НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Кафедра автоматизованих систем обробки інформації та управління

Спеціальність 122 «Комп’ютерні науки та інформаційні технології»

**ЗВІТ**

**виконання самостійної роботи студентів**

з дисципліни: «Технології розробки програмного забезпечення»

на тему: Задача про покриття

Кількість балів: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата захисту: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Члени проекту Кас’янчук А.С.

(підпис) (прізвище та ініціали)

Потильчак Д.Г.

(підпис) (прізвище та ініціали)

Перевірила \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Сперкач М.О.

(підпис) (прізвище та ініціали)

Київ - 2018

**ЗМІСТ**

[1 ВСТУП 2](#_Toc514705334)

[2 ЗАГАЛЬНІ ПОЛОЖЕННЯ 3](#_Toc514705335)

[2.1 Опис предметного середовища 3](#_Toc514705336)

[2.2 Опис функціональної моделі 3](#_Toc514705337)

[3 РІШЕННЯ З ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ 6](#_Toc514705338)

[3.1 Вхідні дані 6](#_Toc514705339)

[3.2 Вихідні дані 6](#_Toc514705340)

[3.3 Опис інформаційного забезпечення 7](#_Toc514705341)

[4 РІШЕННЯ З ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 8](#_Toc514705342)

[4.1 Опис вибраних технологій розробки 8](#_Toc514705343)

[4.2 Архітектура програмного забезпечення 8](#_Toc514705344)

[4.3 Специфікація функцій 17](#_Toc514705345)

[4.4 Вимоги до якості 17](#_Toc514705346)

[5 КЕРІВНИЦТВО КОРИСТУВАЧА 19](#_Toc514705347)

[5.1 Інструкція користувача 19](#_Toc514705348)

[5.2 Методика випробувань 24](#_Toc514705349)

[6 ВИСНОВОК 25](#_Toc514705350)

[Додаток А 26](#_Toc514705351)

[Додаток Б 27](#_Toc514705352)

# 

# **ВСТУП**

Дана задача може використовуватись в різних сферах, наприклад, в комп’ютерних мережах при маршрутизації, коли потрібно відправити дані усім підключеним до мережі пристроям з мінімальними затратами. У нашому випадку, ми маємо початковий комп’ютер та термінальний комп’ютер. При надсиланні даних з початкового комп’ютера в термінальний, дані повинні отримати всі комп’ютери мережі з мінімальними затратами.

Існує декілька алгоритмів розв'язання цієї задачі. Деякі алгоритми повертають нам оптимальні результати, але мають низьку продуктивність, а інші не завжди повертають оптимальні розв’язки, але є дуже продуктивними. Загалом, алгоритм перебору та алгоритм Анрі-Лабодера є точними і для розв'язанні задач з великою кількістю вузлів, потребують багато часу для розв'язання. Жадібний алгоритм розв’язує задачі з великою кількість вузлів за короткий час, але не гарантує точність. Як ми бачимо, у наведених вище алгоритмах є як і мінуси, так і плюси, тому використовувати їх потрібно, пам’ятаючи про це все.

Вхідні дані в програмі генеруються за допомогою генератора. Генератор генерує матрицю заданої розмірності без ізольованих вершин та перевіряє граф на зв'язність.

Пояснювальна записка містить приклад приклади, які виникають при некоректності вхідних даних.

# **ЗАГАЛЬНІ ПОЛОЖЕННЯ**

## **Опис предметного середовища**

Дана задача може використовуватись в різних сферах, наприклад, в комп’ютерних мережах при маршрутизації, коли потрібно відправити дані усім підключеним до мережі пристроям з мінімальними затратами. У нашому випадку, ми маємо початковий комп’ютер та термінальний комп’ютер. При надсиланні даних з початкового комп’ютера в термінальний, дані повинні отримати всі комп’ютери мережі з мінімальними затратами.

Ця задача є актуальною в сфері комп’ютерних мереж, так як оптимізація передача даних по локальній мережі дуже важлива в сучасному світі.

## **Опис функціональної моделі**

Наведемо опис функцій системи у вигляді діаграми використання (Use Case) (рисунок 2.1) та наведемо таблицю з детальним описом функцій системи (таблиця 2.1).

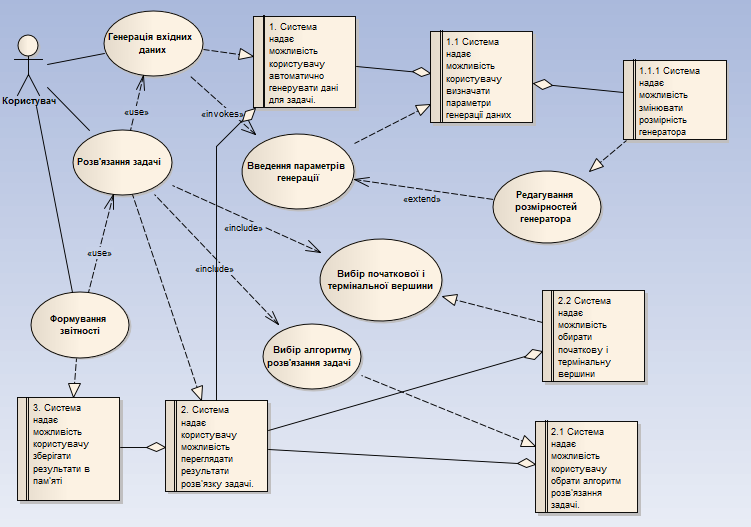


Рисунок 2.1 – Діаграма варіантів використання (UseCase) з описом функціональних вимог

Таблиця 2.1

|  |  |
| --- | --- |
| Назва функції | **Опис** |
| Генерація вхідних даних | Користувач задає розмірність задачі і система автоматично генерує дані задачі. |
| Введення параметрів генерації | Користувач задає параметри для генерації задач |
| Редагування розмірності генератора | Користувач задає розмір задачі, яка буде генерована. |
| Вибір початкової і термінальної вершини | Користувач вибирає початкову і термінальну (кінцеву) вершину для побудови множини шляхів. |
| Розв’язання задачі | Користувач отримує результати розв’язання задачі. |
| Вибір алгоритму розв’язку задачі | Користувач після введення даних обирає один запропоновані алгоритми для рішення задачі |
| Формування звітності | Користувачу надається звіт з детальними текстовими результатами роботи алгоритмів та часу виконання задачі. |

Наведемо формулювання усіх функціональних вимог у відповідності з діаграмою варіантів використання з описом функціональних вимог (рисунок 2.1) в таблиці 2.2.

