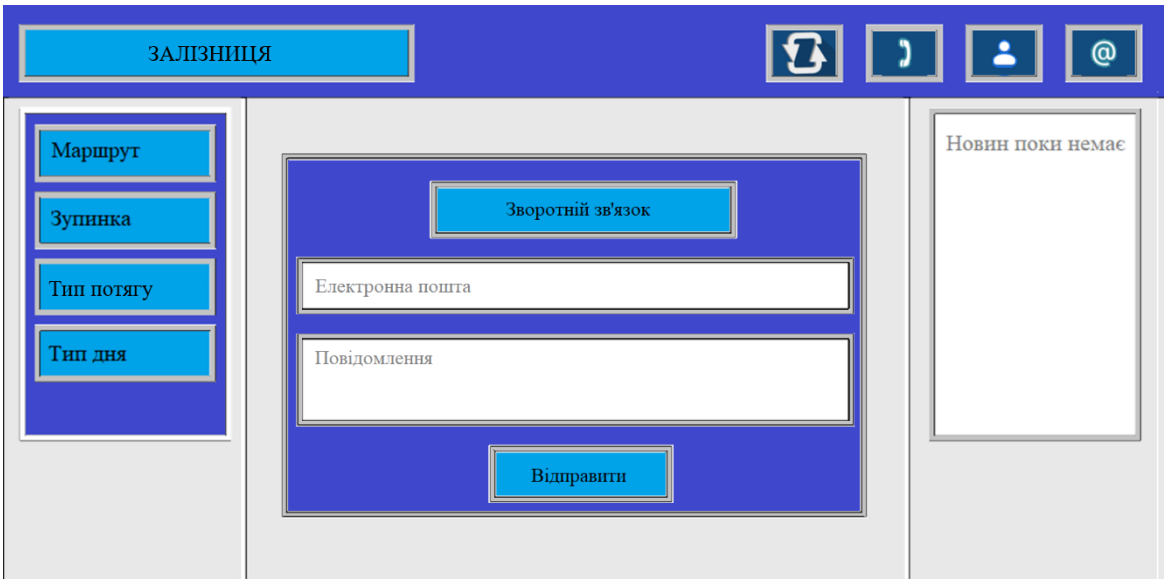


Рисунок 4.5 – Зображення вкладки зворотного зв’язку

У цьому підрозділі було розроблено інструкцію користувача в якій описані дії для повноцінного використання програмного засобу та варіанти взаємодії користувача із ним та його адміністрацією.

4.4 Вимоги до персонального комп’ютера

Розроблений додаток працює на базі браузера, а отже для його використання потрібно мати готовий встановлений браузер для клієнтського використання додатку. У наявності є багато різних версій для смартфонів які працюють на операційних системах типу Android та iOS. Також є браузери для персональних комп’ютерів які працюють на базі Windows, Mac OS X та Linux. Програмний засіб маршрутизації та розкладу існує у вигляді інтернет версії, а тому окрім браузера більше нічого завантажувати не потрібно. Хоч і системні вимоги браузерів між собою можуть сильно відрізнятись, але для використання програмного засобу «Приміські маршрути Вінниці» на базі операційної системи Windows було наведено мінімальні та рекомендовані конфігурації у таблицях 4.2 та 4.3.

Таблиця 4.1 – Мінімальна конфігурація

Тип процесора	32-розрядний (x86) або 64-розрядний (x64) процесор з тактовою частотою 1,8 ГГц
Об’єм оперативної пам’яті	2 ГБ для 32-розрядної системи і 3 ГБ для 64-розрядної системи
Місце на жорсткому диску	1 ГБ
Графічний пристрій	Інтегрований графічний пристрій, сумісний з DirectX9
Операційна система	Windows 7
Браузер	Підтримка HTML5

Таблиця 4.2 – Рекомендована конфігурація

Тип процесора	32-розрядний (x86) або 64-розрядний (x64) процесор з тактовою частотою 2,8 ГГц
Об’єм оперативної пам’яті	4 ГБ для 32-розрядної системи і 6 ГБ для 64-розрядної системи
Місце на жорсткому диску	1 ГБ
Графічний пристрій	Інтегрований графічний пристрій, сумісний з DirectX10
Операційна система	Windows 10
Браузер	Підтримка HTML5

4.5 Висновки до розділу

Для проведення тестування програмного засобу маршрутизації та розкладу з використанням датчиків та трекерів «Залізниця» було розглянуто декілька методів та обрано тестування методом «чорної скриньки». В результаті тестування доведено працездатність та відповідність поставленому завданню. Було розроблено інструкцію користувача, за допомогою якої користувач зможе ознайомитись з основними можливостями програмного засобу та розпочати повноцінне користування. Було визначено рекомендовані конфігурації персональних комп’ютерів на базі Windows для коректної та ефективної роботи програми.

5 ЕКОНОМІЧНА ЧАСТИНА

Науково-технічна розробка має право на існування та впровадження, якщо вона відповідає вимогам часу, як в напрямку науково-технічного прогресу та і в плані економіки. Тому для науково-дослідної роботи необхідно оцінювати економічну ефективність результатів виконаної роботи.

Магістерська кваліфікаційна робота за темою «Розробка методів і програмного засобу для ведення розкладу потягу залізниці із використанням датчиків та трекерів» відноситься до науково-технічних робіт, які орієнтовані на виведення на ринок (або рішення про виведення науково-технічної розробки на ринок може бути прийнято у процесі проведення самої роботи), тобто коли відбувається так звана комерціалізація науково-технічної розробки. Цей напрямок є пріоритетним, оскільки результатами розробки можуть користуватися інші споживачі, отримуючи при цьому певний економічний ефект. Але для цього потрібно знайти потенційного інвестора, який би взявся за реалізацію цього проекту і переконати його в економічній доцільності такого кроку.

Для наведеного випадку нами мають бути виконані такі етапи робіт:

1) проведено комерційний аудит науково-технічної розробки, тобто встановлення її науково-технічного рівня та комерційного потенціалу;

2) розраховано витрати на здійснення науково-технічної розробки;

3) розрахована економічна ефективність науково-технічної розробки у випадку її впровадження і комерціалізації потенційним інвестором і проведено обґрунтування економічної доцільності комерціалізації потенційним інвестором.

5.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Метою проведення комерційного і технологічного аудиту дослідження за темою «Розробка методів і програмного засобу для ведення розкладу потягів залізниці із використанням датчиків та трекерів» є оцінювання науково-технічного рівня та рівня комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням 5-ти бальної системи оцінювання за 12-ма критеріями, наведеними в табл. 5.1 .

Таблиця 5.1 – Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції					
	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено працездатність продукту в реальних умовах
Ринкові переваги (недоліки)					
	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
	Технічні та споживчі властивості продукту значно гірші, ніж в	Технічні та споживчі влас-тивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в	Технічні та споживчі властивості продукту значно кращі, ніж в
	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів

Продовження таблиці 5.1

Ринкові перспективи					
	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
	Активна конкуренція великих компаній на	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
0	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуютьс я у виробництві
1	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
2	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь- які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання науково-технічного рівня та комерційного потенціалу науково-технічної розробки потрібно звести до таблиці.

Таблиця 5.2 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами

Критерії	Експерт (ПІБ, посада)		
	1	2	3
	Бали:		
1. Технічна здійсненність концепції	3	3	4
2. Ринкові переваги (наявність аналогів)	4	3	4
3. Ринкові переваги (ціна продукту)	3	3	3
4. Ринкові переваги (технічні властивості)	3	3	3
5. Ринкові переваги (експлуатаційні витрати)	3	3	3
6. Ринкові перспективи (розмір ринку)	3	4	3
7. Ринкові перспективи (конкуренція)	3	3	4
8. Практична здійсненність (наявність фахівців)	3	3	3
9. Практична здійсненність (наявність фінансів)	3	3	3
10. Практична здійсненність (необхідність нових матеріалів)	3	3	3
11. Практична здійсненність (термін реалізації)	3	4	4
12. Практична здійсненність (розробка документів)	4	3	3
Сума балів	38	38	40
Середньоарифметична сума балів $СБ_c$	38,7		

За результатами розрахунків, наведених в таблиці 5.2, зробимо висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки. При цьому використаємо рекомендації, наведені в табл. 5.3.

Таблиця 5.3 – Науково-технічні рівні та комерційні потенціали розробки

Середньоарифметична сума балів СБ , розрахована на основі висновків експертів	Науково-технічний рівень та комерційний потенціал розробки
41…48	Високий
31…40	Вище середнього
21…30	Середній
11…20	Нижче середнього
0…10	Низький

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Розробка методів і програмного засобу для ведення розкладу потягів залізниці із використанням датчиків та трекерів» становить 38,7 балів, що, відповідно до таблиці 5.3, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вище середнього).

5.2 Розрахунок витрат на проведення науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної роботи на тему «Розробка методів і програмного засобу для ведення розкладу потягів залізниці із використанням датчиків та трекерів», під час планування, обліку і калькулювання собівартості науково-дослідної роботи групуємо за відповідними статтями.

5.2.1 Витрати на оплату праці

До статті «Витрати на оплату праці» належать витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп, науковим, інженерно-технічним працівникам, конструкторам, технологам, креслярам, копіювальникам, лаборантам, робітникам, студентам, аспірантам та іншим працівникам, безпосередньо зайнятим виконанням конкретної теми, обчисленої за посадовими окладами, відрядними розцінками, тарифними ставками згідно з чинними в організаціях системами оплати праці.

Основна заробітна плата дослідників

Витрати на основну заробітну плату дослідників ($З_о$) розраховуємо у відповідності до посадових окладів працівників, за формулою :

$$З_о = \sum_{i=1}^{k} \frac{М_{ni} \cdot t_i}{Т_р}, \quad (5.1)$$

де k – кількість посад дослідників залучених до процесу досліджень;

$М_{ni}$ – місячний посадовий оклад конкретного дослідника, грн;

t_i – число днів роботи конкретного дослідника, дні;

$Т_р$ – середнє число робочих днів в місяці, $Т_р$=21 день.

$$З_о = 25000{,}00 \cdot 60 / 21 = 71428{,}57 \text{ грн.}$$

Проведені розрахунки зведемо до таблиці.

Таблиця 5.4 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Керівник проекту	25000	1190,48	60	71428,57
Інженер-розробник програмного забезпечення	22000	1047,62	60	62857,14
Інженер-розробник програмного забезпечення	22000	1047,62	60	62857,14
Консультант	15000	714,28	30	21428,57
Всього				218571,43

Основна заробітна плата робітників

Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт НДР розраховуємо за формулою:

$$З_р = \sum_{i=1}^{n} С_i \cdot t_i, \quad (5.2)$$

де C_i – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i – час роботи робітника при виконанні визначеної роботи, год.

Витрати на оплату праці в погодинній тарифній ставці відносяться до сум, які сплачуються працівнику за фактично відпрацьований час. Ця система оплати базується на кількості годин або ділянок часу, протягом яких працівник працює.

Погодинна тарифна ставка дозволяє більш гнучко визначати оплату праці, особливо у випадках, коли робочий графік може коливатися. Ця система оплати зазвичай використовується для певних типів робіт або у випадках, коли потрібна гнучкість у визначенні часу роботи працівників.

Погодинну тарифну ставку робітника відповідного розряду C_i можна визначити за формулою:

$$C_i = \frac{M_M \cdot K_i \cdot K_c}{T_p \cdot t_{зм}}, \qquad (5.3)$$

де M_M – розмір прожиткового мінімуму працездатної особи, або мінімальної місячної заробітної плати (в залежності від діючого законодавства), приймемо $M_м$=6700,00 грн;

K_i – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду (табл. Б.2, додаток Б) ;

K_c – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об’єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати.

T_p – середнє число робочих днів в місяці, приблизно T_p = 22 дні;

$t_{зм}$ – тривалість зміни, год.

$C_1 = 6700{,}00 \cdot 1{,}10 \cdot 1{,}65 / (21 \cdot 8) = 72{,}38$ грн.

$З_{р1} = 72{,}38 \cdot 6 = 434{,}30$грн.

Таблиця 5.5 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Тарифний коефіцієнт	Погодинна тарифна ставка, грн	Величина оплати на робітника грн
Установка електронно-обчислювального обладнання	6	2	1,1	72,38	434,30
Підготовка робочого місця дослідника	4	2	1,1	72,38	289,54
Інсталяція програмного забезпечення	2,2	5	1,7	111,87	246,11
Монтаж серверного обладнання	10	5	1,7	111,87	1118,66
Адміністрування системи	10	2	1,1	72,38	1118,66
Всього					3207,27

Додаткова заробітна плата дослідників та робітників

Додаткову заробітну плату розраховуємо як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%}, \quad (5.4)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати. Приймемо 12%.

$З_{дод}$ = (218571,43+ 3207,27) · 12 / 100% = 26613,44 грн.

5.2.2 Відрахування на соціальні заходи

Нарахування на заробітну плату дослідників та робітників розраховуємо як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%} \quad (5.5)$$

де $Н_{зп}$ – норма нарахування на заробітну плату. Приймаємо 22%.

$З_н$ = (218571,43+ 3207,27+ 26613,44) · 22 / 100% = 54646,27 грн.

5.2.3 Сировина та матеріали

До статті «Сировина та матеріали» належать витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби і предмети праці, які придбані у сторонніх підприємств, установ і організацій та витрачені на проведення досліджень.

Витрати на матеріали (*М*), у вартісному вираженні розраховуються окремо по кожному виду матеріалів за формулою:

$$М = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j - \sum_{j=1}^{n} В_j \cdot Ц_{вj}, \quad (5.6)$$

де $Н_j$ – норма витрат матеріалу *j*-го найменування, кг;

n – кількість видів матеріалів;

$Ц_j$ – вартість матеріалу *j*-го найменування, грн/кг;

$К_j$ – коефіцієнт транспортних витрат, ($К_j$ = *1,1 … 1,15*);

$В_j$ – маса відходів *j*-го найменування, кг;

$Ц_{вj}$ – вартість відходів *j*-го найменування, грн/кг.

$М_1$ = 4 · 200,00 · 1,1 - 0,000 · 0,00 = 880 грн.

Проведені розрахунки зведемо до таблиці.

