**Low-Level Architecture and Data Models**

**P10:Odysseum**

|  |  |
| --- | --- |
| **Student ID** | **Name** |
| **25100283** | **Muhammad Affan naved** |
| **25100225** | **Mohammad Haroon Khawaja** |
| **25100212** | **Shahrez Aezad** |
| **25100097** | **Pir M. Shahraiz Chishty** |
| **25100023** | **Luqman Aadil** |

|  |  |  |
| --- | --- | --- |
| **Content** | **Totals** | **Obtained** |
| Architecture diagram | 20 | 10 |
| Architecture justification | 20 | 10 |
| E/R diagram | 20 | 18 |
| E/R diagram description | 20 | 17 |
| Tools and Technologies | 10 | 10 |
| Who did what | 3 | 3 |
| Review checklist | 2 | 2 |
| Overall formatting/template | 5 | 5 |
| GitHub folder structure penalty | -15 | - |
| Late submission penalty | -20 | - |
| **Grand Total** | **100** | **75** |
| **General Comments/Individual Grading:** Update as the high level architecture. Don’t show infrastructure as a layer. | | |

**Table of Contents**

[1.](#_gjdgxs) Introduction 3

[2.](#_30j0zll) System Architecture 4

[2.1 Architecture Diagram—](#_1fob9te)As it is in the prototype code 4

[2.2 Architecture Diagram—](#_3znysh7)As it should-be 4

[3.](#_2et92p0) Data Models 5

[4.](#_tyjcwt) Tools and Technologies 6

[5.](#_3dy6vkm) Who Did What? 7

[6.](#_1t3h5sf) Review checklist 7

# Introduction

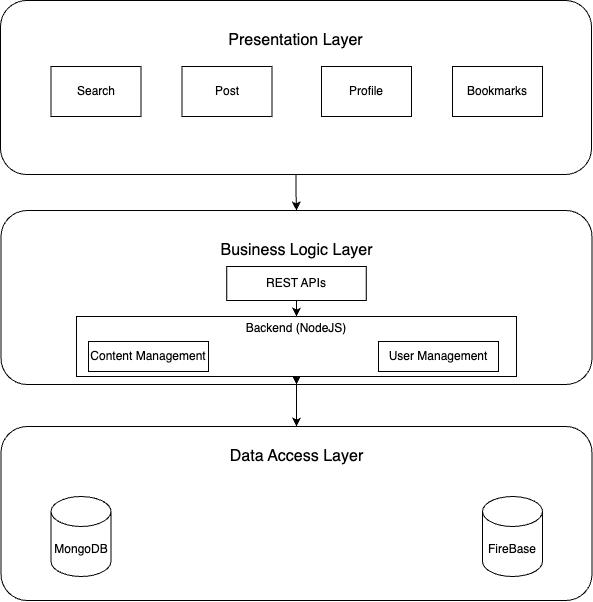
This project aims to develop a travel/social network application to help travelers plan their next trip using just one app. Rather than relying on blogs and pages from different online outlets, the app would be a one-stop solution for all travelers. The application will provide users a platform to search for various tourist destinations they may be interested in visiting and what these destinations offer, such as accommodation, sightseeing, dining, nightlife, historical sites, and tour guides. Combining all these services onto one platform would improve the travel experience and allow users to make well-informed decisions based on destination information.

Nowadays, travelers face a fundamental problem: finding accurate and relevant information. They have to rely on large commercial travel agencies that only have profit-driven goals or on personal connections that provide limited details and advice. This gives travelers an experience far from fulfilling, while local businesses gain limited benefits. The purpose of this app is to serve as a networking app to connect like-minded travelers and local service providers such that both parties benefit, with travelers having a fulfilling experience visiting their destinations and the local business being allowed to boost the economic growth in the region.

