**High level Architecture**

**<P04>:<MANZIL>**

|  |  |
| --- | --- |
| **Student ID** | **Name** |
| **25100235** | **Shahrez Faisal** |
| **25100320** | **Muhammad Usman Arshid** |
| **25100015** | **Omar ibne sajjad** |
| **24100199** | **Umer Inayat** |
| **25100313** | **Muhammad Mehdi** |

|  |  |  |
| --- | --- | --- |
| **Content** | **Totals** | **Obtained** |
| Architecture diagram | 30 | 15 |
| Architecture description | 20 | 12 |
| Architecture justification | 20 | 15 |
| Tools and Technologies | 10 | 10 |
| Hardware Requirements | 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** | **72** |
| **General Comments/Individual Grading:** Update architecture as discussed in the presentation. How components are scalable. Is there any components causing bottleneck? Show external servers/API calls.  Updated is very simple which is not required | | |

**Table of Contents**

[1. Introduction 3](#_Toc179223071)

[2. System Architecture 4](#_Toc179223072)

[2.1 Architecture Diagram 4](#_Toc179223073)

[2.2 Architecture Description 5](#_Toc179223074)

[2.3 Justification of the Architecture 6](#_Toc179223075)

[3. Tools and Technologies 7](#_Toc179223076)

[4. Hardware Requirements 8](#_Toc179223077)

[5. Who Did What? 9](#_Toc179223078)

[6. Review checklist 9](#_Toc179223079)

# Introduction

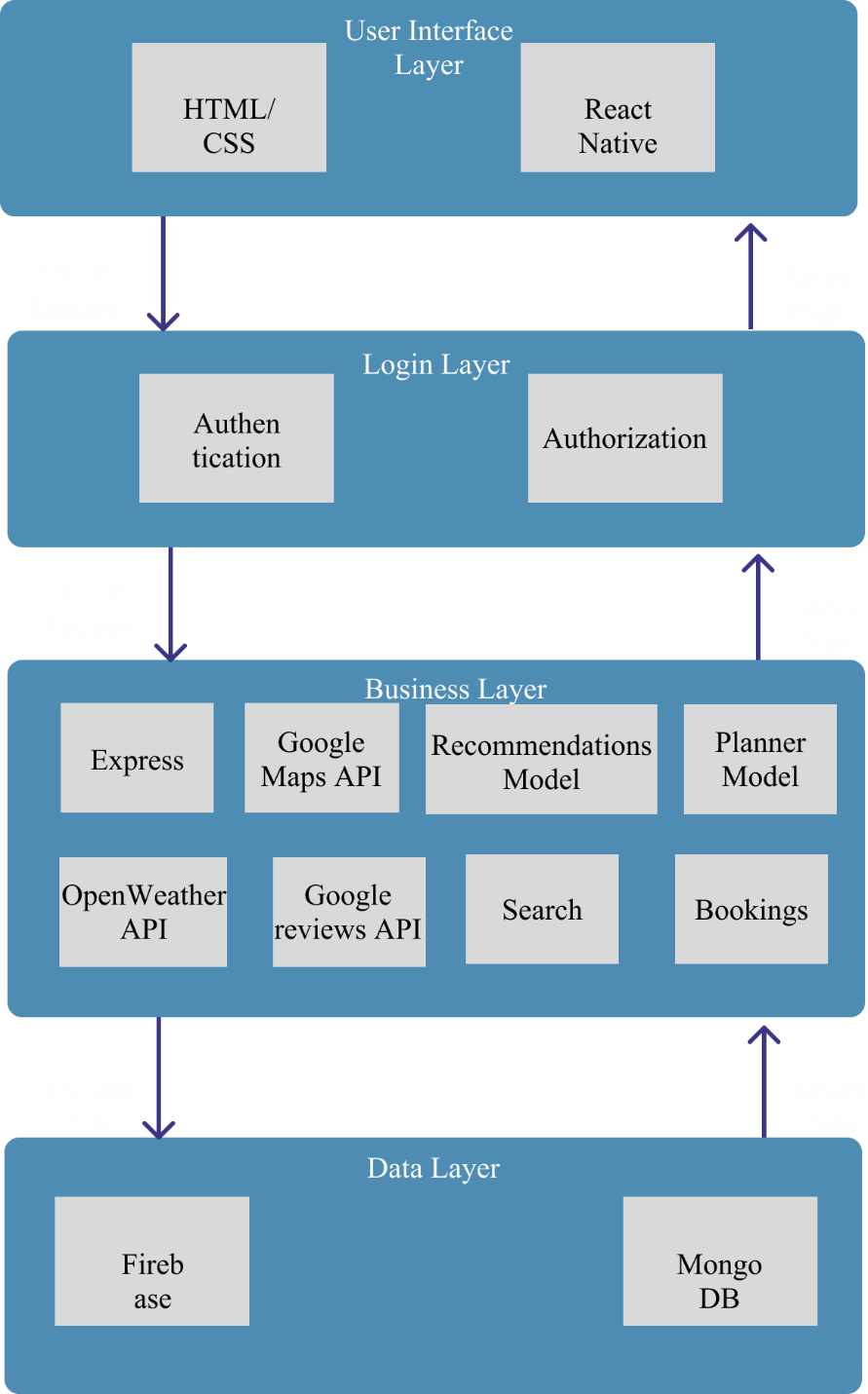
**Manzil** is a mobile application designed to assist users traveling to northern destinations and major cities in Pakistan. It aims to streamline the process by offering a one-stop solution for booking hotels, renting vehicles, checking weather updates, and finding local services and their reviews for better choice of places, such as restaurants, schools, and hospitals.