Таблиця 5.6 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за од	Норма витрат, од	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн
Офісний папір Plus A4-500-80	200	4	0	0	880
Папір для записів Parers Light A5	110	3	0	0	363
Органайзер офісний Office	210	1	0	0	231
Канцелярське приладдя (набір офісного працівника)	175	4	0	0	770
Картридж для принтера Canon LBP 2900	1100	1	0	0	1210
Flesh-пам'ять Kingston 32 GB	180	2	0	0	396
Всього					3850

5.2.4 Розрахунок витрат на комплектуючі

Витрати на комплектуючі *($К_в$)*, які використовують при проведенні НДР на тему «Розробка методів і програмного засобу для ведення розкладу потягів залізниці із використанням датчиків та трекерів» відсутні.

5.2.5 Спецустаткування для наукових (експериментальних) робіт

До статті «Спецустаткування для наукових (експериментальних) робіт» належать витрати на виготовлення та придбання спецустаткування необхідного для проведення досліджень, також витрати на їх проектування, виготовлення, транспортування, монтаж та встановлення.

Балансову вартість спецустаткування розраховуємо за формулою:

$$B_{спец} = \sum_{i=1}^{k} Ц_i \cdot C_{пр.i} \cdot K_i \quad , \qquad (5.7)$$

де $Ц_i$ – ціна придбання одиниці спецустаткування даного виду, марки, грн;

$C_{пр.i}$ –кількість одиниць устаткування відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує доставку, монтаж, налагодження устаткування тощо, ($K_i = 1,10…1,12$);

k – кількість найменувань устаткування.

$B_{спец} = 20000 \cdot 3 \cdot 1,1 = 66000$ грн.

Отримані результати зведемо до таблиці:

Таблиця 5.7 – Витрати на придбання спецустаткування по кожному виду

Найменування устаткування	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
ПК	3	20000	66000
Ноутбук	1	30000	33000
Роутер	1	2000	2200
Серверне обладнання	1	50000	55000
Всього			156200

5.2.6 Програмне забезпечення для наукових (експериментальних) робіт

До статті «Програмне забезпечення для наукових (експериментальних) робіт» належать витрати на розробку та придбання спеціальних програмних

засобів і програмного забезпечення, (програм, алгоритмів, баз даних) необхідних для проведення досліджень, також витрати на їх проектування, формування та встановлення.

Балансову вартість програмного забезпечення розраховуємо за формулою:

$$B_{прг} = \sum_{i=1}^{k} Ц_{iпрг} \cdot C_{прг.i} \cdot K_i \ , \qquad (5.8)$$

де $Ц_{iпрг}$ – ціна придбання одиниці програмного засобу даного виду, грн;

$C_{прг.i}$ – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, ($K_i = 1,10…1,12$);

k – кількість найменувань програмних засобів.

$B_{прг} = 10000 \cdot 4 \cdot 1,1 = 44800$ *грн.*

Отримані результати зведемо до таблиці:

Таблиця 5.7– Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
ОС Windows 11 Professional	4	10000	44800
Прикладний пакет Microsoft Office 2021 Pro	4	5000	22400
Visual Studio Professional 2022	1	2000	2240
Всього			69440

5.2.7 Амортизація обладнання, програмних засобів та приміщень

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо, розраховуємо з використанням прямолінійного методу амортизації за формулою:

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12}, \qquad (5.9)$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$T_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

$A_{обл}$ = (66000,00 · 3) / (3 · 12) = 5500грн.

Проведені розрахунки зведемо до таблиці.

Таблиця 5.8 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
ПК	66000	3	3	5500,00
Ноутбук	33000	3	3	2750,00
Сервер	55000	3	3	4583,33
Робоче місце дослідника	10000	5	3	500,00
Роутер	2200	4	3	137,50
Приміщення лабораторії	212000	20	3	2650,00
ОС Windows 11 Professional	44800	2	3	5600,00
Прикладний пакет Microsoft Office 2021 Pro	22400	2	3	2800,00
Visual Studio Professional 2022	2240	2	3	280,00
Всього				24800,83

5.2.8 Паливо та енергія для науково-виробничих цілей

Витрати на силову електроенергію (B_e) розраховуємо за формулою:

$$B_e = \sum_{i=1}^{n} \frac{W_{yi} \cdot t_i \cdot Ц_e \cdot K_{вni}}{\eta_i}, \quad (5.10)$$

де W_{yi} – встановлена потужність обладнання на визначеному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_e$ – вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо Ц$_e$ = 7,50 грн;

$K_{вni}$ – коефіцієнт, що враховує використання потужності, $K_{вni} < 1$;

η_i – коефіцієнт корисної дії обладнання, $\eta_i < 1$.

$B_e = 0,36 \cdot 480,0 \cdot 7,50 \cdot 0,95 / 0,97 = 1269,28$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 5.9 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
ПК	0,36	480	1269,28
Ноутбук	0,12	360	317,32
Серверне обладнання	0,5	320	1175,26
Робоче місце дослідника	0,15	480	528,87
Роутер	0,05	10	3,67
Всього			3294,39

5.2.9 Службові відрядження

До статті «Службові відрядження» дослідної роботи належать витрати на відрядження штатних працівників, працівників організацій, які працюють за

договорами цивільно-правового характеру, аспірантів, зайнятих розробленням досліджень, відрядження, пов'язані з проведенням випробувань машин та приладів, а також витрати на відрядження на наукові з'їзди, конференції, наради, пов'язані з виконанням конкретних досліджень.

Витрати за статтею «Службові відрядження» розраховуємо як 20…25% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{св} = (З_о + З_р) \cdot \frac{Н_{св}}{100\%}, \quad (5.11)$$

де $Н_{св}$ – норма нарахування за статтею «Службові відрядження», приймемо $Н_{св}$ = 20%.

$В_{св}$ = (218571,43+ 3207,27) · 20 / 100% = 44355,74 грн.

5.2.10 Витрати на роботи, які виконують сторонні підприємства, установи і організації

Витрати за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації» розраховуємо як 30…45% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{сп} = (З_о + З_р) \cdot \frac{Н_{сп}}{100\%}, \quad (5.12)$$

де $Н_{сп}$ – норма нарахування за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації», приймемо $Н_{сп}$= 30%.

$В_{сп}$ = (218571,43+ 3207,27) · 30 / 100% = 66533,61грн.

5.2.11 Інші витрати

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуємо як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$I_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (5.13)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати», приймемо $Н_{ів}$ = 50%.

$І_в$ = (218571,43+ 3207,27) · 50 / 100% = 110889,35 грн.

5.2.12 Накладні (загальновиробничі) витрати

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов'язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов'язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов'язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін.

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуємо як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$В_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (5.14)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо $Н_{нзв}$ = 100%.

$В_{нзв}$ = (218571,43+ 3207,27) · 100 / 100% = 221778,69 грн.

Витрати на проведення науково-дослідної роботи розраховуємо як суму всіх попередніх статей витрат за формулою:

$$В_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + В_{спец} + В_{прг} + А_{обл} + В_е + В_{св} + В_{сп} + І_в + В_{нзв}. \quad (5.15)$$

$В_{заг}$= 1004181,03 грн.

Загальні витрати *ЗВ* на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховується за формулою:

$$ЗВ = \frac{В_{заг}}{\eta}, \quad (5.16)$$

де η - коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, приймемо η =0,7.

ЗВ = 1004181,03/ 0,9 = 1115756,69 грн.

5.3 Розрахунок економічної ефективності науково-технічної розробки при її можливій комерціалізації потенційним інвестором

В ринкових умовах узагальнюючим позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку.

Результати дослідження передбачають комерціалізацію протягом 3-х років реалізації на ринку.

В цьому випадку майбутній економічний ефект буде формуватися на основі таких даних:

ΔN – збільшення кількості споживачів продукту, у періоди часу, що аналізуються, від покращення його певних характеристик; Потенційно такий Програмний засіб можуть придбати залізниці інших країн, відповідно наш прогноз:

1-й рік – 4 користувача;

2-й рік – 6 користувачів;

3-й рік – 5 користувачів.

N – кількість споживачів які використовували аналогічний продукт у році до впровадження результатів нової науково-технічної розробки, приймемо 1 користувач (Укрзалізниця);

$Ц_б$ – вартість програмного продукту у році до впровадження результатів розробки, приймемо 500000 грн;

$\pm\Delta Ц_о$ – зміна вартості програмного продукту від впровадження результатів науково-технічної розробки, приймемо 200000,00 грн.

Можливе збільшення чистого прибутку у потенційного інвестора $\Delta П_i$ для кожного із 3-х років, протягом яких очікується отримання позитивних

результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховуємо за формулою **:**

$$\Delta П_i = (\pm \Delta Ц_o \cdot N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \quad (5.17)$$

де λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2023 році ставка податку на додану вартість складає 20%, а коефіцієнт $\lambda = 0,8333$;

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту. Приймемо ρ =50%;

ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2023 році $\vartheta = 18\%$;

Збільшення чистого прибутку 1-го року:

$\Delta П_1 = (200000 \cdot 1 + 700000 \cdot 4) \cdot 0,83 \cdot 0,5 \cdot (1\text{-}0,18/100\%) =$ 1024590 грн.

Збільшення чистого прибутку 2-го року:

$\Delta П_2 = (200000 \cdot 1 + 700000 \cdot (4+6)) \cdot 0,83 \cdot 0,5 \cdot (1\text{-}0,18/100\%) =$ 2459016 грн.

Збільшення чистого прибутку 3-го року:

$\Delta П_3 = (200000 \cdot 1 + 700000 \cdot (4+6+5)) \cdot 0,83 \cdot 0,5 \cdot (1\text{-}0,18/100\%) =$ 3654371 грн.

Приведена вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \quad (5.18)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

T – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ =0,1;

t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$ПП$ = 1024590/(1+0,1)^1+ 2459016/(1+0,1)^2+ 3654371/(1+0,1)^3= 5709273,10 грн.

Величина початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки:

$$PV = k_{інв} \cdot ЗВ, \tag{5.19}$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$ =2;

$ЗВ$ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 1115756,69 грн.

$PV = k_{інв} \cdot ЗВ$ = 2· 1115756,69= 2231513,39 грн.

Абсолютний економічний ефект $E_{абс}$ для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV \tag{5.20}$$

де $ПП$ – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, 5709273,10тгрн;

PV – теперішня вартість початкових інвестицій, 2231513,39 грн.

$E_{абс} = ПП - PV$ = 5709273,10 - 2231513,39= 3477759,71грн.

Внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$E_в = \sqrt[Тж]{1 + \frac{E_{абс}}{PV}} - 1, \quad (5.21)$$

де $E_{абс}$ – абсолютний економічний ефект вкладених інвестицій, 3477759,71грн;

PV – теперішня вартість початкових інвестицій, 2231513,39грн;

$T_ж$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, 3 роки.

$$E_в = \sqrt[Тж]{1 + \frac{E_{абс}}{PV}} - 1 = (1+3477759,71/\ 2231513,39)^{1/3} = 0,37.$$

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$:

$$\tau_{мін} = d + f, \quad (5.22)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2023 році в Україні d =0,1;

f – показник, що характеризує ризикованість вкладення інвестицій, приймемо 0,18.

$\tau_{мін}$ = 0,1+0,18 = 0,29 < 0,37 свідчить про те, що внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки вища мінімальної внутрішньої дохідності. Тобто інвестувати в науково-дослідну

роботу за темою «Розробка методів і програмного засобу для ведення розкладу потягів залізниці із використанням датчиків та трекерів» доцільно.

Період окупності інвестицій $T_{ок}$ які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$T_{ок} = \frac{1}{E_в}, \quad (5.23)$$

де $E_в$ – внутрішня економічна дохідність вкладених інвестицій.

$T_{ок}$ = 1 / 0,37 = 2,7 року.

$T_{ок} <$ 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

5.4 Висновки до розділу

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Розробка методів і програмного засобу для ведення розкладу потягів залізниці із використанням датчиків та трекерів» становить 38,7 балів, що, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу вище середнього).

Також термін окупності становить 2,7 р., що менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Отже можна зробити висновок про доцільність проведення науково-дослідної роботи за темою «Розробка методів і програмного засобу для ведення розкладу потягів залізниці із використанням датчиків та трекерів».

ВИСНОВКИ

У магістерській кваліфікаційній роботі було розроблено програмний засіб маршрутизації та розкладу з використанням датчиків та трекерів «Залізниця». Додаток призначений для підвищення ефективності ведення розкладу та зручним місцем для перегляду інформації про можливі маршрути. Також створено автоматизовану систему корегування розкладу відповідно до розташування транспортного засобу.

Проведено аналіз стану систем ведення розкладу які є актуальними на даний момент часу. Було проведено аналіз аналогів які на сьогодні є в наявності та проведено порівняння їх між собою та програмним засобом маршрутизації та розкладу з використанням датчиків та трекерів «Залізниця». У результаті чого було доведено доцільність розробки магістерської кваліфікаційної роботи. Також проведено аналіз існуючих підходів до вирішення проставленої задачі, в результаті чого вирішено створити окремий програмний модуль для обробки інформації яка буде надходити від системи датчиків. Також було вирішено використовувати бібліотеку Lodash, яка допоможе краще маніпулювати даними і підвищить ефективність роботі усього програмного продукту. Було встановлено основні завдання, які необхідно виконати для розробки програмного засобу маршрутизації та розкладу руху транспортних засобів.

Розглянуто найпоширеніші способи взаємодії можливого пасажира з програмним засобом маршрутизації та розкладу. Проаналізовано способи безпечного обміну даних та обрано асиметричний метод шифрування RSA. Даний метод найефективніший для збереження конфіденціальної інформації користувача у відкритій мережі інтернет. Було проаналізовано можливість незначної зміни інтерфейсу програми в залежності від обраного користувачем браузеру. На основі цих даних розроблено ескіз графічного інтерфейсу програмного засобу що розробляється, який буде зручним у використанні та не

буде мати зайвого навантаження. Також розроблено формулу обрахування можливої затримки транспорту та створено базові алгоритми.

Детально розглянуто обрану мову програмування JavaScript. Наявна мова програмування повністю задовільняє усі потреби обраної розробки. Також в результаті даного аналізу було прийнято рішення використовувати Visual Studio Code та СУБД PostgreSQL для розробки програмного засобу маршрутизації та розкладу з датчиків та трекерів.

