Форма № 24

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій
(повне найменування факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)
бакалавр
(ступінь вищої освіти)

на тему ВЕБСЕРВІС ДЛЯ ВИРІШЕННЯ ТРАНСПОРТНИХ ЗАДАЧ
WEB SERVICE FOR SOLVING TRANSPORT PROBLEMS

Виконав(ла): студент(ка) 4 курсу, групи КНТ-130
Спеціальності 121 Інженерія програмного
(код і найменування спеціальності)
забезпечення

Освітня програма (спеціалізація)
Інженерія програмного забезпечення

ПЕЧЕРСЬКИЙ М.В.
(ПРІЗВИЩЕ та ініціали)
Керівник СТЕПАНЕНКО О.О.
(ПРІЗВИЩЕ та ініціали)
Рецензент ПОЛЯКОВ М.О.
(ПРІЗВИЩЕ та ініціали)

2024

Форма № 25

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Факультет ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації)

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
“____” _____2024 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ПЕЧЕРСЬКОГО Максима Вячеславовича
(ПРІЗВИЩЕ, ім’я, по батькові)

1. Тема проєкту (роботи) Вебсервіс для вирішення транспортних задач. Web Service for Solving Transport Problems
керівник проєкту (роботи) к.т.н., доцент, СТЕПАНЕНКО Олександр Олександрович,
(науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім’я, по батькові)
затверджені наказом закладу вищої освіти від “ 24 ” квітня 2024 року № 168
2. Строк подання студентом проєкту (роботи) 10 червня 2024 року

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

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз транспортних задач та існуючих методів розв’язання. 2. Вибір та обґрунтування структури системи, яка проєктується. 3. Опис програмного забезпечення. 4. Керівництво програміста. 5. Керівництво оператора.

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

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	СТЕПАНЕНКО О.О., доцент		
Нормоконтроль	ДЕЙНЕГА Л.Ю., ст. викладач		

7. Дата видачі завдання « 15 » квітня 2024 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання
2	Аналіз транспортних задач та існуючих методів розв’язання.	1 тиждень	Розділ 1
3	Вибір та обґрунтування структури системи, яка проєктується.	2 тиждень	Розділ 2
4	Опис програмного забезпечення	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділ 3, 4
6	Тестування та описання розробленої програми.	5 тиждень	Розділ 4, 5
	програмного забезпечення.		
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка) Підпис Максим ПЕЧЕРСЬКИЙ
(підпис) (Ім’я ПРІЗВИЩЕ)

Керівник проєкту (роботи) Олександр СТЕПАНЕНКО
(підпис) (Ім’я ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 119 с., 16 табл., 25 рис., 2 дод., 46 джерел.

ТРАНСПОРТНА ЗАДАЧА, ВЕБ-ЗАСТОСУНОК, ASP.NET, БАЗА-ДАНИХ, MVC-PATTERN.

Об’єкт дослідження – є процес розробки веб-застосунку, що буде реалізувати алгоритм розподілу вантажів, який допомагає зменшити витрати на транспортування вантажів з одного пункту до іншого.

Предмет дослідження – програмні засоби для вирішення проблеми розподілу вантажів, що дає змогу зменшити витрати на транспортування вантажів з одного пункту до іншого. Класична транспортна задача, яку вже розглядали раніше, полягає у пошуку оптимального плану перевезень однорідного вантажу з (m) пунктів відправлення A_1, A_2, …, A_m, у яких знаходиться відповідно a_1, a_2,…,a_m одиниць вантажу, у (n) пунктів призначення B_1, B_2, …, B_n, у кожний з яких потрібно завезти відповідно b_1, b_2, …, b_n. Вартість перевезення одиниць вантажу з пункту (A_i) відправлення у пункт (B_j) призначення відома і складає (C_{ij}).

Транспортна задача полягає у пошуку найбільш вигідного плану перевезення однорідного продукту з пунктів виробництва (чи зберігання) до пунктів споживання, тобто від постачальників до споживачів, ефективність якого будемо оцінювати за критерієм найменшої вартості перевезення. Транспортні витрати визначаються як вартість переїзду від будь-якої точки в будь-яку іншу точку.

Мета роботи – розробка програмного забезпечення, для вирішення транспортної задачі, яка дає можливість автомобільним компаніям доставляти грузи з мінімальними витратами.

Матеріали, методи та технічні засоби: мова програмування C++, платформа .Net, середовище розробки Visual Studio, фреймворк Asp.Net, СКБД EntityFramework із зв’язкою з Microsoft SQL Server.

Було використано такі методи дослідження, як емпіричні: спостереження, опис, вимірювання та експеримент та теоретичні: аналіз, порівняння, формалізація, моделювання, кожен метод допоміг розробити підхід до розробки застосунку.

Результати. Розроблений сервіс для вирішення транспортних задач з використанням сучасних технологій та підходів до побудови архітектури застосунку.

Висновки. Виконано проєктування програмного забезпечення, що реалізує алгоритм розподілу вантажів. Розроблено програмне забезпечення, що реалізує алгоритм розподілу вантажів. Здійснено тестування розробленого програмного забезпечення, що реалізує алгоритм розподілу вантажів.

Галузь використання – розв’язання логістичних задач перевезення вантажів.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 119 pages, 16 tables, 25 figures, 2 appendixes, 46 sources.

TRANSPORT TASK, WEB APPLICATION, ASP.NET, DATABASE, MVC-PATTERN.

The object of research is the process of developing a web application that will implement a cargo distribution algorithm that helps reduce the cost of transporting goods from one point to another.

The subject of the study is software tools for solving the problem of cargo distribution, which helps to reduce the cost of transporting goods from one point to another. The classical transport problem, which has already been considered earlier, is to find an optimal plan for transporting a homogeneous cargo from (m) points of departure A_1, A_2, ..., Am, which have a_1, a_2, ...,am units of cargo, to (n) destinations B_1, B_2, ..., B_n, each of which needs to be delivered b_1, b_2, ..., bn, respectively. The cost of transporting a unit of cargo from point (A_i) of origin to point (B_j) of destination is known and is C_{ij}.

The transport problem is to find the most profitable plan for transporting a homogeneous product from points of production (or storage) to points of consumption, i.e. from suppliers to consumers, the effectiveness of which will be evaluated by the criterion of the lowest cost of transportation. Transportation costs are defined as the cost of moving from any point to any other point.

The aim of the work is to develop software to solve the transport problem, which allows road transport companies to deliver goods at minimal cost.

Materials, methods and tools: C++ programming language, .Net platform, Visual Studio development environment, Asp.Net framework, EntityFramework database with connection to Microsoft SQL Server.

The research methods used were empirical: observation, description, measurement, and experiment, and theoretical: analysis, comparison,

formalisation, modelling, each method helped to develop an approach to application development.

Results. A service for solving transport problems has been developed using modern technologies and approaches to building an application architecture.

Conclusions. The design of software that implements the cargo distribution algorithm has been completed. Software implementing the cargo distribution algorithm has been developed. The developed software implementing the cargo distribution algorithm was tested.

The field of application is solving logistics problems of cargo transportation.

ЗМІСТ

С.

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

IDE – Integrated Development Environment;

MVC – Model-View-Controller;

UI – User Interface;

VS – Visual Studio;

БД – База-даних;

ПЗ – Програмне забезпечення;

СКБД – Система керування базою даних;

ТЗ – Транспортна задача;

ІК – Інтерфейс користувача.

ВСТУП

Транспортна задача - це особливий тип задач лінійного програмування, де метою є мінімізація вартості розподілу продукту з декількох джерел або пунктів призначення до декількох пунктів призначення. Через свою особливу структуру звичайний симплекс-метод не підходить для розв'язання транспортних задач. Ці проблеми вимагають спеціального методу вирішення. Початок транспортної задачі - це місце, з якого відправляються вантажі. Пункт призначення транспортної задачі - це місце, куди відправляються вантажі. Вартість перевезення одиниці вантажу - це вартість перевезення однієї одиниці вантажу з пункту відправлення до пункту призначення [1].

В сучасному світі підприємства та організації постійно стикаються з проблемою транспортних задач. Ці задачі пов'язані з оптимізацією використання ресурсів, зменшенням витрат на перевезення товарів та забезпеченням максимальної ефективності роботи підприємства.

Мета роботи і завдання дослідження. Метою є розробка веб-сервісу, який дозволить вирішувати транспортні задачі шляхом автоматизації обробки та аналізу даних про постачальників, споживачів та вартості перевезення товарів.

Веб-сервіс буде розроблений на платформі ASP.Net. У роботі будуть розглянуті такі питання, як аналіз вимог користувачів, розробка архітектури веб-сервісу, розробка бази даних та інтерфейсу користувача, вибір та розробка алгоритму розв'язання транспортної задачі, тестування та налагодження.

Інформаційне наповнення веб-застосунку буде зберігатися у базі даних (БД), що зробить адміністрування більш зручним та надасть більший арсенал захисту даних із поєднанням надійності зберігання.

Розробка веб-сервісу дозволить підприємствам та організаціям вирішувати транспортні задачі більш ефективно та раціонально, що призведе

до зниження витрат на перевезення товарів та підвищення ефективності роботи підприємства.

Нижче наведені деякі додаткові переваги використання веб-сервісу для вирішення транспортних задач:

– зниження витрат на перевезення товарів;

– підвищення ефективності роботи підприємства;

– зручність адміністрування;

– надійність та безпечність зберігання даних з урахуванням сучасного підходу ґешування;

– широкий спектр можливостей для налаштування поданих даних, що призведе до розрахунку мінімальної кількості палива.

Веб-сервіс є корисним інструментом для вирішення транспортних задач. Він може допомогти підприємствам та організаціям заощадити гроші, підвищити ефективність роботи та забезпечити надійне зберігання даних. На сьогодні програмне забезпечення, що розв’язує транспорту задачу використовується тисячами компаній, зокрема як цивільними так і військовими, у логістичних підприємствах, галузях промисловості та діяльності.

1 АНАЛІЗ ТРАНСПОРТНИХ ЗАДАЧ ТА ІСНУЮЧИХ МЕТОДІВ РОЗВ'ЯЗАННЯ

1.1 Класифікація транспортних задач

Існують два види транспортної задачі:

– транспортна задача по критерію вартості – треба знайти план перевезень з мінімальною вартістю перевезень [2];

– транспортна задача по критерію часу – в якій більш важливим параметром є час перевезення вантажів. Транспортна задача по критерію вартості є окремим випадком задачі лінійного програмування і розв'язується симплекс методом [2].

Транспортні задачі можна класифікувати за такими ознаками:

– кількість джерел і пунктів призначення: транспортні задачі можуть бути одновимірними (одне джерело і один пункт призначення), двовимірними (два джерела і два пункти призначення) або багатовимірними (декілька джерел і декілька пунктів призначення) [3];

– наявність обмежень на обсяги перевезень: транспортні задачі можуть бути з обмеженнями на обсяги перевезень (тобто кожне джерело може перевезти не більше певного обсягу ресурсів, а кожен пункт призначення може прийняти не більше певного обсягу ресурсів) або без обмежень на обсяги перевезень [4];

– наявність обмежень на маршрути перевезень: транспортні задачі можуть бути з обмеженнями на маршрути перевезень (тобто ресурси можуть перевозитися тільки певними маршрутами) або без обмежень на маршрути перевезень [5].

Найбільш поширеним типом транспортних задач є одновимірні задачі з обмеженнями на обсяги перевезень. Ці задачі можна вирішити за допомогою методу простого кроку.

1.2 Огляд існуючих методів вирішення завдання

Для вирішення поставлених задач існує велика кількість методів, які відрізняються за багатьма критеріями, такими як сумісність, зручність та ефективність застосування інструментів, що пропонуються, а також наявність актуальної літератури.

Метою даної роботи є розробка веб-сервісу для вирішення транспортної задачі, яка дозволяє автомобільним компаніям доставляти вантажі з мінімальними витратами.

Зокрема, для визначення технології побудови архітектури застосунку необхідно розглянути велику кількість існуючих фреймворків, таких як Django, Express.js, Flask, Spring, ASP.NET та CakePHP. Однак, незважаючи на різноманіття назв, в першу чергу необхідно визначити головний шаблон архітектури, яких також є багато, таких як MVC, Front Controller, Data Mapper, Active Record та DAO. Отже, вибір технології базується на багатьох факторах [6].

Для вирішення транспортної задачі також існує багато методів, таких як метод потенціалів, симплекс-метод, метод північно-західного кута, метод мінімального елементу, метод апроксимації Фогеля, метод диференціальних рент та інші. Кожен метод має свою особливість, але однією з задач сервісу є надання можливості вирішення актуальними методами з поміж всіх існуючих [7].

Питання збереження даних веб-застосунку також потрібно розглядати з зауваженням декількох аспектів, починаючи з визначення моделі бази даних: ієрархічна, мережна, реляційна. Кожна має як свої переваги так і недоліки, тому є необхідність точного визначення сценарію використання, для подальшого виявлення якомога ефективнішої моделі БД, далі необхідно визначити фреймворк (систему керування базою даних), яких також велика кількість: Spring Boost, DAO, ADO.Framework, Real, Entity Framework та

інші. Обране СКБД має надійно забезпечувати з’єднання, зберігання та управління даними [8].

1.3 Опис класичної транспортної задачі

Транспортна задача – це задача оптимізації, яка полягає в тому, щоб знайти оптимальний розподіл ресурсів між джерелами та пунктами призначення. Вона виникає в багатьох різних областях, таких як логістика, виробництво та енергетика.

Транспортна задача може бути описана матрицею, що зображено на рисунку 1.1.

	A	B	D			E	F	G	H	I	
1			V-prices			5		5			
				V- costs		P1		P2			
										V-Profit	
2	S1	10	Supply	1	C1	4	10	6	0	0	-1
3	S2	30		0	C2	3	20	5	*10*	0	0
4				Demands			20		20		
							D1		D2		

Рисунок 1.1 – Матриця опису транспортної задачі [9]

Кожне поле матриці містить кількість ресурсів, які можна перевезти з джерела в пункт призначення. Важливо, щоб сума ресурсів, які можна перевезти з кожного джерела, була рівною сумі ресурсів, які потрібно доставити в кожен пункт призначення [9].

1.3.1 Постановка задачі

Загальна постановка задачі зводиться до пошуку оптимального плану перевезень однорідного вантажу з (m) пунктів відправлення A_1, A_2, …, A_m, у яких знаходиться відповідно a_1, a_2,…,a_m одиниць вантажу, у (n) пунктів призначення B_1, B_2, …, B_n, у кожний з яких потрібно завезти відповідно b_1, b_2, …, b_n. Вартість перевезення одиниць вантажу з пункту (A_i) відправлення у пункт (B_j) призначення відома і складає (C_{ij}).

1.3.2 Математична модель задачі

Математична модель транспортної задачі полягає в мінімізації функції (1.1):

$$F = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \rightarrow min, \quad (1.1)$$

де m – кількість пунктів відправлення вантажів;

n – кількість пунктів призначення;

c_{ij} – транспортний тариф;

x_{ij} – об’єм перевезень.

Зважаючи на умови(1.2, 1.3, 1.4):

$$\sum_{j=1}^{n} x_{ij} = a_i, i = \overline{1,m}, \quad (1.2)$$

$$\sum_{i=1}^{m} x_{ij} = b_j, j = \overline{1,n}, \quad (1.3)$$

$$x_{ij} \geq 0, i = \overline{1,m}, j = \overline{1,n}, \quad (1.4)$$

де m – кількість пунктів відправлення вантажів;

n – кількість пунктів призначення;

x_{ij} – об’єм перевезень;

a_i – обсяги вантажу;

b_j – велечина заявок вантажу.

Будь-яке невід’ємне рішення системи лінійних рівнянь (1.2), (1.3), яке визначається матрицею X, називається планом перевезень транспортної задачі. План перевезень, що має не більше m + n - 1 відмінних від нуля змінних xij, називається опорним. Якщо число відмінних від нуля компонент в опорному плані, в точності, дорівнює m + n - 1, то план перевезень називається невироджених, якщо менше цього числа, то виродженим [10].

План перевезень X*, при якому функція (1.1) приймає своє мінімальне значення, називається оптимальним планом транспортної задачі [11].

На практиці при перевезенні вантажів можуть виникати такі ситуації:

Кількість одиниць вантажу у постачальників відповідає заявками або попиту з боку споживачів:

$$\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j, \tag{1.5}$$

де m – кількість пунктів відправлення вантажів;

n – кількість пунктів призначення;

a_i – обсяги вантажу;

b_j – велечина заявок вантажу.

Також не можна не зазначити, що дана транспортна задача – є збалансованою, а її математична модель (що зазначена у виразах 1.1 – 1.4) – є закритою.

Потреби в поточному вантажі у споживачі менші, ніж кількість кількість вантажу всіх наявних постачальників:

$$\sum_{i=1}^{m} a_i < \sum_{j=1}^{n} b_j, \tag{1.6}$$

де m – кількість пунктів відправлення вантажів;

n – кількість пунктів призначення;

a_i – обсяги вантажу;

b_j – велечина заявок вантажу.

В наведеному вище випадку, є дисбаланс між пропозицією вантажу постачальниками та попитом споживачів, який виражений у тому, що де-яка частина вантажу постачальників залишається у залишку, проте потреби споживачів повністю задовольняються.

Тож можна зазначити, що математична модель транспортної задачі такого типу буде мати наступний вигляд:

$$\left.\begin{array}{c} F = \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \to min \\ \sum_{j=1}^{n} x_{ij} \le a_i, i = \overline{1,m} \\ \sum_{i=1}^{m} x_{ij} = b_j, j = \overline{1,n} \\ x_{ij} \ge 0, i = \overline{1,m}, j = \overline{1,n} \end{array}\right\}, \quad (1.7)$$

де m – кількість пунктів відправлення вантажів;

n – кількість пунктів призначення;

a_i – обсяги вантажу;

b_j – велечина заявок вантажу;

c_{ij} – транспортний тариф;

x_{ij} – об’єм перевезень.

У випадку коли постачальники не мають достатньої кількості вантажу, щоб задовольнити потреби споживачів:

$$\sum_{i=1}^{m} a_i < \sum_{j=1}^{n} b_j, \quad (1.8)$$

де m – кількість пунктів відправлення вантажів;

n – кількість пунктів призначення;

a_i – обсяги вантажу;

b_j – велечина заявок вантажу.

У даному випадку кожен постачальник зможе продати весь свій вантаж, але частина споживачів отримає менше вантажу, ніж їм потрібно. У моделі транспортної задачі це буде відображено тим, що деякі клітинки таблиці будуть мати значення 0, що означає, що в них не буде жодних перевезень:

$$\left.\begin{array}{c} F = \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij} \cdot x_{ij} \to min \\ \sum_{j=1}^{n} x_{ij} = a_i, i = \overline{1,m} \\ \sum_{i=1}^{m} x_{ij} \geq b_j, j = \overline{1,n} \\ x_{ij} \geq 0, i = \overline{1,m}, j = \overline{1,n} \end{array}\right\}, \quad (1.9)$$

де m – кількість пунктів відправлення вантажів;

n – кількість пунктів призначення;

a_i – обсяги вантажу;

b_j – велечина заявок вантажу;

c_{ij} – транспортний тариф;

x_{ij} – об’єм перевезень.

Настав час зазначити, що математичні моделі, які наведено у виразах 1.7 та 1.9 – мають назву «відкриті» і відповідні їм задачі – незбалансованими. Зважаючи на це, їх необхідно привести до закритого типу.

Тут можливо декілька випадків, в тому разі, якщо відбувається перевищення загального попиту в порівнянні з заказами (вираз 1.3), тоді приведення здійснюється шляхом додавання фіктивного (або умовного) постачальника A_{m+1} з ресурсним запасом:

$$a_{m+1} = \sum_{j=1}^{m} b_j - \sum_{i=1}^{n} a_i, \quad (1.10)$$

де m – кількість пунктів відправлення вантажів;

n – кількість пунктів призначення;

a_i – обсяги вантажу;

b_j – величина заявок вантажу.

Якщо загальні запаси постачальників перевищують попит споживачів (як це показано у другому виразі), то задача може бути перетворена на закриту задачу шляхом додавання фіктивного споживача B_{n+1} з потребами, рівними різниці між загальними запасами постачальників і попитом споживачів:

$$b_{n+1} = \sum_{i=1}^{m} a_i - \sum_{j=1}^{n} b_j, \tag{1.11}$$

де m – кількість пунктів відправлення вантажів;

n – кількість пунктів призначення;

a_i – обсяги вантажу;

b_j – величина заявок вантажу.

Вартість перевезення одиниці продукції від фіктивного постачальника (або споживача) до кожного зі споживачів (постачальника) дорівнює 0 або є набагато більшою за реальні витрати. Це робиться для того, щоб забезпечити баланс між пропозицією і попитом і мінімізувати загальні витрати на перевезення.

1.3.3 Аналіз класичних алгоритмів розв’язання транспортних задач

Для розв’язання транспортних задач, існує велика кількість класичних алгоритмів, кожен з яких має свої переваги та недоліки, зокрема відрізняється ефективністю розрахунків.

Метод подвійної переваги – згідно з процедурою цього методу перед початком заповнення таблиці необхідно позначити будь-якими символами

клітинки, які містять найменшу вартість у рядках, а потім — у стовпчиках. Таблицю починають заповнювати з клітинок, позначених двічі (які містять вартості, що є мінімальними і в рядку, і в стовпчику). Далі заповнюють клітинки, позначені один раз (що містять мінімальні вартості або в рядку, або в стовпчику), а вже потім — за методом мінімальної вартості [12].

Мінімального елементу – цей метод є варіантом методу північно-західного кута, який враховує особливості матриці транспортних витрат $C=c_{ij}$. Він дає змогу побудувати відразу досить економічний план, що скорочує загальне число ітерацій при його подальшій оптимізації (порівняно з методом північно-західного кута). Формальний опис алгоритму полягає в тому, що елементи матриці нумерують, починаючи з мінімального, в порядку збільшення і далі в цьому ж порядку заповнюють матрицю X_0. Нехай елементом з мінімальним порядковим номером виявився $x_{ij}=\min\{a_i,b_i\}$. Можливі три випадки: а) якщо $a_i<b_i$, то незаповнений лишок і-го рядка заповнюємо нулями; б) якщо $a_i>b_i$, то незаповнену частину j-го стовпця заповнюємо нулями; в) якщо $a_i=b_i$, то лишок і-го рядка і j-го стовпця заповнюємо нулями. Далі цей процес повторюють з незаповненою частиною матриці X_0 [13].

Північно-західного кута – ідея цього методу полягає в тому, що заповнення таблиці починають, не враховуючи вартостей перевезень, з лівого верхнього (північно-західного) кута. У клітину записують менше з двох чисел a1 та b1. Далі переходять до наступної клітини в цьому ж рядку або у стовпчику і заповнюють її, і т. д. Закінчують заповнення таблиці у правій нижній клітинці. У такий спосіб значення поставок будуть розташовані по діагоналі таблиці [14].

Апроксимація Фогеля – на кожному кроці знаходять по всіх стовпчиках і всіх рядках різниці між записаними в них мінімальними тарифами. Ці різниці записують в спеціально виділені для цього стовпчики і рядки. Зі знайдених різниць вибирають максимальну. В стовпчику (чи в рядку), якому ця різниця відповідає, знаходять мінімальний тариф. Клітку, в

який він записаний заповнюють на цьому кроці. Якщо мінімальний тариф однаковий для кількох клітин даного рядка (стовпчика), то для заповнення вибирають ту клітину, яка розташована в стовпчику (рядку), що відповідає найбільшій різниці між двома мінімальними тарифами, що знаходяться в цьому стовпці (рядку) [15].

Метод потенціалів – метод потенціалів - це спеціальний метод для знаходження оптимального плану транспортної задачі. Метод потенціалів використовується для розв'язання транспортної задачі. Основою обчислювального процесу при покращенні опорного плану є визначення критерію оптимальності d_{ij}: $d_{ij} = c_{ij}^* - c_{ij}$ [16].

Алгоритм Флойда-Уоршела – це алгоритм, який дозволяє знайти найкоротші шляхи між будь-якими двома вершинами в зваженому графі. Цей алгоритм можна використовувати для вирішення транспортної задачі, оскільки транспортна задача може бути описана як граф, в якому вершини представляють джерела і споживачі, а ребра представляють можливі перевезення. Для вирішення транспортної задачі за допомогою алгоритму Флойда-Уоршела необхідно виконати такі кроки, як обчислення найкоротшого шляху між кожної парою джерела та споживача, а також вибрати найкоротший шлях з мінімальними витратами [17].

