Software Requirements Specification

For

ExploreMate

Prepared By

Specialization	SAP ID	Name
AIML Hons B3	500106941	Kashish Deol
AIML Hons B3	500105627	Milind Vishwakarma
AIML Hons B3	500109640	Sunny Jain
AIML Hons B3	500109566	Pratham Kumar Srivastav

Cluster: Artificial Intelligence and Machine Learning
School Of Computer Science
University of Petroleum & Energy Studies,
Dehradun- 248007, Uttarakhand

Table of Content

1. INTRODUCTION

1.1. Purpose of the Project

The purpose of this project is to create an application that provides personalized travel suggestions by optimizing routes based on user-defined preferences like budget, time, and distance. It seeks to enhance the travel planning experience by offering tailored, dynamic recommendations that address individual needs and constraints.

1.2. Target Beneficiaries

Frequent Travelers: People who travel often, either for business or leisure, and need quick, efficient ways to plan their trips.
Busy Professionals: Individuals who may not have the time to manually research destinations and create detailed itineraries, relying on an automated tool to save time.
Budget-Conscious Travelers: Those who want to maximize their experiences within a limited budget and need optimized recommendations for cost-effective trips.
Travel Agencies: Agencies looking for tools to automate and enhance the experience they offer to their clients by providing tailored itineraries.

1.3. Project Scope

The scope of the **ExploreMate** project involves developing a personalized travel recommendation system in C++ that automates trip planning and optimizes itineraries based on user preferences, including time, location, budget, and destination choices. The system will feature advanced algorithms for shortest-path optimization, ensuring efficient multi-destination planning. This project will allow users to input their travel constraints, and the system will generate cost-effective, time-saving itineraries, while also providing suggestions for attractions and accommodations. The system will incorporate memory-based learning, adapting to users' previous interactions to refine future recommendations. Developed in C++, the focus will be on high performance, scalability, and efficiency, ensuring ExploreMate is capable of serving a wide range of travelers—from frequent travelers and budget-conscious tourists to event planners and travel agencies.

1.4. References

- Lika, B., Kolomvatsos, K., & Hadjiefthymiades, S. (2014). Personalized Recommendations Based on Users' Preferences in Collaborative and Content-Based Filtering. Journal of Travel Research, 48(4), 1-15.
- Ricci, F., Rokach, L., & Shapira, B. (2011). Introduction to Recommender Systems Handbook. Springer.
- Alin, A., & Poenar, C. (2020). Shortest Path Algorithms and Applications in Transportation Networks. Transport Systems Journal.
- Gavalas, D., & Kenteris, M. (2011). A Web-Based Pervasive Recommendation System for Mobile Tourist Guides. Journal of Personalization Research.

2. PROJECT DESCRIPTION

2.1. Data/ Data Structure

The recommendation system primarily deals with graphs, where information is represented as edges that contains multiple attributes such as distance, duration, rating, and connectivity that is accompanied by customizable biases. The data can be broken down into the following categories:

2.2.1. Input Data:

- Current Location: The system accepts user’s current location to start with the processing of next connected location for generating recommendation.

- Duration: Available duration is fetched from user to decide upon the suggestions and where to stop.

2.2.2. Internal Data Structures:

i. Edge representation

- Edges define the connectivity between two nodes (i.e. locations in our application), and carries all the essential attributes such as distance (from current location to next connected location), duration (time estimated to spend on the next connected location), rating (of the connected location), and connectivity (it is the numerical value that describes how many locations are connected to that location).
- Multiple objects for edge class will be created and added in a file that will be accessed further to design the graph class.

ii. Graphs Representation

- This class utilizes multiple data structures available in C++ such as map, lists, tuple, pair that are further integrated with each other to formulate a graph that further uses edges (edge objects) to construct the working of our recommendation system.