Таблиця 2.2

|  |  |  |  |
| --- | --- | --- | --- |
|  | Варіант використання | Функціональна вимога | Пріоритет |
| К  л  і  є  н  т | Генерація вхідних даних | 1. Система надає можливість користувачу автоматично генерувати дані для задачі. | Високий |
| Введення параметрів генерації | 1.1 Система надає можливість користувачу визначати параметри генерації даних | Високий |
| Редагування розмірностей генератора | 1.1.1 Система надає можливість змінювати розмірність генератора | Високий |
| Розв'язання задачі | 2. Система надає користувачу можливість переглядати результати розв'язку задачі. | Високий |
| Вибір алгоритму розв'язання задачі | 2.1 Система надає можливість користувачу обрати алгоритм розв'язання задачі. | Високий |
| Вибір початкової і термінальної вершини | 2.2 Система надає можливість обирати початкову і термінальну вершини | Високий |
| Формування звітності | 3. Система надає можливість користувачу імпортувати результати до текстового файлу. | Високий |

# **РІШЕННЯ З ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ**

## **Вхідні дані**

Опис вхідних даних:

* кількість задач - кількість задач для генерування;
* n - кількість вершин графа;
* w\_min - мінімальна вага ребра (для рандомного генерування);
* w\_max - максимальна вага ребра (для рандомного генерування);
* номер задачі - немар задачі, яка буде розв’язана вибраними алгоритмами;
* початкова вершина;
* термінальна вершина;
* вибір алгоритму - вибір алгоритмів з переліку;

Вхідні дані програми повинні задовольняти *наступним умовам*:

* усі значення повинні бути цілочисельними, не від’ємними;
* w\_min має бути менше w\_max;
* кількість задач, які одночасно генеруються, не повинна перебільшувати 100;
* n не повинна бути в діапазоні від 3 до 100 включно;
* номер задачі повинен бути в межах існуючих задач;
* початкова і термінальна вершина мають існувати в вибраній задачі;
* вибір алгоритму має бути хоча б один;

## **Вихідні дані**

Опис вихідних даних програми:

* множина маршрутів, яка відображається у вигляді “a1->a2->a3 b1->b2->b3”
* Значення цільової функції;
* Час роботи алгоритму у мілісекундах;

Вихідні дані зберігаються в таблиці для легшого порівняння. Формат таблиці наступний:

* номер задачі- номер задачі, яку було розв’язано;
* матриця - граф у вигляді матриці ваг;
* шляхи - множина шляхів;
* результат - значення цільової функції;
* час виконання - час виконання алгоритмів в мілісекундах;

## **Опис інформаційного забезпечення**

Оскільки програний продукт розроблено без використання бази даних, усі вихідні результати генерування та розв’язання записуються у оперативну пам'ять.

# **РІШЕННЯ З ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ**

## **Опис вибраних технологій розробки**

Для розробки цього web-застосунку, було використано технологію ASP.NET Core 2 MVC. Ця технологія дозволяє розробляти мультиплатформенні web - застосунки на базі паттерну MVC (model - view - conroller). Застосунки розроблені по даному паттерну легко масштабуються та тестуються.

Веб-додатки, які потребують комбінування декількох технологій (наприклад, баз даних, HTML-розмітки і виконуваного коду), зазвичай поділяються на ряд шарів або рівнів. Отримані в результаті шаблони природним чином вписуються в концепції MVC.

Інфраструктура ASP.NET Core 2 MVC генерує ясний і відповідний стандартам код розмітки з використанням Razor. За допомогою Razor було створено динамічні сторінки та реалізовано відображеня даних на web-сторінці.

Front-end був реалізований з використанням пакету Bootstrap. Bootstrap — це безкоштовний набір інструментів з відкритим [кодом](https://uk.wikipedia.org/wiki/%D0%9F%D0%BE%D1%87%D0%B0%D1%82%D0%BA%D0%BE%D0%B2%D0%B8%D0%B9_%D0%BA%D0%BE%D0%B4), призначений для створення [веб-сайтів](https://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B0%D0%B9%D1%82)та [веб-додатків](https://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%B7%D0%B0%D1%81%D1%82%D0%BE%D1%81%D1%83%D0%BD%D0%BE%D0%BA), який містить [шаблони](https://uk.wikipedia.org/wiki/%D0%A8%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD_%D1%81%D0%B0%D0%B9%D1%82%D1%83) [CSS](https://uk.wikipedia.org/wiki/CSS) та [HTML](https://uk.wikipedia.org/wiki/HTML) для типографіки, форм, кнопок, навігації та інших компонентів інтерфейсу, а також додаткові розширення [JavaScript](https://uk.wikipedia.org/wiki/JavaScript). Він спрощує розробку динамічних веб-сайтів і веб-додатків.

## **Архітектура програмного забезпечення**

* + 1. **Схема архітектури ПЗ**

***Модель–вигляд–контролер*** (*або Модель–представлення–контролер, англ. Model-view-controller, MVC*) — архітектурний шаблон, який використовується під час проектування та розробки програмного забезпечення (рисунок 4.1).

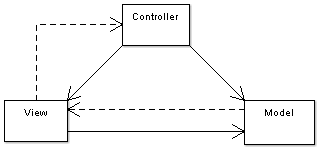


Рисунок 4.1 – Схема паттерну MVC

Цей шаблон передбачає поділ системи на три взаємопов'язані частини: [модель даних](https://uk.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85), вигляд ([інтерфейс користувача](https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BA%D0%BE%D1%80%D0%B8%D1%81%D1%82%D1%83%D0%B2%D0%B0%D1%87%D0%B0)) та модуль керування. Застосовується для відокремлення даних (моделі) від інтерфейсу користувача (вигляду) так, щоб зміни інтерфейсу користувача мінімально впливали на роботу з даними, а зміни в моделі даних могли здійснюватися без змін інтерфейсу користувача.

