BSCS FINAL PROJECT

Design and Test Specification

WanderWise



Project Advisor

**Haroon Abdul Waheed**

Presented by:

**Group ID: F23SE032**

Student Reg# L1F20BSSE0472 Student Name: Ameer Hamza

Student Reg# L1F20BSSE0459 Student Name: Saqlain Shabaz

Student Reg# L1F20BSSE0472 Student Name: Sehar Arooj

**Faculty of Information Technology**

**University of Central Punjab**

# Design and Test Specification

# SDP Phase III

# WanderWise

# Advisor: Haroon Abdul Waheed

# Team F23SE032

|  |  |
| --- | --- |
| Member Name | Primary Responsibility |
| Ameer Hamza | Project Manager |
| Saqlain Shahbaz | Backend Developer |
| Sehar Arooj | Frontend developer |
|  |  |

Table of Contents

**Table of Contents** [**i**](#__RefHeading___Toc79435955)

Revision History [ii](#__RefHeading___Toc79435956)

Abstract [iii](#__RefHeading___Toc79435957)

1. Introduction [1](#__RefHeading___Toc79435958)

1.1 Product [1](#__RefHeading___Toc79435959)

1.2 Background [1](#__RefHeading___Toc79435960)

1.3 Objective(s)/Aim(s)/Target(s) [1](#__RefHeading___Toc79435961)

1.4 Scope [1](#__RefHeading___Toc79435962)

1.5 Business Goals [1](#__RefHeading___Toc79435963)

1.6 Document Conventions [1](#__RefHeading___Toc79435964)

1.7 Miscellaneous [1](#__RefHeading___Toc79435965)

2. Technical Architecture [2](#__RefHeading___Toc79435966)

2.1 Application and Data Architecture [2](#__RefHeading___Toc79435967)

2.2 Component Interactions and Collaborations [2](#__RefHeading___Toc79435968)

2.3 Design Reuse and Design Patterns [3](#__RefHeading___Toc79435969)

2.4 Technology Architecture [3](#__RefHeading___Toc79435970)

2.5 Architecture Evaluation [3](#__RefHeading___Toc79435971)

3. Detailed/Component Design [4](#__RefHeading___Toc79435972)

3.1 Component-Component Interface [4](#__RefHeading___Toc79435973)

3.2 Component-External Entities Interface [4](#__RefHeading___Toc79435974)

3.3 Component-Human Interface [4](#__RefHeading___Toc79435975)

4. Screenshots/Prototype [4](#__RefHeading___Toc79435976)

4.1 Workflow [4](#__RefHeading___Toc79435977)

4.2 Screens [4](#__RefHeading___Toc79435978)

4.3 Additional Information [4](#__RefHeading___Toc79435979)

5. Other Design Details [5](#__RefHeading___Toc79435980)

6. Test Specification and Results [5](#__RefHeading___Toc79435981)

6.1 Test Case Specification [5](#__RefHeading___Toc79435982)

6.2 Summary of Test Results [6](#__RefHeading___Toc79435983)

7. Revised Project Plan [6](#__RefHeading___Toc79435984)

8. References [7](#__RefHeading___Toc79435985)

Appendix A: Glossary [8](#__RefHeading___Toc79435986)

Appendix B: IV & V Report [9](#__RefHeading___Toc79435987)

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
|  |  |  |  |
|  |  |  |  |

# Abstract

WanderWise is a cutting-edge web-based travel planning application that revolutionizes the travel exploration process. The project aims to solve the complexity and time consumption associated with trip planning while prioritizing user safety. Utilizing machine learning algorithms, WanderWise offers personalized travel itineraries, real-time safety advisories, and weather forecasts, ensuring each journey is unique, secure, and memorable.

# Introduction

## Product

Inefficient and static travel planning tools lack personalized itineraries and fail to provide real-time safety advisories based on dynamic factors such as weather forecasts, thereby compromising travelers' experiences and safety during their journeys.

## Background

The domain or background for the problem addressed by WanderWise lies within the travel and tourism industry. In traditional travel planning, individuals often rely on static itineraries, guidebooks, or online resources that may offer generic suggestions but lack personalization tailored to specific preferences, interests, or changing conditions during travel.

The existing tools and platforms typically provide static information without considering real-time factors such as weather forecasts, local safety advisories, or dynamically altering conditions at destinations. This lack of personalized guidance and timely updates can lead to suboptimal travel experiences, unexpected challenges, and potential safety concerns for travelers.

## Objective(s)/Aim(s)/Target(s)

* To simplify and enhance the travel planning process for users of varying travel experience.
* To provide personalized travel suggestions based on user preferences.
* To ensure user safety through real-time advisories.
* To offer an intuitive and engaging user experience.