2.2.3. Output Data

- Available locations
- All possible routes from the current location
- Optimized routes from the available ones

2.2. SWOT Analysis

Strengths:

- **Advanced Algorithms:** ExploreMate uses combinatorial algorithms that provide optimized travel itineraries based on user-defined parameters like budget, time, and destination. This enhances the accuracy and relevance of travel recommendations.
- **Personalization:** The feedback mechanism allows the system to learn from user behavior, enabling it to refine future travel suggestions and create tailored itineraries over time.
- **Efficient Core Processing:** Built using C++, the software benefits from high efficiency in terms of data processing and performance.
- **User-Friendly Design:** Despite being technically advanced, the software is designed to be accessible to users, making it easy for anyone to plan their travel.

Weaknesses:

- **Limited Platform Integration:** Since C++ is mainly used for core processing, cross-platform or web-based integration may require additional development in other languages or frameworks.
- **Potential Learning Curve:** While user-friendly, users may need some time to fully understand the parameters and personalization features in order to optimize their travel plans effectively.
- **Dependence on User Data:** The personalization system heavily relies on user interaction to improve recommendations. New users may not experience the full benefits immediately.

Opportunities:

- **Expansion to Mobile Apps:** ExploreMate can leverage mobile app development to attract more users who prefer on-the-go planning.

- **Collaboration with Travel Agencies or Services:** Integration with airlines, hotels, and tour agencies could expand its ecosystem and provide real-time pricing or availability, making the app even more valuable.

- **AI Integration:** Leveraging AI to predict trends, preferences, or real-time adjustments based on changing circumstances (e.g., weather or flight delays) could further enhance user experience.

- **Global Market Expansion:** The software can be localized into multiple languages and currencies to cater to a global audience.

Threats:

- **Competition in Travel Tech**: There are several existing travel management platforms that already dominate the market.

- **Changing Travel Trends:** Post-pandemic travel behavior is evolving, and if ExploreMate doesn't stay updated with new trends or consumer expectations, it could lose relevance.

- **Privacy Concerns:** Handling sensitive user data (preferences, locations, etc.) must be done securely to prevent breaches and ensure user trust.

2.3. Project Features

- **Multi-Destination Support:** Allows users to plan trips with multiple destinations and suggests the best order and route to visit them, optimizing for cost, time, or convenience.

- **Fast and Efficient Performance:** Built in C++, ensuring high performance, faster computations, and efficient memory management to handle complex travel route calculations and large datasets.

- **Graph-Based Route Visualization**: Provides visual representations of travel routes and destinations, enhancing user experience and understanding of the itinerary plan.

- **Memory Based Recommendation System:** Uses memory-based learning to improve suggestions over time by analysing past user interactions and feedback.

- **Scalability for Travel Agencies:** Designed to handle multiple users and travel groups, offering large-scale planning solutions for travel agencies needing optimized itineraries for clients or attendees.

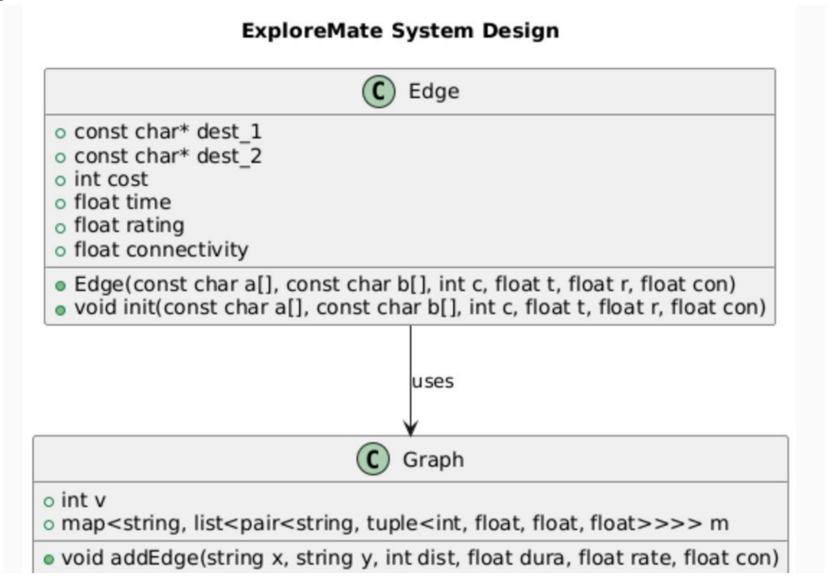
These features combine to offer an intelligent, automated solution for optimizing travel plans while adapting to users' needs and preferences.