Розглянуто декілька методів та обрано тестування методом «чорної скриньки». Було розроблено інструкцію користувача, за допомогою якої користувач зможе ознайомитись з основними можливостями програмного засобу та розпочати повноцінне користування. Було визначено рекомендовані конфігурації персональних комп'ютерів на базі Windows для коректної та ефективної роботи програми.

Було проведено економічні дослідження згідно з якими рівень комерційного потенціалу розробки за темою «Розробка методів і програмного засобу для ведення розкладу потягів залізниці із використанням датчиків та трекерів» становить 38,7 балів, що, свідчить про комерційну важливість проведення даних досліджень. За попередніми оцінками термін окупності становить 2,7 р., що менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Отже можна зробити висновок про доцільність проведення науково-дослідної роботи за темою «Розробка методів і програмного засобу для ведення розкладу потягів залізниці із використанням датчиків та трекерів».

СПИСОК ЛІТЕРАТУРИ

1. Economic Role of Transport Infrastructure: Theory and Models / Claudio Ferrari, Anna Bottasso, Maurizio Conti, Alessio Tei. – Elsevier 1st Edition, 2018. – 15p. – ISBN 978-0128130964.

2. Кошелєв А. О., Розробка алгоритму пошуку потягу на лінії з використанням датчиків та трекерів [Електрон-ний ресурс] / Кошелєв А. О., Черноволик Г. О. // Матеріали всеукраїнської науково-практичної інтернет-конференції молодих вчених та студентів «Сучасні інформаційні системи та технології», 2023 р. – Електрон. текст. дані. – 2023. – Доступ: http://kntu.net.ua/index.php/ukr/Struktura/Kafedri-universitetu/IInformacijnih-tehnologij/Konferenciyi-ta-vidannya

3. Graphical user interface [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Graphical_user_interface.

4. Authentication and Authorization in Web API [електронний ресурс]: https://dotnettutorials.net/lesson/authentication-and-authorization-in-web-api.

5. Знайомство з інтегрованим середовищем розробки Visual Studio Code [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft .com/ru-ru/visualstudio/ide/quickstart-ide-orientation?view=vs-2019.

6. Authentication vs Authorization [електронний ресурс]: https://frontegg.com/blog/authentication-vsauthorization#:~:text=Types%20of%20authorization%20include%20discretionary,being% 20used%20to%20implement%20them.

7. API Visual Viewport [Електронний ресурс] – Режим доступу до ресурсу: https://core.visual.org/#bot-api.

8. Johannes Buchmann. Introduction to Cryptography / Buchmann J. – Springer, 2004. – 352 p. – ISBN 978-3540425806.

9. William Stallings. Cryptography and Network Security: Principles and Practice / Stallings W. – Pearson, 2016. – 784 p. – ISBN 978-0134444284.

10. Christof Paar, Jan Pelzl. Understanding Cryptography: A Textbook for Students and Practitioners / Paar C., Pelzl J. – Springer, 2010. – 464 p. – ISBN 978-3642041006.

11. Richard A. Mollin. RSA and Public-Key Cryptography / Mollin R.A. – Chapman and Hall/CRC, 2002. – 392 p. – ISBN 978-1584883388.

12. 16. Alex Young. Node.js in Action. Second Edition / Young A., Meck B., Cantelon M. – Manning Publications Co., 2017. – 392 p. – ISBN 978-1617292576.

13. Shelley Powers. Learning Node, Second Edition / Powers S. – O'Reilly Media, Inc., 2016. – 400 p. – ISBN 978-1491943120.

14. David Amos. JavaScript Basics: A Practical Introduction to JavaScript / Amos D., Bader D. – JavaScript, Inc., 2021. – 635 p. – ISBN 978-1775093329.

15. Мова програмування JavaScript: особливості й переваги [Електронний ресурс] – Режим доступу до ресурсу: https://ipap.ru/poleznye-stati/4-useful/yazyk-programmirovaniya-javascript-osobennosti-i-preimushchestva.

16. IDE [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Інтегроване_середовище_розробки.

17. Романюк О.Н. Організація баз даних і знань / О.Н. Романюк, Т.О. Савчук // Навчальний посібник. – Вінниця: «УНІВЕРСУМ-Вінниця», 2003. – 123 с. – ISBN 966-641-081-8.

18. Anany Levitin. Introduction to the Design and Analysis of Algorithms / Levitin A. – Pearson, 2012. – 592 p. – ISBN 978-0132316811.

19. Simon Monk. Programming the Raspberry Pi, Second Edition: Getting Started with Python / Monk S. – McGraw-Hill Education, 2015. – 208 p. – ISBN 978-1259587405.

20. Martin Kleppmann. Designing Data-Intensive Applications / Kleppmann M. – O'Reilly Media, 2017. – 616 p. – ISBN 978-1449373320.

21. V.N. Sadasivan. Railway Signalling and Interlocking / Sadasivan V.N. – Sree Saraswathy Press, 2009. – 576 p. – ISBN 978-8185992797.

22. Why did we build Visual Studio Code? [Електронний ресурс] – Режим доступу до ресурсу: https://code.visualstudio.com/docs/editor/whyvscode.

23. Documentation for Visual Studio Code [Електронний ресурс] – Режим доступу до ресурсу: https://code.visualstudio.com/docs.

24. Все, що потрібно знати про бази даних для початківців: MySQL, PostgreSQL, MongoDB [Електронний ресурс] – Режим доступу до ресурсу: https://dan-it.com.ua/uk/blog/vse-shho-potribno-znati-pro-bazi-danih-dlja-pochatkivciv-mysql-postgresql-mongodb/.

25. Тестування програмного забезпечення: типи, види та застосування [Електронний ресурс] – Режим доступу до ресурсу: https://foxminded.ua/testuvannia-prohramnoho-zabezpechennia/.

26. Основи тестування [Електронний ресурс] – Режим доступу до ресурсу: https://qalight.ua/baza-znaniy/shho-take-testuvannya-programnogo-zabezpechennya.

27. Вступ до сучасного Web-дизайну: HTML5+CSS3 / О.М. Шикула. – Київ, 2019. – 68с. – ISBN 978-5-907114-90-6.

28. Програмний засіб і його характеристики [Електронний ресурс] Режим доступу до ресурсу: https://www.centum-d.com/veb-dodatok-yogo-harakteristiki/.

29. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. Вінниця : ВНТУ, 2021. 42 с.

30. Кавецький В. В. Економічне обґрунтування інноваційних рішень: практикум / В. В. Кавецький, В. О. Козловський, І. В. Причепа. Вінниця : ВНТУ, 2016. 113 с.

ДОДАТКИ

Додаток А – Технічне завдання

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

ЗАТВЕРДЖУЮ
Завідувач кафедрою ПЗ
к.т.н., проф.
_____ О. Н. Романюк
" 19 " вересня 2023 р.

Технічне завдання
на магістерську кваліфікаційну роботу
«Розробка методів і програмного засобу для ведення розкладу потягів залізниці із використанням датчиків та трекерів»
за спеціальністю 121 – Інженерія програмного забезпечення

Керівник магістерської кваліфікаційної роботи:
_____ к.т.н., доцент Г. О. Черноволик
" 19 " вересня 2023 р.

Виконав:
_____ студент гр. 3ПІ-17б А. О. Кошелєв
" 19 " вересня 2023 р.

Вінниця – 2023 року

1. Найменування та галузь застосування

Магістерська кваліфікаційна робота: «Розробка методів і програмного засобу для ведення розкладу потягів залізниці із використанням датчиків та трекерів».

Галузь застосування – управління та адміністрування.

2. Підстава для розробки.

Підставою для виконання магістерської кваліфікаційної роботи (МКР) є індивідуальне завдання на МКР та наказ №247 від «18» вересня 2023 р. ректора по ВНТУ про закріплення тем МКР.

3. Мета та призначення розробки.

Метою даної роботи є підвищення ефективності оповіщення пасажирів за допомогою об’єднання усієї інформації в одному місці, а також максимально автоматизовано повідомляти про можливі зміни в розкладі та за необхідності в ручному режимі доповнити інформацію про затримку.

Призначення роботи – розробка і програмна реалізація програмного засобу маршрутизації та розкладу потягів з використанням датчиків та трекерів.

4. Вихідні дані для проведення НДР

Перелік основних літературних джерел, на основі яких буде виконуватись МКР.

1. Economic Role of Transport Infrastructure: Theory and Models / Claudio Ferrari, Anna Bottasso, Maurizio Conti, Alessio Tei. – Elsevier 1st Edition, 2018. – 15p. – ISBN 978-0128130964.

2. Visual Viewport API [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Web/API/Visual_Viewport_API

3. Graphical user interface [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Graphical_user_interface.

4. Романюк О.Н. Організація баз даних і знань / О.Н. Романюк, Т.О. Савчук // Навчальний посібник. – Вінниця: «УНІВЕРСУМ-Вінниця», 2003. – 123 с. – ISBN 966-641-081-8.

5. Технічні вимоги

Модель розробки – ієрархічна; система управління базами даних – PostgreSQL; мова запитів – SQL; вхідні дані – база даних із розкладом руху потягів та коефіцієнтами дорожнього покриття, навантаженості та кривизни; ідентифікаційний номер програмного засобу у відповідному форматі; вихідні дані – перевірений розклад на наявність затримок транспортних засобів; середовище розробки – Visual Studio Code; мова програмування – JavaScript.

6. Конструктивні вимоги

Графічна та текстова документація повинна відповідати діючим стандартам України.

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

1. Пояснювальна записка до БДР;
2. Технічне завдання;
3. 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 магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз завдання і вибір методу вирішення поставленої задачі дослідження	20.09.2023 – 25.09.2023	Виконано
2	Розробка методу та алгоритму обчислення приблизного часу прибуття транспортного засобу	25.09.2023 – 29.09.2023	Виконано
3	Вибір середовища та мови розробки	30.09.2023 – 13.10.2023	Виконано
4	Розробка модулів програми	14.10.2023 – 27.10.2023	Виконано
5	Тестування програми	28.10.2023 – 10.11.2023	Виконано
6	Оцінка економічної ефективності результатів виконаної роботи	11.11.2023 – 24.11.2023	Виконано
7	Оформлення матеріалів до захисту БДР	25.11.2023 – 01.12.2023	Виконано

10. Порядок контролю та прийняття

Виконання етапів магістерської кваліфікаційної роботи контролюється керівником згідно з графіком виконання роботи.

Прийняття магістерської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком.

Додаток Б – Протокол перевірки

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Розробка методів і програмного засобу для ведення розкладу потягів залізниці із використанням датчиків та трекерів

Тип роботи: МКР

Підрозділ : кафедра програмного забезпечення, ФІТКІ

Науковий керівник: к.т.н., доц. Черноволик Г. О.

Оригінальність	86%
Схожість	14%

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 _____ Кошелєв А. О.

Керівник роботи _____ Черноволик Г. О.

Додаток В – Лістинг програми

index.js

```
var map
  , you
  , pos
  , t_0
  , log = ''
  , head = '["latitude","longitude","precision","time"]'
  , zoom
  , time
  , from
  ;

google.maps.event.addDomListener(window, 'load', init);

function init() {
  map = new google.maps.Map( document.getElementById('map')
                       , { zoom: zoom
                         , mapTypeId: google.maps.MapTypeId.ROADMAP
                         });

  if (navigator.geolocation)
    navigator.geolocation.watchPosition(gotPosition, function() {
      noGeolocation('Error: The Geolocation service failed.');
    }, { enableHighAccuracy: true, maximumAge: 10e3, timeout: 20e3 });
  else
    noGeolocation('Error: Your browser doesn\'t support geolocation.');
}

function gotPosition(position) {
  var at  = position.coords
    , off = at.accuracy
    , z
    ;

  pos = ll(at.latitude, at.longitude);
  if (you) you.setPosition(pos); else {
```

```
   t_0 = Math.round(+new Date / 1000);
   you = new google.maps.Marker({ map: map
                                , position: pos
                                , icon: marker(url, s(20, 17), p(10, 8))
                                });
   google.maps.event.addListener(you, 'click', function() {
     location = 'mailto:?subject=GPS%20Track&body='
              + encodeURIComponent(log + ' ]\n}\n');
   });
 }
 if (!zoom) map.setCenter(pos);

 // zoom in, as precision improves (or out again)
 if (off > 2e3) z = 15;
 if (off < 2e3) z = 16;
 if (off < 900) z = 17;
 if (off < 100) z = 18;
 if (z !== zoom) map.setZoom(zoom = z);

 map.panTo(pos);
 save(at);
}

function noGeolocation(message) {
 var opts = { map: map
            , position: ll(60, 105)
            , content: message
            }
   , info = new google.maps.InfoWindow(opts);
 map.setCenter(opts.position);
}

function s(w, h) { return new google.maps.Size(w, h); }
function p(x, y) { return new google.maps.Point(x, y); }
function ll(y, x) { return new google.maps.LatLng(y, x); }
function marker(url, size, hotspot, origin) {
 return new google.maps.MarkerImage(url, size, origin || p(0, 0), hotspot);
```

```
}

function save(at) {
  var lat = at.latitude.toFixed(6) // decimeter precision should be quite enough
    , lng = at.longitude.toFixed(6)
    , pre = at.accuracy.toFixed(0)
    ;
  time = Math.round(new Date / 1000) - t_0;
  log += (log ? ' ,' : '{"time":'+ t_0 +'\n,"head":\n '+ head +'\n,"data":\n [')
       + '[' + lat +','+ lng +','+ pre +','+ time +']\n';
}
```

