CSC 431

Sandsational: Your Go-To Beach Guide

System Architecture Specification (SAS)

Team 2

Nicey Raiyani	Project Manager
Simay Pala	System Architect
Elizabeth Verela	Requirements Engineer

Version History

Version	Date	Author(s)	Change Comments
1.0	04/04/2023	Nicey Raiyani, Simay Pala, Elizabeth Verela	First Draft
2.0	05/02/2023	Nicey Raiyani, Simay Pala, Elizabeth Varela	Second Draft (Feedback Edits)
3.0	05/03/2023	Nicey Raiyani, Simay Pala, Elizabeth Varela	Third Draft (Updates to Diagrams)

Table of Contents

Table of Figures

1. System Analysis

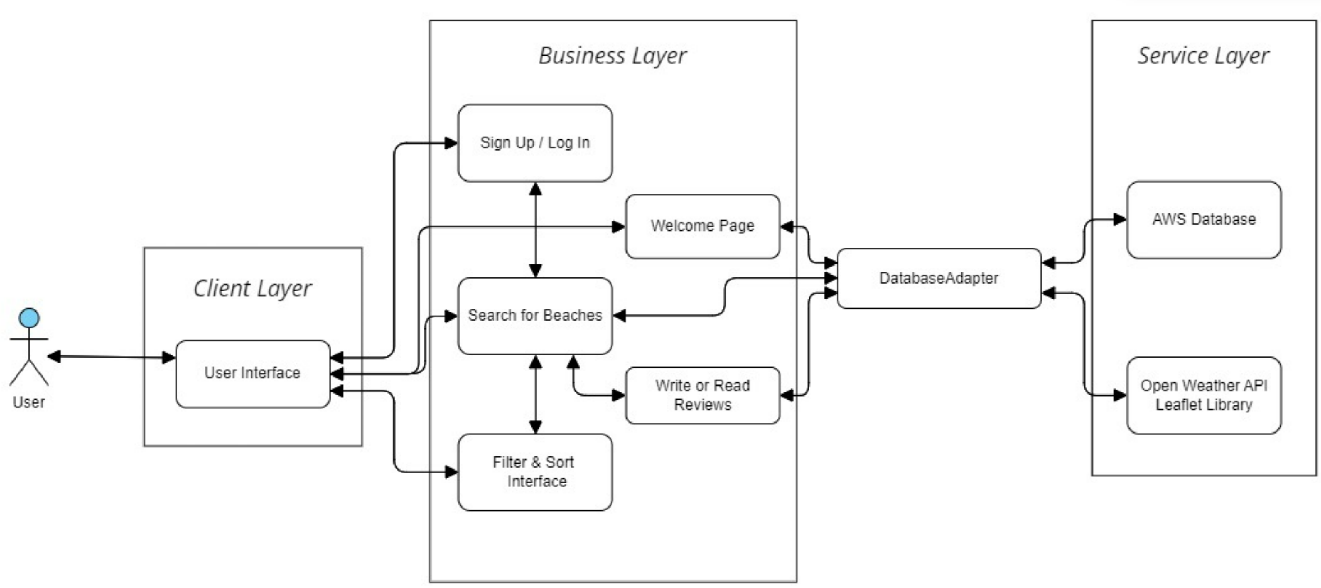
1.1. System Overview

Sandsational is a beach guide application designed to help users find the perfect beach for their preferences and needs. The application provides an easy-to-use interface for users to search, filter, and view beach information, including weather, water temperature, activities, and user reviews. Users can create an account or continue as a guest with limited features. The application is available as a web application and a mobile application for iOS and Android devices.

The interface will be built around four main parts: Beach Search, Beach Information, Beach Reviews, and Account. The Beach Search component is responsible for searching for beaches based on user input and applying filters to the search results. It interacts with the individual Beach Information to display information about each beach, such as its name, location, and available activities and also interacts with the Beach Reviews to display reviews for each beach. The Beach Information represents a beach and has attributes such as name, location, and available activities. It also has methods to retrieve information about the beach, such as its reviews and current weather conditions. The Beach Reviews component is responsible for managing user reviews of beaches, and the Account component is responsible for managing user accounts and storing user-specific information, such as saved reviews and favorite beaches.

The primary objective of the application is to provide an accessible and efficient platform for users to explore and discover beaches based on their preferences and requirements. The system will use data from external sources such as OpenWeather API and Leaflet Library to ensure accurate and up-to-date information. The application utilizes the Angular front-end framework, Ruby for the web application back-end, and Flutter for the mobile application back-end. Data storage and management will be handled using SQL databases and Azure Cloud Storage.