2.4. Design and Implementation Constraints

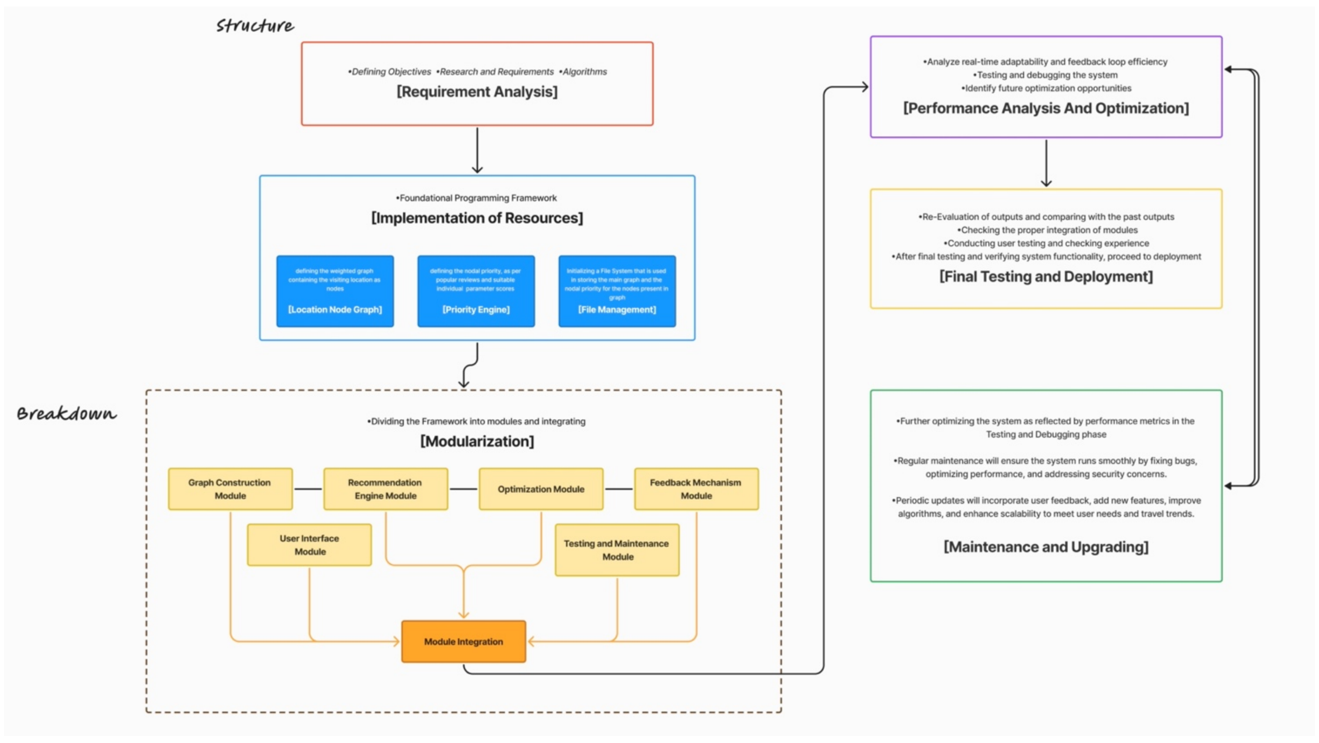
- **Programming Language:** The project must be implemented in C++, which, while providing high performance, also requires careful memory management and error handling.
- **Algorithmic Constraints:** The algorithm must be capable of scaling for large datasets (such as numerous destinations and points of interest). Memory usage and computational time need to be optimized for large graphs representing travel routes.
- **Data Constraints:** Travel data, such as locations, routes, should be sourced from reliable databases or APIs. The availability of real-time data may impact the quality of recommendations.
- **Memory Efficiency**: Given the need for processing large datasets of locations, distances, and travel options, the application must ensure efficient memory usage, avoiding memory leaks and ensuring optimal utilization through C++ features like smart pointers and custom allocators.
- **User Data Privacy**: Since the system may store users' personal travel preferences, ensuring data security and privacy is paramount.

