Software Requirements Specification

For

CityMate: Travel Management System

01/11/2021

Prepared by

Specialization	SAP ID	Name
AI & ML	500075391	Abhijeet Jain
AI & ML	500076058	Aryan Verma
AI & ML	500075878	Chirag Sankhla

UPES
UNIVERSITY WITH A PURPOSE

Department of Informatics
School Of Computer Science
UNIVERSITY OF PETROLEUM & ENERGY STUDIES,
DEHRADUN- 248007. Uttarakhand

Table of Contents

1. INTRODUCTION

CityMate as the name suggests it has something to do with travelling in a city or travelling in any scope . For finding shortest distance between places , usually we go to google maps and get things done but

What if ! we need various stops ,
What if ! we need to plan our daily travels in the quickest way possible.

This is where CityMate comes into play we worked on different algorithms to find the best way people didn't think they needed .

1.1. Purpose of the project

The purpose of the project is to find the best possible algorithm to find shortest way when the user needs to visit several places within a city . User will input locations and the app is suppose to find the most cost efficient way in minimal time .

1.2. Target Beneficiary

The idea of this project came from the unexplored difficulty faced by Indians when they need to send wedding invitations but considering the task having similarities with everyday travel issues it can be useful to everyone from a travel vlogger to a milkman .

1.3. Project Scope

The objective of this project is to find most time and cost efficient algorithm to find shortest path between multiple stops .

1.4. Refrences

- See the TSP world tour problem which has already been solved to within 0.05% of the optimal solution. [1]
- ^ *Ross, I. M.; Proulx, R. J.; Karpenko, M. (6 May 2020). "An Optimal Control Theory for the Traveling Salesman Problem and Its Variants". arXiv:2005.03186 [math.OC].*
- Zomato solving travelling salesman problem https://zenodo.org/record/2656305/files/Online%20FDS.pdf
- Problem solving state representation https://www.geeksforgeeks.org/travelling-salesman-problem-set-1/
- ^ A discussion of the early work of Hamilton and Kirkman can be found in *Graph Theory, 1736–1936* by Biggs, Lloyd, and Wilson (Clarendon Press, 1986).

- *Informed and Uninformed Search Algorithms* *https://www.geeksforgeeks.org/difference-between-informed-and-uninformed-search-in-ai/*
- ^ *Klarreich, Erica (8 October 2020). "Computer Scientists Break Traveling Salesperson Record". Quanta Magazine. Retrieved 13 October 2020.*
- ^ *Karlin, Anna R.; Klein, Nathan; Gharan, Shayan Oveis (30 August 2020). "A (Slightly) Improved Approximation Algorithm for Metric TSP". arXiv:2007.01409 [cs.DS].*
- Hash Map https://www.geeksforgeeks.org/hashing-data-structure/

2. PROJECT DESCRIPTION

2.1. Refrence Alorithm

Dijkstra Algo:

Step 1 : Passing the Adjacency matrix , source and destination to the function
Step 2: initializing distance array and visited array with int max and false
Step 3: setting distance of source equal to 0
Step 4: Starting a loop equal to number of the nodes
Step 5: Calculation of the temporary distances of all neighbour nodes of the active node by summing up its distance with the weights of the edges.
Step 6: calculated distance of a node is smaller as the current one, update the distance and set the current node as antecessor.
Step 7: add the node in path vector to display the path in order

Dynamic Programming :

Step 1: Send the source and Adjacency matrix to the function
Step 2: Create a array to mark the visited city
Step 3: all cities are unvisited, and the visit starts from the source city
Step 4: the TSP distance value is calculated based on a recursive function
Step5: if the distance is less than the minimum distance then added the distance as optimal cost and update the minimum distance
Step 6: Return the optimal cost

2.2. Characterstic of Data

Data of distance between various blocks of University of petroleum and energy Studies is hard coded in a Ordered Map.

Major Data Structures – Graph , Map , Array ,Vector

2.3. SWOT Analysis

Strengths : Fast , Easy to use

Weaknesses : Application is capped to limited locations , for wide usage the user must enter the pairwise distance which might create a bad user experience .