Window.js

```
(function(jQuery) {


  jQuery().ready(function() {

    jQuery('#primary-list').smartmenus({
      subIndicatorsText: "
    });

    jQuery('#primary-mega').smartmenus({
      subIndicatorsText: '',
    });

    jQuery('.menu-button.list').click(function() {
      var $this = jQuery(this),
          $menu = jQuery('#primary-list');
      if ($menu.is(':animated')) {
        return false;
      }
      if (!$this.hasClass('collapsed')) {
        $menu.slideUp(250, function() { jQuery(this).addClass('collapsed').css('display', ''); });
        $this.addClass('collapsed');
      } else {
        $menu.slideDown(250, function() { jQuery(this).removeClass('collapsed'); });
```

```
    $this.removeClass('collapsed');
  }
  return false;
});

jQuery('.menu-button.mega').click(function() {
  var $this = jQuery(this),
      $menu = jQuery('#primary-mega');
  if ($menu.is(':animated')) {
    return false;
  }
  if (!$this.hasClass('collapsed')) {
    $menu.slideUp(250, function() { jQuery(this).addClass('collapsed').css('display', ''); });
    $this.addClass('collapsed');
  } else {
    $menu.slideDown(250, function() { jQuery(this).removeClass('collapsed'); });
    $this.removeClass('collapsed');
  }
  return false;
});

jQuery('#primary .pab-grid .column > .inner').matchHeight();

jQuery('.pab-category.base .column').matchHeight();

jQuery('.pab-category.grid .column .inner').matchHeight();

jQuery('#secondary > .block > .inner').matchHeight({ property: 'min-height' });

jQuery('#tertiary > .block > .inner').matchHeight({ property: 'min-height' });

jQuery('.slider-basic').slick({
  dots: true,
  adaptiveHeight: true
});

jQuery('.slider-headline').slick({
```

```
  dots: true,
  adaptiveHeight: true
});

jQuery('.slider-thumbnail').slick({
  dots: true,
  adaptiveHeight: true,
  asNavFor: '.pager-thumbnail'
});

jQuery('.pager-thumbnail').slick({
  asNavFor: '.slider-thumbnail',
  arrows: true,
  dots: false,
  adaptiveHeight: true,
  focusOnSelect: true
});

jQuery('.slider-playlist').slick({
  dots: true,
  asNavFor: '.pager-playlist'
});

jQuery('.pager-playlist').slick({
  asNavFor: '.slider-playlist',
  dots: false,
  vertical: true,
  focusOnSelect: true,
  responsive: [
    {
      breakpoint: 1441,
      settings: {
        slidesToShow: 3,
        slidesToScroll: 1
      }
    }
  ]
```

```
});

jQuery('.slider-gallery').slick({
  adaptiveHeight: true
});

jQuery('.slider-newsflash').slick({
  arrows: false,
  adaptiveHeight: true,
  vertical: true
});

jQuery('.slider-merging').slick({
  arrows: true,
  dots: false,
  slidesToShow: 1,
  adaptiveHeight: true
});

jQuery(".tab_nav li").click(function(e){
  if (!jQuery(this).hasClass("active")) {
    var tabNum = jQuery(this).index();
    var nthChild = tabNum+1;

    jQuery(this).addClass("active").siblings('li').removeClass('active');
    jQuery(this).parent().next().children("li.active").removeClass('active');
    jQuery(this).parent().next().children("li:nth-child("+nthChild+")").addClass("active");
  }
});

jQuery('ul.accordion .content').hide();
 jQuery('ul.accordion .active .content').show();
jQuery('ul.accordion .active h3').addClass('selected');

jQuery('ul.accordion h3').click(function(){
  if (jQuery(this).hasClass('selected')) {
    jQuery(this).removeClass('selected');
```

```
      jQuery(this).siblings(".content").slideUp();
    } else {
      jQuery('ul.accordion h3').removeClass('selected');
      jQuery(this).addClass('selected');
      jQuery('ul.accordion .content').slideUp();
      jQuery(this).siblings(".content").slideDown();
    }
    return false;
  });

});

})(jQuery);
;
```

Drupal.js

```
(function ($) {
  var cache = {}, uuid = 0;
$.fn.once = function (id, fn) {
  if (typeof id != 'string') {
    // Generate a numeric ID if the id passed can't be used as a CSS class.
    if (!(id in cache)) {
      cache[id] = ++uuid;
    }
    // When the fn parameter is not passed, we interpret it from the id.
    if (!fn) {
      fn = id;
    }
    id = 'jquery-once-' + cache[id];
  }
  // Remove elements from the set that have already been processed.
  var name = id + '-processed';
  var elements = this.not('.' + name).addClass(name);

  return $.isFunction(fn) ? elements.each(fn) : elements;
};
$.fn.removeOnce = function (id, fn) {
```

```
    var name = id + '-processed';
    var elements = this.filter('.' + name).removeClass(name);

    return $.isFunction(fn) ? elements.each(fn) : elements;
  };
})(jQuery);
;

var Drupal = Drupal || { 'settings': {}, 'behaviors': {}, 'locale': {} };

// Allow other JavaScript libraries to use $.
jQuery.noConflict();

(function ($) {

/**
 * Override jQuery.fn.init to guard against XSS attacks.
 *
 * See http://bugs.jquery.com/ticket/9521
 */
var jquery_init = $.fn.init;
$.fn.init = function (selector, context, rootjQuery) {
  // If the string contains a "#" before a "<", treat it as invalid HTML.
  if (selector && typeof selector === 'string') {
    var hash_position = selector.indexOf('#');
    if (hash_position >= 0) {
      var bracket_position = selector.indexOf('<');
      if (bracket_position > hash_position) {
        throw 'Syntax error, unrecognized expression: ' + selector;
      }
    }
  }
  return jquery_init.call(this, selector, context, rootjQuery);
};
$.fn.init.prototype = jquery_init.prototype;
Drupal.attachBehaviors = function (context, settings) {
  context = context || document;
```

```
  settings = settings || Drupal.settings;
  // Execute all of them.
  $.each(Drupal.behaviors, function () {
    if ($.isFunction(this.attach)) {
      this.attach(context, settings);
    }
  });
};
Drupal.detachBehaviors = function (context, settings, trigger) {
  context = context || document;
  settings = settings || Drupal.settings;
  trigger = trigger || 'unload';
  // Execute all of them.
  $.each(Drupal.behaviors, function () {
    if ($.isFunction(this.detach)) {
      this.detach(context, settings, trigger);
    }
  });
};
Drupal.checkPlain = function (str) {
  var character, regex,
      replace = { '&': '&amp;', '"': '&quot;', '<': '&lt;', '>': '&gt;' };
  str = String(str);
  for (character in replace) {
    if (replace.hasOwnProperty(character)) {
      regex = new RegExp(character, 'g');
      str = str.replace(regex, replace[character]);
    }
  }
  return str;
};
Drupal.formatString = function(str, args) {
  // Transform arguments before inserting them.
  for (var key in args) {
    switch (key.charAt(0)) {
      // Escaped only.
      case '@':
```

```
            args[key] = Drupal.checkPlain(args[key]);
          break;
          // Pass-through.
          case '!':
            break;
          // Escaped and placeholder.
          case '%':
          default:
            args[key] = Drupal.theme('placeholder', args[key]);
            break;
        }
        str = str.replace(key, args[key]);
      }
      return str;
    };
    Drupal.t = function (str, args, options) {
      options = options || {};
      options.context = options.context || '';

      // Fetch the localized version of the string.
      if (Drupal.locale.strings && Drupal.locale.strings[options.context] &&
Drupal.locale.strings[options.context][str]) {
        str = Drupal.locale.strings[options.context][str];
      }

      if (args) {
        str = Drupal.formatString(str, args);
      }
      return str;
    };
    Drupal.formatPlural = function (count, singular, plural, args, options) {
      var args = args || {};
      args['@count'] = count;
      // Determine the index of the plural form.
      var index = Drupal.locale.pluralFormula ? Drupal.locale.pluralFormula(args['@count']) :
((args['@count'] == 1) ? 0 : 1);
```

```
  if (index == 0) {
    return Drupal.t(singular, args, options);
  }
  else if (index == 1) {
    return Drupal.t(plural, args, options);
  }
  else {
    args['@count[' + index + ']'] = args['@count'];
    delete args['@count'];
    return Drupal.t(plural.replace('@count', '@count[' + index + ']'), args, options);
  }
};
Drupal.absoluteUrl = function (url) {
  var urlParsingNode = document.createElement('a');

  // Decode the URL first; this is required by IE <= 6. Decoding non-UTF-8
  // strings may throw an exception.
  try {
    url = decodeURIComponent(url);
  } catch (e) {}

  urlParsingNode.setAttribute('href', url);

  // IE <= 7 normalizes the URL when assigned to the anchor node similar to
  // the other browsers.
  return urlParsingNode.cloneNode(false).href;
};
Drupal.urlIsLocal = function (url) {
  // Always use browser-derived absolute URLs in the comparison, to avoid
  // attempts to break out of the base path using directory traversal.
  var absoluteUrl = Drupal.absoluteUrl(url);
  var protocol = location.protocol;

  // Consider URLs that match this site's base URL but use HTTPS instead of HTTP
  // as local as well.
  if (protocol === 'http:' && absoluteUrl.indexOf('https:') === 0) {
    protocol = 'https:';
```

```
  }
  var baseUrl = protocol + '//' + location.host + Drupal.settings.basePath.slice(0, -1);

  // Decoding non-UTF-8 strings may throw an exception.
  try {
    absoluteUrl = decodeURIComponent(absoluteUrl);
  } catch (e) {}
  try {
    baseUrl = decodeURIComponent(baseUrl);
  } catch (e) {}

  // The given URL matches the site's base URL, or has a path under the site's
  // base URL.
  return absoluteUrl === baseUrl || absoluteUrl.indexOf(baseUrl + '/') === 0;
};
Drupal.theme = function (func) {
  var args = Array.prototype.slice.apply(arguments, [1]);

  return (Drupal.theme[func] || Drupal.theme.prototype[func]).apply(this, args);
};

/**
 * Freeze the current body height (as minimum height). Used to prevent
 * unnecessary upwards scrolling when doing DOM manipulations.
 */
Drupal.freezeHeight = function () {
  Drupal.unfreezeHeight();
  $('<div id="freeze-height"></div>').css({
    position: 'absolute',
    top: '0px',
    left: '0px',
    width: '1px',
    height: $('body').css('height')
  }).appendTo('body');
};

/**
```

```
 * Unfreeze the body height.
 */
Drupal.unfreezeHeight = function () {
  $('#freeze-height').remove();
};

/**
 * Encodes a Drupal path for use in a URL.
 *
 * For aesthetic reasons slashes are not escaped.
 */
Drupal.encodePath = function (item, uri) {
  uri = uri || location.href;
  return encodeURIComponent(item).replace(/%2F/g, '/');
};

/**
 * Get the text selection in a textarea.
 */
Drupal.getSelection = function (element) {
  if (typeof element.selectionStart != 'number' && document.selection) {
    // The current selection.
    var range1 = document.selection.createRange();
    var range2 = range1.duplicate();
    // Select all text.
    range2.moveToElementText(element);
    // Now move 'dummy' end point to end point of original range.
    range2.setEndPoint('EndToEnd', range1);
    // Now we can calculate start and end points.
    var start = range2.text.length - range1.text.length;
    var end = start + range1.text.length;
    return { 'start': start, 'end': end };
  }
  return { 'start': element.selectionStart, 'end': element.selectionEnd };
};

/**
```

```
 * Build an error message from an Ajax response.
 */
Drupal.ajaxError = function (xmlhttp, uri, customMessage) {
 var statusCode, statusText, pathText, responseText, readyStateText, message;
 if (xmlhttp.status) {
  statusCode = "\n" + Drupal.t("An AJAX HTTP error occurred.") +  "\n" + Drupal.t("HTTP Result Code: !status", {'!status': xmlhttp.status});
 }
 else {
  statusCode = "\n" + Drupal.t("An AJAX HTTP request terminated abnormally.");
 }
 statusCode += "\n" + Drupal.t("Debugging information follows.");
 pathText = "\n" + Drupal.t("Path: !uri", {'!uri': uri} );
 statusText = '';
 // In some cases, when statusCode == 0, xmlhttp.statusText may not be defined.
 // Unfortunately, testing for it with typeof, etc, doesn't seem to catch that
 // and the test causes an exception. So we need to catch the exception here.
 try {
  statusText = "\n" + Drupal.t("StatusText: !statusText", {'!statusText': $.trim(xmlhttp.statusText)});
 }
 catch (e) {}

 responseText = '';
 // Again, we don't have a way to know for sure whether accessing
 // xmlhttp.responseText is going to throw an exception. So we'll catch it.
 try {
  responseText = "\n" + Drupal.t("ResponseText: !responseText", {'!responseText': $.trim(xmlhttp.responseText) } );