Мета шаблону — гнучкий дизайн програмного забезпечення, який повинен полегшувати подальші зміни чи розширення програм, а також надавати можливість повторного використання окремих компонентів програми. Крім того використання цього шаблону у великих системах сприяє впорядкованості їхньої структури і робить їх більш зрозумілими за рахунок зменшення складності

Відповідно до паттерну MVC, в проекті CoverTask, було створено папки Models, Views і Conrollers (рисунок 4.2). Підключено потрібні бібліотеки (рисунок 4.3). Папка Models містить (рисунок 4.5):

* файл Algorithms.cs - файл, де описані і реалізовані усі алгоритми;
* файл Generator.cs - файл, де описаний і реалізований генератор;
* файл Repository.cs - файл, де описані колекції даних для зберігання згенерованих задач та відповідей на задачі.

Папка View містить (рисунок 4.6):

* файл Index.cshtml - представлення початкової сторінки;
* файл AlgorithmView.cshtml - представлення сторінки розв’язання задачі;
* файл AnswerView.cshtml - представлення сторінки, на якій відображається результати розв’язку;
* файл AnswerListView.cshtml - представлення сторінки, на якій відображається таблиця усіх рішень;
* файл GeneratorView.cshtml - представлення сторінки генератора;
* файл GeneratorListView.cshtml - представлення сторінки, на якій відображається таблиця усіх згенерованих задач;
* файл InfoView.cshtml - представлення сторінки, на якій відображена інформація про авторів і коротка інструкція до користування програмою.

Папка Controllers містить (рисунок 4.4):

* файл HomeController - контролер для взаємодії моделей з представленнями.

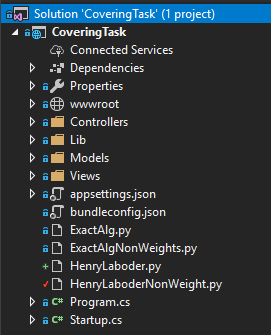


Рисунок 4.2 – Solution

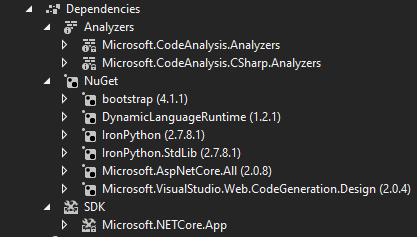


Рисунок 4.3 – Dependencies



Рисунок 4.4 – Вміст папки *Controllers*

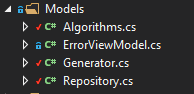
**

Рисунок 4.5 – Вміст папки *Models*

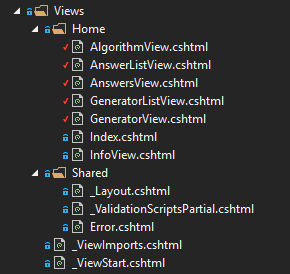
**

Рисунок 4.6 – Вміст папки *Views*

* + 1. **Діаграма класів**

Наведено загальну діаграму класів з урахуванням шаблону архітектури ПЗ та зв’язки між основними класами(рисунок 4.7).

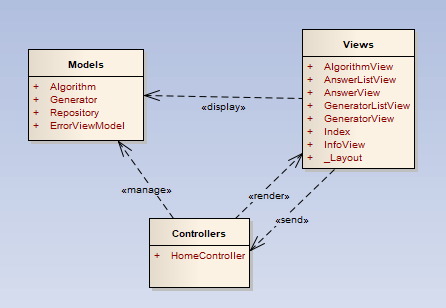


Рисунок 4.7 – Загальна діаграма класів

Діаграма класів для пакету Models з відповідними методами (рисунок 4.8).

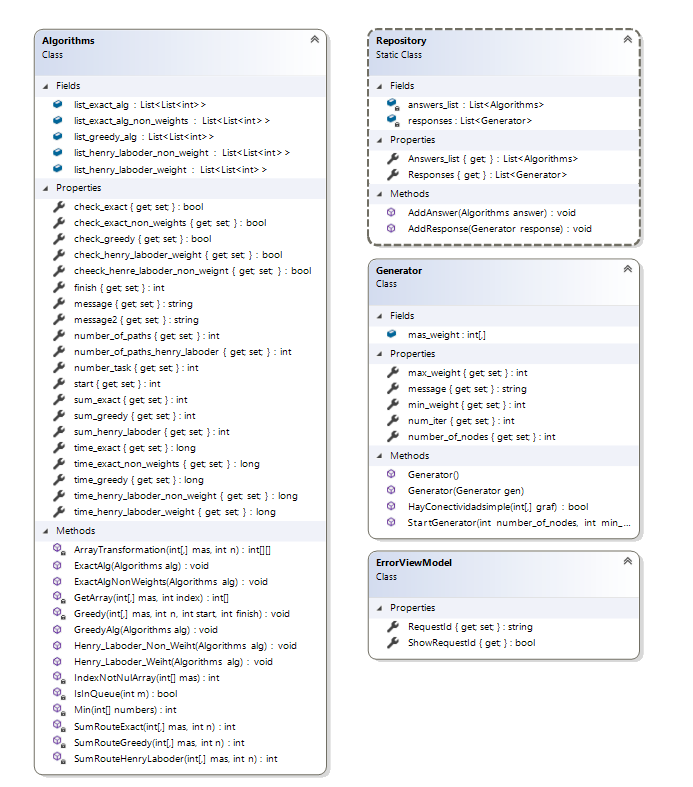


Рисунок 4.8 – Діаграма класів для пакету Models

Діаграма класів для пакету Controllers з відповідними методами (рисунок 4.9).

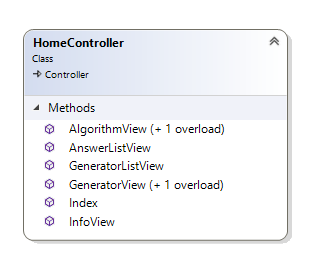


Рисунок 4.9 – Діаграма класів для пакету Controllers

Діаграма класів для пакету Views (рисунок 4.10).

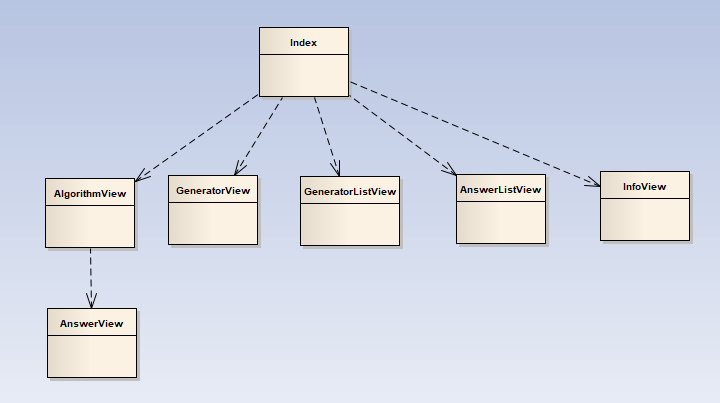


Рисунок 4.10 – Діаграма класів для пакету Views

**3.2.3 Діаграма послідовності**

Діаграма послідовностей для процесів генерації (рисунок 4.11) та вирішення задачі (рисунок 4.12).