Opportunities : CityMate does not qualify enough to be a standalone application but it can be great extension to existing navigation apps .

Threats : When the number of visits increases the computational power required increases exponentially therefore there are chances of unexpected crashes in case of low end devices .

2.4. Project Features

The project has following key features

- A console based interface where user can input the places user wishes to visit.
- Citymate finds the best possible route with the cost of travel in units **.

2.5. User Classes and Characteristics

Considering CityMate a console based single use-case application there is not much a user can perform.

User Module :

- Input – user inputs destinations and pairwise distance between them .

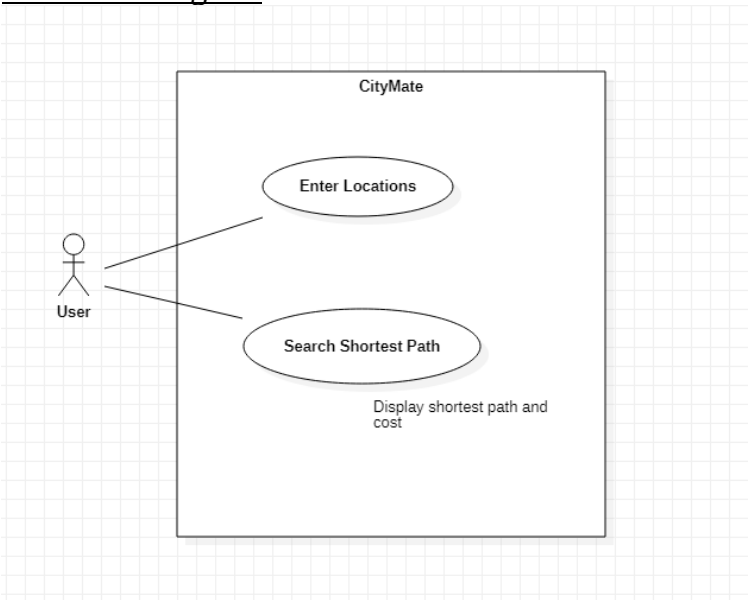
2.6. Design and Implementation Constraints