 } catch (e) {}

 // Make the responseText more readable by stripping HTML tags and newlines.
 responseText = responseText.replace(/<("[^"]*"|'[^']*'|[^'">])*>/gi,"");
 responseText = responseText.replace(/[\n]+\s+/g,"\n");

 // We don't need readyState except for status == 0.
 readyStateText = xmlhttp.status == 0 ? ("\n" + Drupal.t("ReadyState: !readyState", {'!readyState': xmlhttp.readyState})) : "";
```

```
    // Additional message beyond what the xmlhttp object provides.
    customMessage = customMessage ? ("\n" + Drupal.t("CustomMessage: !customMessage", {'!customMessage': customMessage})) : "";

    message = statusCode + pathText + statusText + customMessage + responseText + readyStateText;
    return message;
  };

  // Class indicating that JS is enabled; used for styling purpose.
  $('html').addClass('js');

  // 'js enabled' cookie.
  document.cookie = 'has_js=1; path=/';

  /**
   * Additions to jQuery.support.
   */
  $(function () {
    /**
     * Boolean indicating whether or not position:fixed is supported.
     */
    if (jQuery.support.positionFixed === undefined) {
      var el = $('<div style="position:fixed; top:10px" />').appendTo(document.body);
      jQuery.support.positionFixed = el[0].offsetTop === 10;
      el.remove();
    }
  });

  //Attach all behaviors.
  $(function () {
    Drupal.attachBehaviors(document, Drupal.settings);
  });

  /**
   * The default themes.
   */
```

```
Drupal.theme.prototype = {

 /**
  * Formats text for emphasized display in a placeholder inside a sentence.
  *
  * @param str
  *   The text to format (plain-text).
  * @return
  *   The formatted text (html).
  */
 placeholder: function (str) {
  return '<em class="placeholder">' + Drupal.checkPlain(str) + '</em>';
 }
};

})(jQuery);
;
Drupal.eu_cookie_compliance = {};

 Drupal.eu_cookie_compliance.execute = function () {
  try {
   if (!Drupal.settings.eu_cookie_compliance.popup_enabled) {
    return;
   }

   if (!Drupal.eu_cookie_compliance.cookiesEnabled()) {
    return;
   }

   Drupal.eu_cookie_compliance.updateCheck();
   var status = Drupal.eu_cookie_compliance.getCurrentStatus();
   if ((status === 0 && Drupal.settings.eu_cookie_compliance.method === 'default') || status === null || (Drupal.settings.eu_cookie_compliance.withdraw_enabled && Drupal.settings.eu_cookie_compliance.withdraw_button_on_info_popup)) {
    if (!Drupal.settings.eu_cookie_compliance.disagree_do_not_show_popup || status === null) {
     // Detect mobile here and use mobile_popup_html_info, if we have a mobile device.
```

```
            if (window.matchMedia('(max-width: ' +
Drupal.settings.eu_cookie_compliance.mobile_breakpoint + 'px)').matches &&
Drupal.settings.eu_cookie_compliance.use_mobile_message) {

Drupal.eu_cookie_compliance.createPopup(Drupal.settings.eu_cookie_compliance.mobile_popup_html_in
fo, (status !== null));
            }
            else {

Drupal.eu_cookie_compliance.createPopup(Drupal.settings.eu_cookie_compliance.popup_html_info,
(status !== null));
            }

            Drupal.eu_cookie_compliance.initPopup();
          }
        }
        if (status === 1 && Drupal.settings.eu_cookie_compliance.popup_agreed_enabled) {
          // Thank you banner.

Drupal.eu_cookie_compliance.createPopup(Drupal.settings.eu_cookie_compliance.popup_html_agreed);
          Drupal.eu_cookie_compliance.attachHideEvents();
        }
        else if (status === 2 && Drupal.settings.eu_cookie_compliance.withdraw_enabled) {
          if (!Drupal.settings.eu_cookie_compliance.withdraw_button_on_info_popup) {

Drupal.eu_cookie_compliance.createWithdrawBanner(Drupal.settings.eu_cookie_compliance.withdraw_m
arkup);
          }
          Drupal.eu_cookie_compliance.attachWithdrawEvents();
        }
      }
      catch (e) {
      }
    };

    Drupal.eu_cookie_compliance.initPopup = function() {
      Drupal.eu_cookie_compliance.attachAgreeEvents();
```

```
        if (Drupal.settings.eu_cookie_compliance.method === 'categories') {
          var categories_checked = [];

          if (Drupal.eu_cookie_compliance.getCurrentStatus() === null) {
            if (Drupal.settings.eu_cookie_compliance.select_all_categories_by_default) {
              categories_checked = Drupal.settings.eu_cookie_compliance.cookie_categories;
            }
          }
          else {
            categories_checked = Drupal.eu_cookie_compliance.getAcceptedCategories();
          }
          Drupal.eu_cookie_compliance.setPreferenceCheckboxes(categories_checked);
          Drupal.eu_cookie_compliance.attachSavePreferencesEvents();
        }

        if (Drupal.settings.eu_cookie_compliance.withdraw_enabled &&
Drupal.settings.eu_cookie_compliance.withdraw_button_on_info_popup) {
          Drupal.eu_cookie_compliance.attachWithdrawEvents();
          var currentStatus = Drupal.eu_cookie_compliance.getCurrentStatus();
          if (currentStatus === 1 || currentStatus === 2) {
            $('.eu-cookie-withdraw-button').show();
          }
        }
      }
      Drupal.eu_cookie_compliance.toggleWithdrawBanner = function () {
        var $wrapper = $('#sliding-popup');
        var $tab = $('.eu-cookie-withdraw-tab');
        var $bannerIsShowing = Drupal.settings.eu_cookie_compliance.popup_position ?
parseInt($wrapper.css('top')) === 0 : parseInt($wrapper.css('bottom')) === 0;
        var topBottom = (Drupal.settings.eu_cookie_compliance.popup_position ? 'top' : 'bottom');
        var height = $wrapper.outerHeight();
        if (Drupal.settings.eu_cookie_compliance.popup_position) {
          if ($bannerIsShowing) {
            $wrapper.animate({'top' : -1 * height}, Drupal.settings.eu_cookie_compliance.popup_delay);
          }
          else {
```

```
        $wrapper.animate({'top' : 0}, Drupal.settings.eu_cookie_compliance.popup_delay);
      }
    }
    else {
      if ($bannerIsShowing) {
        $wrapper.animate({'bottom' : -1 * height}, Drupal.settings.eu_cookie_compliance.popup_delay);
      }
      else {
        $wrapper.animate({'bottom' : 0}, Drupal.settings.eu_cookie_compliance.popup_delay);
      }
    }
  };

  Drupal.eu_cookie_compliance.createPopup = function (html, closed) {
    // This fixes a problem with jQuery 1.9.
    var $popup = $('<div></div>').html(html);
    $popup.attr('id', 'sliding-popup');
    if (!Drupal.settings.eu_cookie_compliance.popup_use_bare_css) {
      $popup.height(Drupal.settings.eu_cookie_compliance.popup_height)
          .width(Drupal.settings.eu_cookie_compliance.popup_width);
    }

    $popup.hide();
    var height = 0;
    if (Drupal.settings.eu_cookie_compliance.popup_position) {
      $popup.prependTo('body');
      height = $popup.outerHeight();
      $popup.show()
        .attr({ 'class': 'sliding-popup-top clearfix' })
        .css({ top: -1 * height });
      if (closed !== true) {
        $popup.animate({ top: 0 }, Drupal.settings.eu_cookie_compliance.popup_delay, null, function ()
{
          $popup.trigger('eu_cookie_compliance_popup_open');
        });
      }
    }
```

```
        else {
          if (Drupal.settings.eu_cookie_compliance.better_support_for_screen_readers) {
            $popup.prependTo('body');
          }
          else {
            $popup.appendTo('body');
          }

          height = $popup.outerHeight();
          $popup.show()
            .attr({ 'class': 'sliding-popup-bottom' })
            .css({ bottom: -1 * height });
          if (closed !== true) {
            $popup.animate({bottom: 0}, Drupal.settings.eu_cookie_compliance.popup_delay, null, function () {
              $popup.trigger('eu_cookie_compliance_popup_open');
            });
          }
        }
      };

      Drupal.eu_cookie_compliance.attachAgreeEvents = function () {
        var clickingConfirms = Drupal.settings.eu_cookie_compliance.popup_clicking_confirmation;
        var scrollConfirms = Drupal.settings.eu_cookie_compliance.popup_scrolling_confirmation;

        if (Drupal.settings.eu_cookie_compliance.method === 'categories' && Drupal.settings.eu_cookie_compliance.enable_save_preferences_button) {
          // The agree button becomes an agree to all categories button when the 'save preferences' button is present.
          $('.agree-button').click(Drupal.eu_cookie_compliance.acceptAllAction);
        }
        else {
          $('.agree-button').click(Drupal.eu_cookie_compliance.acceptAction);
        }
        $('.decline-button').click(Drupal.eu_cookie_compliance.declineAction);

        if (clickingConfirms) {
```

```
            $('a, input[type=submit], button[type=submit]').not('.popup-content *').bind('click.euCookieCompliance', Drupal.eu_cookie_compliance.acceptAction);
          }

          if (scrollConfirms) {
            var alreadyScrolled = false;
            var scrollHandler = function () {
              if (alreadyScrolled) {
                Drupal.eu_cookie_compliance.acceptAction();
                $(window).off('scroll', scrollHandler);
              }
              else {
                alreadyScrolled = true;
              }
            };

            $(window).bind('scroll', scrollHandler);
          }

          $('.find-more-button').not('.find-more-button-processed').addClass('find-more-button-processed').click(Drupal.eu_cookie_compliance.moreInfoAction);
        };

        Drupal.eu_cookie_compliance.attachSavePreferencesEvents = function () {
          $('.eu-cookie-compliance-save-preferences-button').click(Drupal.eu_cookie_compliance.savePreferencesAction);
        };
      Drupal.behaviors.eu_cookie_compliance_popup_block_cookies = {
          initialized: false,
          attach: function (context, settings) {
            if (!Drupal.behaviors.eu_cookie_compliance_popup_block_cookies.initialized && settings.eu_cookie_compliance) {
              Drupal.behaviors.eu_cookie_compliance_popup_block_cookies.initialized = true;
              if ((settings.eu_cookie_compliance.method === 'opt_in' && (Drupal.eu_cookie_compliance.getCurrentStatus() === null || !Drupal.eu_cookie_compliance.hasAgreed()))
```

```
            || (settings.eu_cookie_compliance.method === 'opt_out' &&
!Drupal.eu_cookie_compliance.hasAgreed() && Drupal.eu_cookie_compliance.getCurrentStatus() !==
null)
            || (Drupal.settings.eu_cookie_compliance.method === 'categories')
           ) {
            // Split the white-listed cookies.
            var euCookieComplianceWhitelist =
settings.eu_cookie_compliance.whitelisted_cookies.split(/\r\n|\n|\r/g);

            // Add the EU Cookie Compliance cookie.
            var cookieName = (typeof eu_cookie_compliance_cookie_name === 'undefined' ||
eu_cookie_compliance_cookie_name === '') ? 'cookie-agreed' : eu_cookie_compliance_cookie_name;
            euCookieComplianceWhitelist.push(cookieName);

            euCookieComplianceBlockCookies = setInterval(function () {
              // Load all cookies from jQuery.
              var cookies = $.cookie();

              // Check each cookie and try to remove it if it's not white-listed.
              for (var i in cookies) {
                var remove = true;
                var hostname = window.location.hostname;
                var cookieRemoved = false;
                var index = 0;

                remove = !Drupal.eu_cookie_compliance.isWhitelisted(i);

                // Remove the cookie if it's not white-listed.
                if (remove) {
                  while (!cookieRemoved && hostname !== '') {
                    // Attempt to remove.
                    cookieRemoved = $.removeCookie(i, { domain: '.' + hostname, path: '/' });
                    if (!cookieRemoved) {
                      cookieRemoved = $.removeCookie(i, { domain: hostname, path: '/' });
                    }

                    index = hostname.indexOf('.');
```

```
            // We can be on a sub-domain, so keep checking the main domain as well.
            hostname = (index === -1) ? '' : hostname.substring(index + 1);
          }

          // Some jQuery Cookie versions don't remove cookies well.  Try again
          // using plain js.
          if (!cookieRemoved) {
            document.cookie = i + '=; expires=Thu, 01 Jan 1970 00:00:01 GMT; path=/;';
          }
        }
      }, 5000);
    }
  }
}

})(jQuery);
;
```