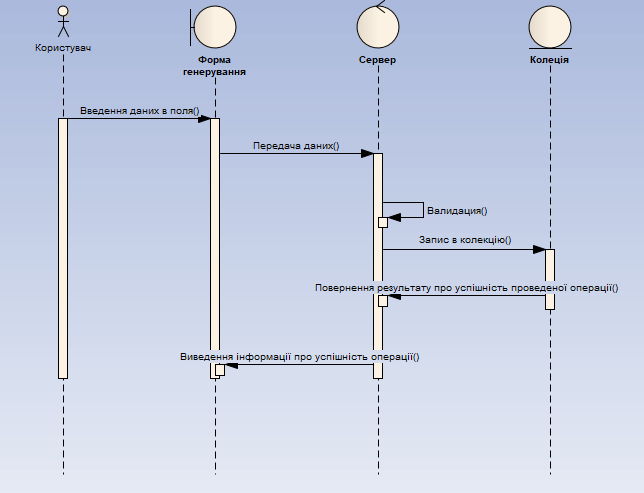


Рисунок 4.11 – Діаграма послідовностей для процесу генерування задачі

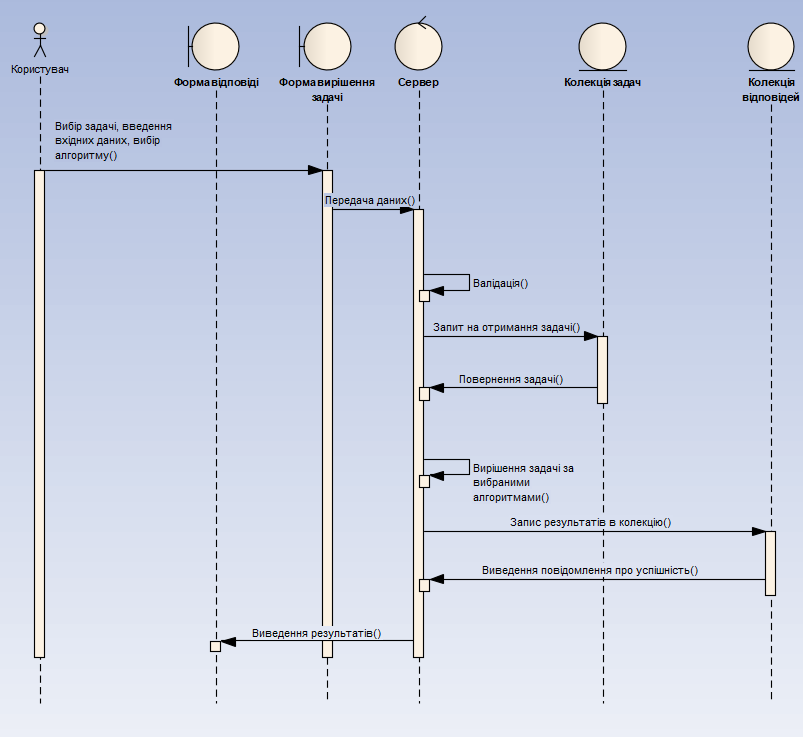
**

Рисунок 4.12 – Діаграма послідовностей для процесу розв’язання задачі

* + 1. **Діаграма розгортання**

Діаграма розгортання (рисунок 2.13) проекту.

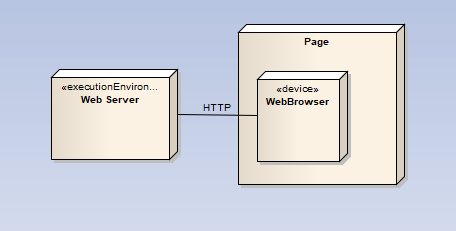


Рисунок 4.13 – Діаграма розгортання

## **Специфікація функцій**

Специфікацію функцій наведено у таблиці 3.1

Таблиця 4.1

|  |  |
| --- | --- |
| **Назва функції** | **Опис** |
| Введення вхідних даних генерування | Користувач вказує вхідні дані для генерування та розв’язування задачі:   * кількість задач; * кількість вершин графа; * мінімальне значення ваг; * максимальне значення ваг; |
| Генерація вхідних даних | Користувач натискає на кнопку і система автоматично генерує дані задачі. |
| Вибір алгоритму розв’язку задачі | Користувач після введення даних обирає один (або декілька) із алгоритмів для розв’язку задачі: |
| Редагування розмірності задачі | Користувач редагує розмірність задачі. |
| Формування кінцевих результатів | Користувач отримує результати розв’язання задачі. |
| Збереження кінцевих результатів | Відповіді автоматично зберігаються. |
| Формування порівнянь результатів | Користувачу надається звіт з детальними табличними результатами роботи алгоритмів та часу виконання задачі. |

## **Вимоги до якості**

* Веб-сайт повинен адаптуватися під будь-яке розширення монітора комп’ютера.
* Кожна веб-сторінка сайту повинна містити кнопку повернення до головної сторінки, кнопку списку розв’язаних задач, кнопку списку задач, кнопку переходу до сторінки з інформацією про програму.
* Сайт повинен бути оформлений українською мовою.
* Графічне оформлення сайту повинне використовувати кольорову палітру, що складається лише з безпечних кольорів.
* Веб-сторінки сайту повинні використовувати системні шрифти.
* Необхідним обладнанням для роботи системи є комп’ютер з встановленим веб-браузером та сервер на якому зберігаються дані для роботи системи. Система повинна використовувати HTTP- синхронний протокол для обміну даними з сервером.
* Для роботи системи на комп’ютерах повинен бути встановленимй браузер з підтримкою HTML,CSS, JavaScript (Google Chrome, Mozilla Firefox, Opera та інші).