Operating System: Windows 10 , 11
Language for implementation: C++

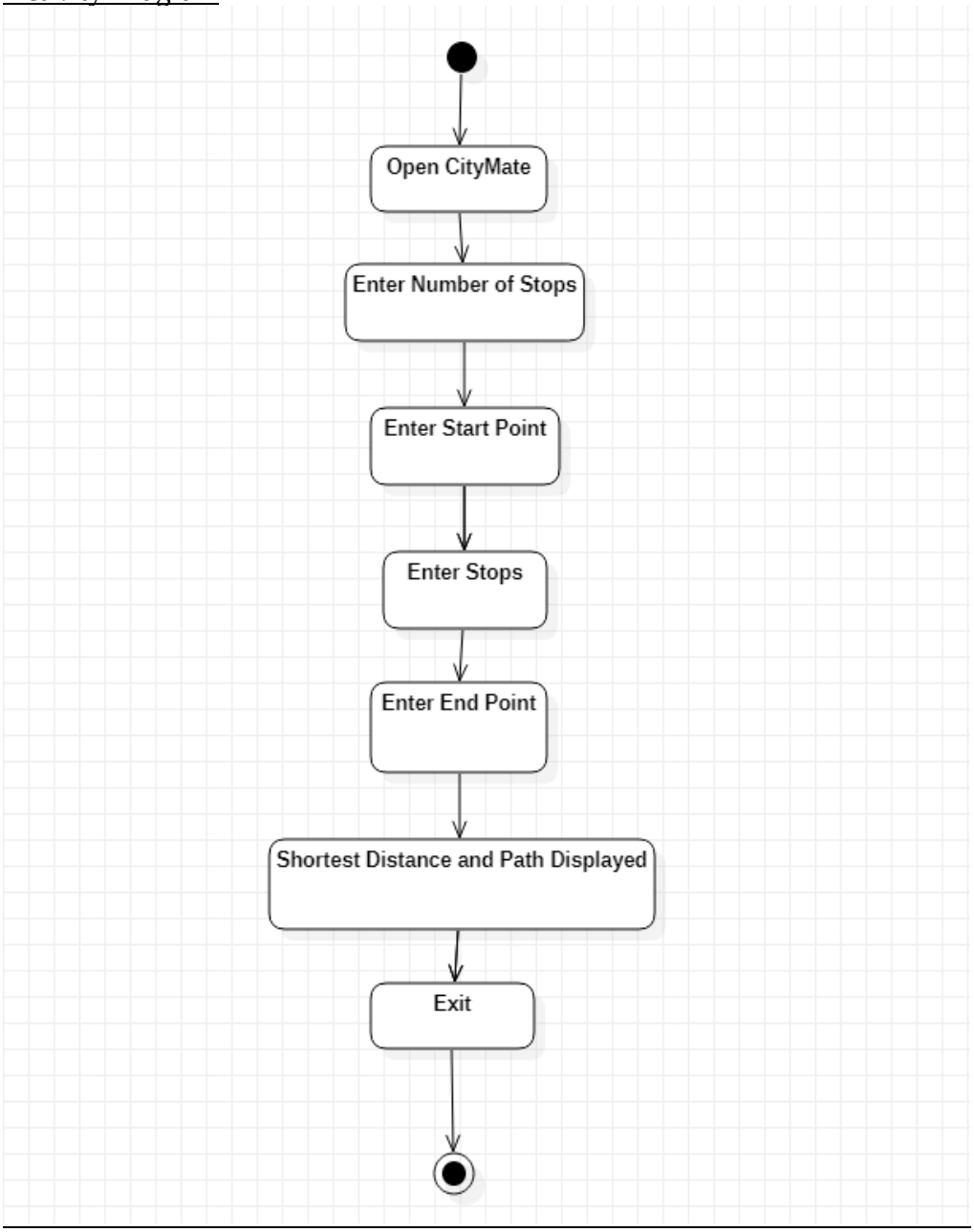
2.7. Design diagrams

UML diagram –

Use Case Diagram



Activity Diagram



3. SYSTEM REQUIREMENTS

3.1. User Interface

A console based interface for user to input start point ,end point and stops in the way .

3.2. Software Interface

The software shall take the input and calculate the shortest distance between start and end point also shortest path with the stops .

4. NON FUNCTIONAL REQUIREMENTS

4.1. Performance Requirements

The software is well tested on windows machines with i5 and i7 intel processors . It might perform a bit slower on low end processors.

4.2. Security Requirements

The Software does not store any sensitive information also no extra permissions are required , it has straight input to output workflow.

4.3. Software Quality Attributes

- Availability : the system is robust and can be used multiple times within a time frame.
- Adaptability : with single use case the software can handle any environment changes until the system requirements are met.
- Reliability : Product is reliable enough to sustain any condition and should give correct results consistently.
- Maintainability : Different versions of the product should be easy to maintain. For development, it should be easy to add code to the existing system, should be easy to upgrade for new features and new technologies from time to time.
- Usability : This can be measured in terms of ease of use. The application should be user-friendly. It should be easy to learn. Navigation should be simple.
- Portability: This can be measured in terms of Costing issues related to porting, Technical issues related to porting, and Behavioral issues related to porting.
- Correctness: The application should be correct in terms of its functionality, calculations used internally and the navigation should be correct. This means that the application should adhere to functional requirement.
- Flexibility: Should be flexible enough to modify. Adaptable to other products with which it needs interaction. Should be easy to interface with other standard 3rd party components.
- Reusability: Software reuse is a good cost-efficient and time-saving development method. Different code library classes should be generic enough to be easily used in different application modules.
- Robustness: The code is able to cope with errors during execution and cope with erroneous input.
- Testability: The system should be easy to test and find defects. If required, it should be easy to divide into different modules for testing.

APPENDIX A : GLOSSARY

CityMate helps optimizing travels with multiple stops while finding the best algorithm we came across cases where in order to find result in a reasonable time the app compromises with a less optimal route.
CityMate is a single use case console based application which can be a great extension to existing navigation apps .