Генетичний алгоритм – це евристичний алгоритм пошуку, який використовується для розв'язання задач оптимізації та моделювання шляхом випадкового підбору, комбінування та варіювання шуканих параметрів з використанням механізмів, аналогічних природному відбору в природі. Є дослідження, що пропонують застосування генетичного алгоритму для розв'язання транспортної задачі. Наприклад, у науковій статті «Генетичний алгоритм для вирішення транспортних задач» В.Н. Луценко описується застосування генетичного алгоритму для розв'язання транспортної задачі. Також існує евристичний алгоритм, що дозволяє генерувати оптимальні розклади для транспортних перевезень. Цей алгоритм повністю задовольняє умови розв'язання логістичних задач і має хороші результати [18].

Даний алгоритм працюють із популяцією розв’язків-особин, а не лише з одним розв’язком. Дана особливість дозволяє їм здійснювати багатосторонній пошук одночасно. Кожна особина представляє потенційний розв'язок задачі.

Створення особини нового покоління включає три основні етапи:

– етап відбору - випадковий вибір двох батьків з популяції для спаровування. Ймовірність вибору особини пропорційна її придатності, яка є мірою того, наскільки добре вона відповідає заданим критеріям;

– етап рекомбінації - обмін генами між двома батьками для отримання потомства;

– етап мутації - випадкова модифікація генів особини для забезпечення різноманітності популяції.

Нове покоління створюється шляхом повторення процесів відбору, рекомбінації та мутації, поки всі хромосоми в новій популяції не замінять хромосоми зі старої.

Для ефективного пошуку генетичного алгоритму необхідний дотримуватися належного балансу між генетичною якістю та різноманітністю серед популяції.

1.4 Огляд різновидів транспортних задач

Основною метою транспортної задачі є визначення такого плану перевезень, при якому загальні витрати на перевезення будуть мінімальними.

Залежно від конкретної ситуації, можуть виникати різні різновиди транспортних задач.

Найбільш поширеними є наступні види.

Класична транспортна задача – у цьому випадку існує певна кількість джерел, які мають певні запаси ресурсів, і певна кількість споживачів, які мають певні потреби в цих ресурсах. Кожне джерело може постачати ресурси лише одному споживачу, а кожен споживач може

отримувати ресурси лише від одного джерела. При цьому загальна кількість ресурсів, доступних у всіх джерелах, дорівнює загальній кількості ресурсів, що необхідні усім споживачам [12].

Транспортна задача з нелінійними витратами – у даному випадку витрати на перевезення не є лінійними функціями від обсягу перевезень.

Транспортна задача з дискретними витратами – в цьому випадку витрати на перевезення є дискретними величинами.

1.5 Огляд існуючих аналогів

1.5.1 Сервіс reshmat

Сервіс reshmat – є онлайн калькулятором для вирішення математичних задач в тому числі транспортної задачі,наведено у рисунку 1.2.

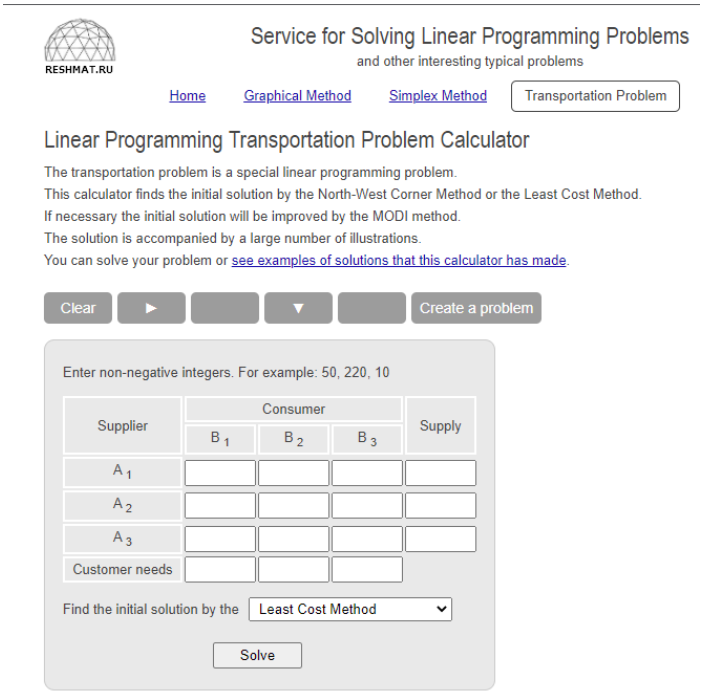


Рисунок 1.2 – Сервіс reshmat[20]

Основними перевагами є можливість змінювати розмір матриці, а також обирати методи знаходження базового розв’язку: мінімальної вартості, північно-західного кута.

Основні недоліками є відсутність можливості пошуку оптимального рішення.

1.5.2 Сервіс matworld

Сервіс matworld – є онлайн калькулятором для вирішення математичних задач в тому числі транспортної задачі, наведено у рисунку 1.3.

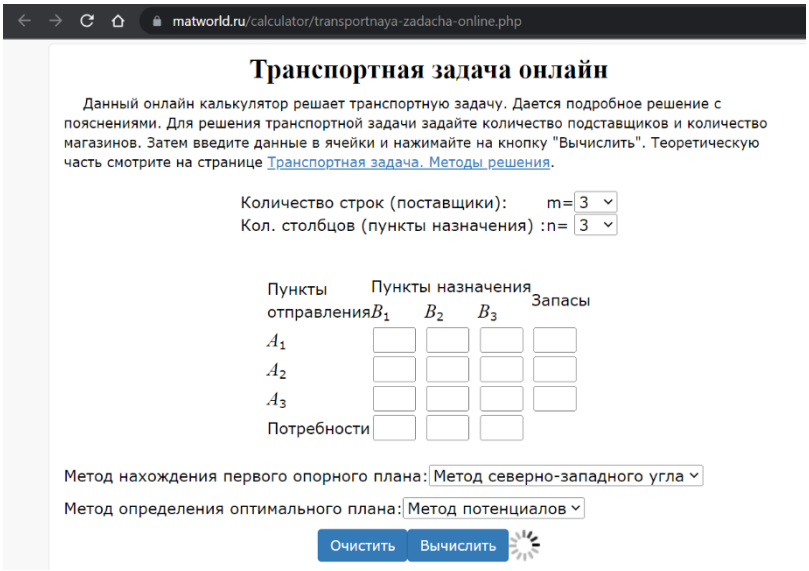


Рисунок 1.3 – Сервіс matworld [21]

Основними перевагами є можливість змінювати розмір матриці, а також обирати методи знаходження базового розв’язку та оптимального рішення.

Основні недоліками є невелика кількість представлених методів.

1.5.3 Сервіс math-pr.com

Сервіс math-pr.com – є онлайн калькулятором для вирішення математичних задач в тому числі транспортної задачі, наведено у рисунку 1.4.

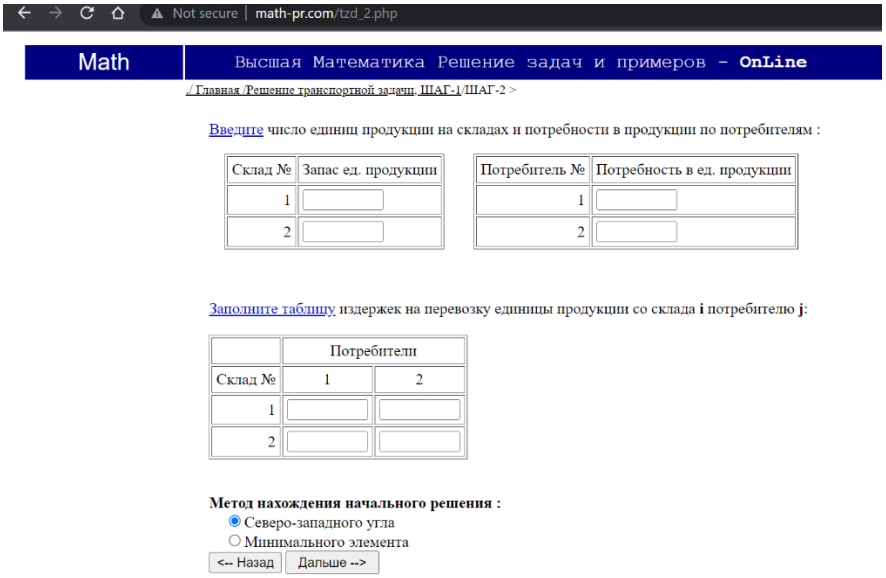


Рисунок 1.4 – Сервіс math-pr.com [22]

Основними перевагами є можливість змінювати розмір матриці, а також обирати методи знаходження базового розв’язку.

Основні недоліками є невелика кількість представлених методів знаходження базового розв’язку та відсутність знаходження оптимального розв’язку.

1.5.4 Сервіс www.easycalculation.com

Сервіс www.easycalculation.com – є онлайн калькулятором для вирішення математичних задач в тому числі транспортної задачі, наведено у рисунку 1.5.

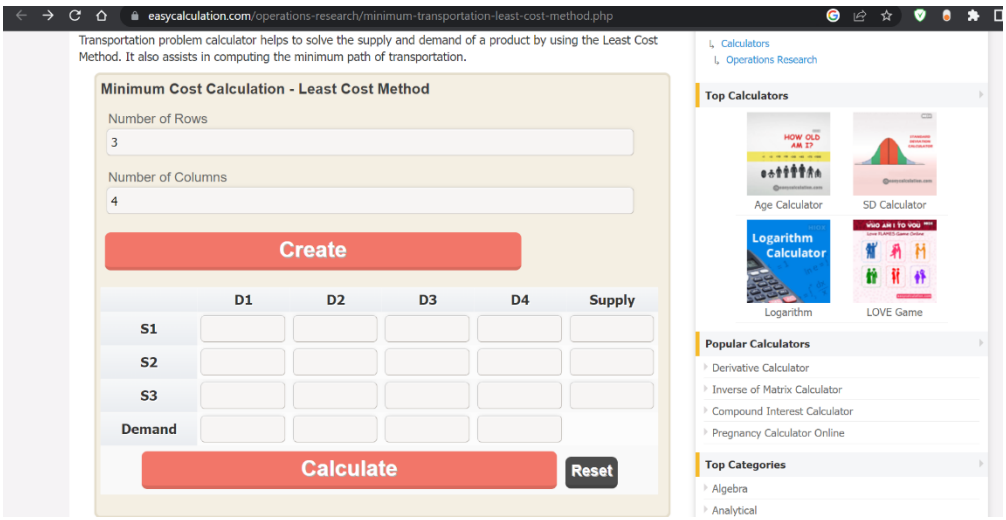


Рисунок 1.5 – Сервіс www.easycalculation.com [23]

Основними перевагами є можливість змінювати розмір матриці.

Основні недоліками є відсутність зміни методу визначення базового розв'язку транспортної задачі, а також відсутність можливості знаходження оптимального розв'язку.

1.5.5 Порівняльна таблиця існуючих та розроблювального програмних продуктів

Вище було наведено найбільш популярні на сьогоднішній день сервіси, які дозволяють вирішувати транспортні задачі. Але крім основних переваг, усі вони мають і недоліки. Наприклад, кількість доступних методів значно обмежена, що, можливо, звужує кругозір прийняття рішень.

Також дійсно існують переваги, характерні для усіх сервісів: найголовніша перевага – це відпрацьовані та протестовані алгоритми розв'язку задач.

У таблиці 1.1 проведено порівняння вже існуючих програмних продуктів та програми, яка буде розроблена в ході виконання роботи.

Таблиця 1.1 – Порівняння програмних продуктів

	reshmat	matworld	math-pr.com	easycalculation.com	Власна розробка
1	**2**	**3**	**4**	**5**	**6**
Зміна розмірів матриці	Динамічна	Перед початком роботи	Перед початком роботи	Перед початком роботи	Динамічна
Кількість методів знаходження базового розв’язку	2	3	2	1	5
Кількість методів знаходження базового розв’язку	0	1	0	0	2
Адаптивність	Ні	Напів-адаптивна	ні	так	так
Редагування контенту сайту	Адміністратором	Адміністратором	Адміністратором	Адміністратором	Адміністратором

1.6 Постановка завдання роботи

Мета бакалаврської роботи – розробка програмного застосунку, який підвищить ефективність розв’язання ТЗ різними способами.

Для досягнення поставленої мети, необхідно вирішити такі **задачі**:

– проаналізувати відомі результати розв’язання транспортних задач;

– обрати серед існуючих алгоритмів ті, які є найбільш актуальними та ефективними, після чого імплементувати їх у програмний застосунок;

– розробити програмну реалізацію розроблених алгоритмів;

– провести дослідження ефективності розроблених алгоритмів.

ПП повинен забезпечувати виконання нижче перерахованих функцій:

– можливість розв’язувати транспортні задачі із знаходження базового розв’язку та оптимального розв’язку;

– підтримку адаптивної верстки, для зручного використання сервісу на різних пристроях;

– панель адміністратора, для зручного редагування, додавання та видалення контенту наповнення сайту;

– зручний та безпечний вхід для адміністратору із дотриманням сучасних норм захисту даних від стороннього втручання;

– адміністратор повинен мати можливість створювати, редагувати та видаляти картки з довідниковою інформацією про методи розв’язку.

Об'єкт дослідження – процес розробки веб-застосунку, що буде реалізувати алгоритм розподілу вантажів, який допомагає зменшити витрати на транспортування вантажів з одного пункту до іншого.

Предмет дослідження – програмні засоби для вирішення проблеми розподілу вантажів, що дає змогу зменшити витрати на транспортування вантажів з одного пункту до іншого.

1.7 Висновка до розділу

Було розглянуто існуючі типи транспортних задач та методи їх розв’язання. Оскільки ефективне планування перевезень за рахунок мінімізації сумарної вартості є важливим завданням транспортної логістики, задачі цього класу мають велике практичне значення та викликають значний

практичний інтерес, який може оптимізувати велику кількість процесів, та принести чималий дохід. З метою розробки програмного застосунку, що буде вирішувати транспортну задачу у декілька способів, була сформульована відповідна постановка задачі та мета дослідження.

Зокрема були розглянуті аналоги для ПЗ, що вже працюють на ринку, було виявлено ряд недоліків та переваг, що в результаті було відзначено у таблиці 1.1.

2 ВИБІР ТА ОБҐРУНТУВАННЯ СТРУКТУРИ СИСТЕМИ, ЯКА ПРОЄКТУЄТЬСЯ

2.1 Вибір інструментарію розробки програмного забезпечення

Одним з головних етапів є вибір інструментарію розробки, тому що саме за допомогою обраних інструментів доведеться втілювати визначені вимоги до програмного продукту.

2.1.1 Огляд особливостей мови програмування

Першочерговим є вибір мови програмування, як головного інструменту реалізації функціоналу.

Розробка буде виконуватися із використанням мови програмування C#, через її універсальні можливості та багатий інструментарій. Мова C# працює на платформі .Net, яка стала новою сучасною середою виконання програм (раніше була COM). C# має великий перелік бібліотек класів, що надають можливість створення застосунків архітектура та спрямування яких є будь якої складності.

Зокрема дана мова є однією із провідних у сфері веб-технологій, через дуже потужний фреймворк ASP.NET.

ASP.NET – технологія створення веб-застосунків і веб-сервісів від компанії Майкрософт. Вона є складовою частиною платформи Microsoft.NET і розвитком старішої технології Microsoft ASP. На цей час останньою версією цієї технології є ASP.NET Core 6.0.

ASP.NET зовні багато в чому зберігає схожість із старішою технологією ASP, що дає змогу розробникам відносно легко перейти на ASP.NET. У той же час внутрішній устрій ASP.NET істотно відрізняється від ASP, оскільки вона заснована на платформі. NET і, отже, використовує всі нові можливості, що надаються цією платформою [24].

ASP.NET пропонує три платформи для створення веб-застосунків: веб-форми, ASP.NET MVC і веб-сторінки ASP.NET. Всі три платформи є

стабільними та зрілими, і ви можете створювати відмінні веб-застосунки з будь-яким з них. Незалежно від вибраної платформи ви отримаєте всі переваги та функції ASP.NET скрізь.

Кожна платформа спрямовано інший стиль розробки. Вибір залежить від поєднання ресурсів програмування (знань, навичок та розробки), типу створюваного додатка та зручного підходу до розробки [25].

Але також необхідно подбати і про інтерфейс застосунка. Більшість сайтів ґрунтується на підході, що за User Interface (UI) відповідає мова гіпертекстової розмітки – HTML, а за зовнішні вигляд – CSS, веб-застосунок, що розробляється не є винятком, в ньому також будуть вказані засади, проте з невеликою зміною, замість стандартного HTML буде використано додаткові можливості, що надає Razor Pages у зв’язці із фреймворком ASP.Net.

Razor Pages - нововведення, що з'явилася в Core.Net 2.0. Razor Page – це сторінка, що складається зі стандартної розмітки (View) та бекенд класу. В якомусь сенсі нагадує Web Forms лише без підтримки збереження стану. Перевага такого рішення очевидна - ми позбавляємося від непотрібного прошарку - моделі сторінки (модель даних у вигляді наприклад Entity це само собою). Бекенд сторінки є контролером і моделлю – класика ООП – інкапсуляція даних та методів роботи з ними в одному об'єкті. Зрештою модель сторінки – це просто клас, немає жодних причин, чому цим класом не може бути контролер [25].

Іншими словами, Razor Pages – більш нестандартне рішення для Інтернету ніж MVC, тепер ми маємо справу з традиційним і логічним поняттям "сторінка" а не з контролерами та моделями розкиданими по всьому проекту. Але оскільки .NET розвиватиметься за напрямом Core.Net, то навряд чи Razor Page з'являться у стандартному фреймворку, незважаючи на те, що найближчі роки більшість проектів залишатиметься на стандартному .NET. Тим не менш, можна зобразити функціональність Razor Pages і на стандартному фреймворку [26].

Але треба зазначити, що можливості Razor Pages будуть впливати лише на інтерфейс сервісу, не ушкоджуючи функціональну модель.

2.1.2 Огляд особливостей обраного середовища

Після того, як мову програмування булу визначено, необхідно провести аналіз наявних середовищ розробки, для того, щоб знайти те, середовище, яке задовільнить більшістю вимог, а також надасть зручний інструментарій. В таблиці 2.1 наведено порівняння середовищ розробки.

Таблиця 2.1 – Порівняння середовищ розробки

	Visual Studio	Visual Studio Code	JetBrains Rider	Sublime Text
1	**2**	**3**	**4**	**5**
Опис середовища	Основне середовище розробки Microsoft для розробки Asp.Net додатків	Безкоштовний кодовий редактор, який надає базовий функціонал для розробки Asp.Net додатків	Середовище розробки зі спеціалізованими інструментами для розробки Asp.Net додатків	Безкоштовний кодовий редактор, який надає базовий функціонал для розробки Asp.Net додатків
Багатоплатформність	Ні, доступний лише для Windows	Так, доступний для Windows, macOS, Linux	Так, доступний для Windows, macOS, Linux	Так, доступний для Windows, macOS, Linux

Продовження таблиці 2.1

1	2	3	4	5
Швидкість модернізації	Висока	Висока	Висока	Середня
Наявність візуального конструктора	Є, має великий набір візуальних елементів	Є, можливість встановлення розширень	Є, має великий набір візуальних елементів	Ні, потрібно встановлювати розширення
Підтримка репозиторіїв	Є, підтримує використання Git	Є, підтримує використання Git	Є, підтримує використання Git	Ні, не підтримує використання Git
Особливості	Повний набір інструментів для розробки Asp.Net додатків, інтеграція з іншими продуктами Microsoft, включаючи Azure та Office 365.	Легкий та швидкий редактор, можливість розширення за допомогою різних плагінів та додатків	Повний набір інструментів для розробки Asp.Net додатків, підтримка різних фреймворків, включаючи .NET Core та Xamarin.	Легкий та швидкий редактор, можливість розширення за допомогою різних плагінів та додатків

Продовження таблиці 2.1

1	2	3	4	5
Ціна	Платний, різні варіанти цін в залежності від функцій та ліцензії	Безкоштовний	Платний, різні варіанти цін в залежності від функцій та ліцензії	Безкоштовний, проте деякі додатки та функції можуть бути платними

2.1.3 Огляд особливостей обраної системи управління базами даних

Після визначення інструментарію для розробки front-end та back-end модулів, прийшов час для визначення СКБД (системи керування бази даних).

У C# є доступ до різноманітних систем керування базами даних, таких як:

– «Microsoft SQL Server» – це система керування базами даних, розроблена компанією Microsoft. Ця СКБД підтримує мови запитів T-SQL та ANSI SQL, а також забезпечує розширені можливості в області аналізу даних та бізнес-інтелекту;

– «MySQL» – це відкрите програмне забезпечення для керування базами даних, розроблене та підтримуване компанією Oracle. MySQL є однією з найпопулярніших СКБД у світі, яка підтримує мову запитів SQL;

– «PostgreSQL» – це об'єктно-реляційна система керування базами даних з відкритим кодом, яка підтримує мову запитів SQL. PostgreSQL забезпечує розширені можливості в області гео-інформатики та обробки масивів даних;

– «Oracle Database» – це система керування базами даних, розроблена компанією Oracle. Вона підтримує мови запитів SQL та PL/SQL, а також має розширені можливості в області аналізу даних та бізнес-інтелекту;

– «SQLite» – це вбудована СКБД, яка зберігає бази даних у вигляді файлів на локальному комп'ютері. SQLite підтримує мову запитів SQL та забезпечує швидкий доступ до даних;

– «MongoDB» – це документ-орієнтована СКБД, яка забезпечує швидкий доступ до даних, особливо для великих обсягів даних. MongoDB підтримує мову запитів JSON та має розширені можливості в області обробки масивів даних.

У наведеному переліку представлені системи керування як реляційними, так і не реляційними базами даних. У застосунку, що розробляється, є потреба мати засоби авторизації, збереження контенту сторінок та інше, то ж найкращим варіантом буде використання саме реляційних баз даних.

Наступним кроком необхідно визначити фреймворк, який буде надавати інструментарій для роботи з реляційними базами даних. Мова програмування C# та платформа .Net має декілька основних фреймворків.

ADO.NET – це набір класів, які надають служби доступу до даних програмістів, які використовують платформу .NET Framework. ADO.NET має багатий набір компонентів для створення розподілених додатків, які спільно використовують дані. Це невід'ємна частина платформи .NET Framework, яка надає доступ до реляційних даних, XML-даних та даних додатків. ADO.NET задовольняє різні потреби розробників, включаючи створення клієнтських додатків баз даних, а також бізнес-об'єктів середнього рівня, що використовуються програмами, засобами, мовами та браузерами[27].

Entity Framework – це ORM Framework із відкритим вихідним кодом для додатків на основі .NET Framework, які підтримує Microsoft. Згідно з документацією Microsoft, Entity Framework було надано для автоматизації всіх типів дій, пов’язаних із базою даних, для нашої програми. За допомогою цього фреймворку розробники можуть отримати доступ до необхідної бази даних і почати роботу з розробкою даних за допомогою предметно-спеціальних об’єктів класу. Крім того, їм не потрібно зосереджуватися на

базових таблицях або стовпцях бази даних, де зберігатимуться фактичні дані. Використовуючи та розуміючи Entity Framework і його структуру, розробники можуть працювати на вищому рівні абстракції, коли вони мають справу з даними, а також можуть створювати та підтримувати програму, орієнтовану на дані, з меншим кодом порівняно з традиційними програмами [28].

Отже, якщо ми хочемо отримати більше контролю над командами та операціями SQL за допомогою необроблених запитів SQL, тоді ADO.NET є хорошим рішенням для початку. Однак, якщо є необхідність швидше розробляти застосунки з зручністю гнучкості коду, Entity Framework буде кращим вибором. У випадку поточного проєкту необхідну гнучкість коду забезпечить Entity Framework.

2.2 Структурна схема програми

Перед початком безпосередньої розробки програмного засобу у середовищі програмування, варто визначитися із структурою застосунку та виділити головні її компоненти.

Веб-застосунок має являти собою сервіс, головна задача якого – надавати можливість вирішувати транспортні задачі різними методами, але, беручи до уваги той факт, що це сервіс, також необхідно впровадити можливість його підтримки, шляхом заповнення та редагування адміністратором. Також не можна забути про пересічного користувача, який, скоріш за все, окрім вирішення поставленої задачі обраним методом, ще хотів би мати інформаційну довідку про алгоритми та етапи роботи доступних методів вирішення транспортних задач.

До структури сервісу належить такі модулі як: сторінка вводу даних задачі, сторінка результату, сторінка з довідниковою інформацією, сторінка з інформацією про проєкт, сторінка входу в обліковий запис користувача-

адміністратора, сторінка панелі адміністратора, сторінка редагування контенту наповнення сайту.

На рисунку 2.1 зображено структурну схему веб-сервісу.