1.2. System Diagram



1.3. Actor Identification

- **Guest User:** The guest user doesn't have an account registered with Sandsational, they can only use the search bar.
- **Registered User:** Once the user registers an account with Sandsational, they have access to all features including searching for beaches, saving their favorite beaches, and providing feedback through reviews and ratings. The registered user has an account associated with a username, and a password they select during registration and can reuse to sign in every time afterwards.
- **Server(Administrator):** Responsible for managing and maintaining the application, including data updates, user management, and system improvements. Also responsible for importing beach information from the databases to be used for the searches.

1.4. Design Rationale

1.4.1. Architectural Style

For the "Sandsational" metasearch engine application, we have decided to use Service-Oriented Architecture (SOA) as the primary architectural style. With SOA, we can break down the application into individual services, each with specific responsibilities such as weather data retrieval, activity information collection, and user interface functionality. These services can communicate with each other to provide a comprehensive search experience for our users.

Our decision to use SOA for the "Sandsational" application was based on its ability to facilitate scalability and evolution. Individual services can be updated or replaced without impacting the entire system. Additionally, SOA enables easy integration with third-party services or data sources, which provides opportunities for expansion in the future.

1.4.2. Design Pattern(s)

- **Facade:** The Facade pattern could be used to simplify the complex interface for the developers. This could be particularly useful for the "Sandsational" application, which may have many different data sources and APIs.
- **Strategy:** The Strategy pattern could be used to allow the user to select different search strategies, such as "find the warmest beach" or "find the best surfing beach." This approach would provide flexibility and allow for easy changes to the search criteria.
- **Factory Method:** This pattern could be used to provide a standard interface for creating objects that implement a particular behavior, such as retrieving weather data or activity information. This approach would provide a flexible and extensible way to add new data sources to the application.

1.4.3. Framework

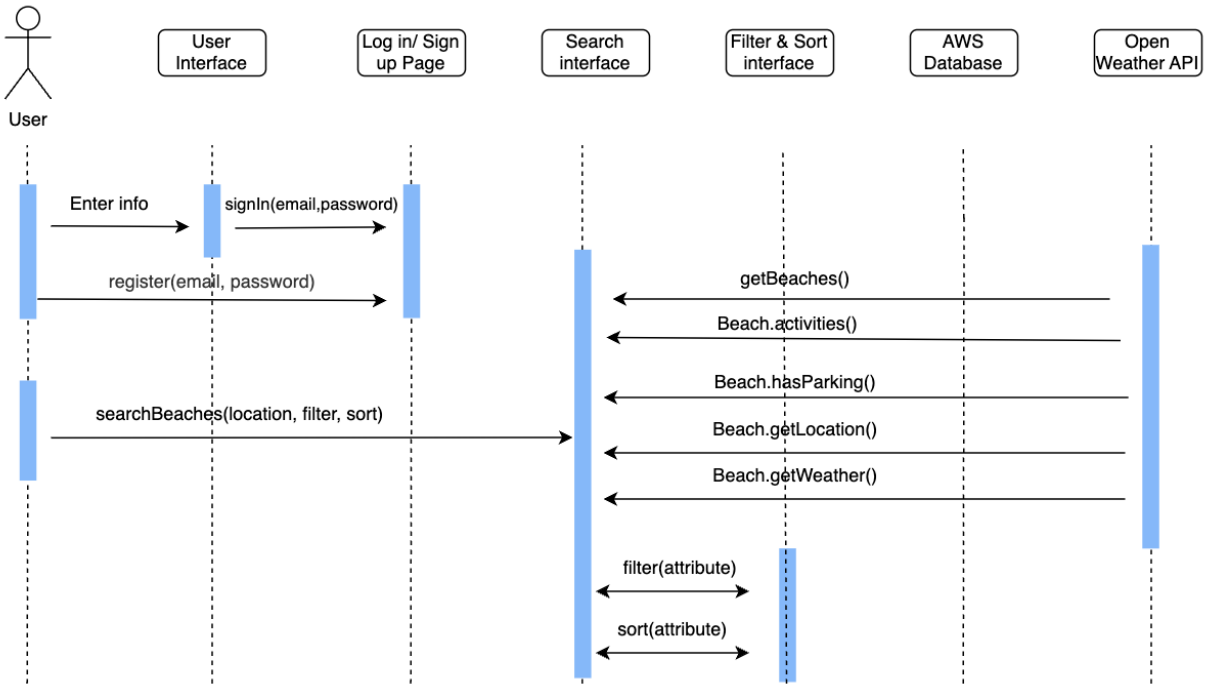
For our metasearch engine application, we have chosen to utilize Ruby on Rails for the backend and Flutter for the frontend development. Ruby on Rails is a widely used web development framework that provides an array of features such as routing, templating, and ORM. These features will be used to handle data from various sources, process it efficiently, and provide a high-performance search experience for users.