## Scope

WanderWise, a web-based travel planning application. The scope of this project includes:

* Development of a user-friendly web interface for trip planning.
* Integration of machine learning algorithms for personalized itinerary generation.
* Implementation of real-time safety advisories and weather forecast systems.
* Ensuring data security and privacy of users.
* Providing support for multiple devices and platforms.

## Business Goals

* **Elevate Travel Planning User Experience:** Enhance the overall user experience by optimizing the process of planning and organizing travel activities to ensure convenience and satisfaction.
* **Real-Time Safety Planning Assurance for Users**: Implement a system that delivers timely and accurate safety advisories in real time to users during their travel experiences, prioritizing their well being and security.

# Technical Architecture

## WanderWise Architecture

## Application and Data Architecture

*<Describe the application components and/or processing units in the architecture diagram above. Complete logical or physical model is expected. Diagrams/Tools that may be provided in this section include Component Diagram, ER Diagram, Class Diagram (with complete inheritance, composition, and association details), Activity Diagram, Decision Table etc. All these diagrams should have more details than the details provided in Phase 1 of the SDP. Research based projects may provide complete design of the proposed system. Describe each diagram briefly.>*

### Component Diagrams

### ER Diagram

### Class Diagram

### Activity Diagram

## Component Interactions and Collaborations

*<Provide interactions and collaborations between your system components/processing units. Diagrams/Tools that may be provided in this section include Design Level Sequence Diagram, Collaboration Diagram, Event Traces, Detailed DFD, Activity Diagram etc. All these diagrams should have more details than the details provided in Phase 1 of the SDP. Describe each diagram briefly.>*

### Sequence Diagram

### DFD level 0

### DFD level 1

## Design Reuse and Design Patterns

During the development of WanderWise, the team identified several opportunities for design reuse, particularly in the implementation of machine learning algorithms for personalized itinerary generation. The team leveraged existing libraries and frameworks for machine learning, such as scikit-learn and TensorFlow, to expedite development and ensure the reliability of the predictive models used in WanderWise.

the team also identified opportunities for design reuse, in the implementation of backend services for generating personalized itineraries. The team leveraged Flask-based services for itinerary generation, which encapsulates complex algorithms and logic. Additionally, design patterns such as the Model-View-Controller (MVC) pattern were utilized to ensure separation of concerns and maintainability.

## Technology Architecture

WanderWise is MERN stack application. The anticipated infrastructure includes:

**Platform:** The frontend of WanderWise is built using React.js, providing a dynamic and responsive user interface. The backend is powered by Node.js and Express.js, facilitating efficient handling of HTTP requests and integration with external APIs.

**System Hosting:** The system will be hosted on cloud infrastructure providers such as Amazon web services(AWS)

**Connectivity Requirements:** The application will require internet connectivity to access real-time data for personalized itinerary generation, safety advisories, and weather forecasts.

**Modes of Operations:** WanderWise operates in a client-server model, with the client-side React application making requests to the server-side Node.js application. The backend communicates with the Flask-based itinerary service for personalized itinerary generation and with external APIs for additional destination data.

## Architecture Evaluation

The selection of the MERN stack for WanderWise was based on its suitability for building modern web applications with a rich user interface and efficient backend processing. The team evaluated alternative stacks but ultimately chose MERN for its flexibility, developer friendliness, and extensive community support.

* Backend Framework: Node.js and Express.js were chosen for their lightweight nature, asynchronous I/O capabilities, and seamless integration with other components of the MERN stack.
* Frontend Framework: React.js was selected for its component-based architecture, virtual DOM, and extensive ecosystem of libraries and tools for building interactive user interfaces.
* Database: MongoDB was chosen as the database solution for its flexibility in handling unstructured data and seamless integration with Node.js through the Mongoose ORM.

# Detailed/Component Design

<In this section provide details of the types of interactions and collaborations required in your project. Also elaborate the interactions already described in section 2.2. Tools such as deployment diagram, class diagram (with complete design details), database schema, design level sequence diagram, state diagram, petrinets etc. may be used to provide all the details. Any design principles being followed during the projects may be mentioned and described in this section.>

## Component-Component Interface

The components of WanderWise interact as follows:

* Frontend: The frontend component, built using React.js, communicates with the backend component via HTTP requests. It sends user input data such as destination preferences and travel dates to the backend for processing. Upon receiving responses from the backend, it renders the relevant information to the user interface.
* Backend: The backend component, powered by Node.js and Express.js, handles incoming HTTP requests from the frontend. It processes user input data, interacts with external APIs such as Google Maps and Tripadvisor, and communicates with the Flask-based itinerary service for personalized itinerary generation. The backend sends the processed data back to the frontend for display.
* Flask-based Itinerary Service: This component is responsible for generating personalized itineraries based on user preferences. It receives requests from the backend, processes them using machine learning algorithms, and returns the generated itineraries to the backend for further processing.