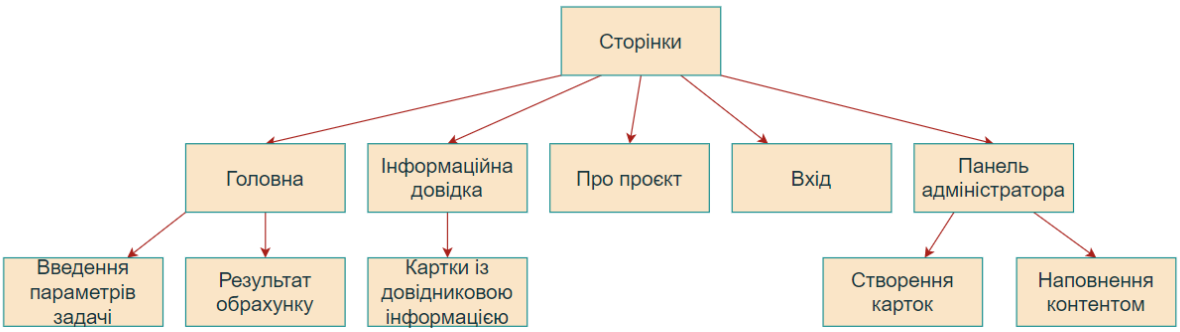


Рисунок 2.1 – Структурна схема програми

У таблиці 2.2 наведено опис модулів реалізованого ПЗ з визначенням їх основних функцій.

Таблиця 2.2 – Опис модулів реалізованого ПЗ з визначенням основних функцій.

Модуль	Опис
1	**2**
Введення параметрів задачі	У даному модулі користувач повинен мати можливість вказати усі необхідні вхідні данні: матрицю тарифів, постачальників та замовників, зокрема користувач також повинен обрати метод обрахунку базового та оптимального плану.
Результату обрахунку	Повинен складатися із матриці базового плану разом із обрахунком загальної суми, а також матриці оптимального плану разом із вказанням остаточної суми.

Продовження таблиці 2.2

1	2
Картки із довідниковою інформацією	Складається із «карток», кожна з яких містить інформацію про алгоритм та особливості методу обрахунку транспортної задачі.
Про проєкт	Містить інформаційні вказівники про проєкт: посилання на репозиторій та контакти автора.
Входу в акаунт	Представляє собою сторінку входу в акаунт адміністратора, де необхідно вказати логін та пароль облікового запису.
Панель адміністратору	Являє собою сторінку із набором інструментів, що покликані впливати на контент наповнення веб-сервісу, а також кнопка для виходу з облікового запису.
Редактор контенту: «Створення карток», «Наповнення контентом»	Є сторінкою, що надає набір засобів для зміни наповнення публічних сторінок веб-застосунок, тобто надає можливість доповнювати, або змінювати текстову та графічну інформацію.

2.3 Перелік алгоритмів розроблюваної системи

В системі, що розробляються реалізується 7 алгоритмів, кожен з яких більш докладно буде розібраний у подальших розділах:

- подвійної переваги;
- мінімального елементу;
- північно-західного кута;
- апроксимація Фогеля;
- метод потенціалів;
- Флойда-Уоршила;

– генетичний алгоритм.

Ці алгоритми допомагають вирішувати транспортні задачі з різною ефективністю та швидкодією, що робить їх корисними інструментами для оптимізації розподілу ресурсів у різних сферах діяльності.

2.4 Засоби розробки

Для реалізації наведеного раніше застосунку було розроблене програмне забезпечення, основною мовою програмування – є C# та платформа .Net.

C# (C Sharp) – є високорівневою об'єктно-орієнтованою мовою програмування, розробленою корпорацією Microsoft, що характеризується сильною типізацією, динамічністю, інтерактивністю, портативністю.

Вибір даної мови програмування обґрунтований наступними перевагами.

Обширність бібліотек – C# має велику вбудовану бібліотеку класів із можливістю розширення використовуючи пакетний менеджер Nuget.

Портативність – мова C# підтримує крос-платформну розробку завдяки ініціативі .NET, зокрема завдяки платформі .NET Core та її спадкоємцю, .NET 5, 6, 7, що дозволяє розробникам створювати додатки для різних операційних систем, таких як Windows, Linux і macOS.

Продуктивність – простота, бібліотеки функції інтеграції процесів, модульні системи тестування та розширені можливості управління дають переваги ефективності розробників, збільшенню швидкості та продуктивності програм ніж при використанні інших мов, розробники можуть зосередитись на унікальному коді під конкретний проект, використовуючи наявні бібліотеки.

Простота – оскільки більшість функцій інкапсюльовано у бібліотеки, то програмувати, читати та розуміти код на C# дуже просто.

Інтерактивність – можливість икористовуватися для створення інтерактивних додатків та інструментів завдяки своїй легкості використання та можливості відразу виконувати код.

Динамічність – не зважаючи на те, що C# є сильно типізованою мовою, вона також підтримує динамічні структури даних і може використовувати рефлексію для маніпуляції об'єктами на етапі виконання.

IoT – C# застосовується для інтернету речей, який стає дедалі актуальнішим.

Велике комьюніті – інклюзивна велика спільнота програмістів, що використовують C# – підтримка програмістів на етапах розробки, наявність експертних тематичних джерел та розробка додаткових бібліотек.

Довіра у великих проектах – останнім часом C# обирають такі корпорації як Microsoft, Google, YouTube, Dropbox, Yahoo, використовуються у галузях освіти, науки, охорони здоров’я, фінансів.

Бібліотеки, що використовуються у проекті, зображені на рисунку 2.2

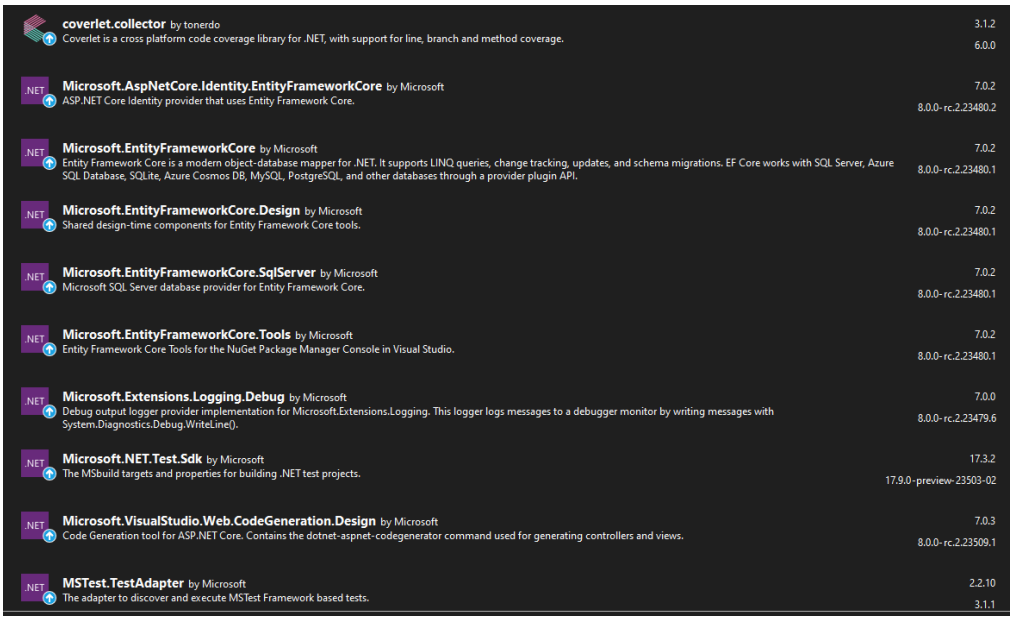


Рисунок 2.2 – Бібліотеки, що використовуються у проєкті

Призначення використаних бібліотек:

– «Coverlet.Collector» – це бібліотека для збору даних покриття коду в проектах на платформі .NET. Вона використовується для вимірювання того, наскільки добре код покривається юніт-тестами (тестами на рівні окремих компонентів, методів, класів тощо);

– «Microsoft.AspNetCore.Identity.EntityFrameworkCore» – це бібліотека, яка надає можливість інтегрувати систему автентифікації та авторизації в ASP.NET Core додатки, використовуючи Entity Framework Core як засіб доступу до бази даних. Ця бібліотека є частиною ASP.NET Core Identity Framework і дозволяє розробникам створювати додатки, які підтримують реєстрацію користувачів, управління ролями, аутентифікацію та авторизацію користувачів, а також інші функції, пов'язані з безпекою і доступом;

– «Microsoft.EntityFrameworkCore» – це бібліотека для Entity Framework Core, яка надає доступ до можливостей об'єктно-реляційного відображення (ORM) в .NET додатках. Entity Framework Core є сучасною версією ORM від Microsoft і дозволяє розробникам зручно взаємодіяти з базами даних в .NET додатках без необхідності написання прямих SQL-запитів;

– «Microsoft.EntityFrameworkCore.Design» – це пакет бібліотеки Entity Framework Core, який містить інструменти розробки і управління кодом бази даних. Цей пакет дозволяє розробникам створювати та керувати моделями даних, базами даних і міграціями без необхідності вручну писати SQL-запити або використовувати інші інструменти;

– «Microsoft.EntityFrameworkCore.SqlServer» – це пакет бібліотеки Entity Framework Core, який надає підтримку для взаємодії Entity Framework Core з сервером бази даних Microsoft SQL Server. Цей пакет дозволяє розробникам створювати додатки, які використовують SQL Server як систему керування базами даних та взаємодіють з нею з використанням Entity Framework Core;

– «Microsoft.EntityFrameworkCore.Tools» – це пакет бібліотеки Entity Framework Core, який містить набір інструментів командного рядка для розробки та управління кодом бази даних в Entity Framework Core. Ці інструменти надають можливість виконувати різні операції, пов'язані з базою даних, такі як створення міграцій, виконання міграцій, створення моделей даних та інші;

– «Microsoft.Extensions.Logging.Debug» – це частина бібліотеки Microsoft.Extensions.Logging, яка надає інтеграцію інфраструктури логування в ASP.NET Core та .NET додатки з виведенням логів у вікно відладки (Debug Output) під час відладки додатка;

– «Microsoft.NET.Test.Sdk» – це пакет бібліотеки, який використовується для підтримки тестування в .NET додатках та проектах. Він забезпечує інфраструктуру для виконання тестів, а також інтеграцію з різними тестовими фреймворками, такими як MSTest, xUnit, NUnit і інші;

– «Microsoft.VisualStudio.Web.CodeGeneration.Design» – це пакет бібліотеки, який надає інфраструктуру для розробки та генерації коду веб-додатків в середовищі Visual Studio. Цей пакет використовується для створення генераторів коду та команд генерації коду, які допомагають розробникам автоматизувати процес створення шаблонів коду для веб-додатків;

– «MSTest.TestAdapter» – це пакет бібліотеки, який використовується для підтримки та інтеграції фреймворка тестування MSTest з іншими інструментами розробки та системами збірки. MSTest є одним з популярних фреймворків тестування в платформі .NET, і MSTest.TestAdapter дозволяє виконувати та керувати тестами, написаними з використанням MSTest.

В якості IDE для розробки програмного продукту було використано Microsoft Visual Studio.

2.5 Архітектура програмного забезпечення

2.5.1 Алгоритм функціонування розробленої програми

Основні алгоритми, які можуть виконуватися при роботі із розробленою програмою зображені на рисунку 2.3.

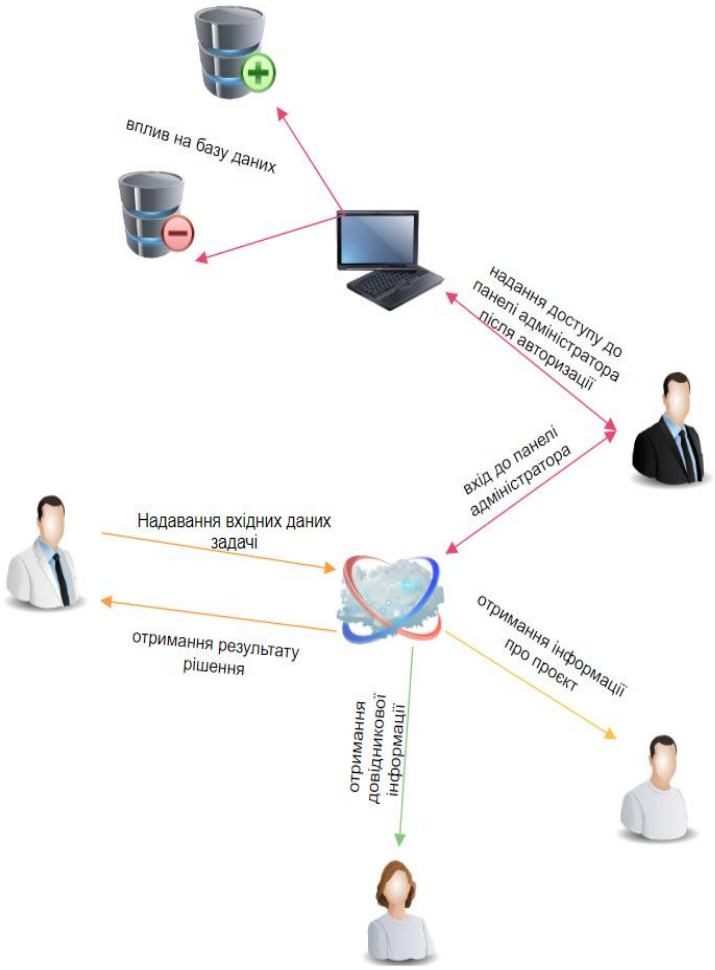


Рисунок 2.3 – Функціональна схема розробленої програми

Як видно зі схеми під час розробки ПП «Веб-застосунок для вирішення транспортних задач», були розроблені наступні алгоритми:

– обрахунок вхідних даних 2-ма з 7 алгоритмами, для отримання опорного та оптимального рішення;

– отримання довідникової інформації;

– авторизації адміністратора;

– зміни даних в БД за допомогою інструментарію панелі адміністратора.

2.5.2 Алгоритм авторизації

Для авторизації на ресурсі необхідно перейти по адресі <домен>/account/login після чого користувачу-адміністратором буде необхідно вести логін та пароль, а також, за бажанням, обрати прапорець «запам’ятати мене». Після вказанням всіх необхідних даних і натискання на кнопку «Увійти» буде виконуватися перевірка, яка при успішному результаті за допомоги Coockie-файлів дозволяє авторизованому користувачу звертатися до області адміністратора, саме у межах цієї області знаходиться панель керування сайту.

2.5.3 Алгоритм обрахунку та отримання результату

Після того, як користувач ввів вхідні данні, обрав алгоритми розрахунку і натиснув на кнопку «Розрахувати», система починаю виконувати розрахунок спочатку опорного плану, після чого відбувається його покращення одним із наявних алгоритмів оптимального рішення. В результаті всі обраховані дані демонструються на сторінці результату.

2.5.4 Алгоритм отримання довідникової інформації

Найпростішим алгоритмом у даному веб-застосунку – є алгоритм отримання, для перегляду, довідникової інформації. Усі данні веб-сервісу, окрім структурних (що вказують на місця розташування окремих компонентів, таких як таблиця, кнопки, тощо), зберігаються у базі даних, це надає гнучкість у підтримці та адмініструванні ПЗ без участі програміста. Кожного разу при спробі звернутися до певної сторінки сайту, відбувається

запит до бази даних, для того, щоб отримати необхідний контент, що в подальшому демонструється користувачу.

2.5.5 Зміни даних в БД за допомогою інструментарію панелі адміністратора

Як було зазначено раніше, усі данні наповнення зберігаються безпосередньо у базі даних, то ж це надає можливість їх зручного редагування. Цією операцією займається адміністратор сайту за допомогою доступної йому панелі адміністратора. Для редагування конкретної сторінка/картки він (адміністратор) повинен обрати відповідний пункт «Edit» біля доступних елементів, після чого буде виконано переадресування на сторінку із можливостями зручного редагування, контенту, що знаходиться в БД на момент початку роботи. Після закінчення внесення змін, необхідно натиснути на кнопку «Save» для збереження і повернення на головну панель сторінки адміністратора.

2.6 Вибір шаблону проєктування

Шаблони проектування, або патерни проектування, - це загально прийняті рішення, що визначають стандартний спосіб розв'язання конкретних проблем проектування в програмному забезпеченні. Вони допомагають покращити якість проектів та зробити їх більш ефективними [29].

У веб-розробці, існує декілька шаблонів проектування, які використовуються для розробки веб-застосунків. Деякі з них:

Model-View-Controller (MVC) - це шаблон проектування, який розділяє логіку програми на три основні компоненти: модель, яка представляє дані, перегляд, який показує дані користувачеві та контролер, який обробляє взаємодію між моделлю та переглядом. Цей шаблон дозволяє

забезпечити розділення відповідальності між компонентами та покращити масштабованість проекту [30].

Шаблон Active Record - це шаблон проектування, який дозволяє зберігати дані в базі даних та виконувати з ними операції через модель. Цей шаблон дозволяє спростити роботу з базою даних та зменшити кількість коду, який потрібно написати [31].

Шаблон Одинак - це шаблон проектування, який дозволяє забезпечити, що клас має тільки один екземпляр, тим самим зменшуючи навантаження на систему. Цей шаблон може бути корисним для роботи з базами даних [32].

Одним із найпопулярнішим патерном проєктування для розробки веб застосунків – є MVC (Model-View-Controller) і в поточному проєкті він буде головуючим. Шаблон проектування MVC – це популярний підхід до розробки веб-застосунків, який дозволяє розділити логіку застосунку на три компоненти: модель, представлення та контролер. Кожен з цих компонентів виконує свої визначені функції, що дозволяє зробити проект більш організованим, зрозумілим та легко змінюваним. Ось деякі переваги використання шаблону проектування MVC:

– розділення логіки застосунку: MVC дозволяє розділити логіку застосунку на три різних компоненти, що робить код більш організованим та простим у змінні;

– перевикористання коду: оскільки логіка застосунку розділена на три компоненти, то кожен з них може бути перевикористаний в інших проектах. Наприклад, ви можете використовувати один і той же контролер в декількох веб-сторінках;

– тестування: Кожен компонент можна легко тестувати окремо, що робить процес тестування більш простим та ефективним;

– розширення: Застосування шаблону проектування MVC дозволяє легко розширювати функціональність проекту. Наприклад, ви можете додати

нові представлення та контролери до проєкту без внесення змін до існуючого коду;

– легкість розробки: Шаблон проектування MVC робить процес розробки веб-застосунків більш легким та простим, оскільки він дозволяє розглядати проект як набір окремих компонентів, а не як єдиний цілий. Це зменшує складність розробки та робить її більш керуючою.

Незважаючи на багато переваг, паттерн MVC має кілька недоліків:

– складність: Реалізація паттерну може бути досить складною, особливо для початківців, оскільки вимагає розуміння концепцій моделі, виду і контролера;

– перенавантаження контролера: Якщо контролер занадто складний, він може стати перевантаженим з великою кількістю обов'язків, що може привести до збоїв у роботі програми;

– велика кількість файлів: Реалізація паттерну MVC може привести до створення великої кількості файлів, що може зробити проект менш зрозумілим та важким для підтримки;

– важко тестувати: Тестування веб-застосунків, реалізованих за допомогою паттерна MVC, може бути складним, оскільки потрібно проводити тестування всіх компонентів окремо;

– збільшення часу виконання: З великою кількістю клієнтів, які взаємодіють з веб-сервером, може збільшуватися час виконання програми, що може призвести до погіршення продуктивності.

Для кращого розуміння цього шаблону проєктування на рисунку 2.4 продемонстровано UML процес, що показує процедуру його роботи так само, як би це відбувалося в реальному застосунку.

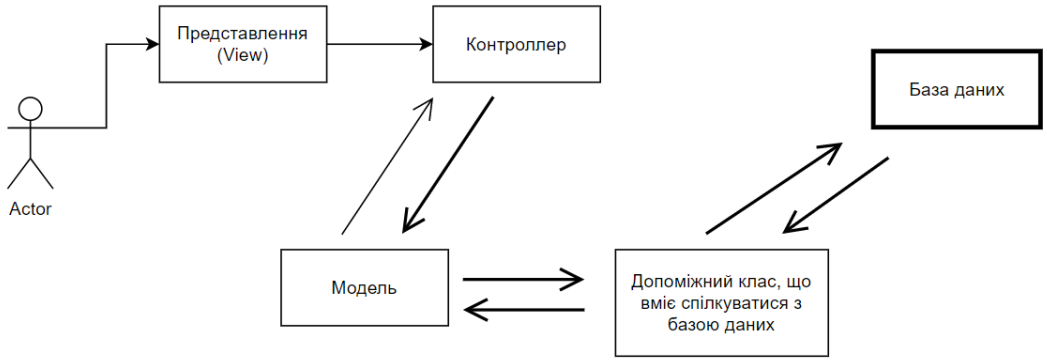


Рисунок 2.4 – UML діаграма, що показує роботу паттерну MVC

Зважаючи на усі переваги і недоліки, було прийнято рішення використовувати архітектуру шаблону MVC для розробки застосунка, це надасть необхідну гнучкість та модульність, а також спростить подальший його розвиток.

2.7 Висновки за розділом 2

У розділі 2 було проведено аналіз структури системи що проєктується, було створено ряд висновків: визначено і обґрунтовано мову та платформу розробки, зокрема було визначено систему керування базами даних, що буде застосовуватися у подальшій розробці, а також визначено структурну схему і проведено детальний опис модулів майбутнього веб-застосунку.

3 ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Проєктування бази даних

Одним з головних етапів у процесі проєктування бази даних – є розробка концептуальної моделі бази даних, оскільки вона дозволяє відобразити всі ключові сутності та їх взаємозв'язки, що будуть використовуватися в системі. Концептуальна модель відображає бізнес-процеси та логіку даних, що використовуються в додатку, незалежно від того, як будуть зберігатися дані.

Розробка концептуальної моделі бази даних передбачає визначення ключових понять, які відображають діяльність компанії чи організації, а також взаємозв'язки між ними. Важливо ретельно вивчити всі бізнес-процеси та визначити, як вони будуть взаємодіяти між собою та з якими даними. Концептуальна модель має детально описати всі відносини та атрибути, що будуть використовуватися в системі, та відобразити структуру бази даних в її цілому.

Розробка концептуальної моделі дозволяє відокремити логіку бізнес-процесів від конкретної реалізації бази даних та забезпечити її більшу гнучкість та масштабованість. Вона також дозволяє забезпечити більшу зрозумілість та простоту взаємодії з базою даних для розробників та користувачів.

Отже, розробка концептуальної моделі бази даних є необхідним етапом в процесі розробки веб-застосунку та дозволяє забезпечити оптимальну структуру бази даних для підтримки бізнес-процесів та ефективної роботи системи.

На рисунку 3.1 наведено концептуальну модель бази даних, що розробляється.

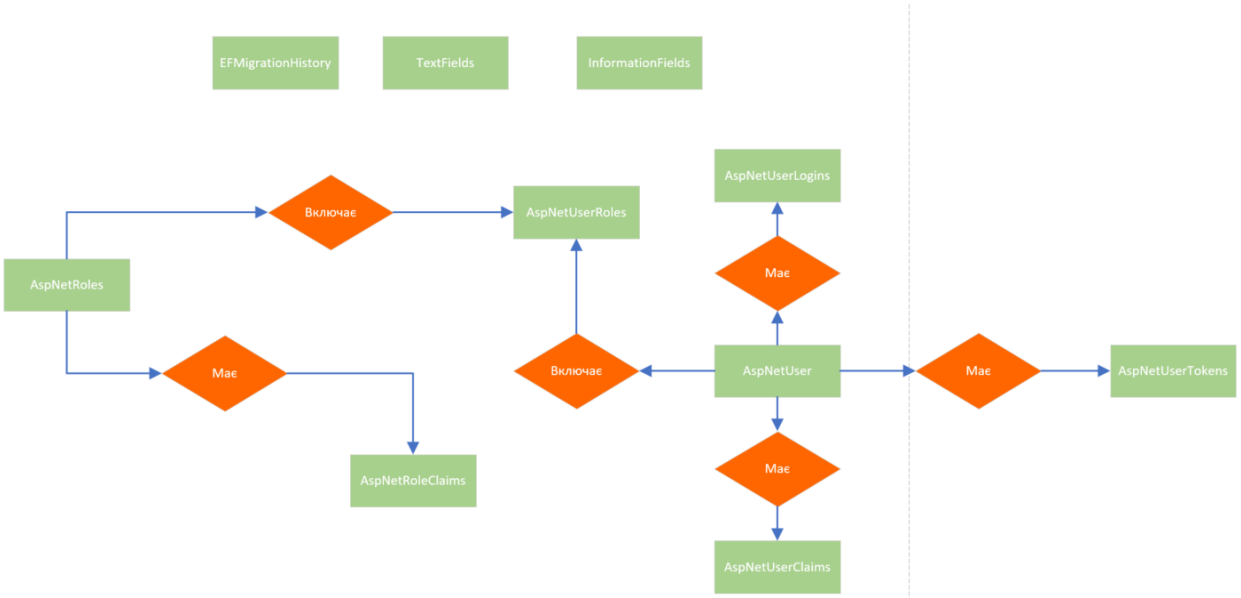


Рисунок 3.1 – Концептуальна модель бази даних

Таким чином маємо діаграму із 10-а сутностями та логічні зв'язки між ними. Дана діаграма має наступні сутності:

– «EFMigrationHistory»;
– «TextFields»;
– «InformationFields»;
– «AspNetRoleClaims»;
– «AspNetRoles»;
– «AspNetUserRoles»;
– «AspNetUserLogins»;
– «AspNetUser»;
– «AspNetUserClaims»;
– «AspNetUserTokens».