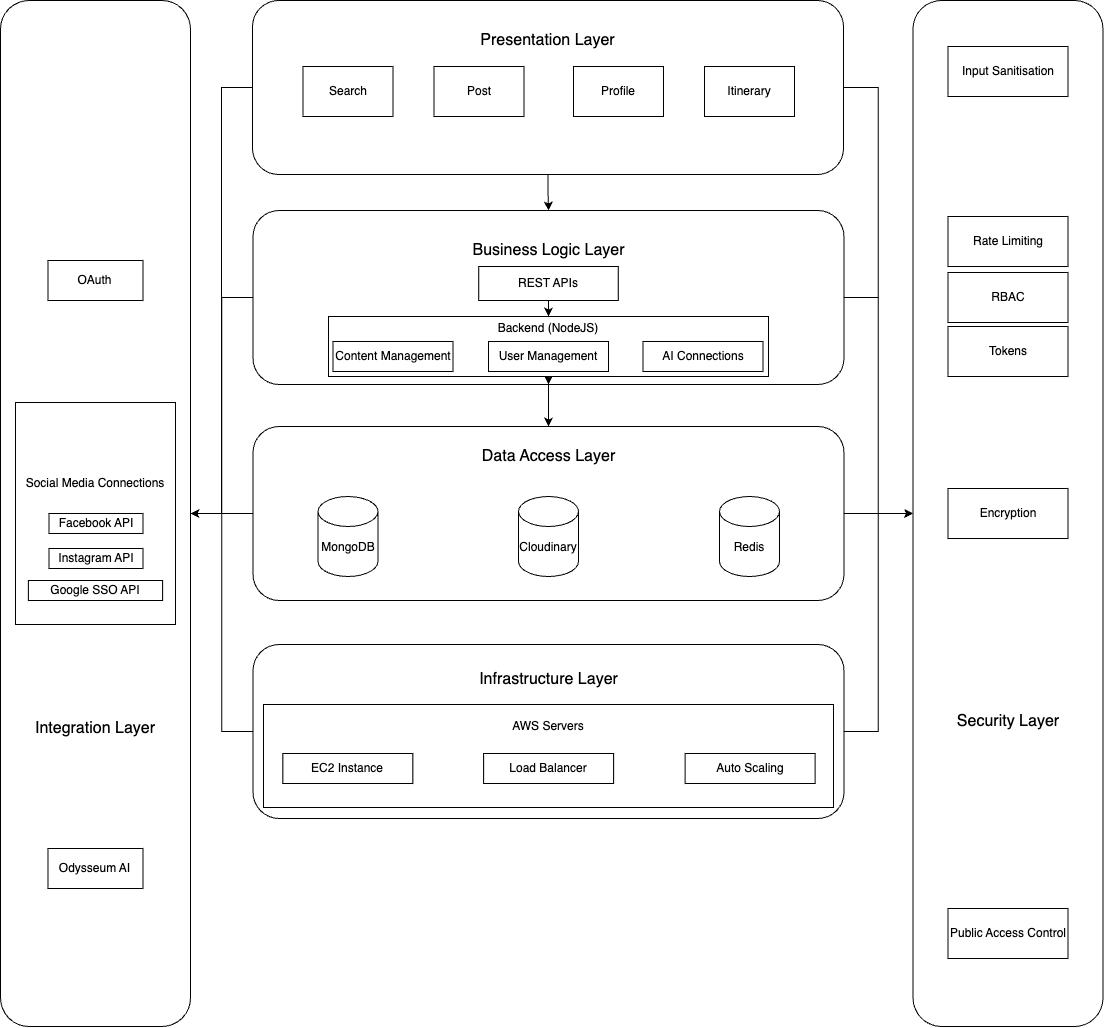
As stated above, potential users of this app include travelers themselves, administrators and local businesses which are but not limited to hotels, restaurants, and tour guides.

# System Architecture

## Architecture Diagram—As it is in the prototype code



## Architecture Diagram—As it should-be



**How the architecture archives design goals**

1. **Separate Layers:**

Each layer has a distinct, individual role, such as the presentation layer that handles user interaction and the business logic layer that handles managing core functionality such as user requests. The distinct layers separate the responsibilities of the system, making the system more maintainable. This would allow our team to work on different layers without affecting the other components.

1. **Server Scalability:**

The architecture will support horizontal scaling due to the infrastructure layer where all the backend components are deployed. This will ensure that the system can efficiently handle large amounts of traffic.

1. **Security:**

The security layer, along with its components that include JWT tokens, encryption, role-based access, rate limiting, etc, ensures that the sensitive data within the databases are protected and access is controlled. Potential security breaches will also be minimized due to the security layer.

1. **Extensibility and Integration:**

The integration layer enables easy interaction with third-party APIs and external services. This makes it easier to add new features, such as social media logins without affecting the rest of the system.

1. **Fault Tolerance and Availability:**

The use of load balancers in our infrastructure ensures that the backend servers distribute services across multiple layers. If one server goes down, the others can contribute by handling its requests and improving server availability.

**Classes**

* + 1. **Presentation Layer**

The presentation layer is responsible for the user interface and user experience of the social media app. It includes the components that users interact with directly, such as forms, buttons, navigation, and visual content.

* Components
  + They include but are not limited to, searching for locations and user profiles, creating posts and itineraries.
    1. **Business Logic Layer**

The business logic layer contains the core application logic that processes user requests, manages data flow, etc. This layer is supposed to handle all server-side operations and ensure that the application behaves as intended.