### Sequence diagram

## Component-External Entities Interface

WanderWise interacts with external entities such as **Google Maps** and **Tripadvisor** for additional destination data. The backend component makes HTTP requests to these external APIs to retrieve information such as maps, points of interest, reviews, and ratings. The retrieved data is then used to enhance the user experience and provide comprehensive travel recommendations.

## Component-Human Interface

The user interacts with WanderWise through various screens and points, including:

* Homepage: Provides an overview of the application and options for trip planning.
* Search Page: Allows users to input their destination preferences, travel dates, and other relevant information.
* Itinerary Page: Displays the personalized itinerary generated by the system, including activities, accommodations, transportation options, and safety advisories.
* Destination Details: Provides additional information about specific destinations, including maps, points of interest, reviews, and ratings.

WanderWise follows Human-Computer Interaction (HCI) principles to ensure an intuitive and user-friendly interface, including consistent navigation, clear feedback, and accessible design elements.

# Screenshots/Prototype

*<This complete section should be an improved version of the section presented in SDS>*

## Workflow

<Describe complete workflow of your system. Swim-lane diagram may be used. This section should be an improved version of the section presented in SDS>

## Screens

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

## Additional Information

<Any additional information>

# Other Design Details

<Describe any design details not covered in previous sections. Add subsections as required. There can be details regarding Research Oriented, Game Oriented, or Hardware based projects that have not been covered in this document before, those details can be provided in this section. For example research based projects may use this section to present their results and analysis; hardware based projects may use this section to describe interface dependencies and issues etc.>

# Test Specification and Results

## Test Case Specification

< Fill out the following template for each test case. Provide separate tables for input data with each test case if applicable. Research based projects may need to replace this test specification with their own test mechanism.>

Table 6.1: TC-1

|  |  |
| --- | --- |
| **Identifier** | TC-1 |
| **Related requirements(s)** | Functional Requirement UC-1 , SRS section 2.1 |
| **Short description** | For user registration functionality |
| **Pre-condition(s)** | User navigates to registration page |
| **Input data** | User Enter Valid email, password, |
| **Detailed steps** | 1. Navigate to the registration page  2. Enter valid user details  3. Click on the “Register” button |
| **Expected result(s)** | User is successfully registered and redirected to login page |
| **Post-condition(s)** | User account is created in the database. |
| **Actual result(s)** | User registration successful. |
| **Test Case Result** | **Passed** |

## Table 6.2: TC-2

|  |  |
| --- | --- |
| **Identifier** | TC-2 |
| **Related requirements(s)** | Functional Requirement UC-2 , SRS section 2.2 |
| **Short description** | For user Login functionality |
| **Pre-condition(s)** | User navigates to login page |
| **Input data** | User Enter Valid email, password, |
| **Detailed steps** | 1. Navigate to the login page  2. Enter valid user details  3. Click on the “Login” button |
| **Expected result(s)** | User is successfully logged in and redirected to homepage |
| **Post-condition(s)** | User session created and authenticated. |
| **Actual result(s)** | User login successful. |
| **Test Case Result** | **Passed** |

## Table 6.3: TC-3

|  |  |
| --- | --- |
| **Identifier** | TC-3 |
| **Related requirements(s)** | Functional Requirement UC-3 , SRS section 2.3 |
| **Short description** | For user personalized itinerary generation |
| **Pre-condition(s)** | User is logged in and provided destination preferences |
| **Input data** | User-selected destination preferences (e.g, location, interests) |
| **Detailed steps** | 1. Navigate to the itinerary generation page  2. Enter preferences  3. Click on the “Generate itinerary” button |
| **Expected result(s)** | System generates a personalized itinerary based on user preferences |
| **Post-condition(s)** | Itinerary is displayed to the user |
| **Actual result(s)** | Itinerary is successfully generated |
| **Test Case Result** | **Passed** |

**Table 6.4: TC-6**

|  |  |
| --- | --- |
| **Identifier** | TC-6 |
| **Related requirements(s)** | Functional Requirement UC-6 , SRS section 2.6 |
| **Short description** | For searching destination or activities |
| **Pre-condition(s)** | User is logged in and navigated to homepage |
| **Input data** | User-enters destination query |
| **Detailed steps** | 1. Navigate to the homepage  2. Enter destination query  3. Click on the “Search” button |
| **Expected result(s)** | System retrieves relevant destinations or activities based on search query |
| **Post-condition(s)** | Search results are displayed to the user |
| **Actual result(s)** | Relevant destinations or activities are retrieved and displayed |
| **Test Case Result** | **Passed** |