The app leverages machine learning (ML) to provide personalized recommendations and supports real-time services like live weather updates and itinerary management. The primary users include travelers, hotel management, and app administrators.

# System Architecture

## Architecture Diagram

<Draw a diagram of the system architecture. Remember that your system’s architecture may be defined using multiple architectural patterns.>



## Architecture Description

The system architecture for **Manzil** consists of the following key components:

1. **Mobile Application (User Interface):**
   * The front-end mobile app is built using **React Native** for both iOS and Android platforms. It interacts with the backend through REST APIs, enabling users to search for locations, book services, and get personalized recommendations.
2. **Backend Services (API Layer):**
   * The backend is hosted on **AWS EC2 instances**, using **Node.js** with **Express.js** for REST API implementation. Express handles all API requests related to hotel bookings, weather updates, and ML-based suggestions.
   * The backend communicates with the database through **Mongoose**, an ODM (Object Data Modeling) library, to manage MongoDB interactions efficiently.
3. **Database:**
   * A **MongoDB** database stores user data, booking history, reviews, hotel information, and other related information. Secure access is ensured via encrypted connections using **SSL/TLS**.
4. **Machine Learning Models:**
   * **Machine Learning models** are integrated into the backend, providing AI-based recommendations for hotels, restaurants, and tourist destinations. These models are trained using historical user interactions and preferences to deliver personalized results.
5. **External Services:**
   * **External APIs** such as Google Maps, OpenWeather, and others provide live weather updates, location data, traffic information, and other real-time services.
6. **Interaction Between Subsystems:**

* The **mobile app** sends requests to the backend via **REST APIs**. The backend, powered by **Node.js** and **Express.js**, processes these requests, communicates with the **MongoDB** database via **Mongoose**, retrieves or updates the data, and returns the response to the mobile app.
* **Machine learning models** are triggered based on user interactions to provide personalized recommendations.
* **External APIs** are integrated into the backend to fetch real-time data like weather and traffic updates, which are then sent back to the mobile app for display to the user.

## Justification of the Architecture

**Pros:**

* **Scalability:** The architecture allows scaling of backend services to accommodate more users without affecting performance.
* **Modularity:** Separate layers for the frontend, backend, and database ensure that individual components can be maintained and updated independently.
* **AI Integration:** The architecture supports easy integration of machine learning models, enhancing user experience.
* **Security:** With secure API endpoints, encrypted data storage, and role-based access control, the architecture ensures data privacy.

