***Requirement Gathering and Analysis Phase***
***Solution Architecture***

| | |
|---|---|
| *Date* | *05july2024* |
| *Team ID* | SWTID1720162063 |
| *Project Name* | *Project - Cab Booking App* |
| *Maximum Marks* | |

## Overview

The RideEase cab booking app aims to provide an efficient, user-friendly platform for booking cab rides, managing driver information, and ensuring secure and reliable transactions. The solution architecture will leverage the MERN stack (MongoDB, Express.js, React.js, Node.js) to build a scalable and robust application.

## Components

1. **User Interface**: React.js-based web application providing an intuitive and responsive user experience.
2. **Web Server**: Node.js with Express.js handling server-side logic and API requests.
3. **API Gateway**: Manages client requests and routes them to appropriate services.
4. **Authentication Service**: Ensures secure user authentication and authorization using JWT.
5. **Database**: MongoDB for storing user data, ride details, and other critical information.
6. **Cloud Database**: MongoDB Atlas for scalable and reliable cloud storage.
7. **File Storage**: AWS S3 for storing user profile images and ride history.
8. **External APIs**:
    - Payment Gateway: Stripe and PayPal for processing payments.
    - Notification Services: Twilio and SendGrid for email and SMS notifications.
    - Map Services: Google Maps and Mapbox for location and navigation.
9. **Machine Learning Models** (Optional): TensorFlow and Scikit-Learn for analyzing driver behavior.
10. **Infrastructure**: Kubernetes and Docker for container orchestration and deployment on AWS or IBM Cloud.

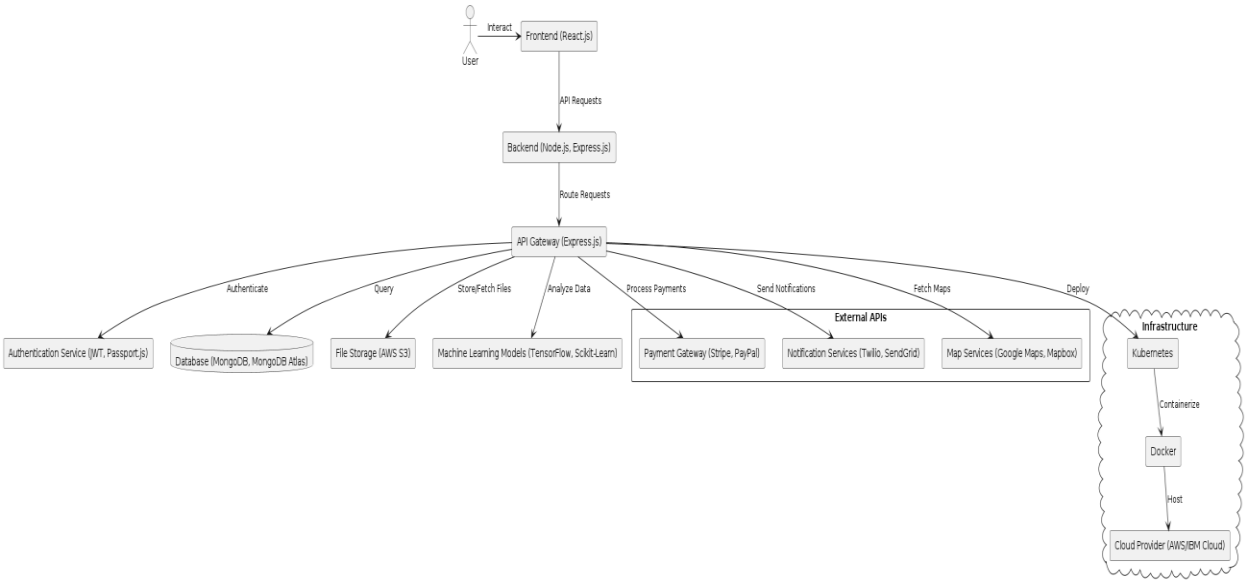



*Figure 1: Architecture and data flow of the cab booking application*

## Requirement Gathering and Analysis Phase
## Solution Requirements (Functional & Non-functional)

| | |
|---|---|
| Date | 05 july 2024 |
| Team ID | SWTID1720162063 |
| Project Name | Project - Cab Booking App |
| Maximum Marks | |

### Functional Requirements Table

| FR No. | Functional Requirement (Epic) | Sub Requirement (Story / Sub-Task) |
|---|---|---|
| FR-1 | User Registration | Registration through Form |

| | | |
|---|---|---|
| | | Registration through Gmail |
| | | Registration through LinkedIN |
| FR-2 | User Confirmation | Confirmation via Email |
| | | Confirmation via OTP |
| FR-3 | User Profile Management | Update Email |
| | | Update Name |
| | | Update Password |
| FR-4 | Ride Booking | Browse Available Cabs |
| | | Specify Pick-up and Drop-off Locations |
| | | Choose Ride Options (e.g., Ride-sharing, Premium) |
| FR-5 | Booking Confirmation | Display Cab Information |
| | | Display Driver Details |
| | | Provide Booking ID |
| FR-6 | Ride History | View Past Rides |
| | | View Upcoming Rides |
| FR-7 | Feedback | Provide Ride Ratings |
| | | Provide Driver Ratings |
| FR-8 | User Logout | Log out from the Application |
| FR-9 | Admin System Management | Control System Functionalities |
| | | Configure System Settings |
| FR-10 | Admin Rider Management | Create Rider Accounts |
| | | Update Rider Accounts |
| | | Delete Rider Accounts |
| FR-11 | Admin Cab Management | Add New Cabs |
| | | Update Cab Details |
| | | Remove Inactive Cabs |
| FR-12 | Admin Driver Management | Add New Drivers |
| | | Update Driver Profiles |
| | | Manage Driver Ratings |
| FR-13 | Admin Logout | Log out from the Application |

**Non-functional Requirements**

We can provide descriptions for the non-functional requirements to ensure they meet the overall system standards.