Наступним етапом після розробки концептуальної моделі слід виконати опис наведених в ній сутностей та побудувати ER-діаграму.

Сутність EFMigrationHistory (MigrationId, ProductVersion) – Сутність для підтримки технології «Міграції», що надає Entity Framework. Опис полів сутності наведено у таблиці 3.1.

Таблиця 3.1 – Опис полів сутності EFMigrationHistory

№	Назва поля	Тип даних	Розмір
1	MigrationId	nvarchar	150
2	ProductVersion	nvarchar	32

Для збереження даних контенту сторінок сайту використовується сутність TextFields, зокрема вона зберігає й мета данні відповідної сторінки сайту. Опис полів сутності наведено у таблиці 3.2.

Таблиця 3.2 – Опис полів сутності TextFields

№	Назва поля	Тип даних	Розмір
1	Id	uniqueidentifier	-
2	CodeWord	nvarchar	MAX
3	Title	nvarchar	MAX
4	Text	nvarchar	MAX
5	Subtitle	nvarchar	MAX
6	TitleImagePath	nvarchar	MAX
7	MetaTitle	nvarchar	MAX
8	MetaDescription	nvarchar	MAX
9	KeyWords	nvarchar	MAX
10	DateAdded	datetime2	7

Сутність InformationFields (id, title, Text, Subtitle, TitleImagePath, MetaTitle, MetaDescription, keyWords, DateAdded) – Сутність для збереження даних, що знаходяться в «інформаційних картках», що описують, наприклад алгоритм роботи методу рішення транспортної задачі. Опис полів сутності наведено у таблиці 3.3.

Таблиця 3.3 – Опис полів сутності InformationFields

№	Назва поля	Тип даних	Розмір
1	**2**	**3**	**4**
1	Id	uniqueidentifier	-
2	Title	nvarchar	MAX
3	Text	nvarchar	MAX
4	Subtitle	nvarchar	MAX

Продовження таблиці 3.3

1	2	3	4
5	TitleImagePath	nvarchar	MAX
6	MetaTitle	nvarchar	MAX
7	MetaDescription	nvarchar	MAX
8	KeyWords	nvarchar	MAX
9	DateAdded	datetime2	7

Сутність AspNetUserLogins (LoginProvider, ProviderKey, ProviderDisplayName, UserId) – є сутністю яка містить дані про зовнішні посилання на входи користувачів. Кожен запис в цій таблиці містить ідентифікатор користувача, провайдера зовнішнього входу (наприклад, Facebook або Google) та ідентифікатор посилання на вхід для даного провайдера. Ця таблиця використовується для забезпечення зовнішнього входу в систему за допомогою облікових записів з інших додатків чи сервісів. Опис полів сутності наведено у таблиці 3.4.

Таблиця 3.4 – Опис полів сутності AspNetUserLogins

№	Назва поля	Тип даних	Розмір
1	LoginProvider	nvarchar	450
2	ProviderKey	nvarchar	450
3	ProviderDisplayName	nvarchar	MAX
4	UserId	nvarchar	450

Сутність AspNetUser (Id, UserName, NormalizedUserName, Email, NormalizedEmail, EmailConfirmed, PasswordHash, SecurityStamp, ConcurrencyStamp, PhoneNumber, PhoneNumberConfirmed, TwoFactorEnabled, LockoutEnd, LockoutEnabled, AccessFailedCount) – є сутністю в базі даних Microsoft ASP.NET Identity Framework, яка містить дані про користувачів системи. Кожен запис в цій сутності містить унікальний ідентифікатор користувача, його ім'я, адресу електронної пошти, пароль та інші атрибути, такі як ім'я, прізвище, дата народження та інше.

Наведена сутність використовується для автентифікації та авторизації користувачів в системі. Крім того, вона може містити додаткові поля, які використовуються для зберігання додаткової інформації про користувачів, яка необхідна для функціонування додатків на базі ASP.NET Identity Framework. Опис полів сутності наведено у таблиці 3.5.

Таблиця 3.5 – Опис полів сутності AspNetUser

№	Назва поля	Тип даних	Розмір
1	Id	nvarchar	450
2	UserName	nvarchar	256
3	NormalizedUserName	nvarchar	256
4	Email	nvarchar	256
5	NormalizedEmail	nvarchar	256
6	EmailConfirmed	bit	-
7	PasswordHash	nvarchar	MAX
8	SecurityStamp	nvarchar	MAX
9	ConcurrencyStamp	nvarchar	MAX
10	PhoneNumber	nvarchar	MAX
11	PhoneNumberConfirmed	bit	-
12	TwoFactorEnabled	bit	-
13	LockoutEnd	datetimeoffset	7
14	LockoutEnabled	bit	-
15	AccessFailedCount	int	-

Сутність AspNetUserTokens (UserId, LoginProvider, Name, Value) –є сутністю в базі даних Microsoft ASP.NET Identity Framework, яка містить токени, що використовуються для підтвердження ідентифікації користувачів у системі.

Кожен запис цієї сутності містить інформацію про токен, який був створений для конкретного користувача, його тип та значення. Тип токена може бути використаний для відрізнення токенів, які використовуються для різних цілей, наприклад, для підтвердження електронної пошти або зміни пароля.

Опис полів сутності наведено у таблиці 3.6.

Таблиця 3.6 – Опис полів сутності AspNetUserTokens

№	Назва поля	Тип даних	Розмір
1	LoginProvider	nvarchar	450
2	Name	nvarchar	450
3	Value	nvarchar	MAX
4	UserId	nvarchar	450

Сутність AspNetUserClaims (Id, UserId, ClaimType, ClaimValue) – є однією з сутностей бази даних, яка використовується в системі ASP.NET Identity для зберігання клеймів користувача. Клейми представляють собою інформацію про користувача, яка необхідна для аутентифікації та авторизації в додатку. Використання клеймів дозволяє розширити базовий функціонал системи аутентифікації та авторизації, додавши до неї додаткову інформацію про користувача, яка може використовуватись у додатку. Наприклад, це можуть бути дані про користувача, такі як його електронна адреса, ім'я та прізвище, назва компанії, до якої він належить, або ролі, які він виконує в системі.

Опис полів сутності наведено у таблиці 3.7.

Таблиця 3.7 – Опис полів сутності AspNetUserClaims

№	Назва поля	Тип даних	Розмір
1	Id	int	-
2	ClaimType	nvarchar	MAX
3	ClaimValue	nvarchar	MAX
4	UserId	nvarchar	450

Сутність AspNetRoleClaims (Id, RoleId, ClaimType, ClaimValue) – являє собою одну з сутностей бази даних, яка зберігає права доступу користувачів до певних ресурсів в межах ролі в системі ASP.NET Identity. Вона зберігає клейми (claims) для кожної ролі, які можуть містити інформацію про дозволи користувачів на доступ до певних дій або ресурсів, наприклад, право на редагування певної сторінки веб-сайту або на доступ до конкретного API.

AspNetRoleClaims використовується для зберігання інформації про дозволи ролі, що дозволяє управляти правами доступу до даних та функцій в системі. Вона містить поля, які зберігають ідентифікатор ролі, тип клейма (наприклад, роль користувача або адміністратора) та значення клейма. Ця таблиця використовується в парі з AspNetRoles та AspNetUserRoles для надання рівня доступу користувачам в системі ASP.NET Identity.

Опис полів сутності наведено у таблиці 3.8.

Таблиця 3.8 – Опис полів сутності AspNetRoleClaims

№	Назва поля	Тип даних	Розмір
1	Id	int	-
2	ClaimType	nvarchar	MAX
3	ClaimValue	nvarchar	MAX
4	RoleId	nvarchar	450

Сутність AspNetRoles (Id, Name, NormalizedName, ConcerrencyStamp) – є частиною архітектури безпеки ASP.NET та містить інформацію про ролі користувачів в системі. Вона використовується для надання доступу до певних ресурсів системи залежно від ролі користувача.

У таблиці AspNetRoles зберігається список ролей, які визначаються в аплікації. Кожна роль має унікальний ідентифікатор (RoleId) та назву (Name). Для кожної ролі можуть бути задані додаткові атрибути, наприклад, опис або дату створення.

AspNetRoles використовується разом з іншими таблицями ASP.NET Identity для забезпечення аутентифікації та авторизації користувачів в системі.

Опис полів сутності наведено у таблиці 3.9.

Таблиця 3.9 – Опис полів сутності AspNetRoles

№	Назва поля	Тип даних	Розмір
1	**2**	**3**	**4**
1	Id	nvarchar	450

Продовження таблиці 3.9

1	2	3	4
2	Name	nvarchar	256
3	NormalizedName	nvarchar	256
4	ConcurrencyStamp	nvarchar	MAX

Сутність AspNetUserRoles (UserId, RoleId) – є частиною архітектури безпеки ASP.NET та використовується для зв'язку між сутностями AspNetUsers та AspNetRoles. Вона містить інформацію про приналежність користувачів до різних ролей в системі.

У таблиці AspNetUserRoles зберігаються записи, які містять ідентифікатори користувачів (UserId) та ідентифікатори ролей (RoleId), до яких вони належать. Кожен запис відповідає одній ролі, до якої приналежить один користувач.

AspNetUserRoles використовується разом з іншими таблицями ASP.NET Identity для забезпечення аутентифікації та авторизації користувачів в системі. Дані з цієї таблиці використовуються для визначення прав доступу користувача до різних ресурсів системи, які обмежені ролями.

Опис полів сутності наведено у таблиці 3.10.

Таблиця 3.10 – Опис полів сутності AspNetUserRoles

№	Назва поля	Тип даних	Розмір
1	Id	nvarchar	450
2	Name	nvarchar	256

Для кращої візуалізації зв’язків та сутностей концептуальної моделі, застосуємо ER-діаграму.

Діаграма прив'язки сутності ER - це блок-схеми, які ілюструють, як «сутності» (люди, об'єкти або концепції) ставляться один до одного в системі. ER-діаграма - це та модель, яка найчастіше використовуються для розробки або налагодження реляційних баз даних в областях, бізнес-інформаційних систем і досліджень. Вона використовує набір геометричних

символів, таких як прямокутник, ромб, овал і лінії, для відображення взаємозв'язку об'єктів, відносин і їх атрибутів. Ця ER-діаграма зв'язана зі структурою даних DSD, які фокусуються на відносинах елементів всередині сутностей, а не на відносинах між самими об'єктами. Схеми ER також часто використовуються в поєднанні з діаграм потоків даних DFD, які відображають інформацію для процесів або систем[33].

Діаграми «сутність-зв'язок» (або ERD) — невід'ємна складова процесу моделювання будь-яких систем, включаючи прості та складні бази даних, проте застосовувані в них фігури та способи нотації можуть запросто ввести в оману будь-кого.

Концептуальні моделі даних дають загальне уявлення у тому, що має входити до складу моделі. Концептуальні ER-діаграми можна брати за основу логічних моделей даних. Їх також можна використовувати для створення відносин спільності між різними ER-моделями, поклавши їх в основу інтеграції [34].

Отже, ER-діаграма є важливим інструментом при побудові бази даних, оскільки вона дозволяє візуалізувати структуру даних та зв'язки між ними. За допомогою ER-діаграми можна визначити сутності, атрибути та зв'язки між ними, а також зрозуміти, як вони пов'язані між собою. При побудові ER-діаграми необхідно враховувати основні принципи нормалізації баз даних, щоб забезпечити їх ефективність та надійність. Нормалізація допоможе уникнути дублювання даних та забезпечити їх консистентність. Правильне використання ER-діаграм допоможе забезпечити ефективність та надійність бази даних.

На рисунку 3.2 представлено ER-діаграму концептуальної моделі предметної області системи, що розробляється.

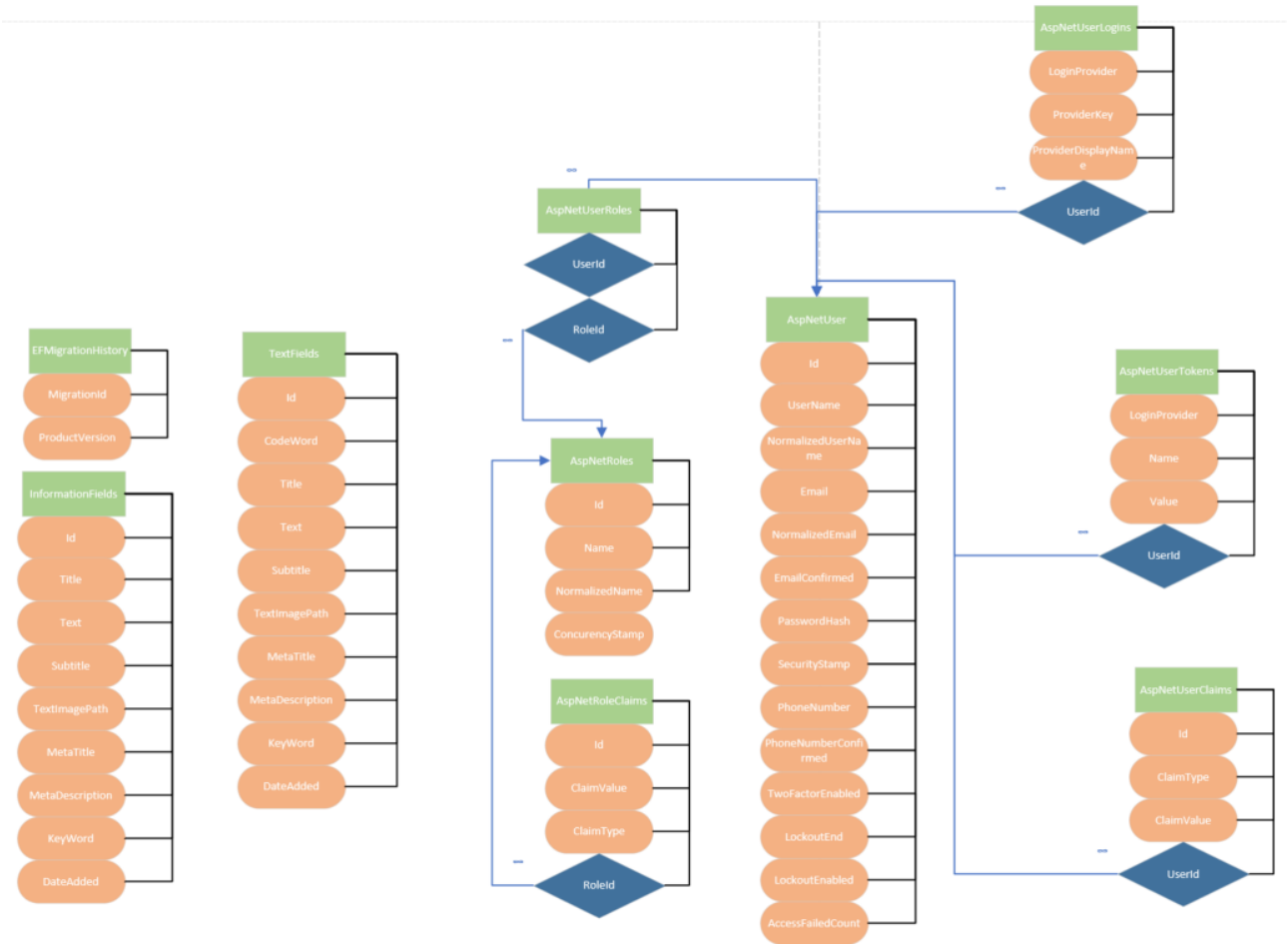


Рисунок 3.2 – ER-діаграма концептуальної моделі

3.2 Вхідні данні

Вхідні дані – це дані, які надходять до системи і які необхідні для її подальшої роботи. Вхідні данні реєструються в програмі за допомогою форм, або за допомогою імпортування з файлу.

Данні про умови транспортну задачу:

– матриця тарифів транспортної задачі;

– масив постачальників;

– масив споживачів;

– тип алгоритму обрахунку базового плану;

– тип алгоритму обрахунку покращеного плану.

Данні про користувача:

– логін;

– пароль.

Данні адміністратора:

– інформаційне наповнення контенту сторінок сайту;

– інформаційне наповнення довідникових карток.

3.3 Вихідні данні

Вихідними даними є сформований план перевезень продукції, у якому враховані потреби всіх клієнтів та постачальників, а також мінімальна вартість витрат на перевезення.

4 КЕРІВНИЦТВО ПРОГРАМІСТА

4.1 Призначення й умови застосування програми

Програма призначена для вирішення транспортних задач різними методами, а також можливість адміністрування даного веб-сервісу.

Програма виконує наступні функції (з точки зору пересічного користувача):

– надає зручний інтерфейс для вказання вхідних даних транспортної задачі та передбачає обрання способу знаходження базового та оптимального плану;

– надає результат розрахунку базового та оптимального плану у вигляді вихідних матриць та обрахованих сум витрат перевезення;

– надає інформаційну довідку у вигляді «карток» для ознайомлення із алгоритмом роботи методів обрахунку;

– надання інформаційної довідки про розроблений проєкт.

Програма виконує наступні функції (з точки зору адміністратору):

– надає можливість виконати авторизацію, щоб зайти до панелі адміністратора;

– надає можливість вийти із облікового запису адміністратору;

– надає можливість виконувати редагування наповнення контенту наповнення сайту, тобто редагувати данні, які знаходяться у базі даних;

– виконувати додавання «інформаційних карток», тобто виконує їх створення;

– надає можливість редагування «інформаційних карток»;

– надає можливість видаляти «інформаційні картки».

Даний веб-застосунок розробляється на платформі Asp.Net, отже для нормального функціонування програми, повинні виконуватися наступні умови до апаратної частини сервера Asp.Net:

– процесор: Мінімальним вимогам до процесора на сервері Asp.Net є наявність 64-бітного процесора з тактовою частотою не менше 1.4 ГГц. Процесор повинен мати достатню кількість ядер (не менше 4) та пам'яті кешу (не менше 8 МБ), щоб забезпечити ефективну обробку запитів;

– пам'ять: Мінімальним обсягом оперативної пам'яті на сервері Asp.Net є 2 ГБ. Однак, для високонавантажених веб-додатків може знадобитися більш великий обсяг пам'яті;

– дисковий простір: На сервері повинен бути достатній обсяг вільного місця на жорсткому диску, щоб забезпечити зберігання даних, включаючи файли конфігурації, журнали і кеш;

– операційна система: Asp.Net підтримується різними операційними системами, включаючи Windows Server 2019, Windows Server 2016 і Windows 10;

– браузер: Для взаємодії з веб-додатком, що працює на сервері Asp.Net, необхідний браузер, який підтримує HTML5 та CSS3;

– інтернет-з'єднання: Для доступу до веб-додатку, що працює на сервері Asp.Net, необхідне стабільне інтернет-з'єднання з достатньою швидкістю передачі даних.

Для нормального функціонування програми, повинні виконуватися наступні умови до апаратної частини клієнту:

– процесор: Мінімальні вимоги до процесора на клієнтському комп'ютері для запуску Asp.Net застосунку не є дуже високими. Найчастіше достатньо процесора з тактовою частотою від 1 ГГц та з підтримкою SSE2;

– пам'ять: Мінімальним обсягом оперативної пам'яті на клієнтському комп'ютері є 512 МБ. Однак, для більш комфортної роботи з веб-додатком може знадобитися більш великий обсяг пам'яті;

– відеокарта: Для відображення графіки та відео в Asp.Net застосунку необхідна відеокарта з підтримкою DirectX 9 або вище;

– дисковий простір: На клієнтському комп'ютері повинен бути достатній обсяг вільного місця на жорсткому диску, щоб забезпечити зберігання даних, включаючи файли кешу та тимчасові файли;

– операційна система: Asp.Net підтримується різними операційними системами, включаючи Windows 10, Windows 8.1, Windows 7 та macOS;

– браузер: Для взаємодії з Asp.Net застосунком необхідний браузер, який підтримує HTML5 та CSS3. Рекомендовано використовувати найновіші версії Google Chrome, Mozilla Firefox, Microsoft Edge або Safari.

Умови до програмної частини, для нормального функціонування програми:

– встановлення .NET Framework: Для запуску Asp.Net застосунка необхідно встановити відповідну версію .NET Framework. Найновіша версія .NET Framework на даний момент - .NET Framework 4.8;

– використання середовища Visual Studio: Для розробки Asp.Net застосунка рекомендується використовувати середовище Visual Studio. Для цього необхідно встановити відповідну версію Visual Studio;

– використання ASP.NET Core: Рекомендовано використовувати ASP.NET Core, оскільки він є більш сучасним та ефективним фреймворком для розробки веб-додатків, ніж традиційний ASP.NET;

– використання баз даних: Для збереження та обробки даних в Asp.Net застосунку можна використовувати різні бази даних, включаючи SQL Server, MySQL, Oracle та інші;

– використання різних бібліотек та фреймворків: Для розширення функціональності Asp.Net застосунку можна використовувати різні бібліотеки та фреймворки, наприклад, jQuery, Bootstrap, AngularJS та інші;

– безпека: Для забезпечення безпеки Asp.Net застосунку рекомендується використовувати різні методи захисту від атак, такі як шифрування даних, валідація введених даних та інші;

– тестування: Для забезпечення якості Asp.Net застосунку варто виконувати тестування застосунку, включаючи модульне тестування та інтеграційне тестування.

4.2 Характеристики програми

Веб-застосунок, що використовує фреймворк ASP.Net для створення та розгортання на веб-сервері.

Цей застосунок являє собою серверну частину. Клієнтом виступає веб-браузер, за допомогою якого відбувається взаємодія із серверною частиною та відображення інтерфейс користувача, який дозволяє вводити дані транспортної задачі та відображати результати обчислень, виконаних на сервері.

На стороні сервера реалізовані методи, які розв'язують транспортну задачу, а також панель адміністратора, яка дозволяє додавати, видаляти та редагувати дані, що являються собою інформаційне та довідкове наповнення веб-ресурсу.

Застосунок використовує базу даних для зберігання даних про транспортну задачу та контент, який буде відображатися на сторінках застосунку.

У цьому веб-сервісі реалізовані різні методи захисту від атак, такі як шифрування даних, валідація введених моделей даних, обмеження доступу до панелі адміністратора та інші, щоб забезпечити безпеку даних та користувачів.

Загалом, цей ASP.Net застосунок можна охарактеризувати як веб-додаток, який дозволяє вирішувати транспортну задачу та має адміністративну панель для наповнення контентом.

4.3 Звертання до програми

Звертання до програми відбувається за допомогою будь-якого сучасного веб-браузеру, за умови наявності мережі інтернет, зокрема через адаптивну верстку – надається зручна можливість використання веб-застосунку і на мобільних платформах без обмежень в користувацькому функціоналі, як для звичайного користувача так і для адміністратора ресурсу.

4.4 Початкові та вихідні дані

До початкових даних, відносяться системна інформація, яка описує та надає можливість коректного функціонування веб-застосунку, зокрема:

– файл конфігурацій – в ньому зберігається загальна інформація проєкту (назва, контактні данні та інше), а також рядок з’єднання із базою даних;

– перший авторизований користувач із роллю адміністратора, данні про якого вказуються під час створення бази даних.

До вхідних даних належать:

– матриця тарифів - це двовимірний масив, що містить вартість перевезення одиниці товару з кожного постачальника до кожного пункту зберігання, де кожен елемент відповідає конкретній комбінації постачальника та пункту зберігання;

– масив потреб - це одновимірний масив, що містить кількість одиниць товару, які необхідно перевезти до кожного пункту зберігання;

– масив постачальників - це одновимірний масив, що містить кількість одиниць товару, які може перевезти кожен постачальник;

– дані для адміністративної панелі - це дані, які зберігаються у базі даних та включають в себе текстовий та мультимедійний контент, який можна змінювати за допомогою панелі адміністратора. Ці дані можуть

включати такі елементи, як тексти сторінок, зображення, відео, посилання та інші.

4.5 Алгоритмічна база

Розроблений сервіс підтримує наступні алгоритми:

- подвійної переваги;
- мінімального елементу;
- північно-західного кута;
- апроксимація Фогеля;
- метод потенціалів;
- Флойда-Уоршила;
- генетичний алгоритм.

Для кожного із наведених алгоритмів в кодовій базі проєкту, що розроблявся є власний клас [35]. В таблиці 4.1 наведено назви алгоритмів і назви класів, які їм відповідають.

