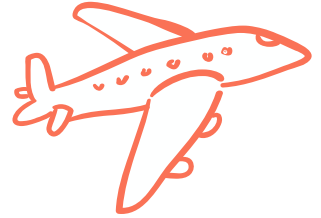# Wanderlust Travel Recommender
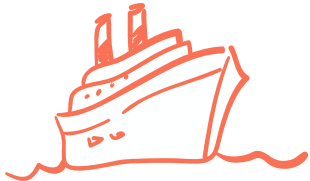
Sydney Divozzo, Cecile Darwiche, Michael Sampino

Quick Recap: The Problem

Conventional travel planning is time-consuming and overwhelming, often leading to information overload and difficulty in finding the right activities tailored to your interests in a new place.

Quick Recap: Our Solution - Wanderlust Travel Recommender

Efficiency

Save time with a user-friendly platform that simplifies and accelerates the planning process.

Exploration

Receive recommendations aligned with individual preferences for a more enjoyable travel experience.

Personalization

Discover new destinations and activities tailored to your interests.

System Overview

1

Three-Tier System

- **Client Tier:** Users interact through intuitive web and mobile interfaces.
- **Server Tier:** Application logic processes user data and generates recommendations.
- **Database Tier:** Ensures efficient data management and scalability.

2

Event-Driven Architecture

- Utilizes asynchronous communication for enhanced system responsiveness.
- Events like user interactions are handled in real-time, improving flexibility.

4

Key Benefits:

- Personalized travel recommendations.
- Scalability to accommodate diverse user preferences.
- Adaptable to evolving requirements through its flexible architecture.

Description of the System's Components
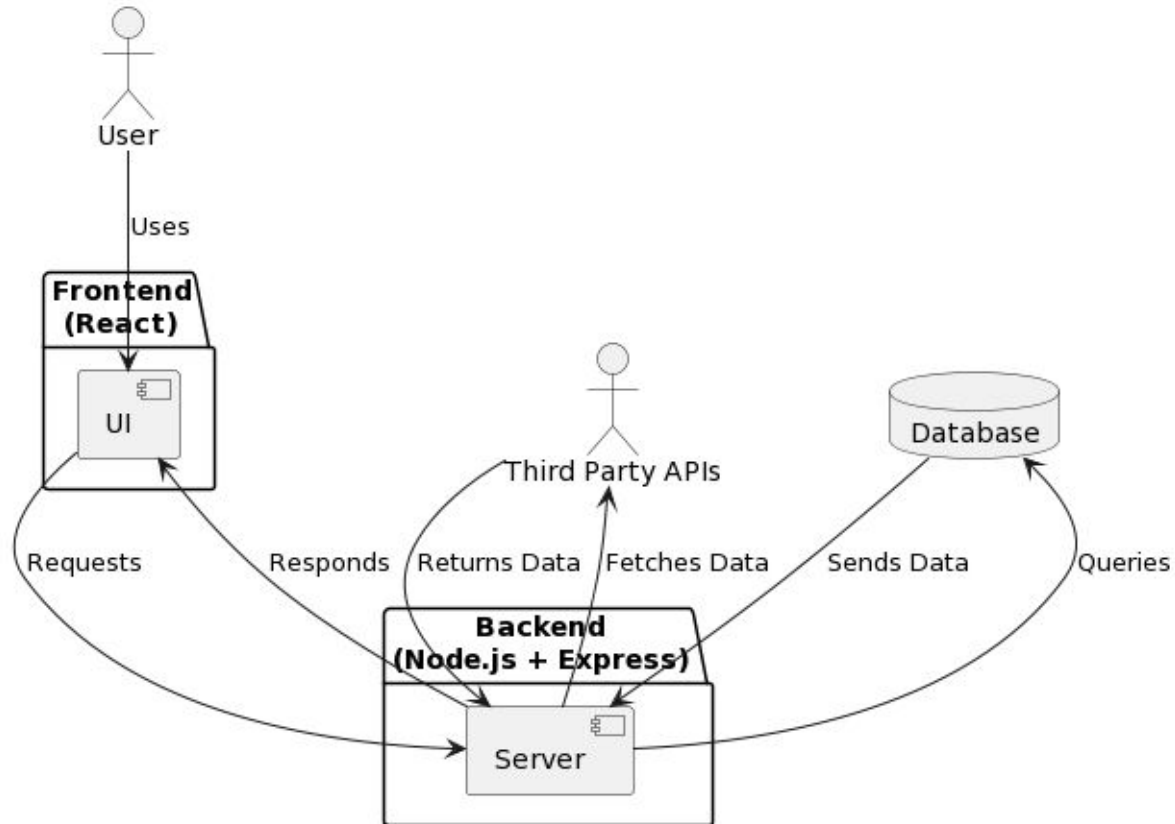
Actor Identification