Додаток Г.

ІЛЮСТРАТИВНІ ЧАСТИНА

РОЗРОБКА МЕТОДІВ І ПРОГРАМНОГО ЗАСОБУ ДЛЯ ВЕДЕННЯ РОЗКЛАДУ ПОТЯГІВ ЗАЛІЗНИЦІ ІЗ ВИКОРИСТАННЯМ ДАТЧИКІВ ТА ТРЕКЕРІВ

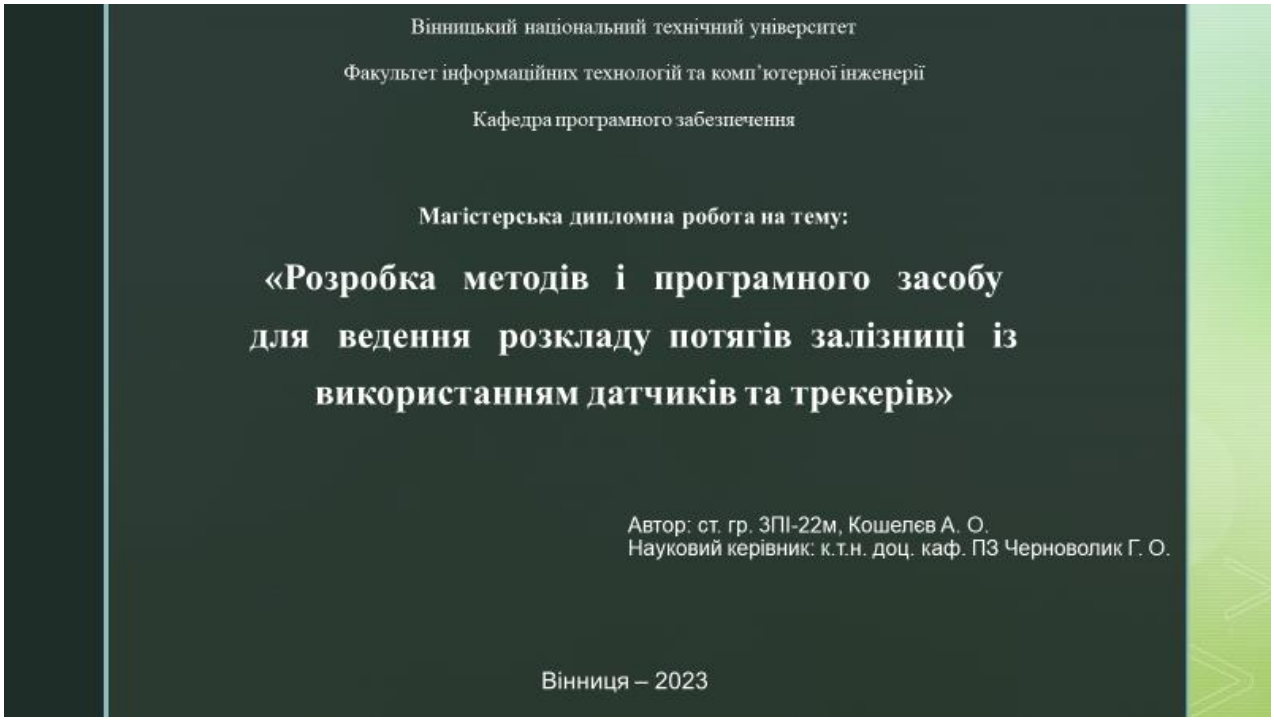


Рисунок В.1 – Титульний слайд

Рисунок В.2 – Актуальність теми

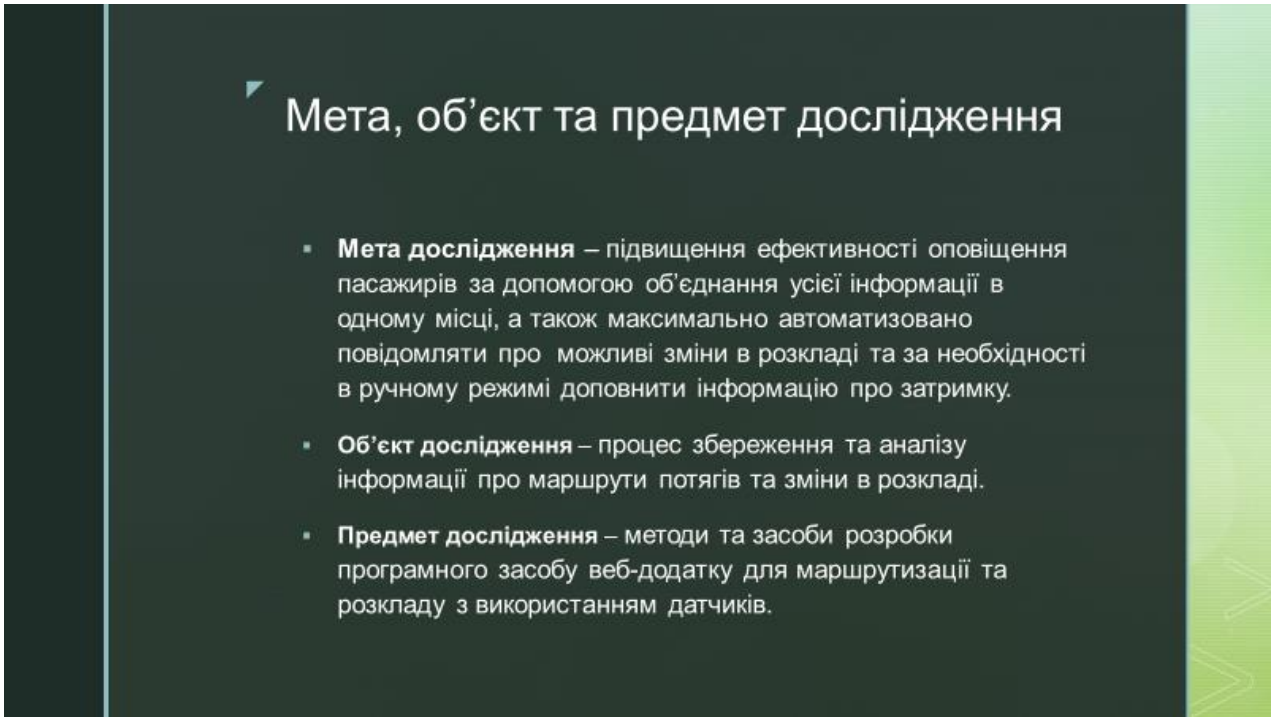


Рисунок В.3 – Мета, об’єкт та предмет дослідження

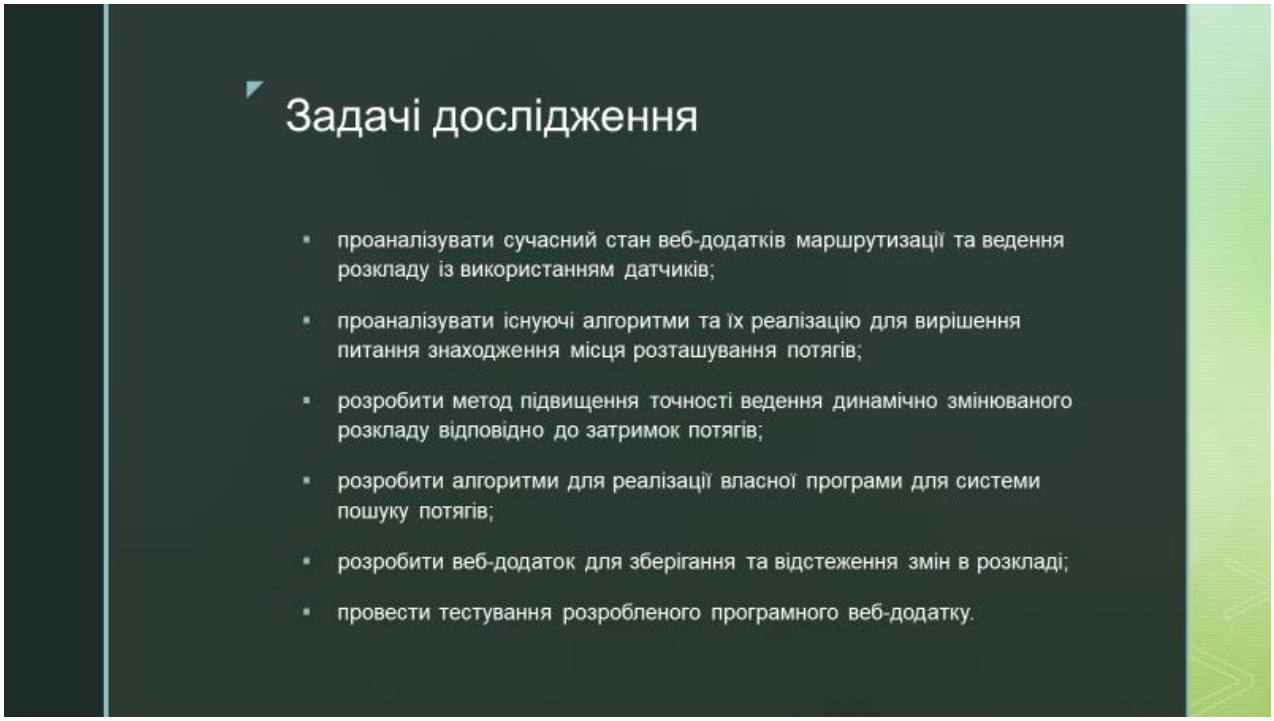


Рисунок В.4 – Задачі дослідження

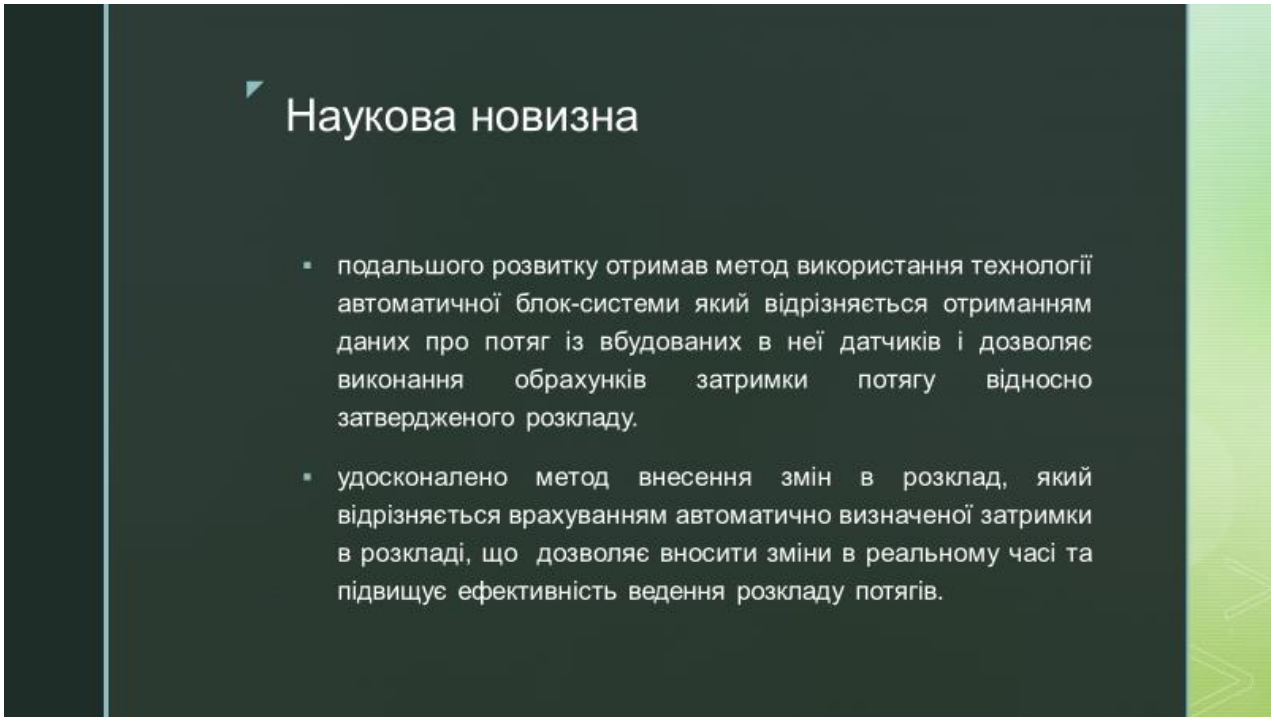


Рисунок В.5 – Наукова новизна

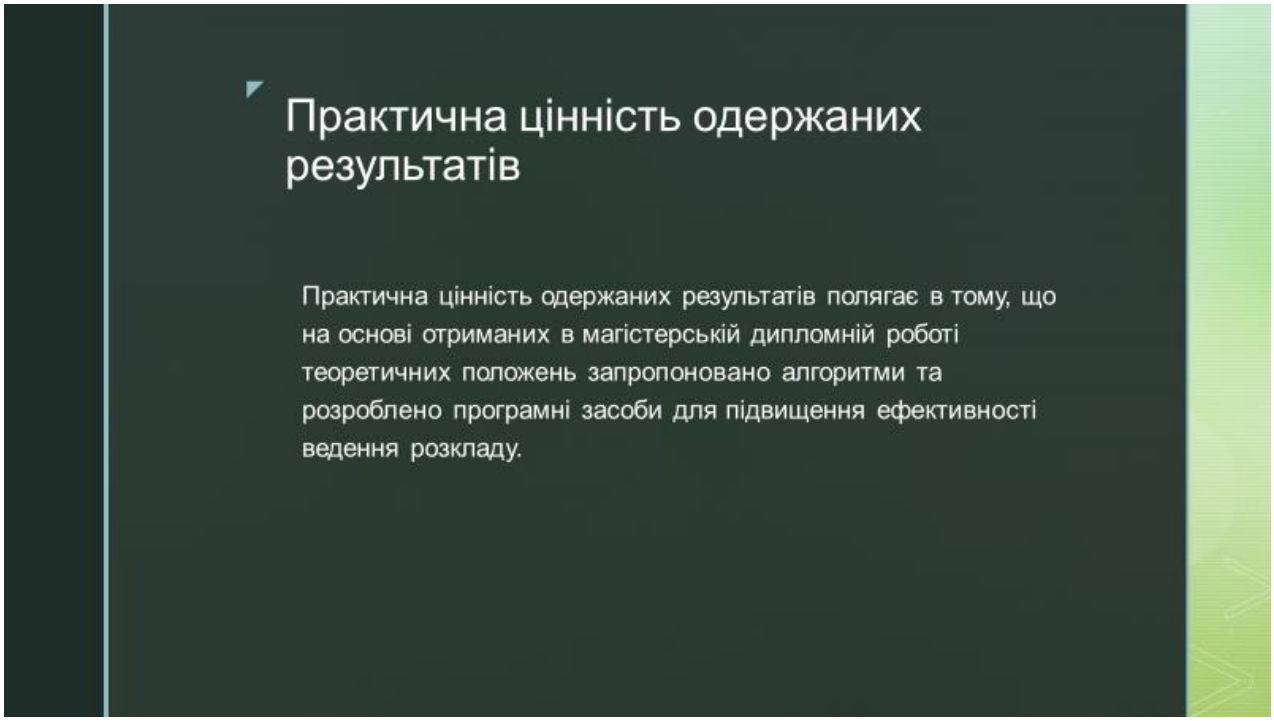


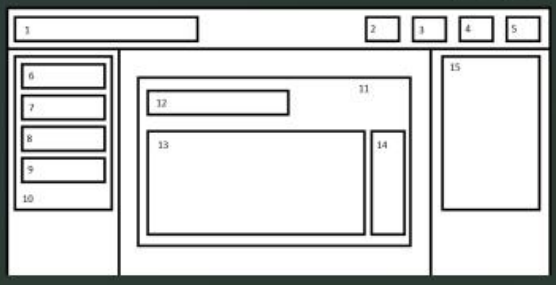
Рисунок В.6 – Практична цінність одержаних результатів

Порівняльний аналіз аналогів

Критерій	Мій Гнівань	BUSFOR.UA	Poïzdato	Rozklad.in.ua	Вінницький трамвай	Залізниця
Розрахунок часу до прибуття наступного транспортного засобу	0	0	0	0	1	1
Реєстрування на сайті	1	1	0	1	1	1
Можливість переглянути маршрут на карті	0	0	0	1	0	0
Визначення можливої затримки транспорту	0	0	0	0	0	1
Зворотній звязок	0	1	1	0	1	1
Підсумковий результат	1	2	1	2	3	4

Рисунок В.7 – Порівняльний аналіз аналогів

Структура графічного інтерфейсу. Головне вікно додатку



1. Назва веб-додатку.
2. Пункт «Зворотній зв'язок».
3. Пункт «Контакти».
4. Пункт «Вхід».
5. Пункт «Реєстрування».
6. Кнопка «Маршрут».
7. Кнопка «Зупинка».
8. Кнопка «Тип потягу».
9. Кнопка «Тип дня».
10. Область вибору типу відображення розкладу.
11. Область відображення розкладу.
12. Назва області «Маршрут».
13. Список розкладу.
14. Панель перевірки затримки.
15. Область відображення новин.