Таблиця 4.1 – Алгоритми і класи які їм відповідають

№	Назва алгоритма	Назва класу
1	Подвійної переваги	DoublePreference
2	Мінімального елементу	MinElemet
3	Північно-західного кута	NordWest
4	Апроксимація Фогеля	VogelApproximation
5	Метод потенціалів	OptimizationMethodPotentials
6	Флойда-Уоршила	FloydWarshall
7	Генетичний алгоритм	GeneticAlgorithm

Після реалізації кожного алгоритму було проведено UNIT-тестування [36], для визначення коректності та ефективності роботи кожного класа.

Unit тести – це автоматичні тести, які перевіряють невеликі частини коду, такі як функції або методи, ізольовано від решти системи. Вони дають

змогу розробникам переконатися, що кожна частина коду працює правильно і відповідає очікуваній поведінці. Вони пишуться мовою програмування і можуть бути запущені автоматично для швидкої перевірки коду при кожній його зміні [37].

Для виконання тестування на платформі .Net існує багато фрейворків, найпопулярнішими є NUnit[38] та XUnit[39].

Також важно зазначити, що все тести були написані дотримуючись AAA-структури тестового метода [40].

AAA структура тестового метода складається з трьох частин:

Arrange – частина метода, де створюються всі тестові дані.

Act – частина тестового метода, де відбувається виклик метода, який ми тестуємо.

Assert – частина метода, де відбуваються всі тестові перевірки.

Результат тестування наведено на рисунку 4.1.

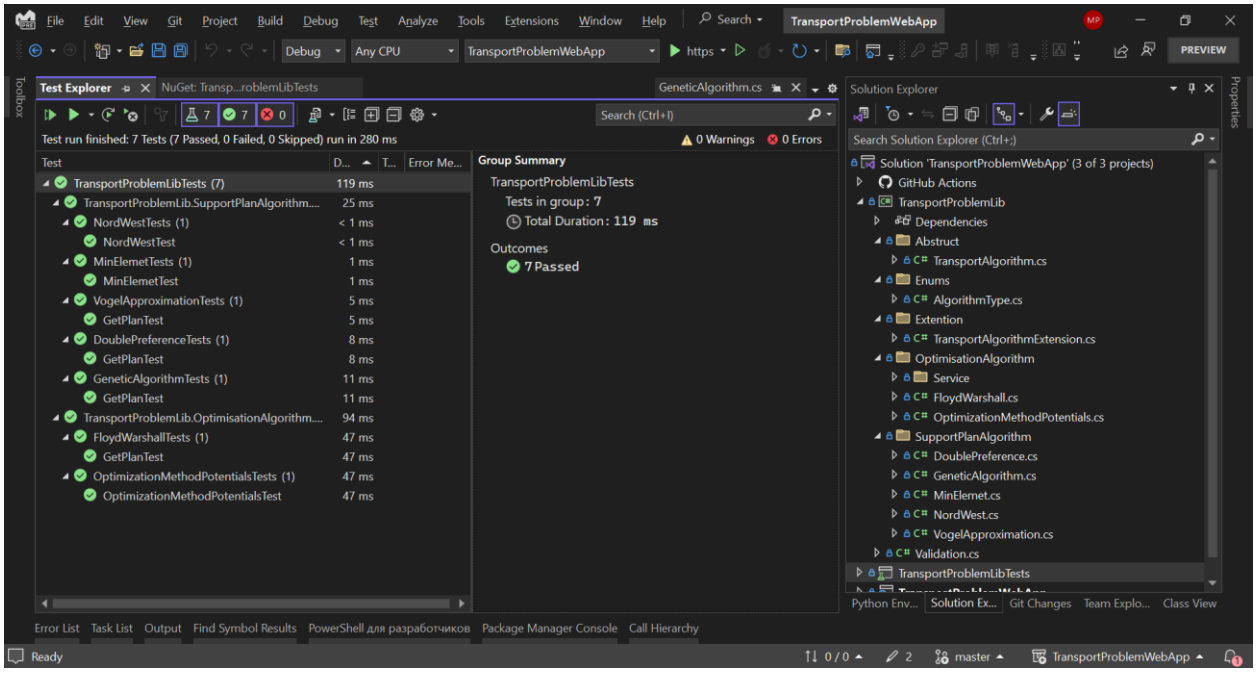


Рисунок 4.1 – Результат проведення тестування

Головним параметром, який оцінюється в даному тестування – є час виконання, він напряму залежить від пристрою на якому відбувається виконання програмного коду, але загальна тенденція проглядається добре.

Як бачимо найдовшим серед всіх алгоритмів виявився алгоритм Флойда-Уоршила, при тому що його точність бажає бути кращою.

Наступним за швидкістю виконання виявився генетичний алгоритм, його швидкість майже в 2 рази випереджає алгоритм Флойда-Уоршила, хоч і все одно є достатньо великою, в порівнянні з іншими алгоритмами, але точність розрахунків при цьому є найкращою.

Кожному алгоритму на вхід надавалися однакові данні, що складалися із матриці-цін перевезення, масив-потреб, та масив-постачальників. Данні, що використовувалися для тестування наведені на рисунку 4.2.

```
30 references
internal static class SeedData
{
    12 references | 7/7 passing
    public static double[,] Prices { get; }=
        {
            { 4,3,4,11,9 },
            { 3,4,7,15,8 },
            { 7,4,2,8,15}
        };

    9 references | 7/7 passing
    public static double[] Reserves { get; } = { 40, 30, 10 };
    9 references | 7/7 passing
    public static double[] Needs { get; } = { 20, 21, 7, 24, 8 };
}
```

Рисунок 4.2 – Данні на яких відбувалося тестування

Тепер варто зазначити точність розрахунку, в таблиці 4.2 наведено перелік алгоритмів та значень.

Таблиця 4.2 – Алгоритми і розраховані значення

№	Назва алгоритма	Значення
1	Подвійної переваги	464
2	Мінімального елементу	464
3	Північно-західного кута	659
4	Апроксимація Фогеля	476
5	Метод потенціалів	451
6	Флойда-Уоршила	464
7	Генетичний алгоритм	241

Як можна побачити найкращий результат показав генетичний алгоритм, але варто зауважити, через нюанси його роботи, значення не постійне і може відрізнятися з кожним запуском.

5 КЕРІВНИЦТВО ОПЕРАТОРА

5.1 Призначення програми

Застосунок приймає матрицю тарифів, масив потреб та постачальників, і знаходить оптимальний спосіб перевезення товарів з постачальників до пунктів зберігання з мінімальним загальним витратами.

Крім того, застосунок має адміністративну панель, яка дозволяє адміністраторам змінювати вміст сайту без необхідності редагування вихідного коду. Це забезпечує можливість зміни текстового та мультимедійного контенту, такого як зображення, відео, посилання та інші, що дозволяє адаптувати сайт до потреб користувачів.

Отже, загальне призначення цього веб-застосунку – надання користувачам інструменту для вирішення транспортної задачі та можливість змінювати вміст сайту за допомогою адміністративної панелі.

5.2 Умови виконання програми

Умови виконання програми, можуть бути розглянуті з трьох різних аспектів: апаратної, програмної та інфраструктурної.

Апаратні умови виконання програми включають наявність достатньо потужного сервера з операційною системою Windows та підтримкою середовища виконання ASP.NET, достатню кількість оперативної пам'яті, місце для зберігання даних та інші технічні характеристики, необхідні для безперебійного функціонування програми.

Програмні умови виконання програми включають наявність необхідного програмного забезпечення, такого як веб-сервер (наприклад, IIS), база даних (наприклад, MS SQL Server, що використовується наразі), середовище виконання ASP.NET та інші залежно від конкретних вимог програми.

Інфраструктурні умови виконання програми включають забезпечення безпеки даних, резервного копіювання, захисту від злому та зловживань, а також розвитку та підтримки інфраструктури.

Отже, для нормального функціонування програми, яка вирішує транспортну задачу та має адміністративну панель для зміни вмісту сайту, необхідні достатньо потужний сервер з операційною системою Windows та підтримкою середовища виконання ASP.NET, необхідне програмне забезпечення, таке як веб-сервер, база даних та середовище виконання ASP.NET, а також безпека даних, резервне копіювання, захист від злому та зловживань, а також підтримка та розвиток інфраструктури.

5.3 Виконання програми

Виконання програми залежить від багатьох факторів, таких як апаратна та програмна частини сервера та клієнта, наявність необхідних бібліотек, налаштування серверу, обсяг та складність обчислювальних завдань, які виконує програма, тощо.

Розберемо декілька з них:

– апаратні фактори - це все, що пов'язано з обладнанням, на якому запущена програма. Це можуть бути характеристики сервера, який виконує програму (наприклад, об'єм пам'яті, кількість процесорів, швидкість дискової підсистеми), а також характеристики клієнтських пристроїв (наприклад, об'єм пам'яті, швидкість процесора, якість підключення до Інтернету). Якщо апаратні фактори не задовольняють потреби програми, то це може призвести до сповільнення роботи програми, відмови у запуску або інших проблем;

– програмні фактори - це характеристики програмного забезпечення, на якому запущена програма. Це може бути операційна система, налаштування сервера, якість написання коду програми, наявність інших програм, які працюють одночасно з даною програмою. Якщо програмні фактори не оптимальні, то це може призвести до різноманітних

проблем з виконанням програми, таких як некоректна робота, відмови, збої і тощо;

– людські фактори - це все, що пов'язано з користувачами програми, розробниками, тестувальниками та іншими людьми, які займаються програмою. Наприклад, недостатнє навчання користувачів може призвести до того, що вони не зможуть ефективно користуватися програмою. Неправильна комунікація між розробниками може призвести до неправильного розуміння вимог до програми та різноманітних компонентів.

Якщо всі умови виконання програми були задовільнені, програма може бути успішно виконана. Програма, яка вирішує транспортну задачу, приймає матрицю тарифів та масив потреб та постачальників, обчислює найкоротші маршрути від постачальників до споживачів і знаходить оптимальний план перевезення. Крім того, панель адміністратора дозволяє змінювати наповнення сайту та зберігати ці дані у базі даних.

Під час виконання програми можуть виникати різні помилки та проблеми, такі як збої серверу, проблеми зі з'єднанням мережі, некоректні дані, введені користувачем, та інші. Однак, з відповідними заходами забезпечення безпеки та моніторингу виконання програми, можна забезпечити стабільну та безперебійну роботу застосунку.

Крім того, ефективність виконання програми може залежати від різних факторів, таких як оптимізація коду, розмір оброблюваних даних, кількість запитів до бази даних, потреби в ресурсах комп'ютера (таких як оперативна пам'ять, процесор, диск), швидкість інтернет-з'єднання тощо.

Також важливо враховувати можливі взаємодії з іншими програмами або сервісами, які можуть впливати на роботу нашої програми, а також можливі відмови апаратної частини, які можуть призвести до зупинки програми або втрати даних.

Отже, виконання програми є складним процесом, який залежить від багатьох факторів, і вимагає ретельного планування та тестування, щоб забезпечити її ефективність та стабільність.

5.4 Повідомлення оператору

Веб-застосунок, що вирішує транспортну задачу, має вбудовану систему моніторингу роботи, то при виявленні помилок у роботі програми або її компонентів, система може створити повідомлення, яке буде автоматично відправлене оператору. Оператор отримає це повідомлення та зможе швидко втрутитися для вирішення проблеми, забезпечивши безперебійну роботу програми.

Приклад такого повідомлення наведено на рисунку 5.1, де в панелі адміністратора продемонстровано обробку події, коли не всі обов'язкові поля введенні і система за допомоги концепції «валідації моделі» підказує користувачу про не коректні дії з його боку.

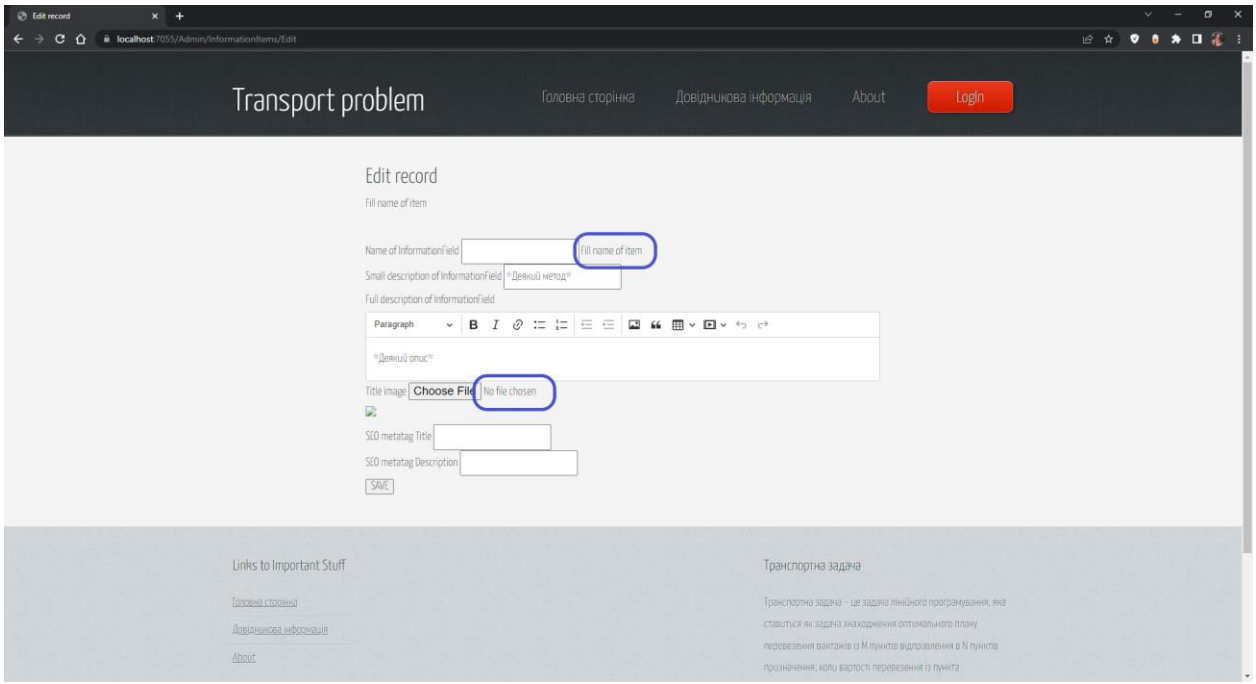


Рисунок 5.1 – Приклад повідомлення оператору

Крім того, програма може містити інтерфейс для відправки повідомлень оператору вручну. Наприклад, якщо виникне потреба в зміні налаштувань програми або вирішенні проблем, які не можна автоматично

виявити, користувач може створити повідомлення та відправити його оператору, який зможе відповісти на запит та надати допомогу.

Отже, повідомлення оператору є важливою складовою програми, що допомагає забезпечити її безперебійну роботу та швидке вирішення проблем.

5.5 Опис прикладу роботу застосунку

Головна сторінка розробленого застосунку складається із меню навігації (показано на рисунку 5.2).

Пункти головного меню:

– «Головна сторінка»;
– «Довідникова інформація»;
– «About»;
– «Login».

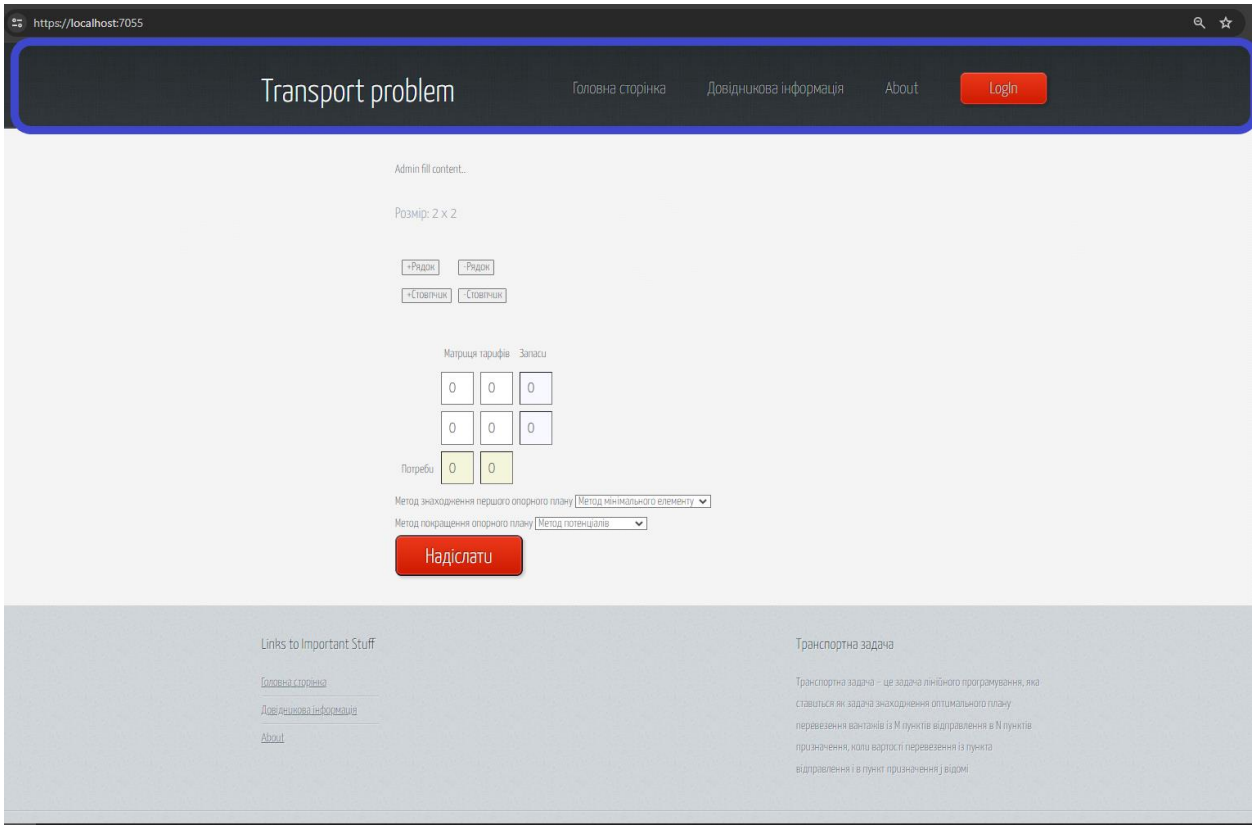


Рисунок 5.2 – Рисунок головного вікна із позначеними меню навігації

Використовуючи пункт «Головна сторінка» можна потрапити на головну сторінку ресурсу сайту з будь-якого місця.

Використовуючи пункт «Довідникова інформація» виконується переадресація на сторінку із інформаційними картками, кожна з яких містить дані певної теми (буде наведено далі).

Нижче панелі навігації знаходиться основна частина головної сторінки (наведено на рисунку 5.3), яка складається із наступних складових:

– контент, що заповнюється адміністратором сайту (розглянемо далі);

– вхідні данні для розрахунку;

– кнопка надіслати.

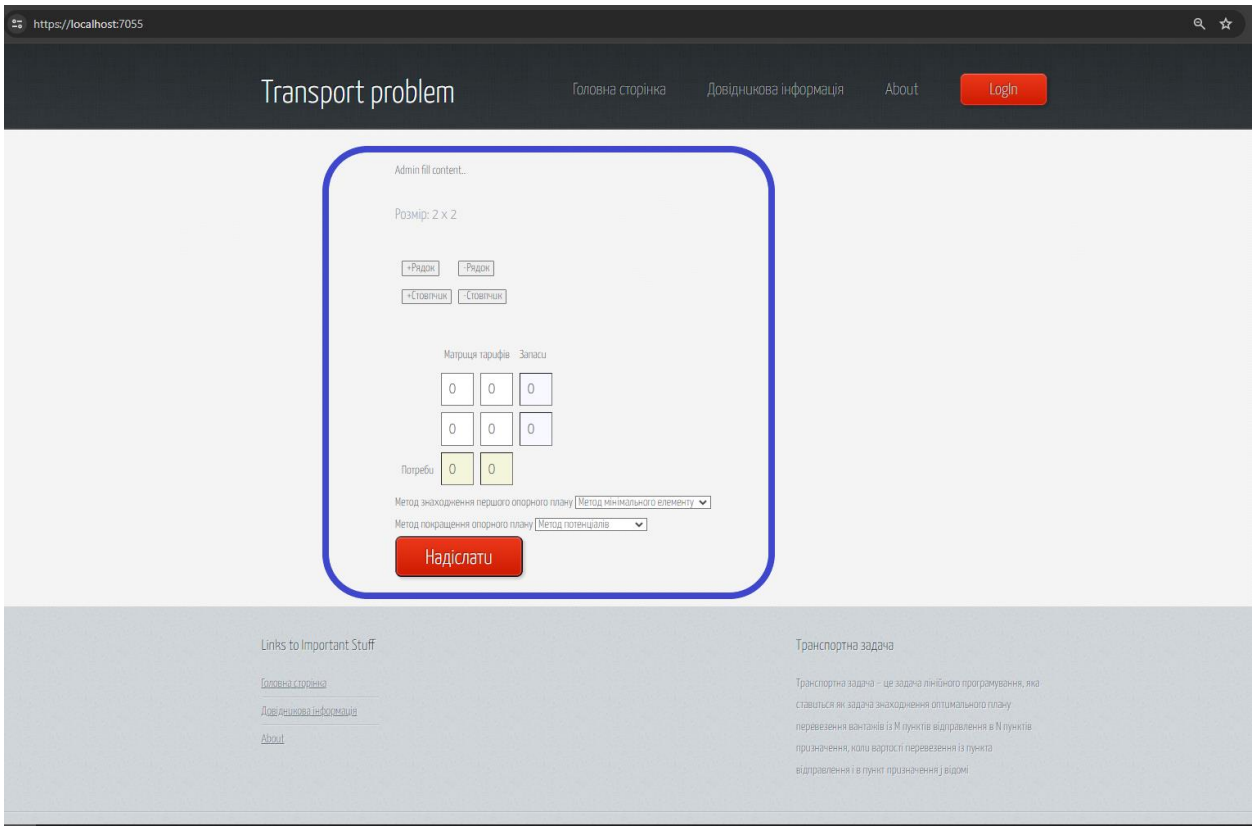


Рисунок 5.3 – Головна сторінка із позначеною головною частиною

Розглянемо інтерфейс введення даних транспортної задачі більш докладно. Головним елементом вхідних даних – є матриця перевезень, та тарифи, для збільшення або зменшення кількості елементів на формі

присутні кнопки збільшення/зменшення кількості рядків та стовпців. На рисунку 5.4 наведено приклад матриці із збільшеною кількістю рядків і стовпців.

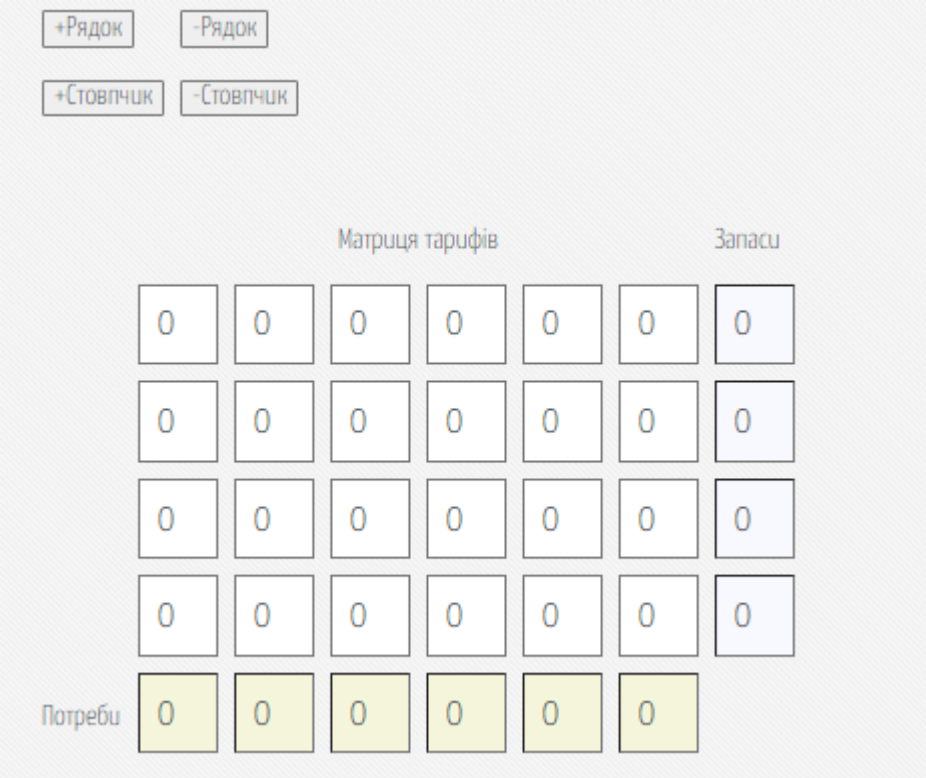


Рисунок 5.4 – Результат зміни розмірів матриці

Під матрицею тарифів знаходиться елемент, використовуючи який можна вибрати метод знаходження (розрахунку) опорного плану, а одразу під ним, є можливість вибрати метод оптимізації, показано на рисунку 5.5.