1. **КЕРІВНИЦТВО КОРИСТУВАЧА**

## **Інструкція користувача**

**Головна сторінка**

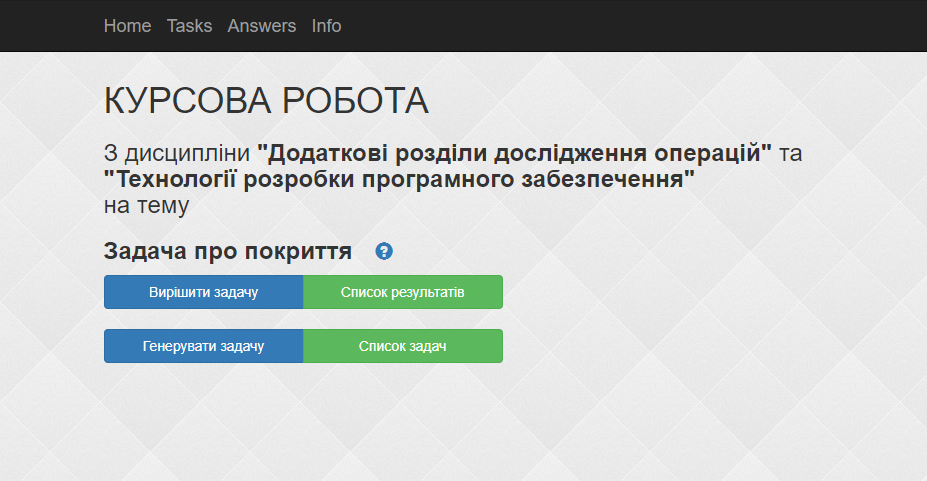
На головні сторінці відображена інформація про курсову роботу та кнопки: “Вирішити задачу”, “Список результатів”, “Генерувати задачу”, “Список задача” (рисунок 5.1).

Рисунок 5.1 – Головна сторінка програми”

**Як генерувати задачу?**

Для генерації задачі потрібно натиснути на синю кнопку “Генерувати задачу”. Після цього відкриється сторінка (рисунок 5.2) з полями для введення чисел. В поле “Кількість вузлів” потрібно ввести число в діапазоні від 3 до 100. Це число визначає розмір матриці ваг. В поле “Мінімальна вага ребра” і “Максимальна вага ребра” потрібно ввести числа від 1 до 1000 та від 2 до 1000 відповідно. Ці числа визначають діапазон не нульових ваг в матриці ваг. В полі “Скільки ви хочете генерувати задач” по замовчуванню число дорівнює одиниці, але можна ввести число від 1 до 100. Це число визначає кількість генерованих задач. Після заповнення кожного поля, натисніть на кнопку “Генерувати”. Задачі будуть генеровані.

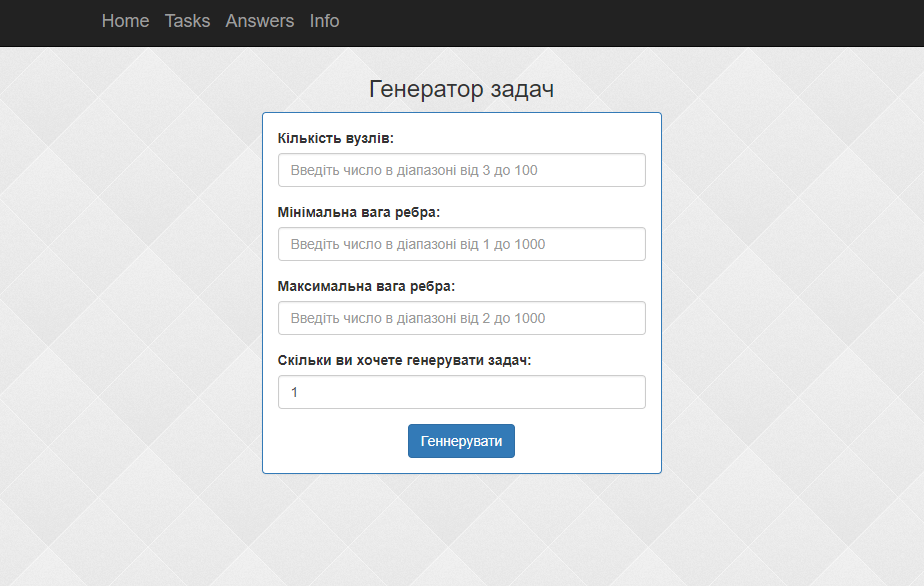


Рисунок 5.2 – Сторінка генератора

**Де переглянути генеровані задачі?**

Для перегляду генерованих задач, натисніть на кнопку “Список задач”. Якщо раніше не було згенеровано задач, то відкриється наступна сторінка (рисунок 5.3). Якщо задачі раніше генерувались, то відкриється наступна форма (рисунок 5.4). При натисканні на кнопку “Переглянути матрицю”, відкриється обрана матриця.