Рисунок В.8 – Структура графічного інтерфейсу головного вікна додатку

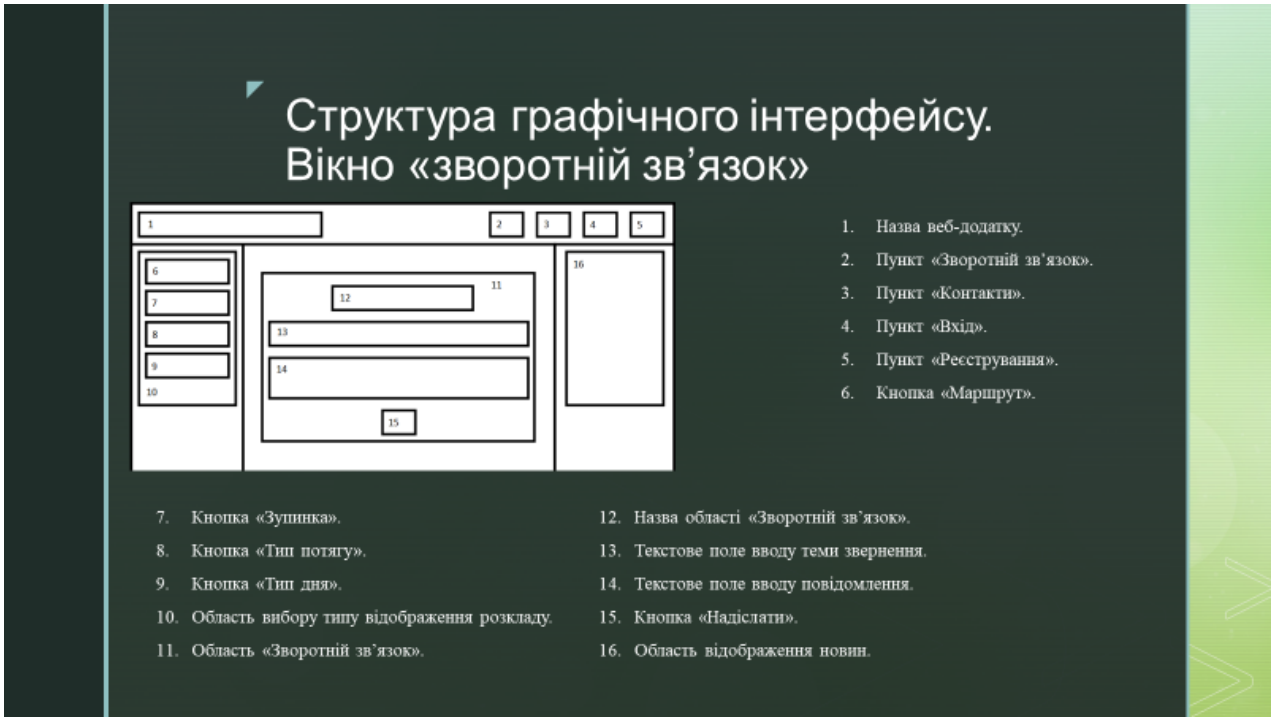


Рисунок В.9 – Структура графічного інтерфейсу вікна зворотного зв'язку

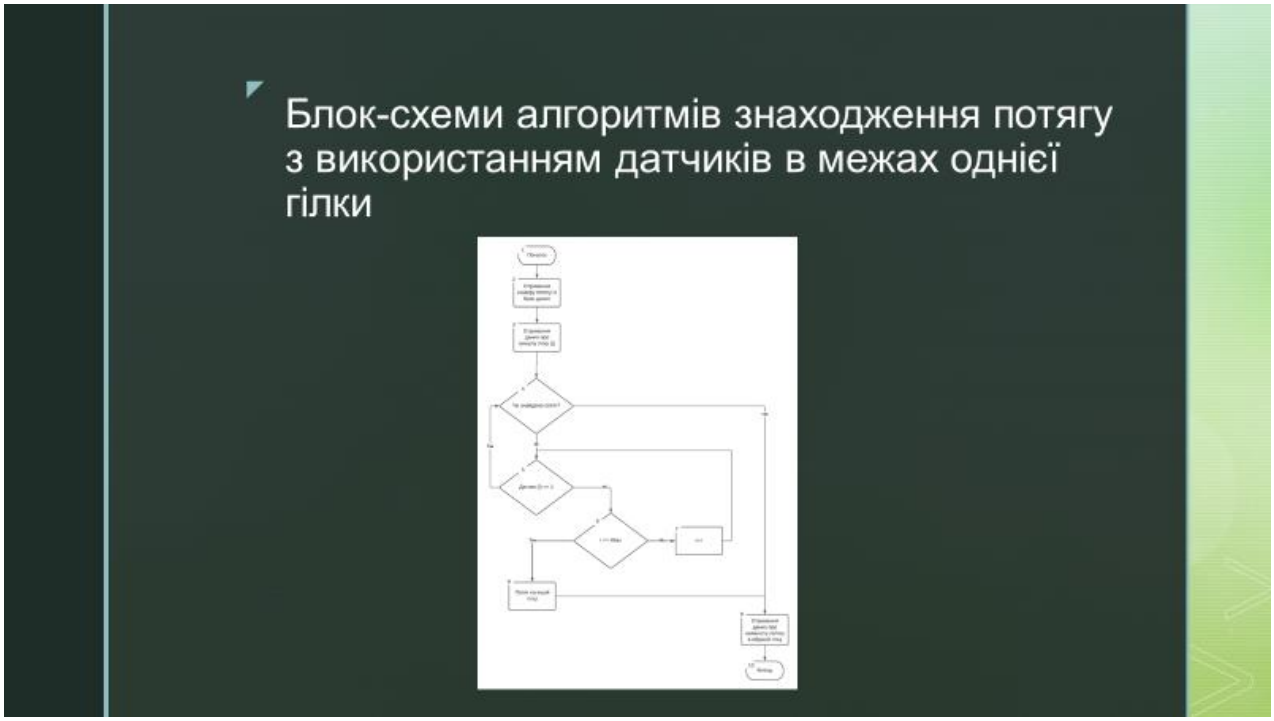


Рисунок В.10 – Блок-схеми алгоритмів знаходження потягу з використанням датчиків в межах однієї гілки

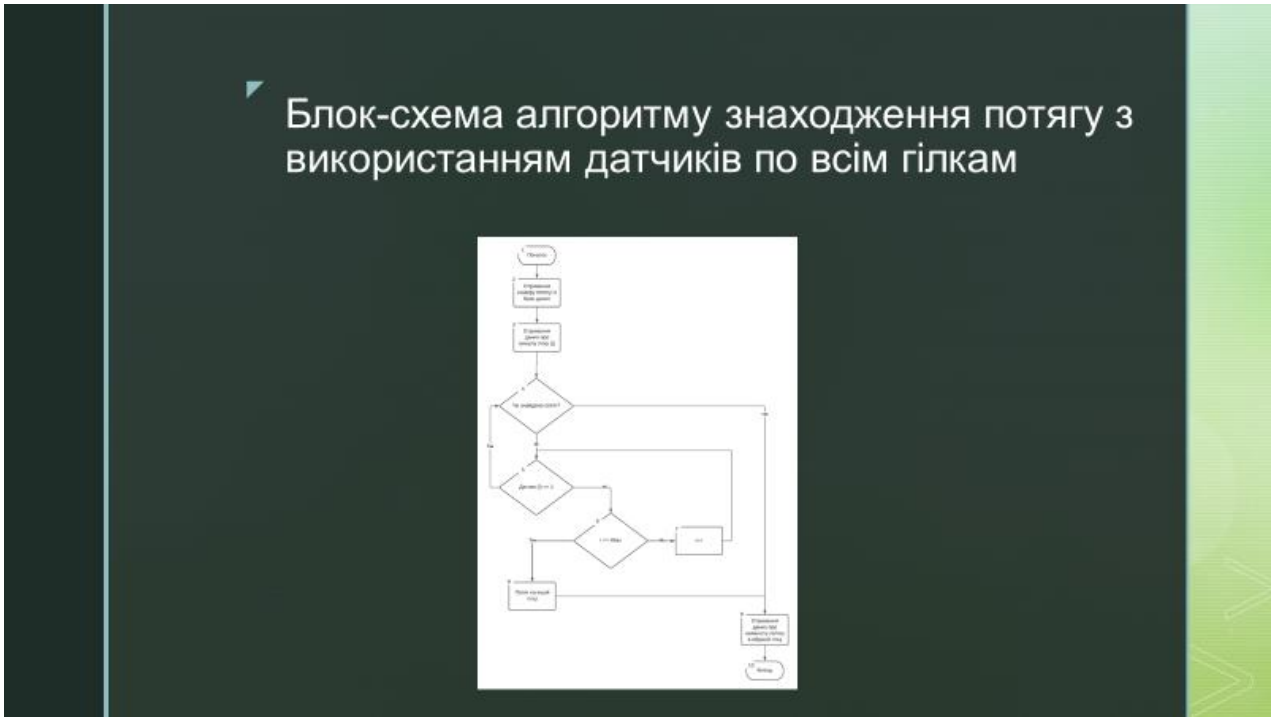


Рисунок В.11 – Блок-схема алгоритму знаходження потягу з використанням датчиків по всім гілкам