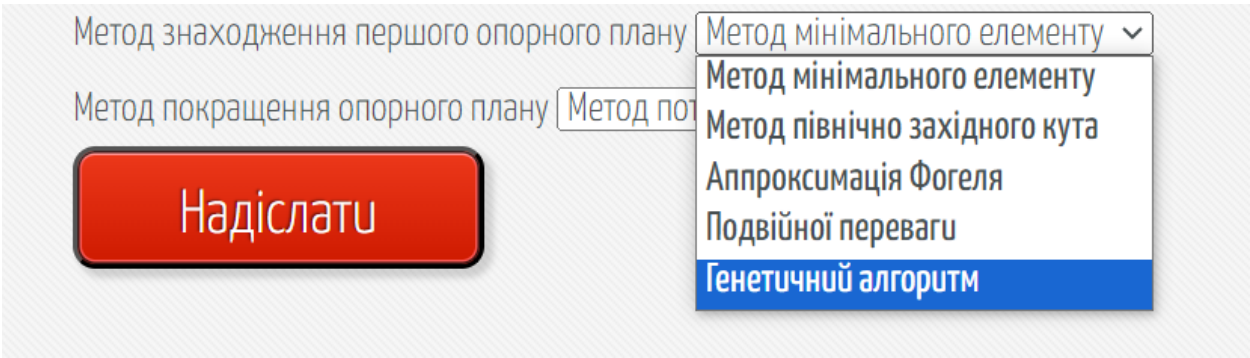
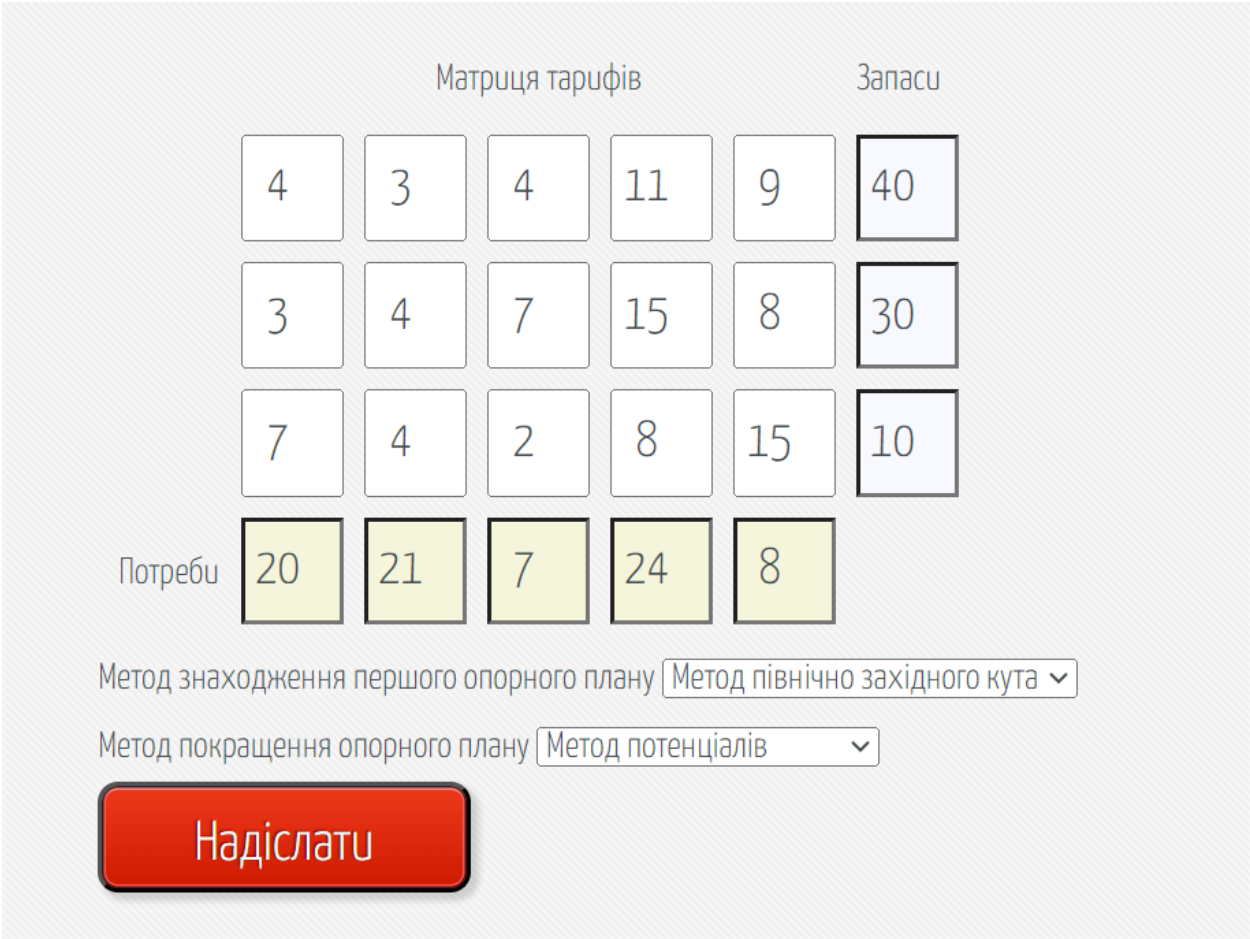


Рисунок 5.5 – Приклад вибору алгоритму розрахунку опорного плану

На рисунку 5.6 наведено приклад заповнення даними для розрахунку.



На рисунку 5.6 – Приклад заповнення вхідними даними

В самому кінці сторінки знаходиться «підвал» сторінки (рисунок 5.7).

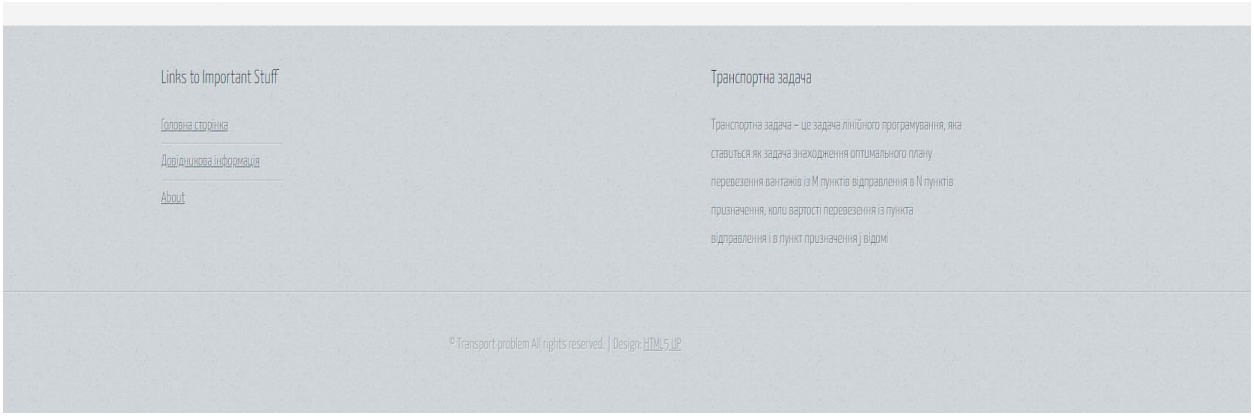


Рисунок 5.7 – Наведено «підвал» сторінки

Після введення даних та натискання на кнопку «Надіслати», виконується переадресація на сторінку, де користувач може побачити результат обрахунку, показано на рисунку 5.8.

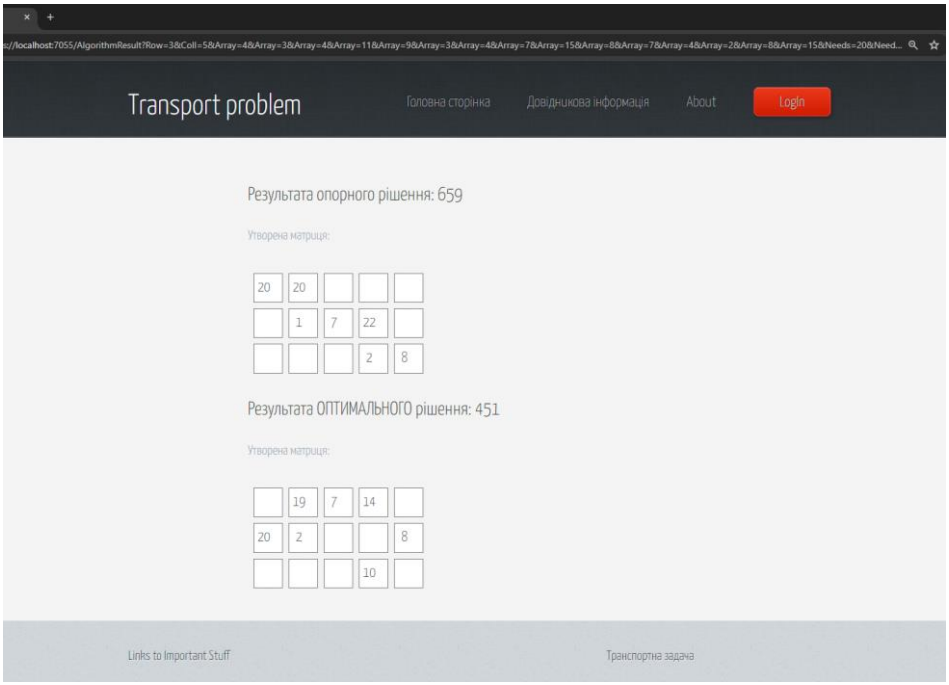


Рисунок 5.8 – Результат виконання обрахунку

Як можна побачити, результат представлений у вигляді двох матриць, що описують опорний та оптимальний план із розрахованою вартістю.

Також на сайті передбачена панель адміністратора для керування наповнення контентом сайту, на рисунку 5.9 наведено приклад редагування головної сторінки сайту.

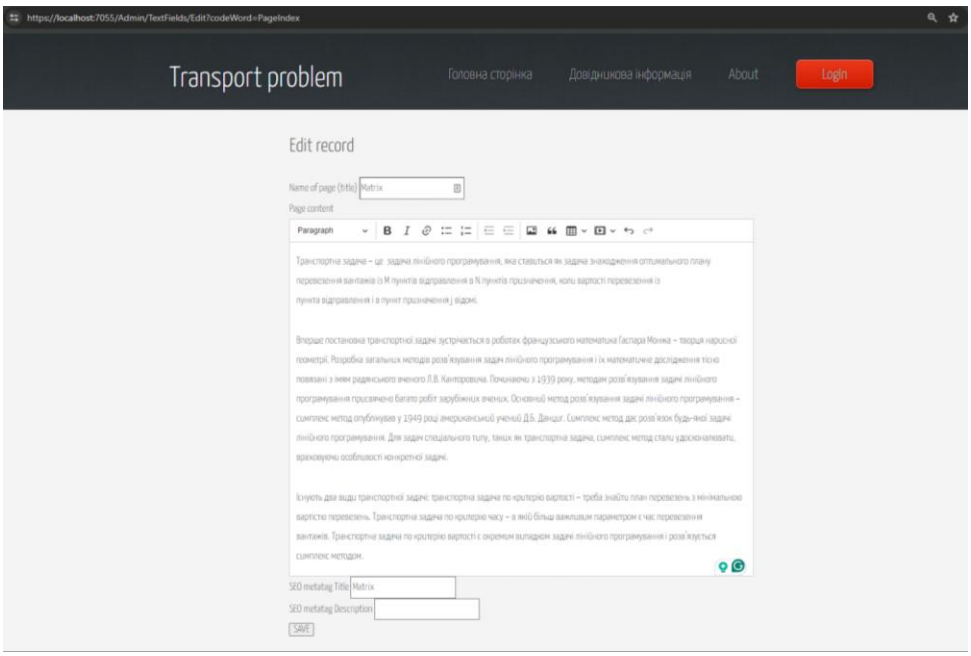


Рисунок 5.9 – Приклад редагування наповнення головної сторінки.

За схожим принципом адміністратор сервісу може редагувати інформаційні картки, що знаходяться на сторінці «Довідникова інформація», наведено на рисунку 5.10.

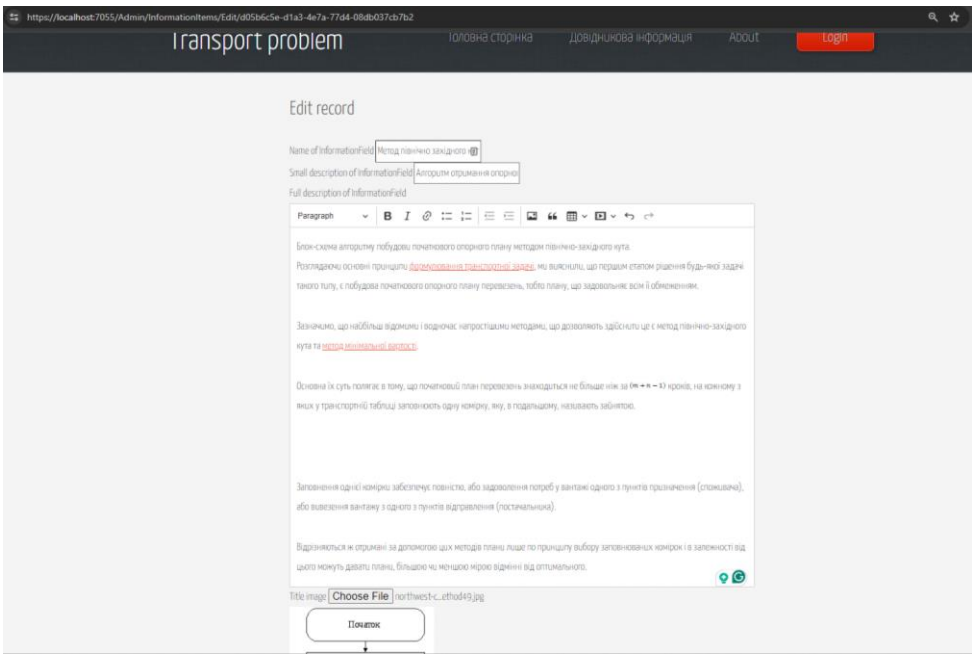


Рисунок 5.10 – Приклад редагування інформаційної картки

6 АНАЛІЗ РОЗГЛЯНУТИХ АЛГОРИТМІВ РОЗВ'ЯЗАННЯ ТРАНСПОРТНИХ ЗАДАЧ

Транспортна задача – це задача лінійного програмування, яка ставиться як задача знаходження оптимального плану перевезення вантажів із M пунктів відправлення в N пунктів призначення, коли вартості перевезення із пункту відправлення i в пункт призначення j відомі [41].

6.1 Метод подвійної переваги

Метод подвійної переваги при знаходженні опорного плану транспортної задачі зазвичай використовується в тому випадку коли таблиця вартостей ТЗ достатньо велика і перебір всіх її елементів є досить трудоємкою процедурою. Також слід відмітити, що у багатьох випадках рішення транспортної задачі методом потенціалів з використання методу подвійної переваги виявляється більш простим, зручним і швидким, в порівнянні з іншими методами [42].

Розглянемо алгоритм вирішення транспортної задачі даним методом:

– на першому кроці, у кожному стовпці транспортної таблиці відзначають знаком «V» комірку з найменшою вартістю;

– потім те ж проробляють в кожному рядку. В результаті виконання другого кроку, деякі комірки транспортної таблиці будуть містити відмітку «VV»;

– на останньому кроці, у комірки з подвійними відмітками (подвійна перевага) поміщають максимально можливі обсяги перевезень, щоразу виключаючи з розгляду відповідні стовпці або рядки. Потім розподіляють перевезення по комірках з одиничною відміткою. Решта перевезення розподіляють згідно алгоритму методу мінімального елемента.

Іншими словами, цей алгоритм можна звести до того, що перед початком заповнення таблиці необхідно позначити будь-якими символами

клітинки, які містять найменшу вартість у рядках, а потім – у стовпчиках. Таблицю починають заповнювати з клітинок, позначених двічі (які містять вартості, що є мінімальними і в рядку, і в стовпчику). Далі заповнюють клітинки, позначені один раз (що містять мінімальні вартості або в рядку, або в стовпчику), а вже потім – за методом мінімальної вартості [43].

6.2 Метод мінімального елементу

Даний метод (також відомий як метод мінімальної вартості) побудований на аналізі матриці вартості перевезень, тому дозволяє побудувати опорне рішення, яке є досить близьким до оптимального, або навіть відразу знайти оптимальний план [44].

Головна ідея методу полягає у наступному:

– виявлення комірки транспортної таблиці з найменшим тарифом на перевезення грузу (якщо виявиться, що є кілька осередків з однаковими та мінімальними тарифами – вибираємо будь-яку з них). У цей осередок виписуємо максимально можливий обсяг вантажу (X_{ij}), який можна доставити з відповідного цього осередку складу до відповідного споживача;

– обсяги запасів та потреб зменшуються на величину вантажу. Якщо запаси складу вичерпані, повністю викреслюємо цей рядок таблиці. Якщо потреби споживача повністю задоволені, повністю викреслюємо цей стовпець таблиці;

– продовжуємо так само, поки всі запаси не будуть вичерпані, а всі потреби задоволені;

– у результаті отримаємо опорний план перевезень для транспортної задачі.

6.3 Метод північно-західного кута

Цей метод є найбільш відомим і водночас найпростішим. Основна суть полягає в тому, що початковий план перевезень знаходиться не більше ніж за m + n - 1 кроків, на кожному з яких у транспортній таблиці заповнюють одну комірку, яку, в подальшому, називають зайнятою.

Розглянемо алгоритм:

а) на першому кроці, із співвідношення $x_{11} = \min(a_1, b_1)$ знаходимо значення об'єму перевезень від першого постачальника до першого споживача. Зазначимо, що при цьому, можливі три варіанти:

1) якщо $a_1<b_1$, то $x_{11}=b_1$, рядок i=1 виключається з подальшого розгляду (запаси постачальника A_1 повністю вичерпані), а потреби першого споживача b_1 (стовпець j=1) зменшується на величину a_1;

2) якщо $a_1>b_1$, то $x_{11}=b_1$, стовпець j=1 виключається з подальшого розгляду (потреби споживача B_1 повністю задоволені), а наявність вантажу у першого постачальника (рядок i = 1) зменшується на величину b_1;

3) якщо $a_1=b_1$, то $x_{11}=a_1=b_1$, рядок $i_1=1$ і стовпець $j_1=1$ виключаються з подальшого розгляду. Даний варіант призводить до виродження вихідного плану;

б) потім аналогічні операції проробляють з частиною таблиці, починаючи з її північно-західного кута;

в) на останньому кроці, залишиться один рядок і один стовпець. Після заповнення комірки, яка знаходиться на їх перетині, процес побудови початкового опорного плану методом північно-західного кута завершується;

г) після цього, отриманий план необхідно перевірити на вродженість. Якщо кількість зайнятих комірок дорівнює (m+n-1), то план є невироджених, в іншому випадку – виродженим. Якщо план вироджений, тобто кількість зайнятих комірок виявилася меншою за число (m+n-1), то незаповнені

комірки з мінімальними вартостями перевезень заповнюються нулями, щоб загальна кількість зайнятих комірок стала рівною (m+n-1).

6.4 Метод апроксимація Фогеля

Сенс апроксимації Фогеля у знаходженні різниці (за модулем) між парою мінімальних тарифів у кожному рядку та стовпці. Потім рядок або стовпець з найбільшою різницею заповнюються у напрямку від клітини з мінімальним тарифом до клітини з максимальним.

Далі розглянемо кроки алгоритму:

– насамперед знаходимо для кожного рядка транспортної таблиці абсолютні різниці (за модулем, тобто без знаку) між двома мінімальними тарифами у цьому рядку. Те саме робимо і для кожного стовпця: шукаємо в ньому два мінімальні тарифи і знаходимо їх різницю по модулю. Знайдені різниці виписуємо у доданий стовпець (D_i) та доданий рядок (D_j);

– серед обчислених різниць (рядками і стовпцями) вибираємо найбільшу;

– потім у рядку або стовпці, якому відповідає максимальна різниця, шукаємо клітку з мінімальним тарифом. Заповнюємо її максимально можливим обсягом вантажоперевезення;

– потім повторюємо всі вищеописані дії знову, тільки не враховуючи заповнені клітини і обрані раніше різниці (D_i і D_j). І так доти, доки не буде повністю знайдено опорний план;

– в результаті ми отримуємо опорний план.

6.5 Метод потенціалів

Даний розподільний метод одержання оптимального розв’язку є досить трудомістким. У кожному допустимому розв’язку для кожної вільної змінної необхідно будувати цикли й визначати зміни цільової функції [45].

Також варто зазначити, що за для знаходження оптимального плану задачі, необхідно відштовхуватися від початкового плану, знайденого, наприклад, методом північно-західного кута.

Алгоритм розв’язання складається з наступних кроків:

– відповідно до вихідних даних складається транспортна матриця розмірністю n на m, де n - кількість джерел живлення, m - кількість споживачів;

– знаходиться допустимий розв’язок, наприклад методом найменшої питомої вартості;

– для допустимого розв’язку кожному i-му рядку й кожному j-му стовпцю транспортної матриці привласнюється значення потенціалу V_i і U_j (i=1,2,...n; j=1,2,...m). Для кожної базисної змінної сума потенціалів дорівнює питомій вартості $V_i+U_j=c_{ij}$;

– довільно задавшись значенням одного з потенціалів, за виразом $V_i+U_j=c_{ij}$, який є справедливий для базисних змінних, обчислюються значення інших потенціалів;

– для всіх вільних змінних перевіряється співвідношення суми потенціалів з питомою вартістю. Якщо для всіх вільних змінних $V_i+U_j<c_{ij}$, то отриманий розв’язок є оптимальним;

– якщо є вільні змінні, для яких $V_i+U_j>c_{ij}$, то вибирається кожна із цих вільних змінних і переводиться в базис. Для цього будується цикл перерахунку (замкнута ламана лінія), початкова вершина якого лежить у клітинці обраної вільної змінної. Інші вершини циклу лежать у клітинках, які відповідають базисним змінним. Початковій вершині циклу привласнюється знак "+", що відповідає збільшенню змінної. Далі знаки вершин циклу чергуються. Знаки "+" відповідають збільшенню базисних змінних, знаки "-" - їхньому зменшенню;

– з від’ємних вершин циклу вибирається вершина з найменшим значенням базисної змінної й на цю величину змінюються всі змінні, які лежать у вершинах циклу. В додатних вершинах змінні збільшуються, в

від’ємних - зменшуються. При цьому обрана вільна змінна стає базисною, а найменша за величиною базисна змінна в від’ємній вершині циклу стає вільною (рівною нулю);

– для знову отриманого розв’язку обчислювальна процедура повторюється, починаючи з пункту 3.

6.6 Метод Флойда-Уоршила

Даний алгоритм є динамічний алгоритм знаходження найкоротших відстаней між усіма вершинами зваженого орієнтованого графа. Він є ефективним для розрахунку найкоротших шляхів у щільних графах, коли має місце велика кількість пар ребер між парами вершин. У разі розріджених графів з ребрами невід'ємної ваги найкращим вибором є використання алгоритму Дейкстри для кожного можливого вузла.

Алгоритм складається з наступних кроків:

– ініціалізація: Створити матрицю суміжності для графа, де елемент (i, j) представляє вагу ребра від вершини i до вершини j. Якщо між вершинами немає ребра, встановити вагу на нескінченність. Заповнити головну діагональ матриці нулями;

– основний цикл: Для кожної вершини графа розглянути всі пари вершин (i, j) і оновити вагу шляху від i до j через поточну вершину k, якщо такий шлях коротший;

– завершення: Після завершення виконання алгоритму матриця суміжності містить найкоротші відстані між всіма парами вершин у графі.

6.7 Генетичний алгоритм

Генетичні алгоритми виникли в результаті спостереження і спроб копіювання природних процесів, що відбуваються в світі живих організмів, зокрема, еволюції та пов'язаної з нею селекції (природнього відбору)

популяцій живих істот. Звичайно, при подібному зіставленні нейронних мереж і генетичних алгоритмів слід звертати увагу на принципово різну тривалість протікання згадуваних природних процесів, тобто на надзвичайно швидку обробку інформації в нервовій системі і дуже повільний процес природньої еволюції. Однак при комп'ютерному моделюванні ці відмінності виявляються несуттєвими. Генетичний алгоритм являє собою метод, що відображає природну еволюцію методів вирішення проблем, і в першу чергу задач оптимізації. Генетичні алгоритми — це процедури пошуку, засновані на механізмах природного відбору і спадкоємства. У них використовується еволюційний принцип виживання найбільш пристосованих особин [46].

Класичний генетичний алгоритм (який також називається елементарним чи простим генетичним алгоритмом) складається з наступних кроків:

- ініціалізація, або вибір вихідної популяції хромосом;
- оцінка пристосованості хромосом в популяції;
- перевірка умови зупинки алгоритму;
- селекція хромосом;
- застосування генетичних операторів;
- формування нової популяції; вибір «найкращої» хромосоми.