2.5. Design Diagrams

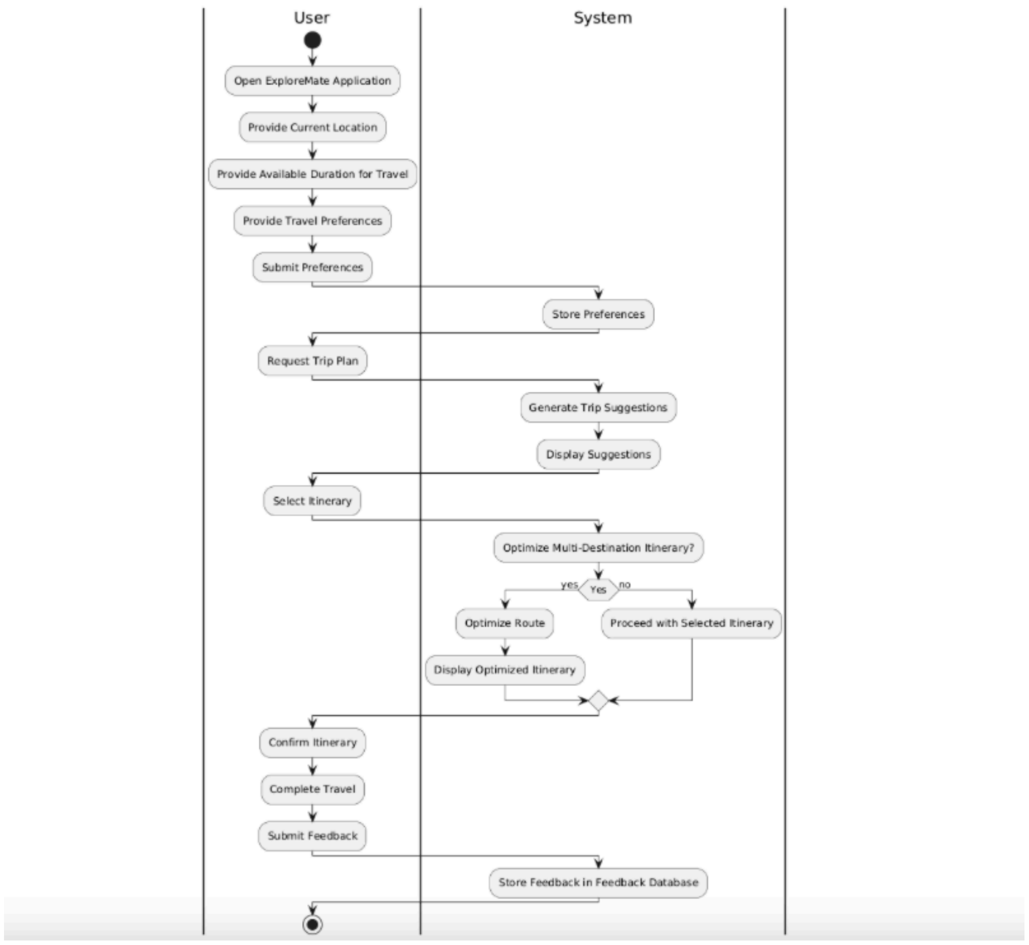
1. Class Diagram


ExploreMate System Design
Edge
const char* dest_1
const char* dest_2
int cost
float time
float rating
float connectivity
Edge(const char a[], const char b[], int c, float t, float r, float con)
void init(const char a[], const char b[], int c, float t, float r, float con)
uses
Graph
int v
map<string, list<pair<string, tuple<int, float, float, float>>>> m
void addEdge(string x, string y, int dist, float dura, float rate, float con)