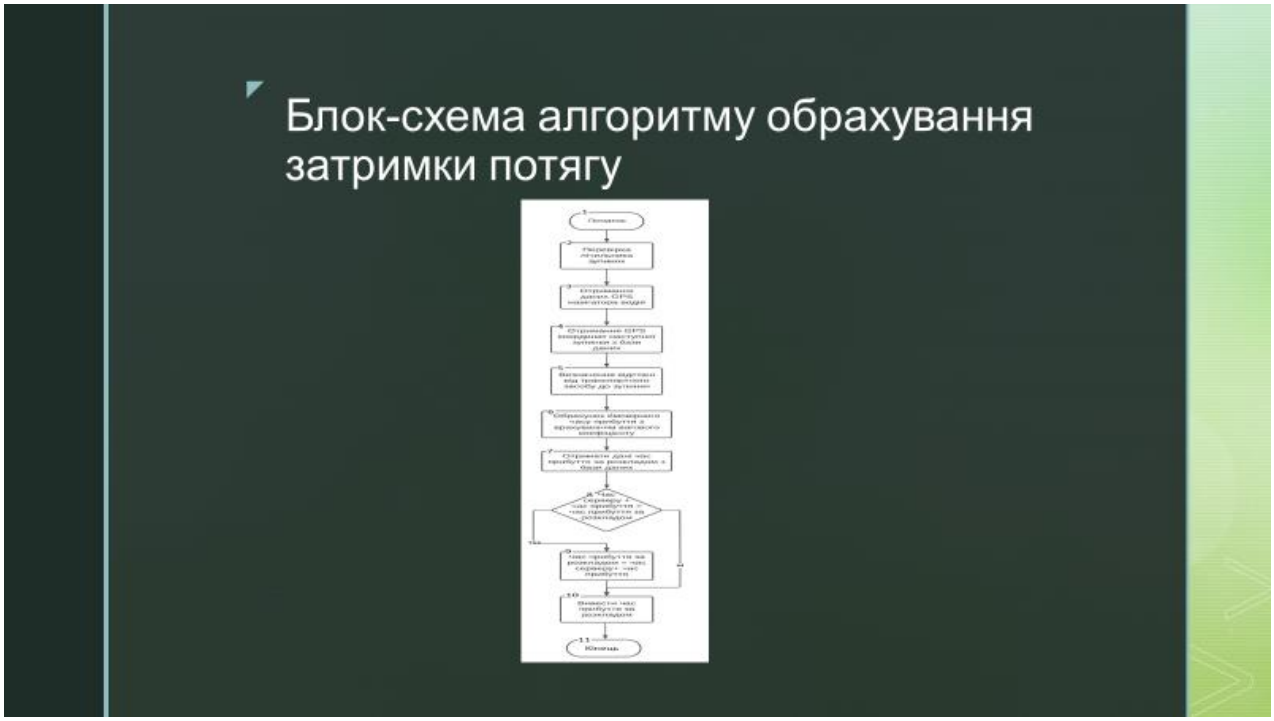


Рисунок В.12 – Блок-схема алгоритму обрахування затримки потягу

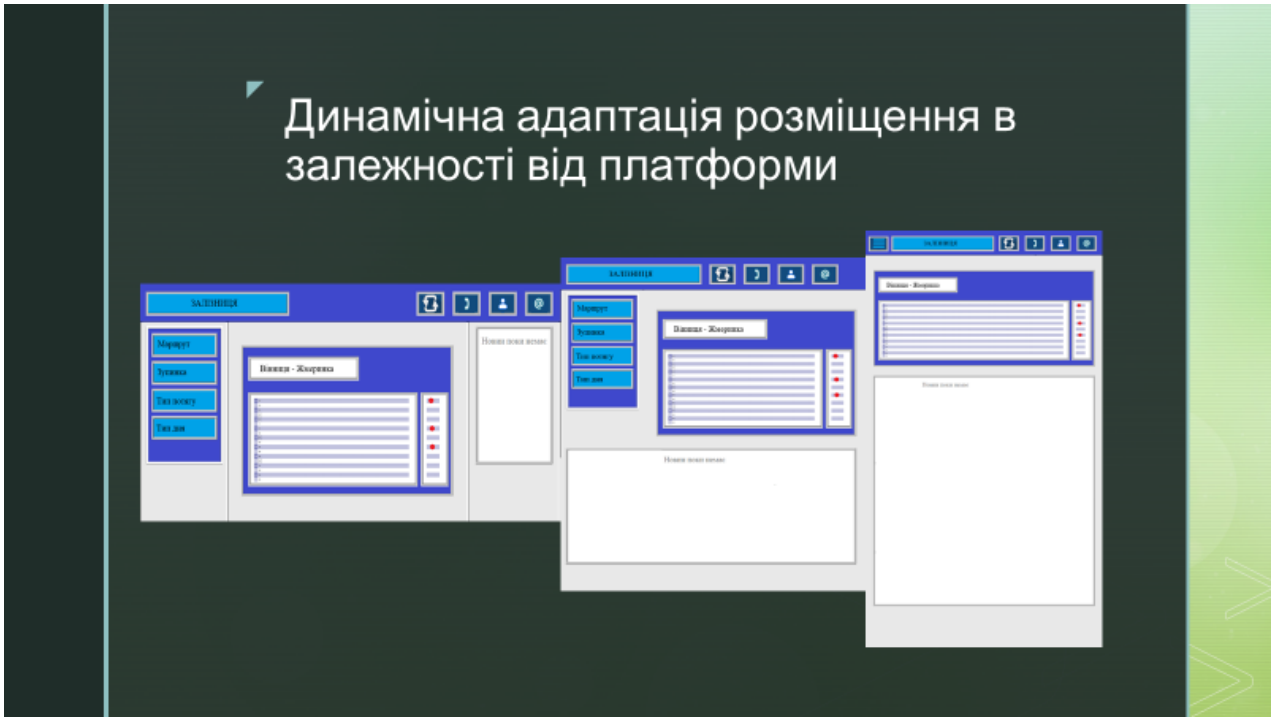


Рисунок В.13 – Динамічна адаптація розміщення в залежності від платформи

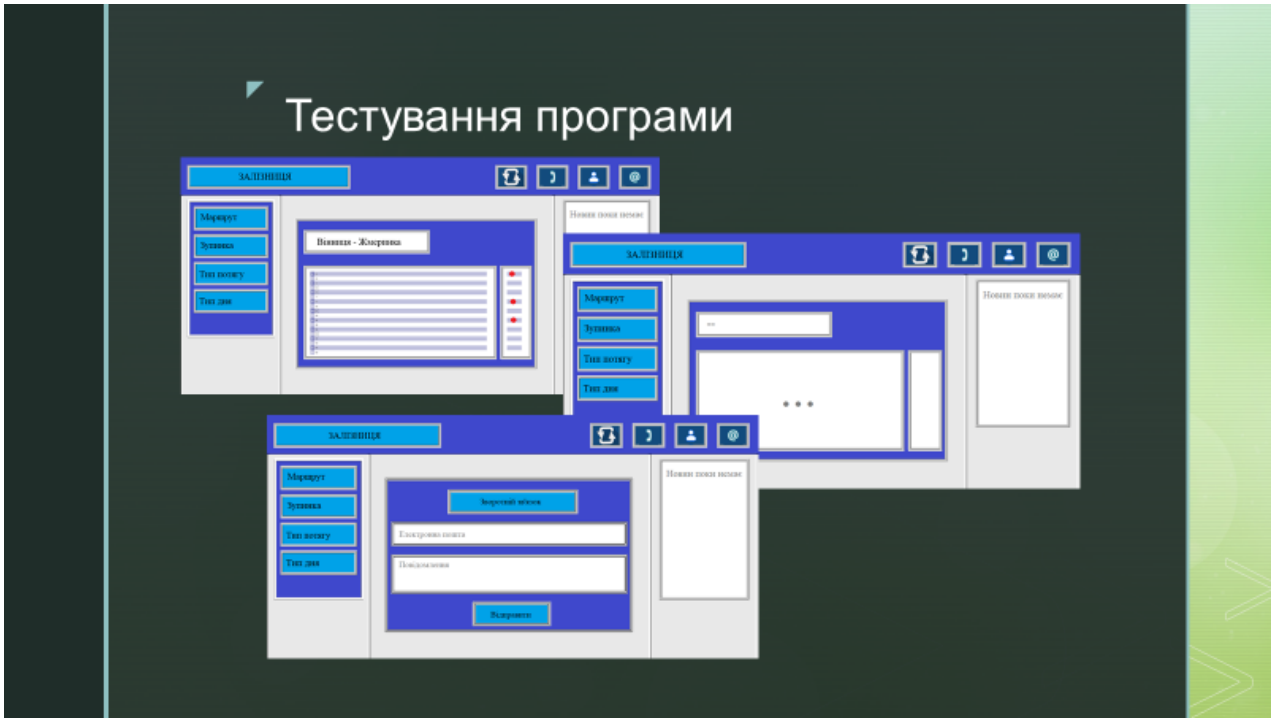


Рисунок В.14 – Тестування програми

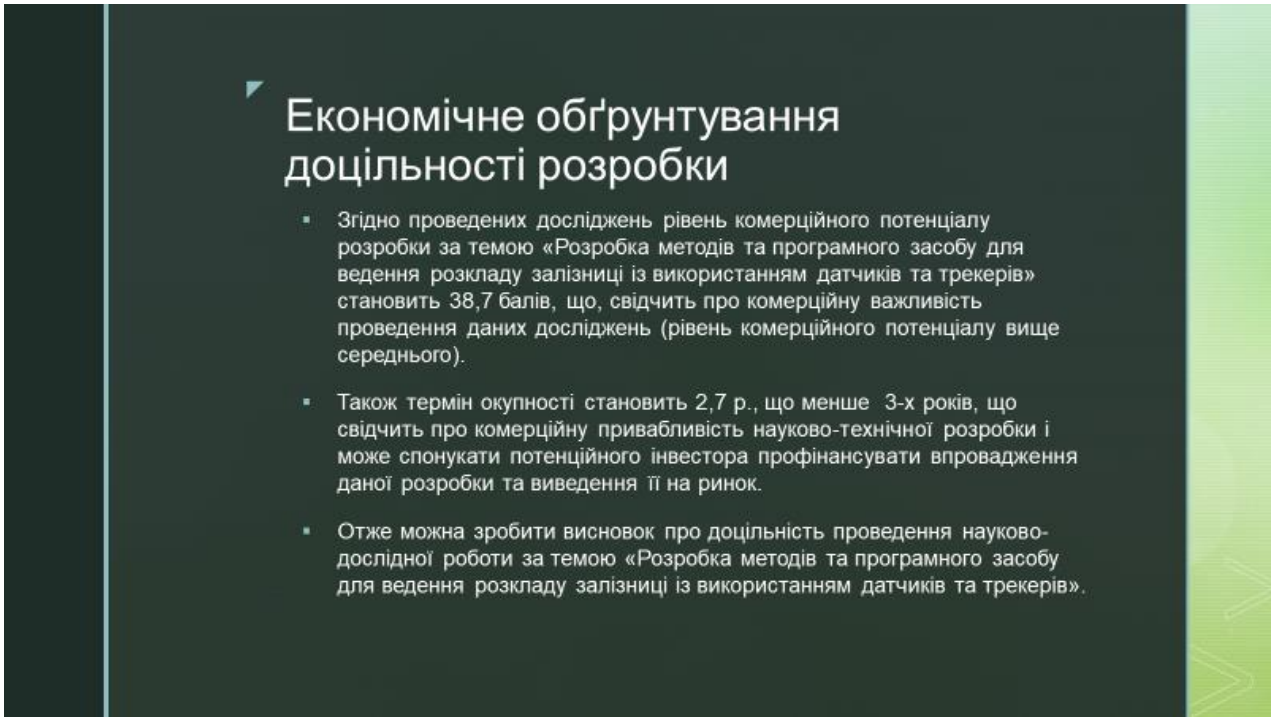


Рисунок В.15 – Економічне обґрунтування доцільності розробки

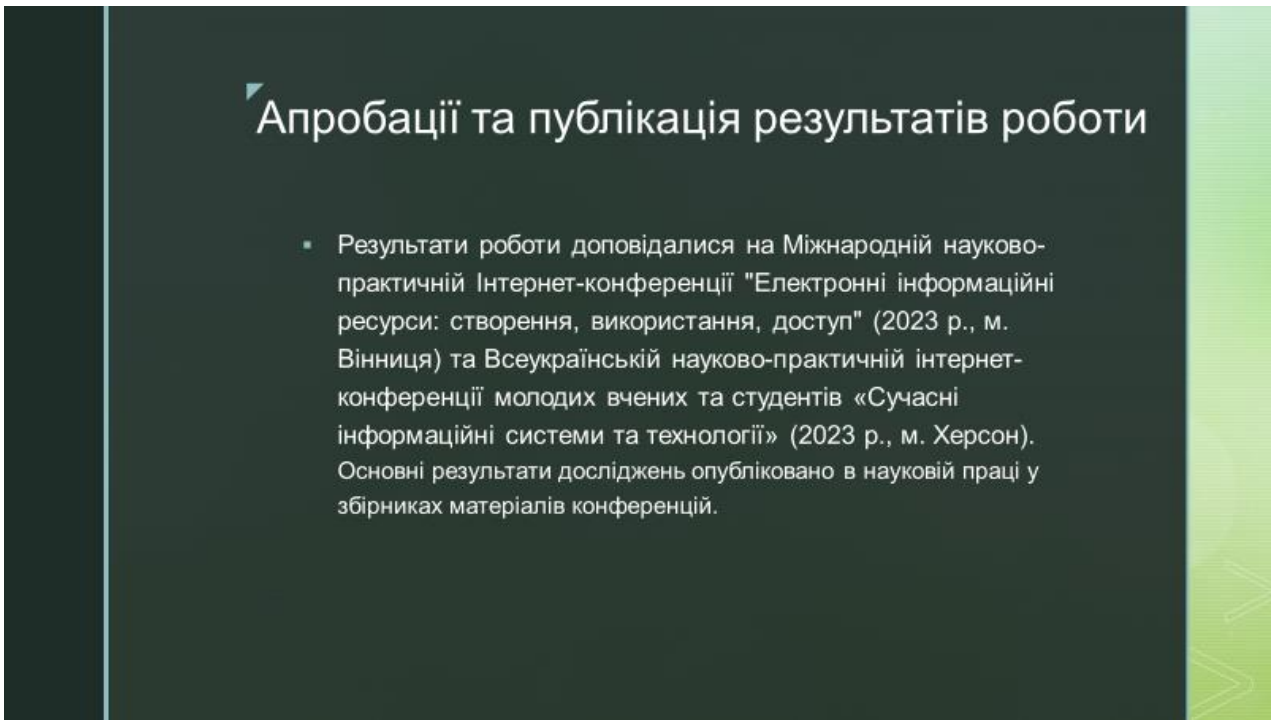


Рисунок В.16 – Апробації та публікації результатів роботи

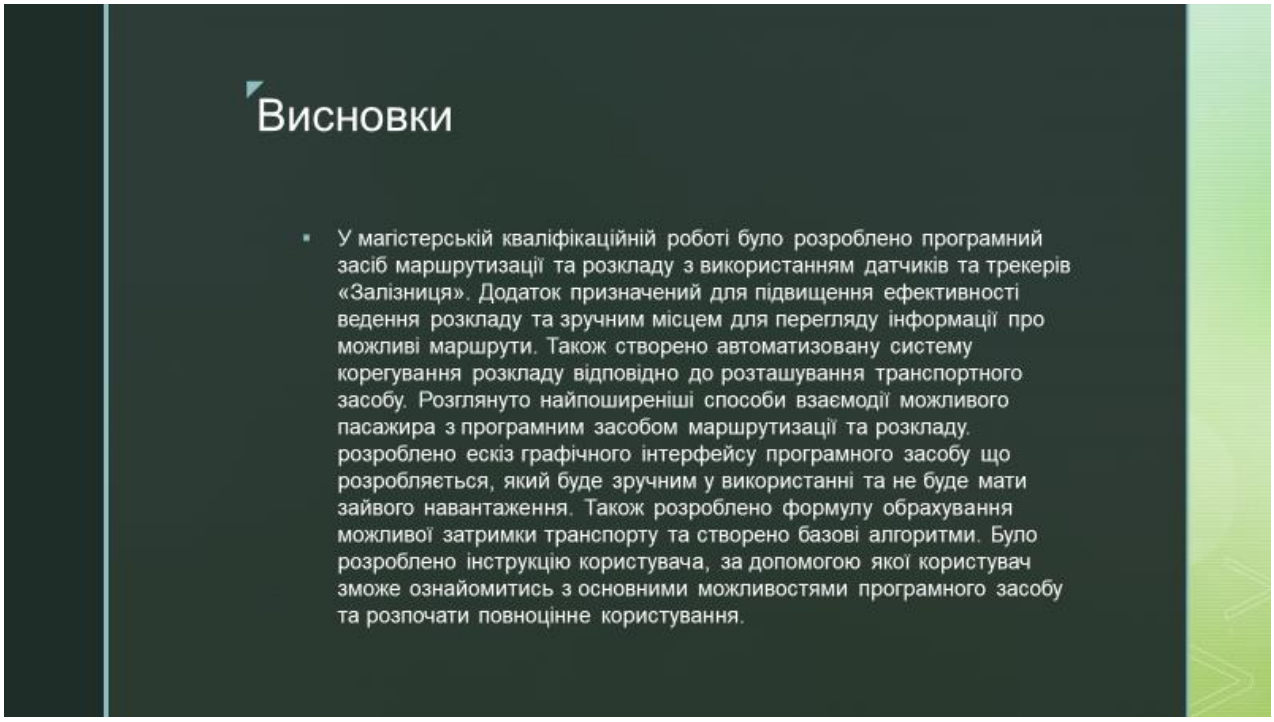


Рисунок В.17 – Висновки

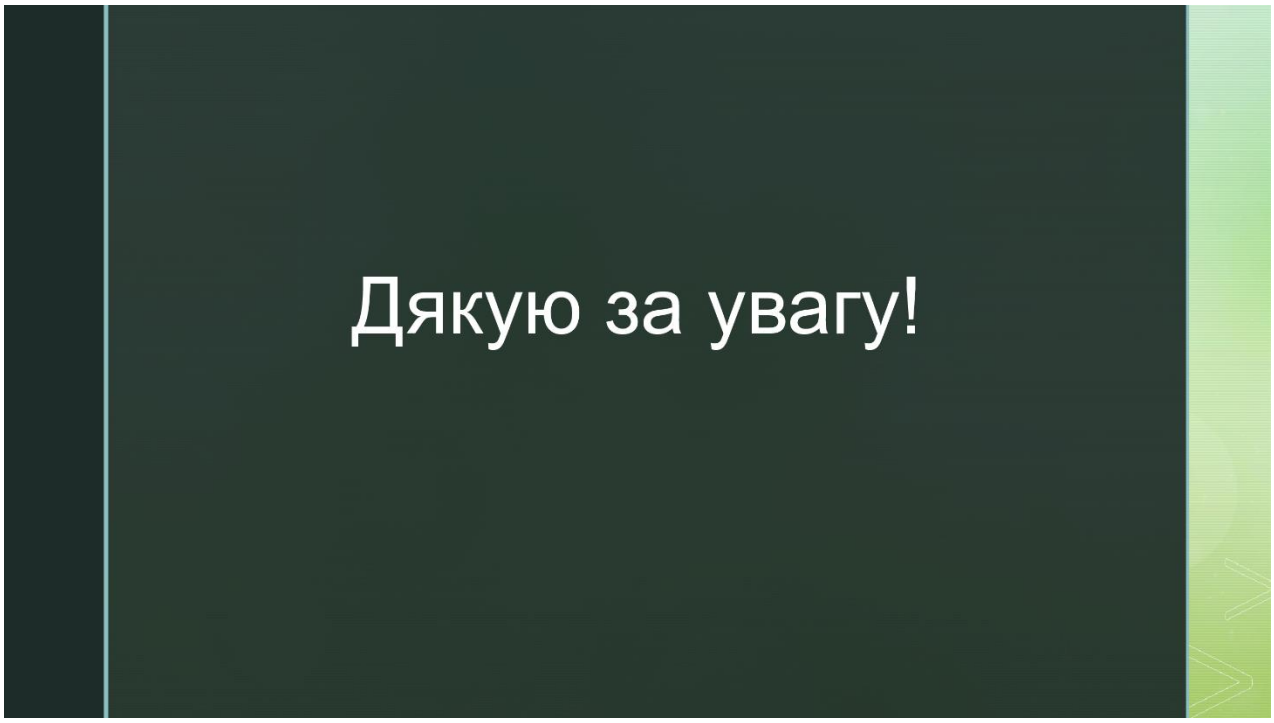


Рисунок В.18 – Фінальний слайд