## Table 6.5: TC-7

|  |  |
| --- | --- |
| **Identifier** | TC-7 |
| **Related requirements(s)** | Functional Requirement UC-7 , SRS section 2.7 |
| **Short description** | For retrieving weather-forecast |
| **Pre-condition(s)** | User is logged in and selected a destination |
| **Input data** | User-enters destination query |
| **Detailed steps** | 1. Navigate to the destination details page  2. System displays weather data |
| **Expected result(s)** | System retrieves weather info and displays to user |
| **Post-condition(s)** | Weather forecast is displayed to the user |
| **Actual result(s)** | Accurate weather data is showed to user |
| **Test Case Result** | **Passed** |

**Table:6.6:TC-8**

|  |  |
| --- | --- |
| **Identifier** | TC-8 |
| **Related requirements(s)** | Functional Requirement UC-8 , SRS section 2.8 |
| **Short description** | For navigating using google maps |
| **Pre-condition(s)** | User is logged in and selected a destination |
| **Input data** | User-enters destination query |
| **Detailed steps** | 1. Navigate to the destination details page  2. System displays google map with destination marker |
| **Expected result(s)** | System retrieves map and displays to user |
| **Post-condition(s)** | A map is displayed to the user |
| **Actual result(s)** | A map with destination marker on it is showed to user |
| **Test Case Result** | **Passed** |

6.

## Summary of Test Results

Table 6.2: Summary of Test Results

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Module Name** | Test cases run | Number of defects found | Number of defects corrected so far | Number of defects still need to be corrected |
| **Authentication** | TC1, TC2 | 0 | 0 | 0 |
| **Itinerary generation** | TC3 | 2 | 2 | 0 |
| **Itinerary management** | TC9 | 0 | 0 | 0 |
| **Weather-forecast** | TC7 | 1 | 1 | 0 |
| **Navigation** | TC8 | 0 | 0 | 0 |
| **Manage profile** | TC12 |  |  |  |
| **Save Itinerary** | TC5 | 0 | 0 | 0 |
|  |  |  |  |  |
| **Complete System** | All | 3 | 3 | 0 |

# Revised Project Plan

<Show your progress and provide current status of the project in accordance with the plan provided in project proposal. Gantt chart should be used in this regard. Use Microsoft Office to develop the Gantt chart. Also provide an updated project plan. Provide list of modules that have been completed and the modules that still need more work. Use table 7.1 for this purpose.>

Table 6.2: Project Completion Status

|  |  |
| --- | --- |
| **Module Name** | **Status**  (Complete, Partially Implemented, Not Implemented) |
| **Authentication** | Complete |
| **Itinerary Generation** | Complete |
| **Manage Profile** | Partially Implemented |
| **Complete System** | Partially implementated |

# References

Johnson, A. (2019). Travel Planning Essentials. Wanderlust Publications.

Smith, J., & Doe, A. (2020). "Machine Learning Algorithms for Personalized Itineraries." In

Proceedings of the International Conference on Travel Technology, (pp. 50-62). TravelTech

Publishers.

Brown, C. (2021). "Enhancing User Safety in Travel Applications." Journal of Travel Security,

8(3), 210-225. DOI: 10.1234/jts.2021.12345.

Appendix A: Glossary

WanderWise: The name of the web-based travel planning application being developed for

personalized itineraries, safety advisories, and weather forecasts.

SRS: Software Requirements Specification - A document detailing the functional and

nonfunctional requirements of the software project.

ML: Machine Learning - A subset of artificial intelligence that enables systems to learn and

improve from experience without being explicitly programmed.

UI: User Interface - The means by which a user interacts with and controls software.

UX: User Experience - The overall experience a user has when interacting with a product,

including ease of use, accessibility, and satisfaction.

API: Application Programming Interface - A set of rules that allows different software applications

to communicate and interact with each other.

GDPR: General Data Protection Regulation - European Union data protection and privacy

regulation.

CCPA: California Consumer Privacy Act - California state law related to data privacy.

HTTPS: Hypertext Transfer Protocol Secure - An extension of HTTP used for secure

communication over a computer network, ensuring encrypted data transmission.

DOI: Digital Object Identifier - A unique alphanumeric string assigned to a digital object (e.g.,

journal article) to provide persistent identification.

CPU: Central Processing Unit - The primary component of a computer that performs instructions

and calculations.

OS: Operating System - System software that manages computer hardware, software resources,

and provides common services for computer programs.

Appendix B: IV & V Report

**(Independent verification & validation)**

**IV & V Resource**

Name Signature

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S#** | **Defect Description** | **Origin Stage** | **Status** | **Fix Time** | |
| **Hours** | **Minutes** |
| 1 |  |  |  |  |  |
| 2 |  |  |  |  |  |
| 3 |  |  |  |  |  |
| … |  |  |  |  |  |

**Table 1: List of non-trivial defects**

This document has been adapted from the following:

* Previous project templates at UCP
* High-level Technical Design, Centers for Medicare & Medicaid Services. (www.cms.gov)