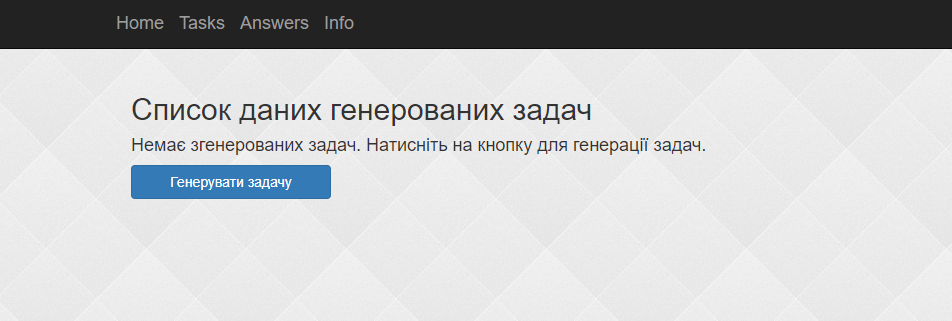


Рисунок 5.3 – Сторінка списку генерованих задач

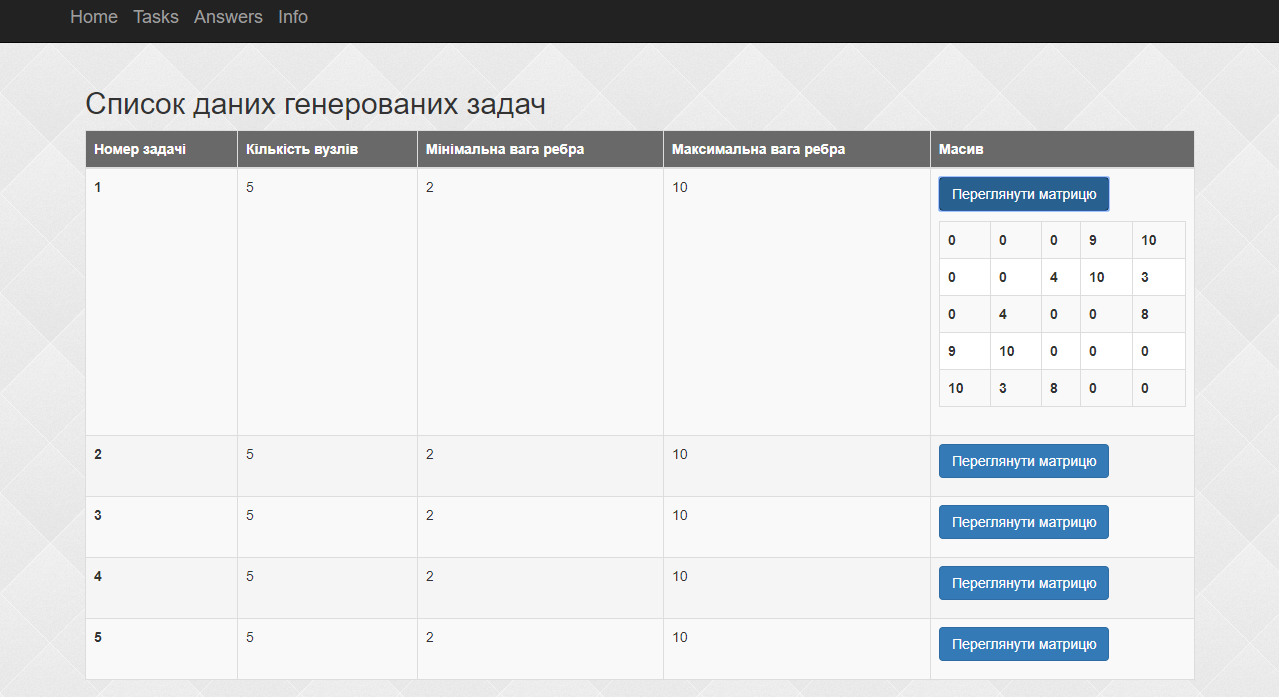


Рисунок 5.4 – Сторінка списку генерованих задач

**Як розв’язати задачу?**

Для розв’язання задачі, натисніть на кнопку “Вирішити задачу”. Після цього відкриється сторінка розв’язання задачі (рисунок 5.5). В поле “Номер задачі” введіть число з підсказки. В поле “Початкова вершина” і “Термінальна вершина” введіть номер вершини графа вибраної задачі. Виберіть один або декілька алгоритмів із запропонованих та натисніть на кнопку “Вирішити задачу”. Після цього відкриється сторінка з результатами (рисунок 5.6).