6.8 Порівняльна таблиця алгоритмів

Порівняння алгоритмів наведено у таблиці 6.1.

Таблиця 6.1 – Порівняння алгоритмів розрахунку транспортної задачі.

Назва алгоритму	Час виконання	Кількість пам’яті, що необхідна застосунку, для використання алгоритму	Точність розрахунків
1	**2**	**3**	**4**
Подвійної переваги	9ms	18Mb	92.95%
Мінімального елементу	56ms	17Mb	92.95%
Північно-західного кута	6ms	18Mb	63.49%
Апроксимація Фогеля	9ms	18Mb	79.25%
Метод потенціалів	12ms	17Mb	87.134%
Флойда-Уоршила	5ms	18Mb	94.95%
Генетичний алгоритм	16ms-48ms	18Mb	100%

ВИСНОВКИ

Бакалаврська робота присвячена задачі маршрутизації транспортних засобів з використанням різних алгоритмів. Метою є мінімізація сумарної вартості транспортних перевезень продукції споживачам від постачальників.

У першому розділі здійснено аналіз задачі маршрутизації транспортних засобів та надано огляд існуючих методів і алгоритмів їх розв’язання. Було наведено перелік аналогів програмних продуктів, що вирішують задачі маршрутизації транспортних засобів використовуючи обмежену кількість алгоритмів. З метою збільшення ефективності роботи відомих алгоритмів розв’язання даного типу задач, була сформульована відповідна постановка задачі та мета дослідження.

У другому розділі було наведено вибір та обґрунтування структури системи що проєктується. Було наведено перелік існуючого інструментарію розробки, який включає платформи на основі якої розробляється застосунок. Наведено та обґрунтовано використання СКБД, а також перелік алгоритмів, що включає:

– подвійної переваги;
– мінімального елементу;
– північно-західного кута;
– апроксимація Фогеля;
– метод потенціалів;
– Флойда-Уоршила;
– генетичний алгоритм.

У третьому розділі міститься опис та обґрунтування архітектури системи, що включає опис проєктування бази даних застосунку.

У четвертому та п’ятому розділі наведено опис програмного забезпечення. Крім того, розписано покрокова інструкція роботи з програмою, що розв’язує задачу маршрутизації транспортних засобів.

У результаті роботи над бакалаврською роботою було:

– проаналізовано прикладів веб-версій для розв’язання транспортних задач;

– розроблено програмний застосунок для вирішення транспортних задач із використанням усіх найпопулярніших алгоритмів;

– проведено дослідження ефективності розроблених алгоритмів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Transportation problem [Electronic resource]. – Access mode: https://www.oreilly.com/library/view/quantitative-techniques-theory/9789332512085/xhtml/ch5sec1.xhtml#:~:text=The%20transportation%20problem%20is%20a,to%20a%20number%20of%20destinations.

2. Транспортна задача [Електрон. ресурс]. – Режим доступу: https://ela.kpi.ua/bitstream/123456789/26217/1/Transportna_zadacha.doc#:~:text=%D0%86%D1%81%D0%BD%D1%83%D1%8E%D1%82%D1%8C%20%D0%B4%D0%B2%D0%B0%20%D0%B2%D0%B8%D0%B4%D0%B8%20%D1%82%D1%80%D0%B0%D0%BD%D1%81%D0%BF%D0%BE%D1%80%D1%82%D0%BD%D0%BE%D1%97%20%D0%B7%D0%B0%D0%B4%D0%B0%D1%87%D1%96,%D0%BF%D0%B0%D1%80%D0%B0%D0%BC%D0%B5%D1%82%D1%80%D0%BE%D0%BC%20%D1%94%20%D1%87%D0%B0%D1%81%20%D0%BF%D0%B5%D1%80%D0%B5%D0%B2%D0%B5%D0%B7%D0%B5%D0%BD%D0%BD%D1%8F%20%D0%B2%D0%B0%D0%BD%D1%82%D0%B0%D0%B6%D1%96%D0%B2.

3. Travelling Salesman Problem using Dynamic Programming [Electronic resource]. – Access mode: https://www.geeksforgeeks.org/travelling-salesman-problem-using-dynamic-programming/.

4. Branch and Bound Algorithm [Electronic resource]. – Access mode: https://www.geeksforgeeks.org/branch-and-bound-algorithm/.

5. Dynamic Programming or DP [Electronic resource]. – Access mode: https://www.geeksforgeeks.org/dynamic-programming/.

6. What web development technologies should you use? [Electronic resource]. – Access mode: https://www.orientsoftware.com/technologies/web-technologies/.

7. Statistical Methods for Solving Transportation Problems [Electronic resource]. – Access mode:

https://www.researchgate.net/publication/334677518_Statistical_Methods_for_Solving_Transportation_Problems.

8. Save data back to the database in .NET Framework applications [Electronic resource]. – Access mode: https://learn.microsoft.com/en-us/visualstudio/data-tools/save-data-back-to-the-database?view=vs-2022.

9. An Introductiontothe Mass Transportation Theoryandits Applications [Electronic resource]. – Access mode: https://www.math.cmu.edu/cna/lecturenotesfiles/notes3.pdf.

10. Транспортна задача лінійного програмування [Електрон. ресурс]. – 2022. – Режим доступу: https://www.mathros.net.ua/transportna-zadacha-matematychna-postanovka-zadachi.html#transport-task-math-model.

11. Транспортна задача [Електрон. ресурс]. – Режим доступу: https://lib.chmnu.edu.ua/pdf/posibnuku/226/8.pdf.

12. Методи побудови опорного плану [Електрон. ресурс]. – Режим доступу:
https://wiki.cuspu.edu.ua/index.php/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4%D0%B8_%D0%BF%D0%BE%D0%B1%D1%83%D0%B4%D0%BE%D0%B2%D0%B8_%D0%BE%D0%BF%D0%BE%D1%80%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BF%D0%BB%D0%B0%D0%BD%D1%83_%D0%A2%D0%97.#:~:text=%D0%9C%D0%B5%D1%82%D0%BE%D0%B4%20%D0%BF%D0%BE%D0%B4%D0%B2%D1%96%D0%B9%D0%BD%D0%BE%D1%97%20%D0%BF%D0%B5%D1%80%D0%B5%D0%B2%D0%B0%D0%B3%D0%B8,-%D0%AF%D0%BA%D1%89%D0%BE%20%D1%80%D0%BE%D0%B7%D0%BC%D1%96%D1%80%D0%BD%D1%96%D1%81%D1%82%D1%8C%20%D0%B7%D0%B0%D0%B4%D0%B0%D1%87%D1%96&text=%D0%97%D0%B3%D1%96%D0%B4%D0%BD%D0%BE%20%D0%B7%20%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D0%B4%D1%83%D1%80%D0%BE%D1%8E%20%D1%86%D1%8C%D0%BE%D0%B3%D0%BE%20%D0%BC%D0%B5%D1%82%D0%BE%D0%B4%D1%83,%2C%20%D0%B0%20%D0%BF%D0%BE%D

1%82%D1%96%D0%BC%20%E2%80%94%20%D1%83%20%D1%81%D1%82%D0%BE%D0%B2%D0%BF%D1%87%D0%B8%D0%BA%D0%B0%D1%85.

13. Математичні методи оптимізації [Електрон. ресурс]. – Режим доступу: https://studfile.net/preview/3757244/page:8/.

14. Методи побудови опорного плану транспортної задачі [Електрон. ресурс] – 2024 – Режим доступу: https://fingal.com.ua/content/view/460/76/1/2/.

15. Транспортна задача [Електрон. ресурс]. – Режим доступу: https://www.researchgate.net/publication/274476731_Transportna_zadaca.

16. Розв'язок транспортної задачі. Метод потенціалів. [Електрон. ресурс]. – Режим доступу: https://www.youtube.com/watch?v=Hf3RrvLVGRM.

17. What is Floyd-Warshall Algorithm [Electronic resource]. – Access mode: https://www.upperinc.com/glossary/route-optimization/floyd-warshall-algorithm/.

18. Застосування генетичних алгоритмів для вирішення задачі транспортних перевезень [Електрон. ресурс]. – Режим доступу: http://journals.nupp.edu.ua/mist/article/view/472.

19. Транспортна задача [Електрон. ресурс]. – Режим доступу https://financial.lnu.edu.ua/wp-content/uploads/2019/09/ME-lektsiia-6.pdf.

20. Сервіс reshmat [Електрон. ресурс]. – Режим доступу: http://surl.li/gwlcp.

21. Сервіс matworld [Електрон. ресурс]. – Режим доступу: http://surl.li/gwlct.

22. Сервіс math-pr.com [Електрон. ресурс]. – Режим доступу: http://math-pr.com.

23. Сервіс www.easycalculation.com [Electronic resource]. – Access mode: https://www.easycalculation.com.

24. ASP.NET - Introduction [Electronic resource]. – Access mode: https://www.tutorialspoint.com/asp.net/asp.net_introduction.htm.

25. Огляд платформи ASP.Net [Електрон. ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/aspnet/overview.

26. Додавання функціональність Razor Pages до стандартного .NET [Електрон. ресурс]. – Режим доступу: https://habr.com/en/articles/464831/

27. ADO.NET [Electronic resource]. – Access mode: https://learn.microsoft.com/en-us/dotnet/framework/data/adonet/.

28. The Ultimate Comparison of ADO.NET and Entity Framework [Electronic resource]. – Access mode: https://blog.devart.com/ado-net-vs-entity-framework.html.

29. Design patterns [Electronic resource]. – Access mode: https://refactoring.guru/design-patterns.

30. MVC Design Pattern [Electronic resource]. – Access mode: https://www.geeksforgeeks.org/mvc-design-pattern/.

31. Overview of the Active Record Pattern [Electronic resource]. – Access mode: https://blog.savetchuk.com/overview-of-the-active-record-pattern.

32. Singleton Framework [Electronic resource]. – Access mode: https://refactoring.guru/design-patterns/singleton.

33. ER-діаграма – це. Опис, види, правила побудови [Електрон. ресурс]. – Режим доступу: https://hi-news.pp.ua/kompyuteri/14668-er-dagrama-ce-opis-vidi-pravila-pobudovi.html.

34. Що таке діаграма ER і як її створити? [Електрон. ресурс]. – Режим доступу: https://www.lucidchart.com/pages/ru/erd-%D0%B4%D0%B8%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0.

35. Object-Oriented programming (C#) [Electronic resource]. – Access mode: https://learn.microsoft.com/en-us/dotnet/csharp/fundamentals/tutorials/oop.

36. Модульне тестування [Електрон. ресурс]. – Режим доступу: https://qalight.ua/baza-znaniy/modulne-testuvannya/.

37. Що таке Unit тести і як їх писати [Електрон. ресурс]. – Режим доступу: https://foxminded.ua/yunit-testy/.

38. NUnit [Electronic resource]. – Access mode: https://nunit.org/.

39. About xUnit.net [Electronic resource]. – Access mode: https://xunit.net/.

40. Мистецтво юніт-тестування в .NET [Електрон. ресурс]. – Режим доступу: https://dou.ua/forums/topic/40477/.

41. Транспортна задача [Електрон. ресурс]. – Режим доступу: https://ela.kpi.ua/bitstream/123456789/26217/1/Transportna_zadacha.doc.

42. Побудова опорного плану транспортної задачі методом подвійної переваги [Електрон. ресурс]. – Режим доступу: https://www.mathros.net.ua/pobudova-opornogo-planu-transportnoi-zadachi-metodom-podvijnoi-perevagy.html.

43. Побудова опорного плану транспортної задач [Електрон. ресурс]. – 2015. – Режим доступу: https://studfile.net/preview/2398487/page:13/.

44. Розв’язання транспортної задачі методом мінімальної вартості [Електрон. ресурс]. – Режим доступу: https://www.mathros.net.ua/metod-minimalnogo-elementa.html.

45. Метод потенціалів [Електрон. ресурс]. – Режим доступу: https://elib.lntu.edu.ua/sites/default/files/elib_upload/%D0%95%D0%9D%D0%9F_%D0%9C%D0%9C%D0%9E%D0%95%D0%A1_19/page20.html.

46. Генетичні алгоритми. Ключові поняття і методи реалізації [Електрон. ресурс] – Режим доступу: http://www.znannya.org/?view=ga_general.

ДОДАТОК А

Текст програми

```
Program Diplom;
// FloydWarshall
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using TransportProblemLib.Abstruct;

namespace TransportProblemLib.OptimisationAlgorithm
{
   public class FloydWarshall : TransportAlgorithm
   {
      private double[,] _supportPlan;
      public FloydWarshall(double[] reserves, double[] needs, double[,] supportPlan) : base(reserves, needs)
      {
         _supportPlan = (double[,])supportPlan.Clone();
      }

      public override double[,] GetPlan()
      {
         int m = _supportPlan.GetLength(0); // Кількість постачальників
         int n = _supportPlan.GetLength(1); // Кількість клієнтів

         // Матриця шляхів
         int[,] paths = new int[m, n];

         // Алгоритм Флойда-Уоршелла
         for (int k = 0; k < m; k++)
         {
            for (int i = 0; i < m; i++)
            {
               for (int j = 0; j < n; j++)
               {
                  if (_supportPlan[i, j] > _supportPlan[i, k] + _supportPlan[k, j])
                  {
                     _supportPlan[i, j] = _supportPlan[i, k] + _supportPlan[k, j];
                     paths[i, j] = k;
                  }
               }
            }
         }
```

```
                return _supportPlan;
            }
        }
    }
    // OptimizationMethodPotentials
    public class OptimizationMethodPotentials : TransportAlgorithm
    {

        private readonly double[,] _matrix;
        private readonly double[,] _prices;
        private Point[] _allowed;
        public OptimizationMethodPotentials(double[] reserves, double[] needs,
double[,] supportPlan, double[,] prices)
            : base(reserves, needs)
        {
            _matrix = (double[,])supportPlan.Clone();
            _prices = (double[,])prices.Clone();
        }
        public override double[,] GetPlan()
        {
            var helpMatrix = CreateHelpMatrix();

            double[] U = new double[Reserves.Length];
            double[] V = new double[Needs.Length];

            FindUV(U, V, helpMatrix);

            var sMatrix = CreateSMatrix(helpMatrix, U, V);

            while (!IsAllPositive(sMatrix))
            {
                Roll(sMatrix);

                for (int i = 0; i < helpMatrix.GetLength(0); i++)
                {
                    for (int j = 0; j < helpMatrix.GetLength(1); j++)
                    {
                        if (_matrix[i, j] == double.PositiveInfinity)
                        {
                            helpMatrix[i, j] = _prices[i, j];
                            _matrix[i, j] = 0;
                            continue;
                        }
                        helpMatrix[i, j] = double.IsNaN(_matrix[i, j]) ? double.NaN :
_prices[i, j];
```

```
            }
        }

        FindUV(U, V, helpMatrix);
        sMatrix = CreateSMatrix(helpMatrix, U, V);
    }

    return _matrix;
}
private void Roll(double[,] sMatrix)
{
    Point minInd = new Point();
    double min = double.MaxValue;
    int k = 0;
    _allowed = new Point[Reserves.Length + Needs.Length];

    for (int i = 0; i < Reserves.Length; i++)
        for (int j = 0; j < Needs.Length; j++)
        {
            if (_matrix[i, j] == _matrix[i, j])
            {
                _allowed[k].X = i;
                _allowed[k].Y = j;
                k++;
            }
            // find max abs
            if (sMatrix[i, j] < min)
            {
                min = sMatrix[i, j];
                minInd.X = i;
                minInd.Y = j;
            }
        }
    //search cicle
    _allowed[_allowed.Length - 1] = minInd;
    Point[] Cycle = GetCycle(minInd.X, minInd.Y);
    double[] Cycles = new double[Cycle.Length];
    bool[] bCycles = new bool[Cycle.Length];
    for (int i = 0; i < bCycles.Length; i++)
        bCycles[i] = i == bCycles.Length - 1 ? false : true;
    min = double.MaxValue;

    // find min element
    for (int i = 0; i < Cycle.Length; i++)
    {
```

```
                Cycles[i] = _matrix[Cycle[i].X, Cycle[i].Y];
                if ((i % 2 == 0) && (Cycles[i] == Cycles[i]) && (Cycles[i] < min))
                {
                    min = Cycles[i];
                    minInd = Cycle[i];
                }
                if (Cycles[i] != Cycles[i]) Cycles[i] = 0;
            }
            // diff-add
            for (int i = 0; i < Cycle.Length; i++)
            {
                if (i % 2 == 0)
                {
                    Cycles[i] -= min;
                    _matrix[Cycle[i].X, Cycle[i].Y] -= min;
                }
                else
                {
                    Cycles[i] += min;
                    if (double.IsNaN(_matrix[Cycle[i].X, Cycle[i].Y]))
                    {
                        _matrix[Cycle[i].X, Cycle[i].Y] = 0;
                    }
                    _matrix[Cycle[i].X, Cycle[i].Y] += min;
                }
            }
            _matrix[minInd.X, minInd.Y] = double.NaN;
        }

        private Point[] GetCycle(int x, int y)
        {
            Point Beg = new Point(x, y);
            FindWay fw = new FindWay(x, y, true, _allowed, Beg, null);
            fw.BuildTree();
            Point[] Way = Array.FindAll(_allowed, p => (p.X != -1) && (p.Y != -
1));
            return Way;
        }

        private bool IsAllPositive(double[,] sMatrix) =>
Array.TrueForAll(sMatrix.Cast<double>().ToArray(), x => x > 0);

        private double[,] CreateHelpMatrix()
        {
            var matrix = new double[Reserves.Length, Needs.Length];
```

```
            for (int i = 0; i < matrix.GetLength(0); i++)
            {
                for (int j = 0; j < matrix.GetLength(1); j++)
                {
                    matrix[i, j] = (!double.IsNaN(_matrix[i, j])) ? _prices[i, j] :
double.NaN;
                }
            }

            return matrix;
        }

        private bool IsAllTrue(bool[] lst) => Array.TrueForAll(lst, x => x);
        private void FindUV(double[] U, double[] V, double[,] helpMatr)
        {
            //to check if Ui Vi is calculated, we will use an array of booleans
            //even 2 arrays. in one indication of whether the corresponding
potential has been calculated
            //in the second, did we go through the line / line of this potential
            //the algorithm will allow for a finite number of iterations to calculate
all the potentials
            bool[] U1 = new bool[Reserves.Length];
            bool[] U2 = new bool[Reserves.Length];
            bool[] V1 = new bool[Needs.Length];
            bool[] V2 = new bool[Needs.Length];

            var FirstStep = (bool[] isCalc, bool[] isVisited, ref int index) =>
            {
                for (int i = isCalc.Length - 1; i >= 0; i--)
                {
                    if (isCalc[i] && !isVisited[i])
                    {
                        index = i;
                    }
                }
            };
            var SecondStep = (bool[] isCalc, bool[] isVisited, ref int index,
double[] UorV) =>
            {
                for (int i = isCalc.Length - 1; i >= 0; i--)
                {
                    if (!isCalc[i] && !isVisited[i])
                    {
                        index = i;
```

```
            UorV[index] = 0;
            isCalc[index] = true;
            break;
        }
    }
};

// until all elements of arrays V1 and U1 are equal to true
while (!(IsAllTrue(V1) && IsAllTrue(U1)))
{
    int i = -1;
    int j = -1;

    FirstStep(V1, V2, ref i);
    FirstStep(U1, U2, ref j);

    if ((j == -1) && (i == -1))
    {
        SecondStep(V1, V2, ref i, V);
    }
    if ((j == -1) && (i == -1))
    {
        SecondStep(U1, U2, ref j, U);
    }

    if (i != -1)
    {
        for (int j1 = 0; j1 < U1.Length; j1++)
        {
            if (!U1[j1]) U[j1] = helpMatr[j1, i] - V[i];
            if (U[j1] == U[j1]) U1[j1] = true;
        }
        V2[i] = true;
    }

    if (j != -1)
    {
        for (int i1 = 0; i1 < V1.Length; i1++)
        {
            if (!V1[i1]) V[i1] = helpMatr[j, i1] - U[j];
            if (V[i1] == V[i1]) V1[i1] = true;
        }
        U2[j] = true;
    }
```

```
            }
        }

        private double[,] CreateSMatrix(double[,] helpMatrix, double[] U, double[] V)
        {
            double[,] HM = new double[U.Length, V.Length];

            for (int i = 0; i < U.Length; i++)
            {
                for (int j = 0; j < V.Length; j++)
                {
                    HM[i, j] = helpMatrix[i, j];
                    if (HM[i, j] != HM[i, j])
                        HM[i, j] = _prices[i, j] - (U[i] + V[j]);
                }
            }
            return HM;

        }

    }

    using System;

    using System.Collections.Generic;

    using System.Drawing;

    using System.Linq;

    using System.Text;

    using System.Threading.Tasks;


    namespace TransportProblemLib.OptimisationAlgorithm.Service

    {

        internal class FindWay

        {

            private Point _root;

            private Point _begining;

            private FindWay _father;

            private FindWay[] _childrens;

            private Point[] _mAllowed;
```

```
        private bool _flag;
        public FindWay(int x, int y, bool flag, Point[] mAllowed, Point Beg,
FindWay Father)
        {
            _begining = Beg;
            _flag = flag;
            _root = new Point(x, y);
            _mAllowed = mAllowed;
            _father = Father;
        }
        public bool BuildTree()
        {
            Point[] ps = new Point[_mAllowed.Length];
            int Count = 0;
            for (int i = 0; i < _mAllowed.Length; i++)
            {
                if (_flag)
                {
                    if (_root.Y == _mAllowed[i].Y)
                    {
                        Count++;
                        ps[Count - 1] = _mAllowed[i];
                    }
                }
                else
                  if (_root.X == _mAllowed[i].X)
                {
                    Count++;
                    ps[Count - 1] = _mAllowed[i];
                }
```

```
                }
                FindWay fwu = this;
                _childrens = new FindWay[Count];
                int k = 0;
                for (int i = 0; i < Count; i++)
                {
                    if (ps[i] == _root)
                    {
                        continue;
                    }
                    if (ps[i] == _begining)
                    {
                        while (fwu != null)
                        {
                            _mAllowed[k] = fwu._root;
                            fwu = fwu._father;
                            k++;
                        };
                        for (; k < _mAllowed.Length; k++)
                        {
                            _mAllowed[k] = new Point(-1, -1);
                        }
                        return true;
                    }

                    if (!Array.TrueForAll(ps, p => (p.X == 0 && p.Y == 0)))
                    {
                        _childrens[i] = new FindWay(ps[i].X, ps[i].Y, !_flag,
_mAllowed, _begining, this);
                        bool result = _childrens[i].BuildTree();
```

```
                if (result)
                {
                    return true;
                }
            }
        }
        return false;
    }

}
}
namespace TransportProblemLib
{
    public static class Validation
    {
        public static bool IsUpdate(ref double[] needs, ref double[] reserves)
        {
            double reservesSum = reserves.Sum();
            double needsSum = needs.Sum();

            if (reservesSum == needsSum)
            {
                return false;
            }
            if (reservesSum > needsSum)
            {
                UpdateArray(ref needs, needsSum, reservesSum);
            }
            else
            {
```

```
                UpdateArray(ref reserves, needsSum, reservesSum);
            }
            return true;
        }

        private static void UpdateArray(ref double[] lst, double value1, double
value2)
        {
            Array.Resize(ref lst, lst.Length + 1);
            lst[lst.Length - 1] = Math.Abs(value1 - value1);
        }
    }
}
using System.Diagnostics;
using TransportProblemLib.Abstruct;

namespace TransportProblemLib.Extention
{
    public static class TransportAlgorithmExtension
    {
        public static double GetSum(this TransportAlgorithm alg, double[,]
plan, double[,] prices)
        {
            double sum = 0;
            for (int i = 0; i < plan.Length; i++)
            {
                int j = (i - i % alg.Needs.Length) / alg.Needs.Length;
                int k = i % alg.Needs.Length;
                if (plan[j, k] == plan[j, k])
                    sum += plan[j, k] * prices[j, k];
```

```
            }
            return sum;
        }

        public static T[,] ToMatrix<T>(this List<List<T>> lists)
        {
            T[,] matrix = new T[lists.Count, lists.First().Count];
            for (int i = 0; i < matrix.GetLength(0); i++)
            {
                for (int j = 0; j < matrix.GetLength(1); j++)
                {
                    matrix[i, j] = lists[i][j];
                }
            }

            return matrix;
        }
    }
}
namespace TransportProblemLib.Abstruct
{
    public abstract class TransportAlgorithm
    {
        public TransportAlgorithm(double[] reserves, double[] needs)
        {
            Reserves = (double[])reserves.Clone();
            Needs = (double[])needs.Clone();
        }

        public virtual double[] Reserves { get; protected set; }
```

```
        public virtual double[] Needs { get; protected set; }
        public abstract double[,] GetPlan();

        protected void NullToNan(double[,] resultMatrix)
        {
            for (int i = 0; i < resultMatrix.GetLength(0); i++)
            {
                for (int j = 0; j < resultMatrix.GetLength(1); j++)
                {
                    if (resultMatrix[i, j] == 0)
                    {
                        resultMatrix[i, j] = double.NaN;
                    }
                }
            }
        }
    }
}

using Microsoft.AspNetCore.Mvc;
using System.Drawing.Drawing2D;
using TransportProblemLib.Enums;
using TransportProblemWebApp.Model;
using TransportProblemWebApp.Service.MinElementAlgorithm;

namespace TransportProblemWebApp.Controllers
{
    public class AlgorithmResultController : Controller
    {
        private readonly ITransportAlgorithm _transportAlgorithm;
```

```
        public AlgorithmResultController(ITransportAlgorithm transport)
        {
            _transportAlgorithm = transport;
        }

        public IActionResult Index(MatrixViewModel model)
        {
            var result = _transportAlgorithm.GetResultModel(model);

            return View(result);
        }
    }
}
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore;
using TransportProblemWebApp.Domain.Entities;
using TransportProblemWebApp.Service;

namespace TransportProblemWebApp.Domain
{
    public class AppDbContext :IdentityDbContext
    {
        public AppDbContext(DbContextOptions<AppDbContext> options)
            :base(options) { }
        public DbSet<TextField> TextFields { get; set; }
        public DbSet<InformationField> InformationFields { get; set; }

        protected override void OnModelCreating(ModelBuilder builder)
```

```
        {
            base.OnModelCreating(builder);
            builder.Entity<IdentityRole>().HasData(new IdentityRole
            {
                Id = "e217e9d7-ec23-4868-acc9-81737b460ee7",
                Name = AdminAreaConfig.Name,
                NormalizedName = AdminAreaConfig.Name.ToUpper(),
            });

            builder.Entity<IdentityUser>().HasData(new IdentityUser
            {
                Id = "6edae83a-f414-41f5-8a48-9c1f77c776b7",
                UserName = AdminAreaConfig.Name,
                NormalizedUserName = AdminAreaConfig.Name.ToUpper(),
                Email = AdminAreaConfig.Email,
                NormalizedEmail = AdminAreaConfig.Email.ToUpper(),
                EmailConfirmed = true,
                PasswordHash = new
PasswordHasher<IdentityUser>().HashPassword(null,
AdminAreaConfig.Password),
                SecurityStamp = string.Empty

            });

            builder.Entity<IdentityUserRole<string>>().HasData(new
IdentityUserRole<string>
            {
                RoleId = "e217e9d7-ec23-4868-acc9-81737b460ee7",
                UserId = "6edae83a-f414-41f5-8a48-9c1f77c776b7"
            });
```

```
            builder.Entity<TextField>().HasData(new TextField
            {
                Id = new Guid("c55c5de5-966d-40a9-8476-cfd6851d0d57"),
                CodeWord = "PageIndex",
                Title = "MAIN",
            });

            builder.Entity<TextField>().HasData(new TextField
            {
                Id = new Guid("60467613-3d6a-4521-b8de-2735b98b23c6"),
                CodeWord = "PageInformations",
                Title = "Information",
            });

            builder.Entity<TextField>().HasData(new TextField
            {
                Id = new Guid("9d02c058-a1e3-4111-af64-d1aba6b2e268"),
                CodeWord = "PageAbout",
                Title = "About",
            });
        }
    }
}
```