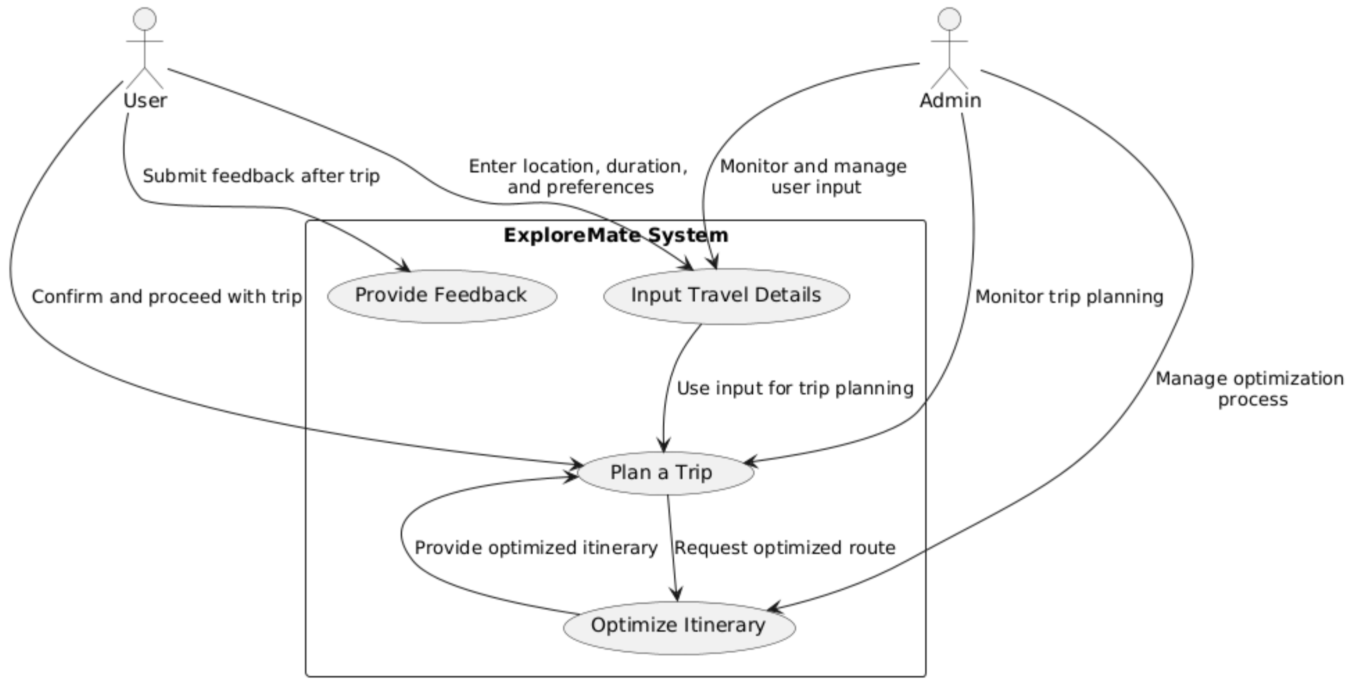
2. Work Flow diagram


Structure
•Defining Objectives •Research and Requirements •Algorithms
[Requirement Analysis]
•Foundational Programming Framework
[Implementation of Resources]
defining the weighted graph containing the visiting location as nodes
[Location Node Graph]
defining the nodal priority, as per popular reviews and suitable individual parameter scores
[Priority Engine]
Initializing a File System that is used in storing the main graph and the nodal priority for the nodes present in graph
[File Management]
Breakdown
•Dividing the Framework into modules and integrating
[Modularization]
Graph Construction Module
Recommendation Engine Module
Optimization Module
Feedback Mechanism Module
User Interface Module
Testing and Maintenance Module
Module Integration
•Analyze real-time adaptability and feedback loop efficiency
•Testing and debugging the system
•Identify future optimization opportunities
[Performance Analysis And Optimization]
•Re-Evaluation of outputs and comparing with the past outputs
•Checking the proper integration of modules
•Conducting user testing and checking experience
•After final testing and verifying system functionality, proceed to deployment
[Final Testing and Deployment]
•Further optimizing the system as reflected by performance metrics in the Testing and Debugging phase
•Regular maintenance will ensure the system runs smoothly by fixing bugs, optimizing performance, and addressing security concerns.
•Periodic updates will incorporate user feedback, add new features, improve algorithms, and enhance scalability to meet user needs and travel trends.
[Maintenance and Upgrading]