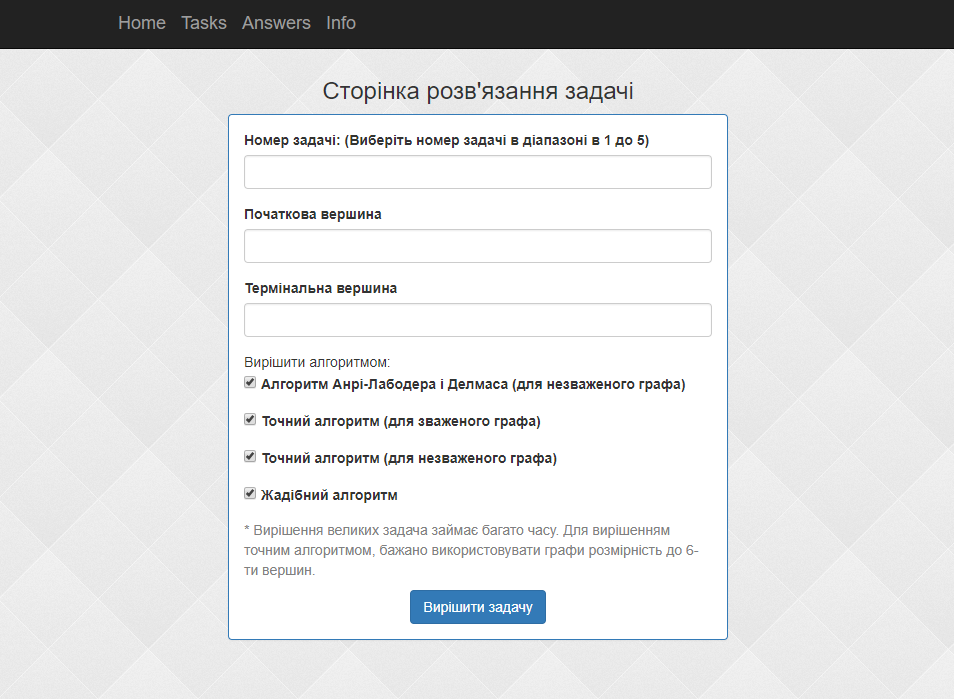


Рисунок 5.5 – Сторінка розв’язання задачі

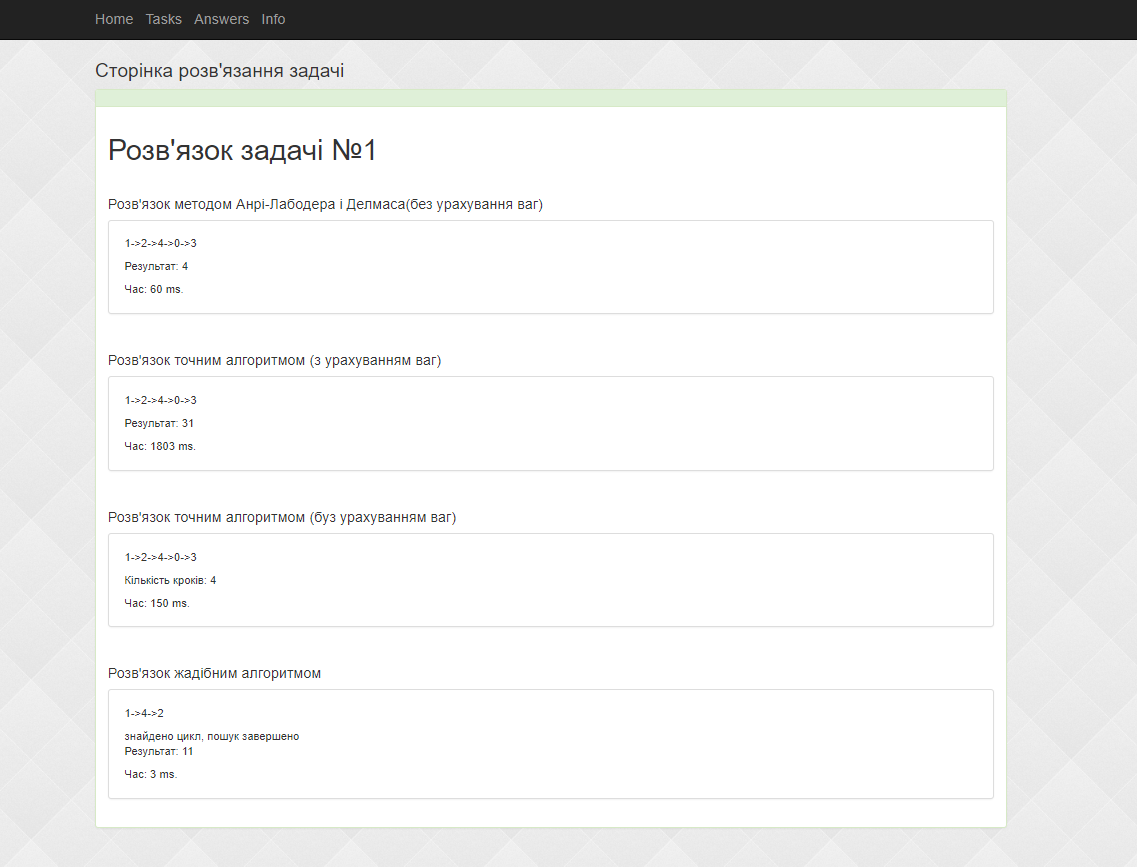


Рисунок 5.6 – Сторінка з розв’язком

**Як переглянути результати, розв’язаних раніше, задач?**

Для перегляду результатів розв’язання задач, натисніть на кнопку “Список результатів”. Якщо раніше не було розв’язано задач, то відкриється наступна сторінка (рисунок 5.7). Якщо задачі раніше розв’язувались, то відкриється наступна форма (рисунок 5.8). При натисканні на кнопку “Переглянути матрицю”, відкриється обрана матриця.

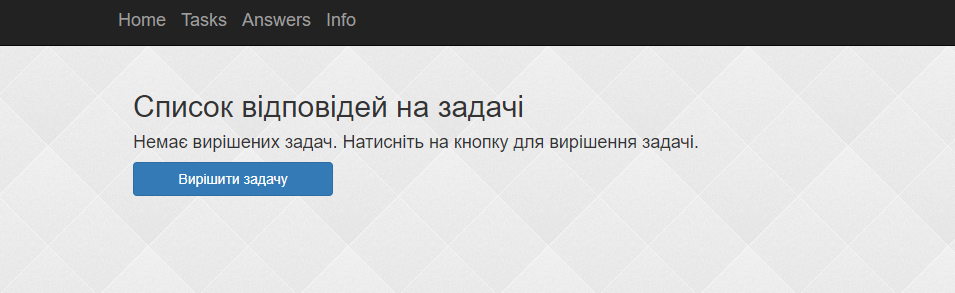


Рисунок 5.7 – Сторінка списку розв’язаних задач

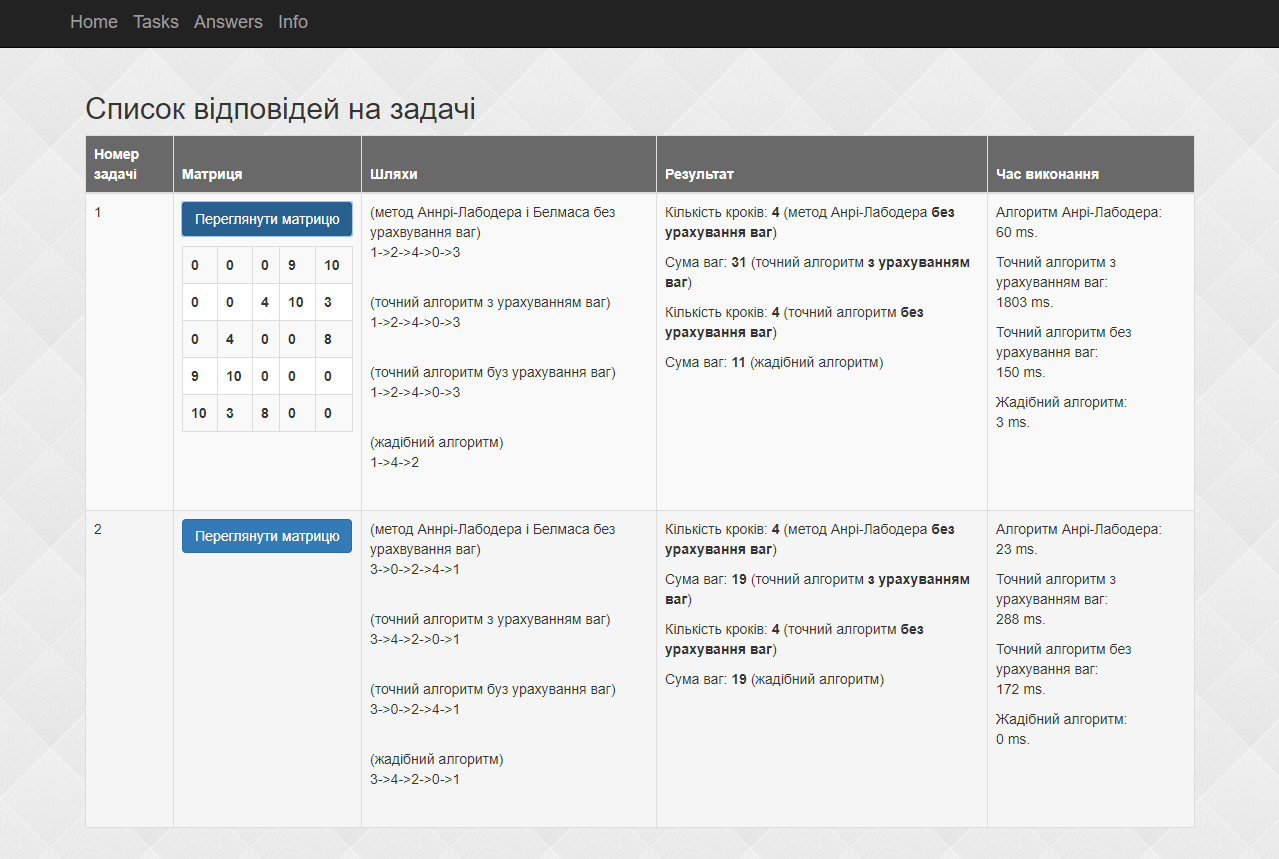


Рисунок 5.8 – Сторінка списку розв’язаних задач

**Де подивитись інформацію про авторів, курсову роботу та почитати коротку інструкцію по користуванню застосунком?**

На головній сторінці натисніть на кнопку “” або на кнопку “Info” в шапці сайту. Після цього відкриється наступна сторінка (рисунок 5.9).