* Components
  + Backend: Contains the main express server which is implemented using NodeJS and express routing to provide REST APIs for requests from the front end.
  + Each API endpoint allows the server to send, receive, delete or update data based on the request type (whether it was to make changes to a user or content)
    1. **Data Access Layer**

The data access layer manages data storage and retrieval. It abstracts the underlying database systems, providing a clean interface for the business logic layer to interact with data.

* Components
  + Database: This will be a NoSQL database (MongoDB) that will store all user data, from login credentials to posts and recommendations.
  + Image/Video Storage: Since this is a social media app, we will use Firebase to store all images or videos users upload to make a post.
  + Cache System: Optionally, Redis may be used for caching frequently requested data that does not change (e.g. fetch user profile, comments, message) in order to reduce unnecessary calls to the database.
    1. **Integration Layer**

The integration layer facilitates interactions with external services and APIs. This layer manages data exchange between the social media app and third-party systems, such as authentication providers like Google.

* Components
  + The only components here would be the third-party providers whose APIs and other services may be used. These include Google OAuth, connecting the account to other social media apps etc.
    1. **Security Layer**

The security layer encompasses mechanisms that protect data and ensure secure access to application resources. This layer ensures that only authorized users can access certain functionalities and that data is protected during transmission.

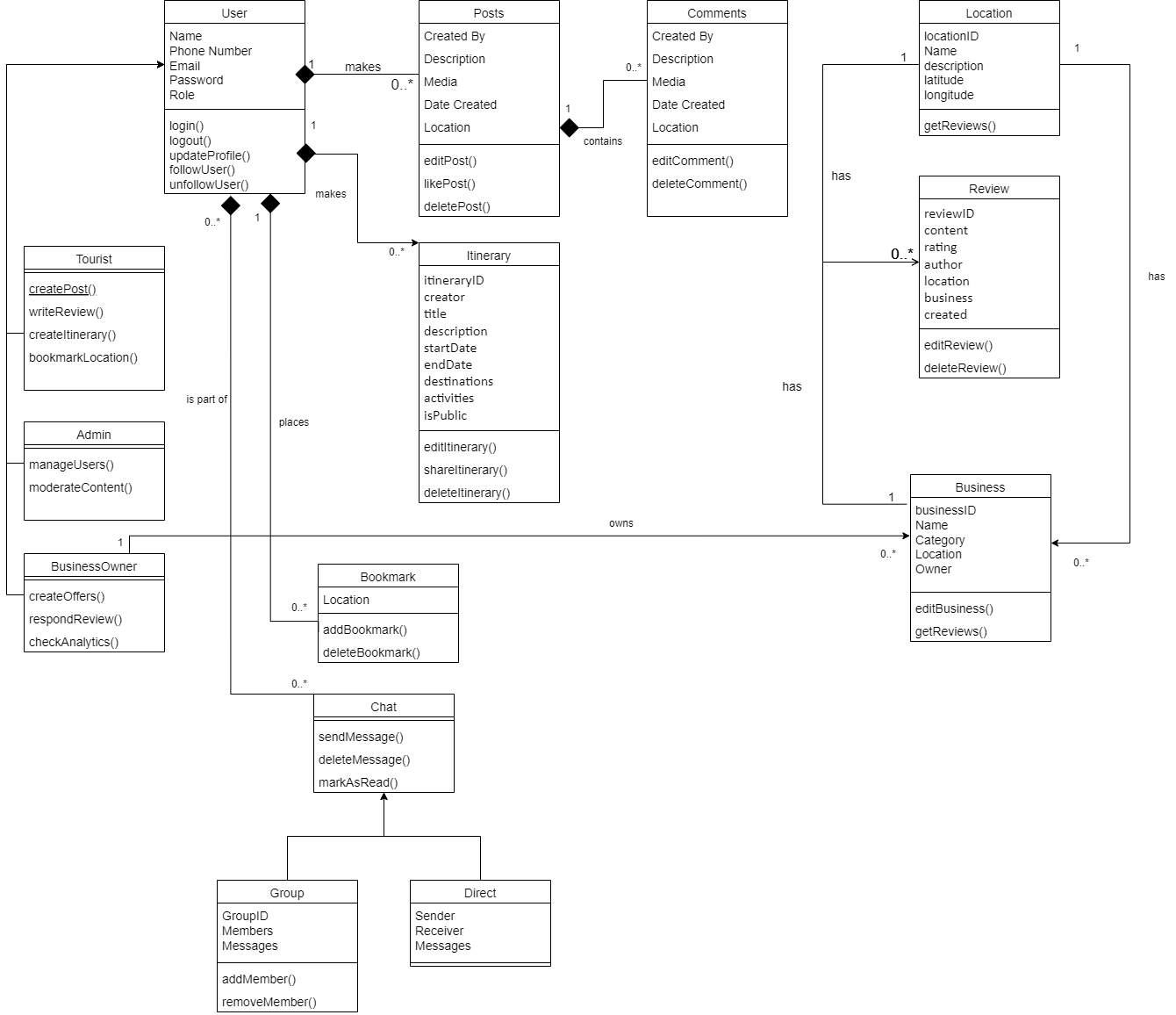
* Components
  + JWT Tokens: Used for stateless authentication, allowing users to access protected routes after successfully logging in.
  + Encryption: Ensures sensitive data (e.g. passwords) is stored securely and is only accessible to authorized users.
  + Role-Based Access Control: Ensure that only certain users are allowed to access sensitive functionalities of the backend system. (e.g. admins)
  + Input Validation and Sanitization: Ensure all user inputs are validated to prevent injection attacks and strip out harmful code or scripts from inputs to avoid Cross-Site Scripting (XSS) attacks.
  + Rate Limiting: Prevent brute force attacks by limiting the number of API requests from a single IP or user