**Cons:**

* **Complexity:** The use of multiple technologies and seperate layers can lead to higher maintenance overhead.
* **Latency:** External APIs (for weather, traffic) may introduce latency if the response times are slow.

**Justification:**

This architecture is appropriate because it ensures that **Manzil** can handle high loads (e.g., concurrent bookings), provide real-time information, and meet non-functional requirements like availability and performance. By leveraging cloud infrastructure (AWS), we can ensure scalability and fault tolerance. Machine learning models help the app offer personalized experiences, which are crucial for enhancing user satisfaction.

# Tools and Technologies

* **React Native** (v0.71): For building cross-platform mobile applications.
* **Express** (v4.x): For developing backend REST APIs.
* **Node.js** (v18.x): As the runtime environment for the backend.
* **MongoDB** (v6.x): For database management.
* **Mongoose** (v6.x): For managing MongoDB through object modeling.
* **Firebase:** For real-time database and authentication services.
* **AWS EC2:** Hosting backend services.
* **TensorFlow** (v2.10): For implementing ML models.
* **Google Maps API**: For map services and location-based features.
* **Google Reviews API**: For review services of different places (hotels, restaurants, etc.).
* **OpenWeather API:** For live weather updates.
* **GitHub:** For version control and collaboration.
* **GitHub Boards:** For project management and issue tracking.

# Hardware Requirements

**Development Machines:**

1. **Processor**: Intel Core i5 or higher / AMD Ryzen 5 or higher (Quad-core or above)
2. **RAM**: 8 GB (to handle multiple development environments, virtual machines, and tools)
3. **Storage**: 512 GB SSD (for fast read/write speeds during development)
4. **Graphics**: Integrated or dedicated GPU (optional, required for ML model training and testing)
5. **Operating System**: Windows 10/11, macOS, or Linux (Ubuntu 20.04 or higher)
6. **Internet Connection**: High-speed internet (for access to cloud services, repositories, APIs, etc.)
7. **Development Tools**: Node.js, Express, React Native, MongoDB, Firebase, Git, Visual Studio Code or any preferred IDE.

**Deployment Servers:**

1. **AWS EC2 Instance**:
   * **Type**: t2.micro (for free tier) or t3.medium (for production use, capable of handling backend requests)
   * **vCPU**: 2 vCPUs (scalable based on traffic)
   * **RAM**: 4 GB (for smooth operation of backend services)
   * **Storage**: 50-100 GB EBS (Elastic Block Storage) for application deployment, logs, and temporary files
   * **OS**: Ubuntu 20.04 LTS or Amazon Linux 2 (recommended for deployment)
2. **MongoDB Atlas** (Cloud Database):
   * **Cluster**: Free-tier or M10 cluster (for production-level deployment)
   * **Storage**: 10-20 GB (scalable based on data growth)
3. **Firebase**:
   * **Services**: For real-time database and authentication, integrated with Node.js backend
4. **Load Balancer**: AWS Elastic Load Balancer (optional, for auto-scaling to handle high traffic)
5. **Backup and Recovery**: Cloud storage (S3 or equivalent) for periodic backups of data and logs.

These hardware configurations ensure smooth development and deployment of the "Manzil" app across various environments.

# Who Did What?

|  |  |
| --- | --- |
| **Name of the Team Member** | **Tasks done** |
| Shahrez Faisal | 3 |
| Muhammad Usman Arshid | 1, 2.3 |
| Omar Ibne Sajjad | 4 (hardware requirements) |
| Umer Inayat Khan | 2.1 |
| Muhammad Mehdi | 2.2 |

# Review checklist

|  |  |
| --- | --- |
| **Section** **Title** | **Reviewer Name(s)** |
| 3. Tools and Technologies | Muhammad Usman Arshid |
| 4. Hardware Requirements | Shahrez Faisal |
| 1. Introduction | Umer Inayat |
| 2. System Architecture | Omar Ibne Sajjad |