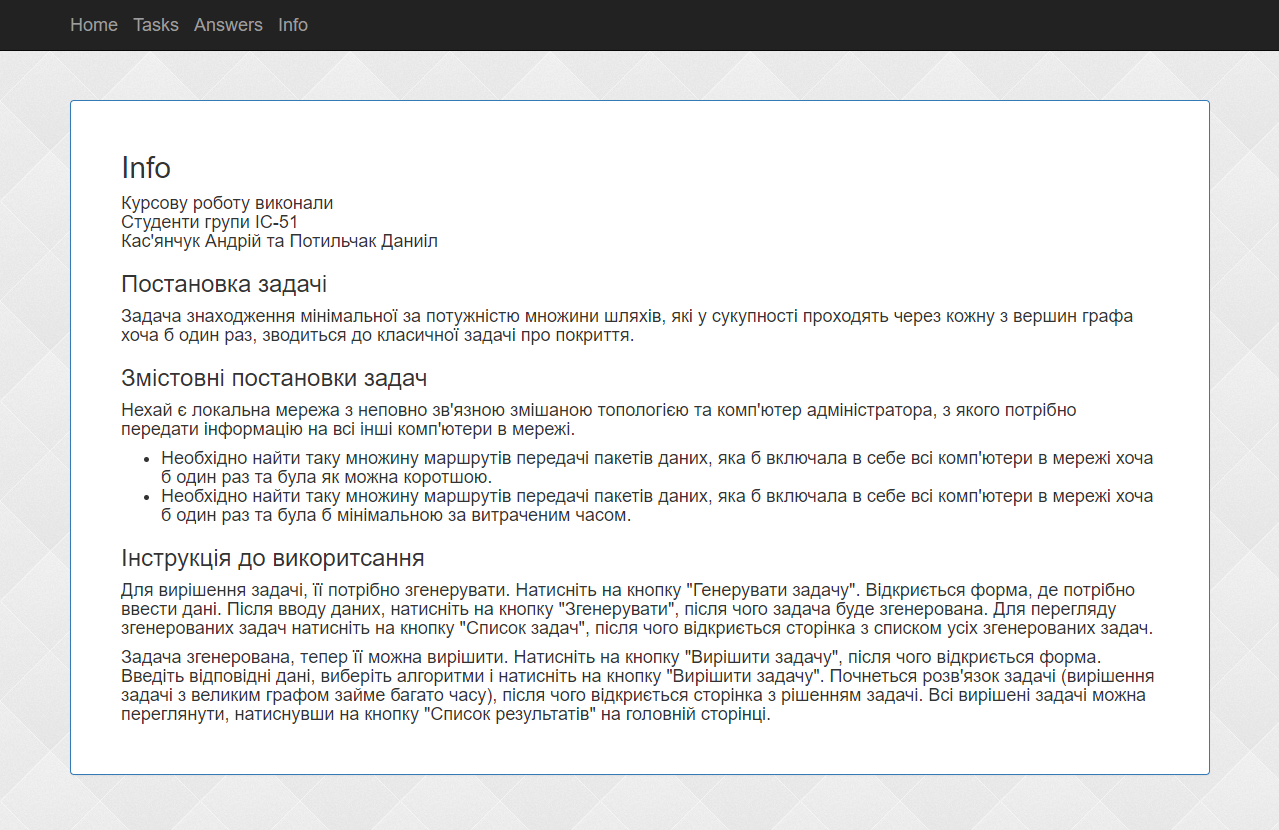


Рисунок 5.9 – Сторінка Info

## **Методика випробувань**

В даному ПЗ використовується валідація введених даних, отже користувач не може ввести дані, які ПЗ не зможе обробити. Таким чином, методика випробувань у даному ПЗ зводиться до генерації графів різних розмірностей та перевірки правильності рішень за допомогою стороннього ПЗ або інших методів.

# **ВИСНОВОК**

Для вирішення поставленої задачі у ході курсової роботи у рамках дисциплін «Дослідження операцій» та «Технології розробки програмного забезпечення» було розроблено програму, яка розв’язує задачі покриття усіх вершин графа.

# **Додаток А**

А.1 Календарний план

**КАЛЕНДАРНИЙ ПЛАН ВИКОНАННЯ ПРОЕКТУ**

|  |  |  |  |
| --- | --- | --- | --- |
| **№ з/п** | **Назва етапів виконання проекту** | **Строк виконання етапів проекту** | **Примітка** |
| *1.* | *Вивчення рекомендованої літератури* | 18.03.2018 |  |
| *2.* | *Аналіз існуючих методів розв’язання задачі* | 30.03.2018 |  |
| *3.* | *Постановка та формалізація задачі, виявлення вимог до реалізації проекту* | 6.04.2018 |  |
| *4.* | *Розробка інформаційного забезпечення* | 13.04.2018 |  |
| *5.* | *Створення моделі бази даних* | 18.04.2018 |  |
| *6.* | *Розробка програмного забезпечення* | 24.04.2018 |  |
| *7.* | *Налагодження та тестування програми* | 13.05.2018 |  |
| *8.* | *Оформлення пояснювальної записки роботи* | 14.05.2018 |  |
| *9.* | *Захист роботи* | 17.05.2018 |  |

А.2 Початковий план робіт

А.3 Кінцевий план робіт

# **Додаток Б**

Б.1 Програмний код

Посилання на репозиторій GitHub з проектом CoverTask: https://github.com/Kasyanchyk/CoveringTask.git