**1.1.6 Infrastructure Layer**

The infrastructure layer represents the environment where all backend components are deployed. It encompasses the physical and virtual resources required to run the application, including servers, load balancers, and databases.

* Components
  + Load Balancer: Distributes incoming traffic across multiple backend servers to ensure optimal resource utilization and availability.
  + EC2: This is a cloud compute service provided by amazon which will host the backend.
  + Backend Servers: These servers host the backend services, which can be scaled horizontally based on traffic demand (auto-scaling).

# Data Models



**Description**

1. User

This model holds all user data. It consists of three types of users

* 1. Tourist: The main user who will navigate the app to look for locations and businesses and review them when they visit.
  2. Business Owner: These users will advertise their business in certain locations to incoming tourists.
  3. Admin: They hold special privileges and can manipulate both tourist and business owner data including their accounts and posts.

1. Posts

This model contains the data for a post which a tourist or a business owner can make. It can contain images or videos too.

1. Comment

This model contains all the comments made by different users under a post.

1. Location

This model contains all popular tourist locations for different people to view and add to their itineraries.

1. Review

This model will store data containing reviews for a location or a business so that other users can view and make well informed decisions on whether a certain place is worth visiting.

1. Business

This model contains data of a specific business including its locations, owner, description, reviews etc.

1. Chat

This model will hold all chat interactions between different users who are communicating directly or in a group. It is divided into two models:

1. Group: Two people can chat with one another directly.
2. Direct: 3 or more people can chat together.
3. Bookmark

This model contains data of a location which has been bookmarked by a user for future reference.

1. Itinerary

This model stores data related to a user created travel itinerary so that they may share it with other users.

# Tools and Technologies

**Frontend**:

* React Native (Version: [0.75])

**Backend**:

* Node.js (Version: [v22.9.0])

**Database**:

* MongoDB (Version: [8.0])
* Firebase (Version: [10.14.1])

**Deployment/Cloud Hosting**:

* AWS or Azure (AWS EC2, S3, or Azure App Service)

**APIs**:

* Integration with various third-party APIs (e.g., Google Maps API, weather services)

**Version Control**:

* GitHub for code repository and collaboration

**Code Editor/IDE**:

* Visual Studio Code (Version: [1.94.2])
* Postman (API testing) (Version: [v11.16])
* Expo Go (App Testing) (Version: SDK 51)

**Package Management:**

* NPM(node package manager) (Version: [10.9.0])

# Who Did What?

|  |  |
| --- | --- |
| **Name of the Team Member** | **Tasks done** |
| Muhammad Affan naved | Section 3 |
| Mohammad Haroon Khawaja | Section 3 |
| Shahrez Aezad | Section 2 |
| Pir M. Shahraiz Chishty | Section 4, Section 1 |
| Luqman Aadil | Section 4, Section 1 |

# Review checklist

Before submission of this deliverable, the team must perform an internal review. Each team member will review one or more sections of the deliverable.

|  |  |
| --- | --- |
| **Section** **Title** | **Reviewer Name(s)** |
| 2,3 | Pir M. Shahraiz Chishty |
| 2 | Shahrez Aezad |
| 2,4 | Muhammad Affan Naved |
| 1,4 | Mohammad Haroon Khawaja |
| 2,3 | Luqman Aadil |