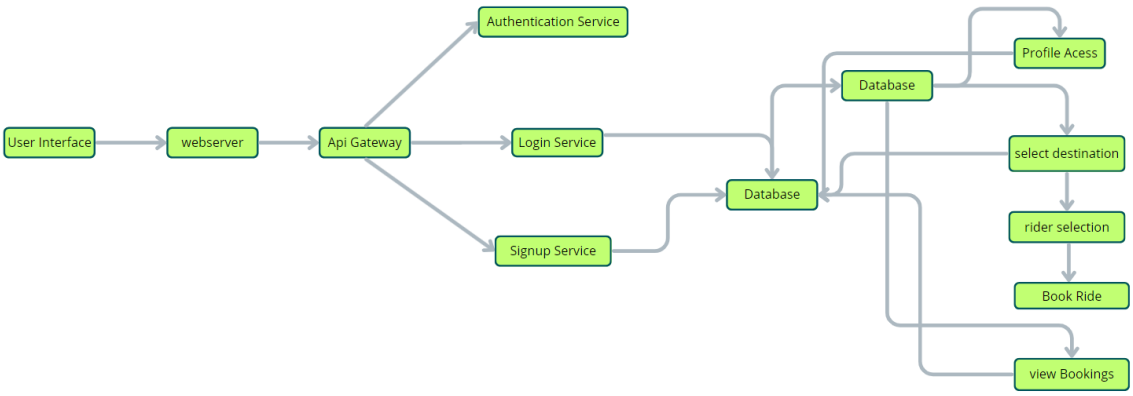
**Non-functional Requirements Table**

| FR No. | Non-Functional Requirement | Description |
|---|---|---|
| NFR-1 | Usability | The application should have an intuitive and user-friendly interface, ensuring ease of use for both riders and admins. |
| NFR-2 | Security | The system must ensure data protection and privacy through encryption, secure authentication methods, and regular security audits. |
| NFR-3 | Reliability | The application should be reliable and consistently perform its intended functions without failure. Regular backups and recovery procedures should be in place. |
| NFR-4 | Performance | The application should provide fast response times and handle multiple transactions efficiently. It should be optimized for both mobile and web platforms. |
| NFR-5 | Availability | The system should have high availability, with minimal downtime, and should be accessible to users 24/7. |
| NFR-6 | Scalability | The system should be scalable to accommodate an increasing number of users and transactions without performance degradation. This includes the ability to add more servers or services as needed. |

**Requirement Gathering and Analysis Phase**
**Technology Stack (Architecture & Stack)**

| | |
|---|---|
| Date | 05july2024 |
| Team ID | SWTID1720162063 |
| Project Name | Project - Cab Booking Ap |
| Maximum Marks | |

**Technical Architecture:**




## Table-1: Components & Technologies

| S.No | Component | Description | Technology |
|---|---|---|---|
| 1 | User Interface | How users interact with the application | HTML, CSS, JavaScript, React.js |
| 2 | Web Server | Hosts the user interface, serving dynamic web pages | Node.js, Express.js |
| 3 | API Gateway | Central entry point for client requests | Express.js |
| 4 | Authentication Service | Manages user authentication and authorization | JWT, Passport.js |

| | | | |
|---|---|---|---|
| 5 | Database | Stores persistent data related to rides and users | MongoDB |
| 6 | Cloud Database | Cloud-based database service | MongoDB Atlas |
| 7 | File Storage | Storing user profile images and ride history | AWS S3 |
| 8 | External API-1 | Payment gateway integration | Stripe API, PayPal API |
| 9 | External API-2 | Email and SMS notification service | Twilio API, SendGrid API |
| 10 | External API-3 | Map and location services | Google Maps API, Mapbox API |
| 11 | Machine Learning Model | Driver behavior analysis (optional) | TensorFlow, Scikit-Learn |
| 12 | Infrastructure (Server) | Local and cloud server deployment configuration | Kubernetes, Docker, AWS EC2, IBM Cloud |

**Table-2: Application Characteristics**

| S.No | Characteristics | Description | Technology |
|---|---|---|---|
| 1 | Open-Source Frameworks | List the open-source frameworks used | React.js, Node.js, Express.js, Mongoose |
| 2 | Security Implementations | Security and access controls, encryption methods | JWT, Passport.js, SSL/TLS, OAuth, IAM |
| 3 | Scalable Architecture | Justify the scalability of architecture (microservices, containerization) | Docker, Kubernetes, Microservices |
| 4 | Availability | Ensure high availability with load balancers and distributed servers | AWS ELB, NGINX, HAProxy |
| 5 | Performance | Design considerations for performance (caching, CDNs, number of requests per sec) | Redis Cache, AWS CloudFront, CDN |