3. Activity Diagram


User
System
Open ExploreMate Application
Provide Current Location
Provide Available Duration for Travel
Provide Travel Preferences
Submit Preferences
Store Preferences
Request Trip Plan
Generate Trip Suggestions
Display Suggestions
Select Itinerary
Optimize Multi-Destination Itinerary?
yes
Yes
no
Optimize Route
Proceed with Selected Itinerary
Display Optimized Itinerary
Confirm Itinerary
Complete Travel
Submit Feedback
Store Feedback in Feedback Database

4. **Use-Case Diagram**


User
Admin
ExploreMate System
Submit feedback after trip
Enter location, duration, and preferences
Monitor and manage user input
Confirm and proceed with trip
Provide Feedback
Input Travel Details
Monitor trip planning
Use input for trip planning
Manage optimization process
Plan a Trip
Provide optimized itinerary
Request optimized route
Optimize Itinerary

3. SYSTEM REQUIREMENTS

3.1. <u>User Interface</u>

- Input Handling: User will enter the system interactively through a console-based input giving their current location and the duration they have for travelling.
- Output Display: The output will be locations that could be travelled in the available duration. These locations will be listed based upon their ratings, and several other factors.

3.2. <u>Software Interface</u>

- Operating System – This system is more flexible and can be used with general-purpose operating systems that support C++, such as Windows and Linux.
- Development Environment – The program uses standard C++ libraries like <iostream>, <fstream>, <vector>, <string>, <algorithm>, and <cmath>. These can compile with any C++ compiler, including GCC or MSVC.

3.3. <u>Protocols</u>

No network protocols are involved with this code. All operations are local to the machine, reading images from disk and processing them in memory in line.

4. NON-FUNCTIONAL REQUIREMENTS

4.1. <u>Performance Requirements</u>

- Efficiency – It should be really efficient on low-end systems. Thus, it doesn't rely on any external libraries so it stays lightweight.

4.2. Security Requirements

- File Handling Security – It supports file input but doesn't deal with sensitive data, databases, etc. It is required to only handle errors properly, namely an unsupported format, as well as a file that can't be opened.

4.3. Software Quality Attributes

- Maintainability – The code has a modularity structure such that the improvement technique for every technique is encapsulated in its own separate function, making it easier to maintain or extend by incorporating new filters or processing methods.

- Portability – The program will be written in standard C++ and will just run in any platform which supports C++ with minimal changes.

- Usability – The console-based interface is primitive but of course not sophisticated in functionalities compared to a graphical interface. A user has to enter a valid current location and numeric input for available duration.

- Reliability – The system implements the basic error handling, so that errors are handled without causing any system crashes.

5. OTHER REQUIREMENTS

- Portability – The developed image enhancement system must be portable, running on environments with the presence of Windows and Linux with minimal, if any change at all. This includes low-end devices where computation resources will be limited.

- Maintainability – Such a codebase should be well commented, modular, and easy to extend or adapt functionality by future developers or users of the system. Clean code practices must be followed to facilitate understanding and modification.

- Usability – The system should be user-friendly enough for a general user but also provide options for professionals. Feedback mechanisms are supposed to be simple and intuitive so that even a non-technical user can enhance pictures without much effort.

- Deployment Considerations – The system should have guidelines on how to deploy in different environments including providing a lightweight version for low-resource devices and a high-performance version for resource-rich systems.

This means the additional requirements ensure the system is versatile and easy to maintain, adaptable for user needs and their environments.