Flutter is an open-source mobile application development framework that enables developers to create high-quality, natively compiled applications for mobile, web, and desktop platforms from a single codebase. We will utilize Flutter to build the application's front end, providing an intuitive and user-friendly interface for our users.

To ensure high scalability, security, and reliability of our application, we have decided to host it on Amazon Web Services (AWS). AWS offers a comprehensive suite of cloud-based services that will assist us in handling traffic spikes, deploying the application quickly and efficiently, and providing high availability and security to our users.

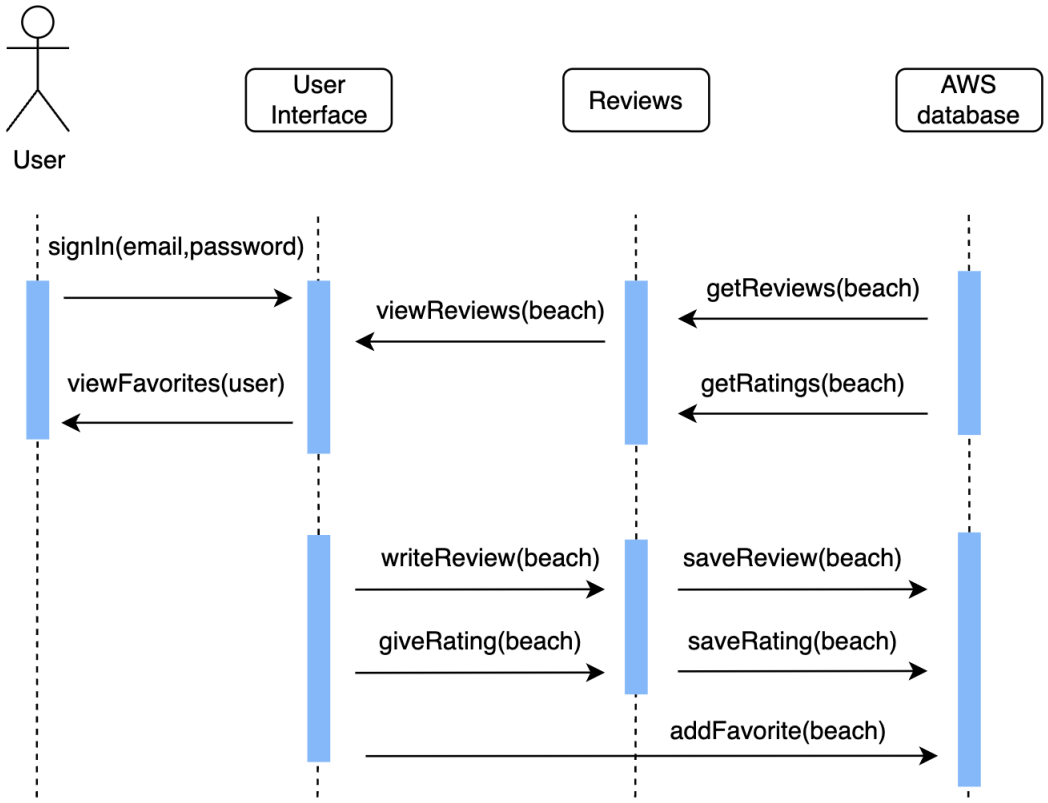
2. Functional Design

2.1. Searching Beaches with Filtering and Sorting



1. Registered user types in their login information to sign in.
2. New user can access the search without signing in, or choose to register a new account.
3. User can search for beaches by selecting the features(filters) they want and later choose to sort the beaches according to their desired attributes.
4. The search interface fetches the beach information along with the activities, parking details, and any other information saved about the beach from the Open Weather API database.
5. Search Interface interacts with the Filter & Sort interface to sort and filter the beaches according to the attributes chosen by the user.
6. The search interface then displays the list of beaches and information about them according to the filters applied and sorting preferences.

2.2. Viewing and Writing Reviews



1. User needs to sign in to write reviews.
2. A new user, however, can view other reviews without having to sign in.
3. Users can write reviews and ratings for beaches of their choice.
4. These reviews are then saved into the AWS database for further reference.
5. The reviews and ratings are fetched from the AWS database to be displayed to the user along with the beach that they are viewing.
6. User has the option to favorite beaches and then view a list of the beaches they have favorited.
7. The rating can also be used for the aforementioned sort functionality when users want to see their search results ranked from highest to lowest rated beaches or vice versa.

3. Structural Design