New users	Individuals who wish to create an account.
Registered users	Individuals looking for personalized travel recommendations.
Administrators	System operators responsible for managing the application's backend and ensuring data integrity.

Architectural Style

Client Tier

Web and mobile applications used by travelers

Server Tier

Application logic and REST API controller

Database Tier

Storage and retrieval of travel-related data

Design Patterns

- Factory Method Pattern
  - Allows for the creation of complex objects
- Strategy Pattern
  - Enables interchangeable algorithms within the recommendation engine
- Facade Pattern
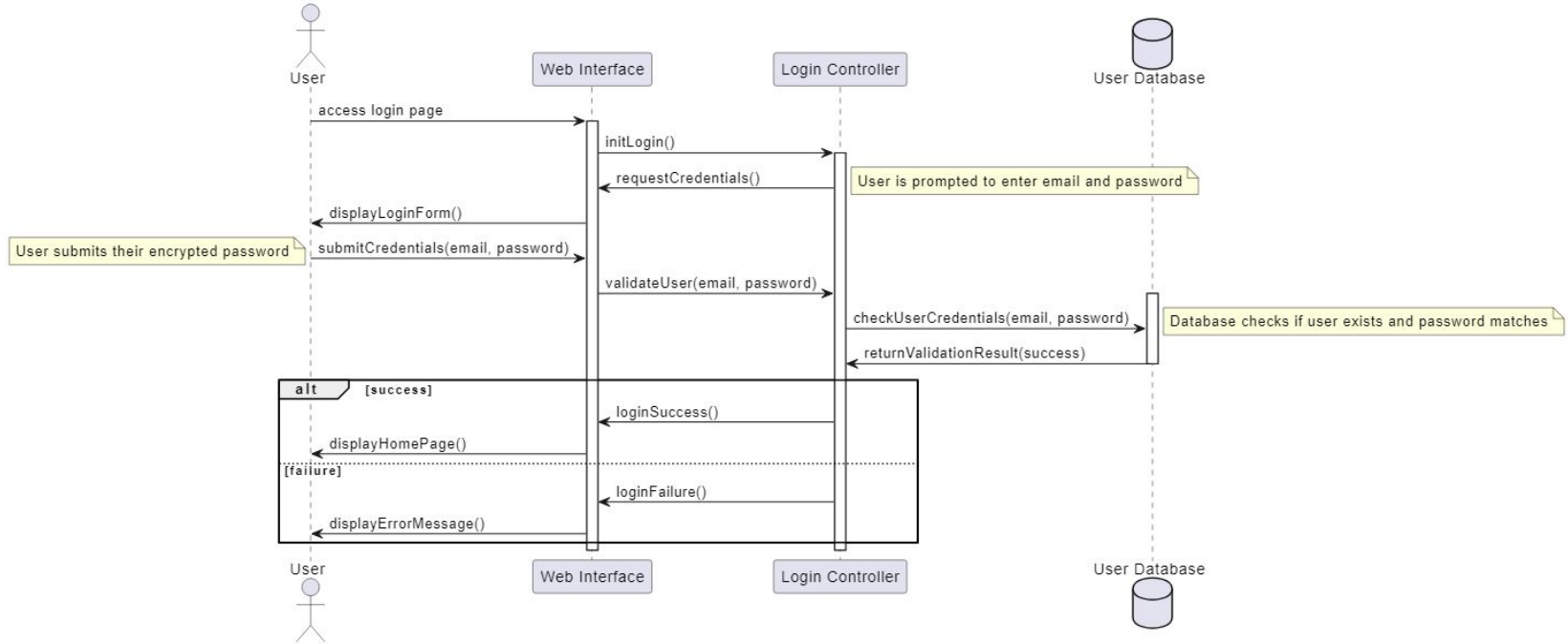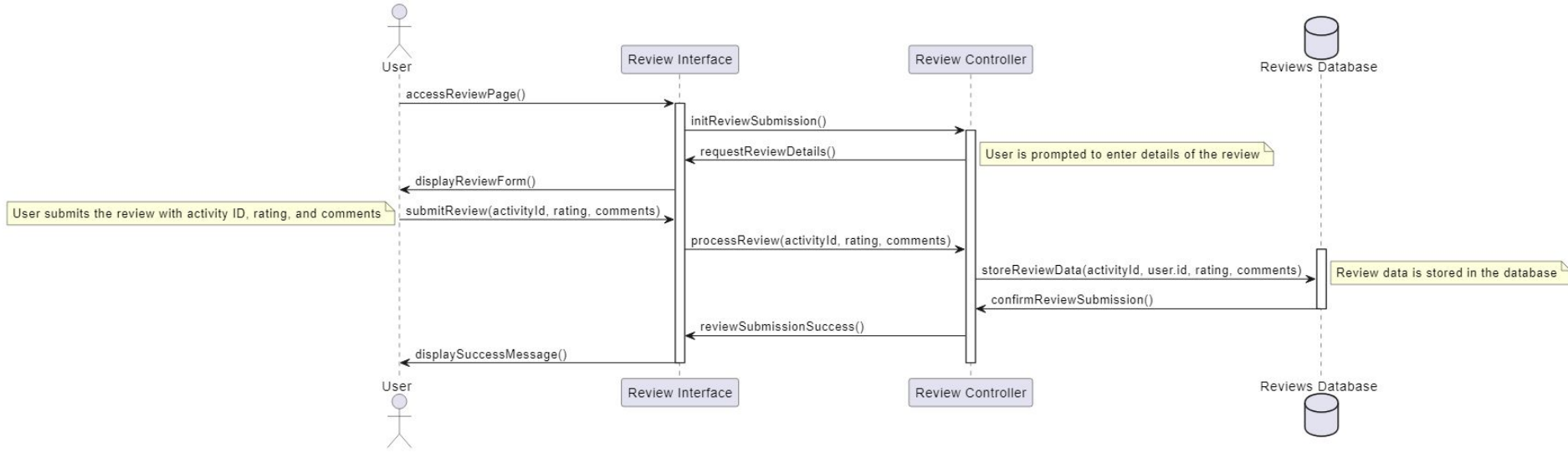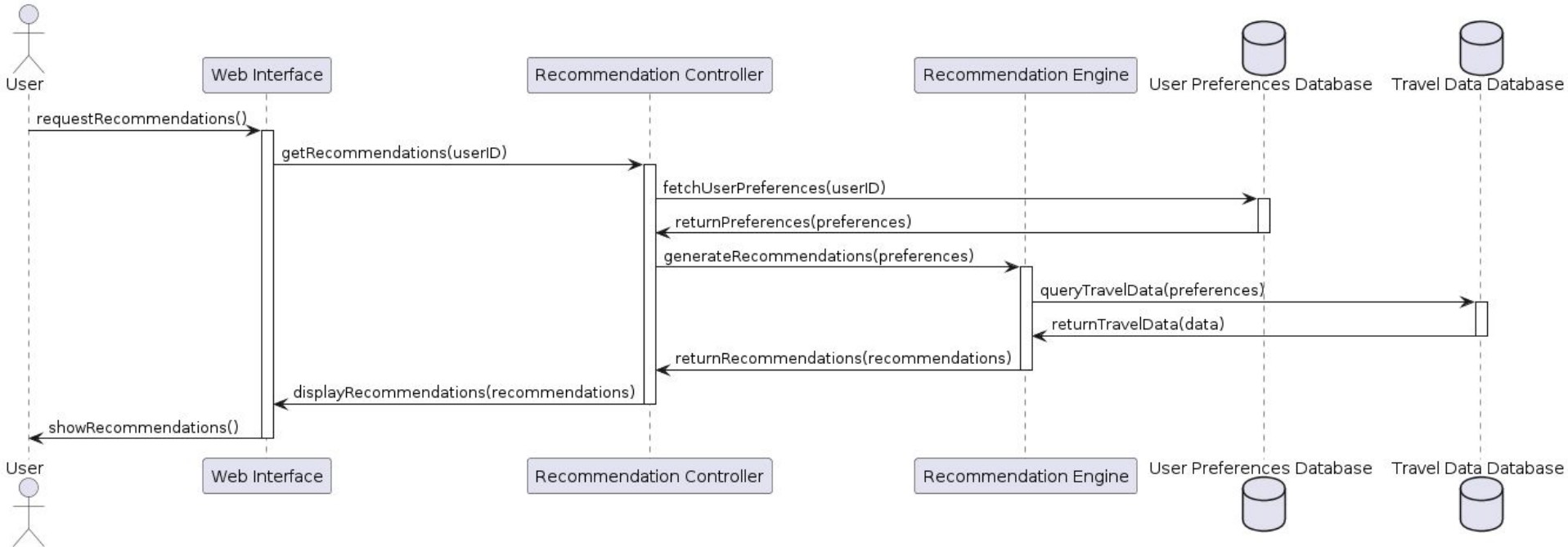 - Simplifies interactions with complex subsystems