ДОДАТОК Б

Слайди презентації

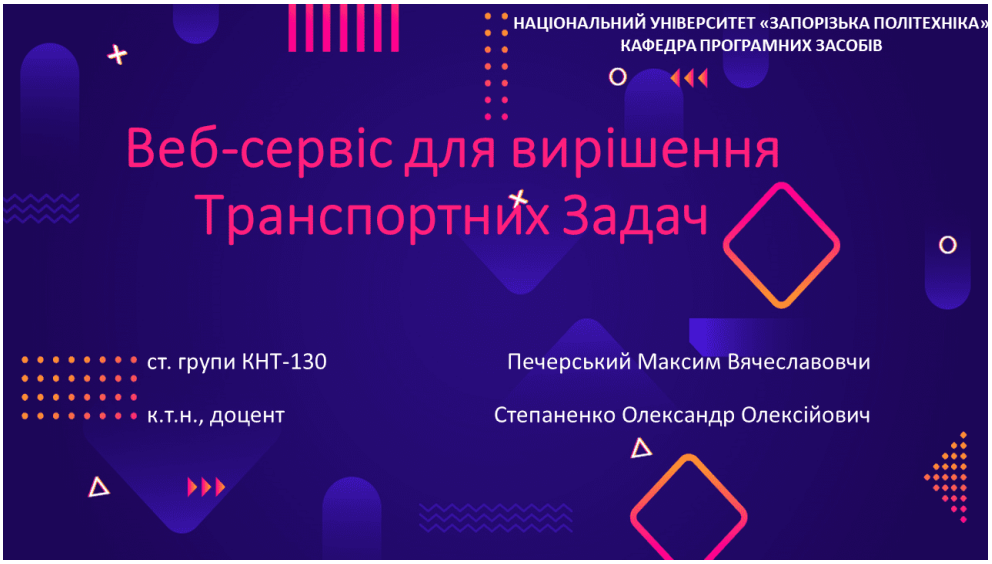


Рисунок Б.1 – Слайд 1

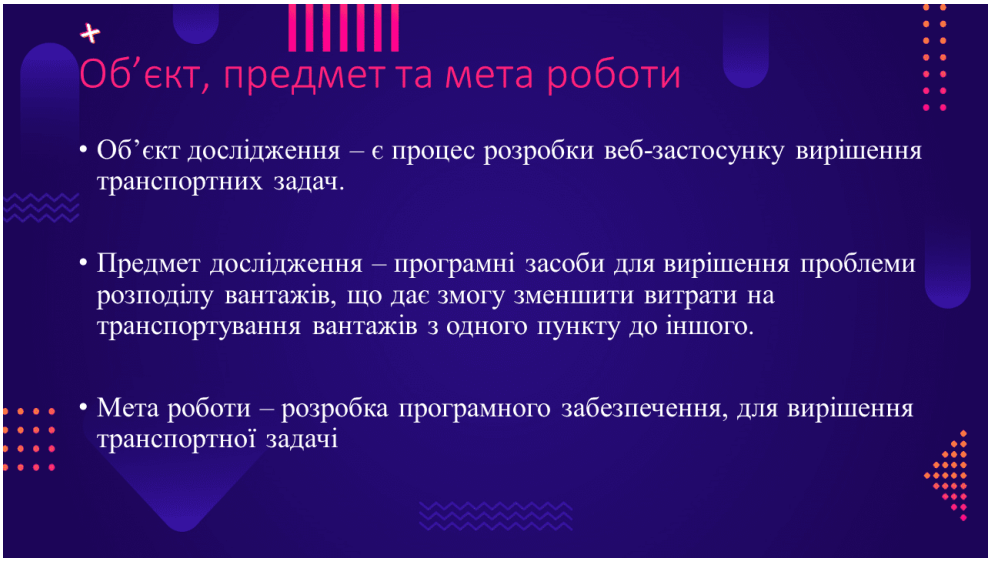


Рисунок Б.2 – Слайд 2

	reshmat	matworld	math-pr.com	easycalculation .com	Власна розробка
1	2	3	4	5	6
Зміна розмірів матриці	Динамічна	Перед початком роботи	Перед початком роботи	Перед початком роботи	Динамічна
Кількість методів знаходження базового розв'язку	2	3	2	1	5
Кількість методів знаходження базового розв'язку	0	1	0	0	2
Адаптивність	Ні	Напів-адаптивна	ні	так	так
Редагування контенту сайту	Адмін.	Адмін.	Адмін.	Адмін.	Адмін.

Рисунок Б.3 – Слайд 3

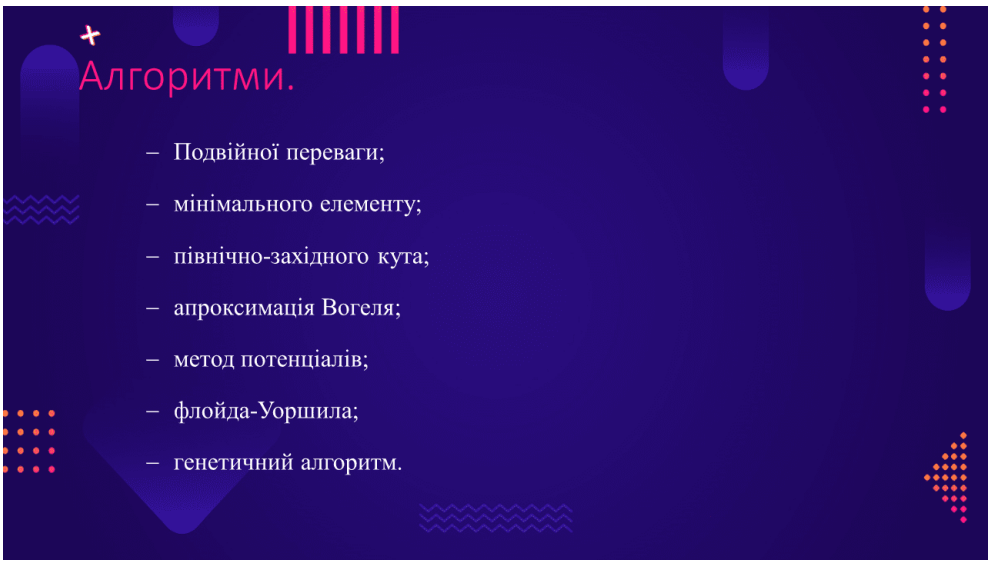


Рисунок Б.4 – Слайд 4

Рисунок Б.5 – Слайд 5

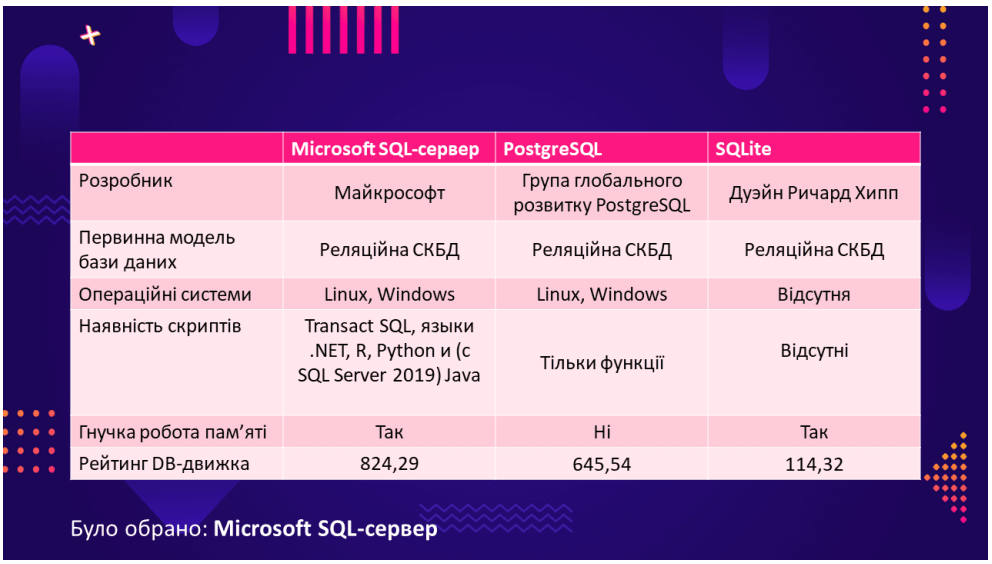


Рисунок Б.6 – Слайд 6

Рисунок Б.7 – Слайд 7

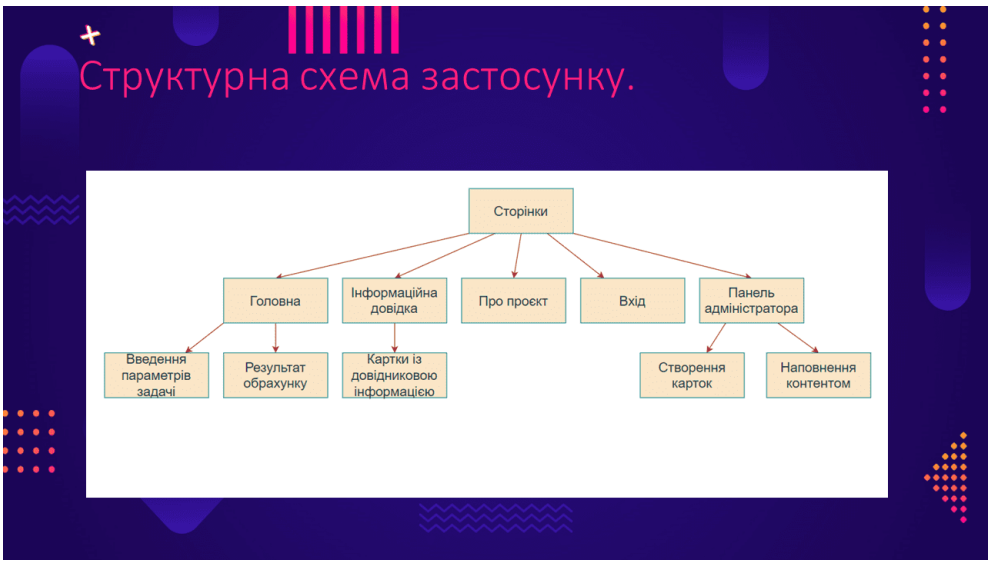


Рисунок Б.8 – Слайд 8

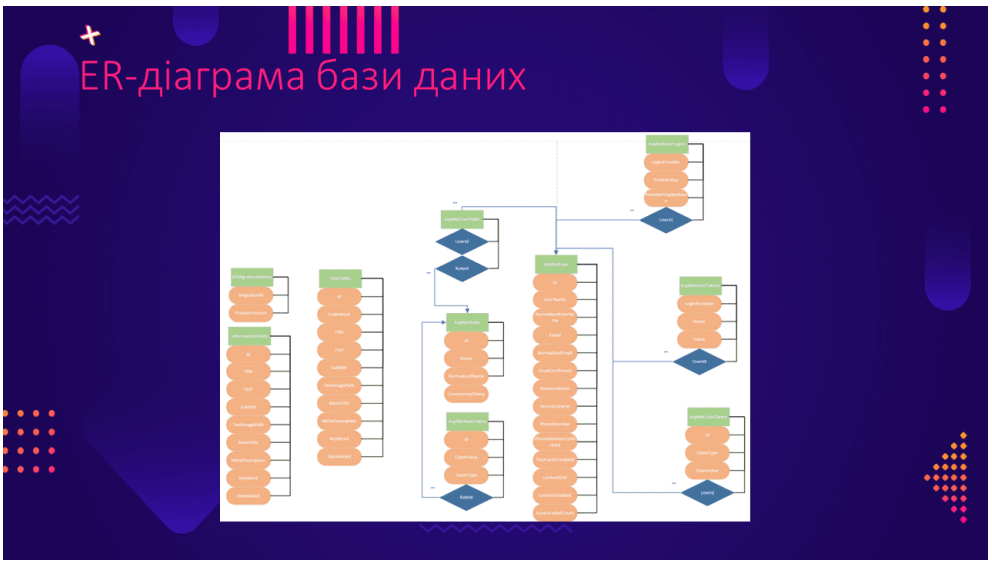


Рисунок Б.9 – Слайд 9

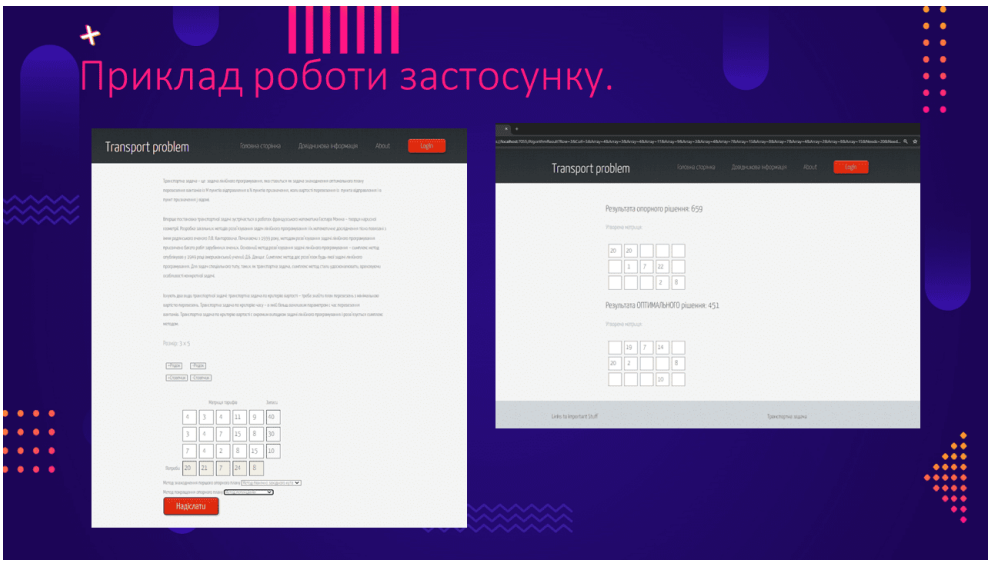


Рисунок Б.10 – Слайд 10

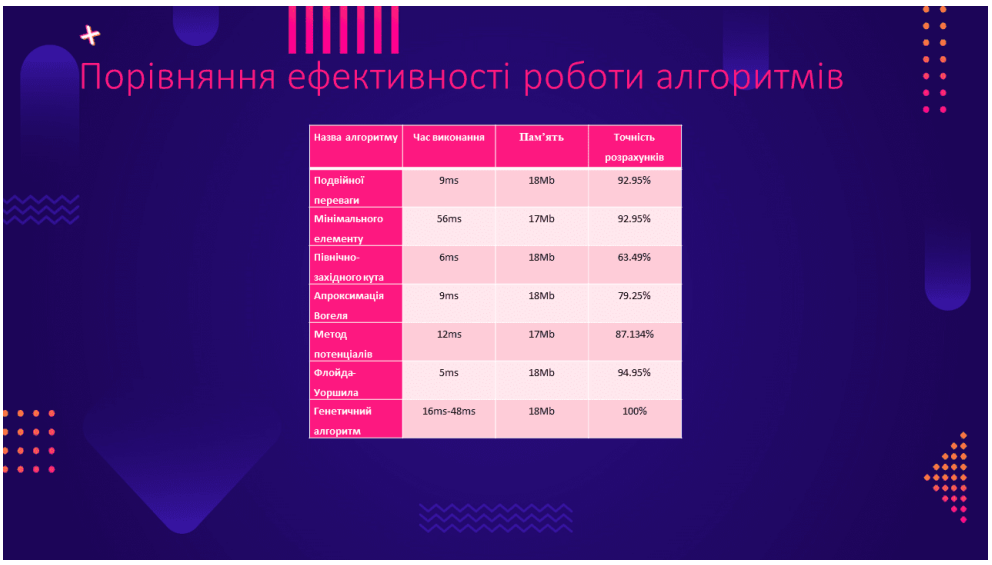


Рисунок Б.11 – Слайд 11

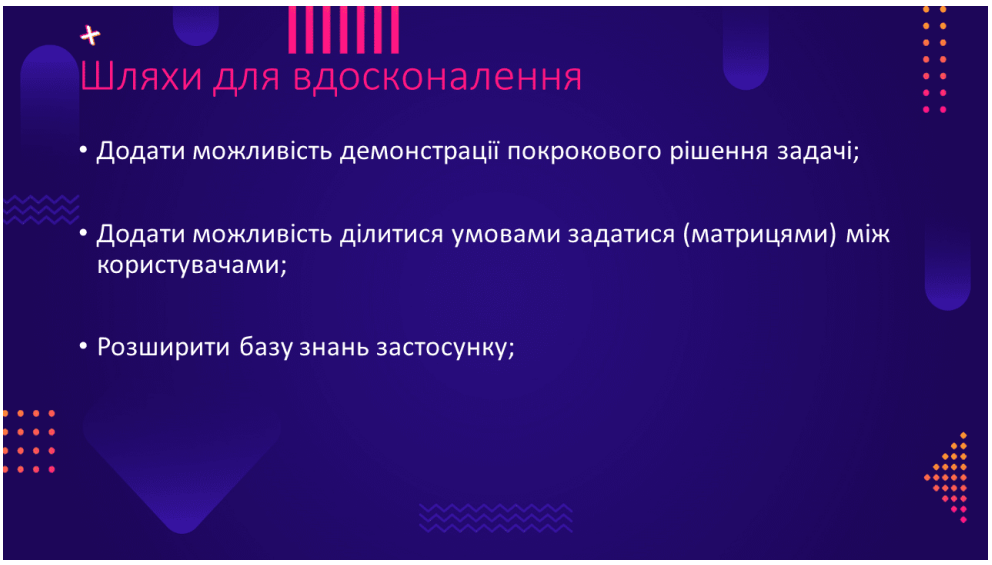


Рисунок Б.12 – Слайд 12

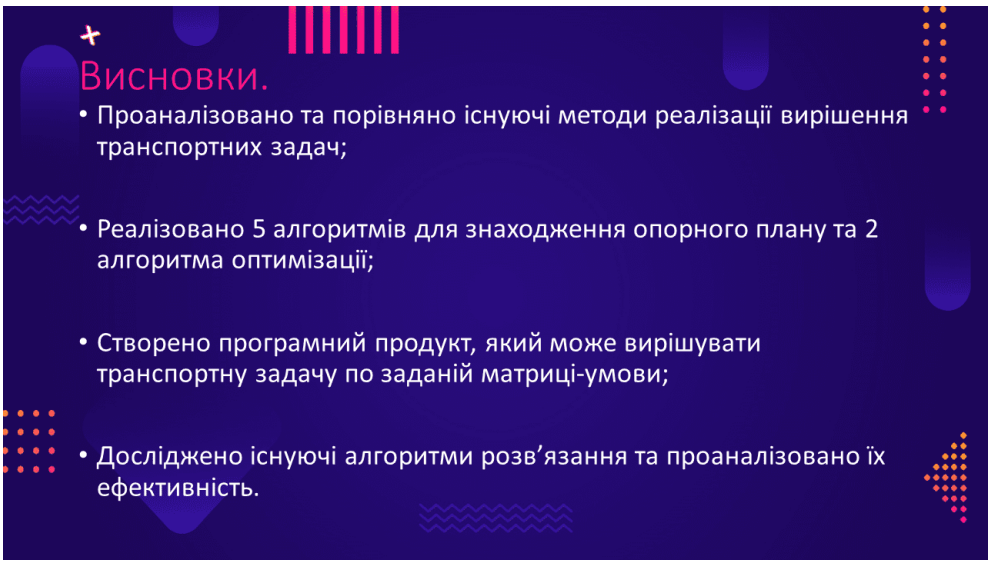


Рисунок Б.13 – Слайд 13

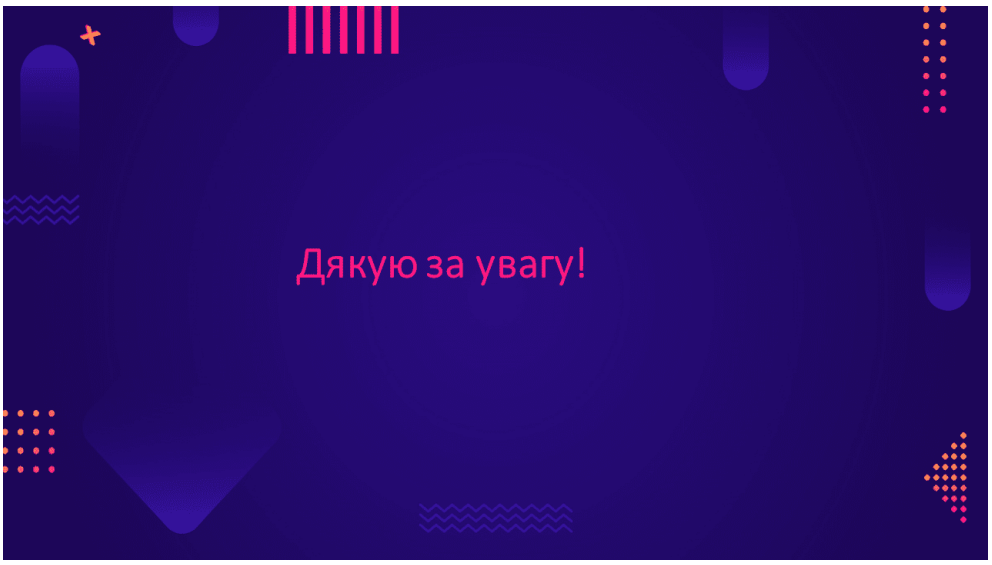


Рисунок Б.14 – Слайд 14