Frameworks

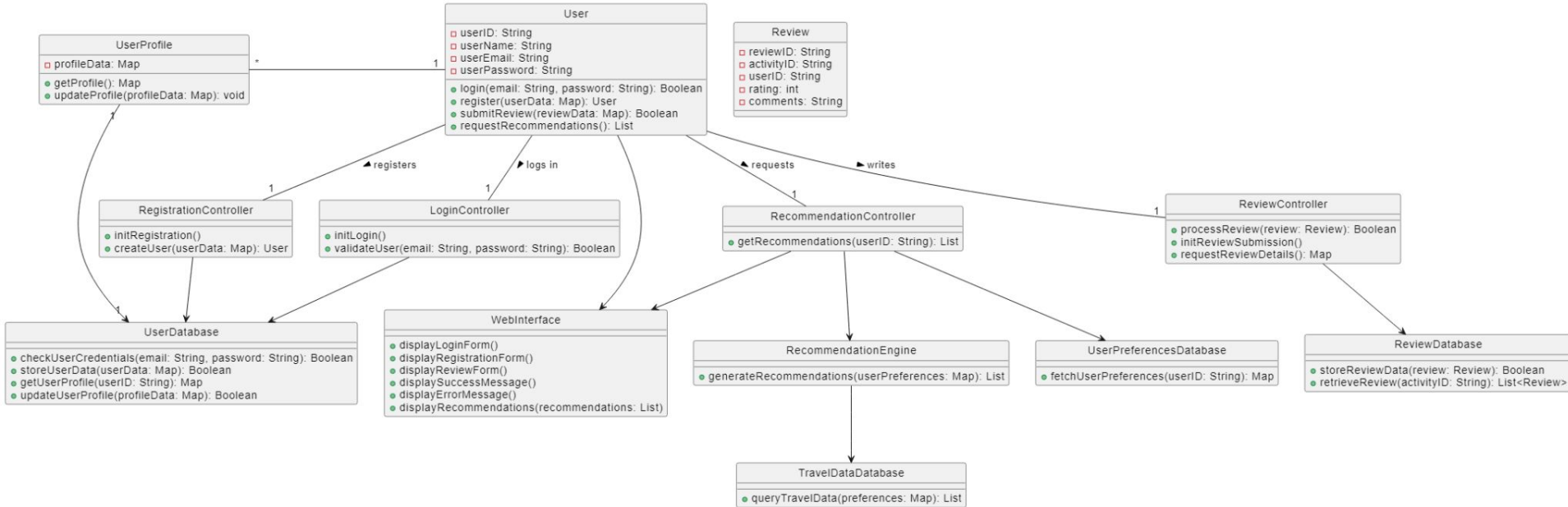# User Registration Sequence Diagram
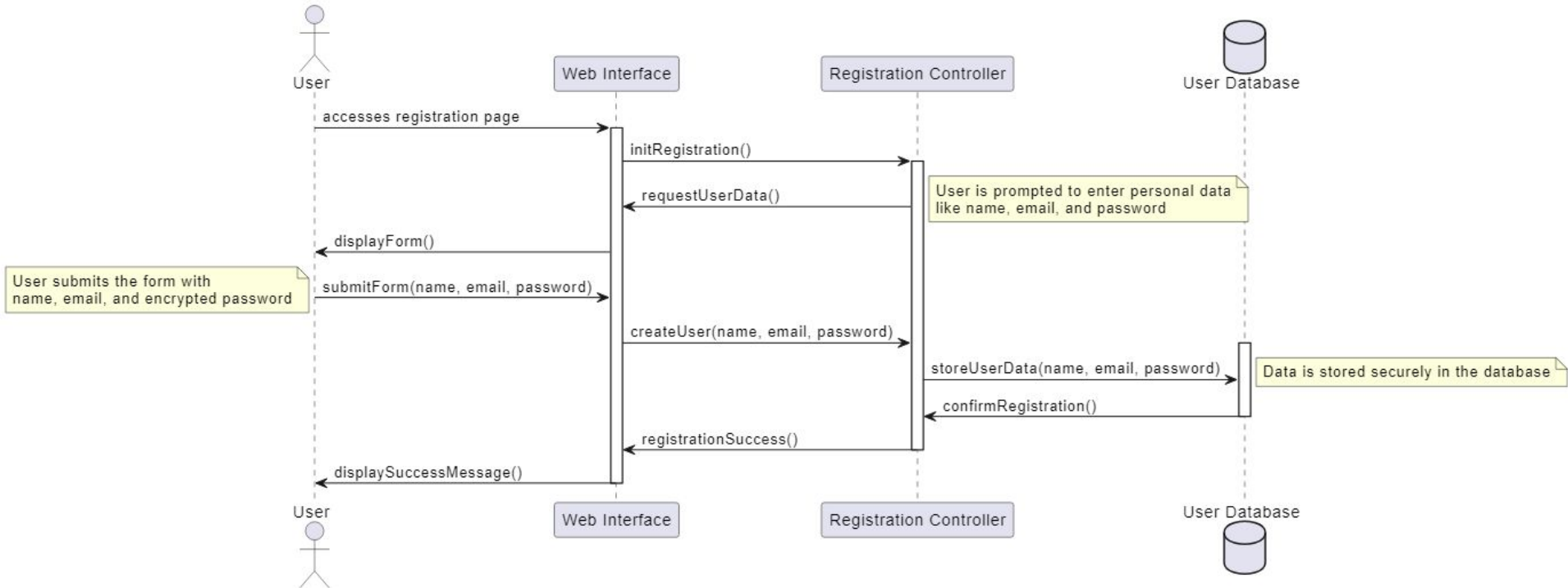
User Login Sequence Diagram

User — Web Interface — Login Controller — User Database

access login page
initLogin()
requestCredentials()

User is prompted to enter email and password

displayLoginForm()

User submits their encrypted password

submitCredentials(email, password)

validateUser(email, password)

checkUserCredentials(email, password)

Database checks if user exists and password matches

returnValidationResult(success)

alt [success]

loginSuccess()

displayHomePage()

[failure]

loginFailure()

displayErrorMessage()

Submit Review
Sequence Diagram

User

Review Interface

Review Controller

Reviews Database

accessReviewPage()

initReviewSubmission()

requestReviewDetails()

User is prompted to enter details of the review

displayReviewForm()

User submits the review with activity ID, rating, and comments

submitReview(activityId, rating, comments)

processReview(activityId, rating, comments)

storeReviewData(activityId, user.id, rating, comments)

Review data is stored in the database

confirmReviewSubmission()

reviewSubmissionSuccess()

displaySuccessMessage()

User

Review Interface

Review Controller

Reviews Database

Recommendation Engine Sequence Diagram

Class Diagram

Mockups

Wanderlust
TRAVEL RECOMMENDER

get started

Wanderlust

email

password

ready

Don't have an account?
Create Account

Wanderlust

name

age

gender

email

password

ready

Mockups

Thank you!
any questions?