**NaugeCX**

**Functional Requirements Document**

**For**

Implementation of

"City Transport Ticketing and Real-Time Bus Tracking Application”

16/01/2024

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date Version Author Change** | | | |
| 16/01/2024 | 1 | Apaar Jain | 1st Draft |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Table of Contents**

**1 Purpose ....................................................................................................................................4**

**2 Project Information..................................................................................................................4**

2.1 Project Description.................................................................................................................... 4

2.2 Project Approach ...................................................................................................................... 4

2.3 Goals, Objectives, and Scope ................................................................................................... 5

2.4 Business Drivers ....................................................................................................................... 5

2.5 Stakeholders............................................................................................................................. 5

2.6 Assumptions, Dependencies, and Constraints .......................................................................... 5

2.7 Risks......................................................................................................................................... 6

2.8 Costs ........................................................................................................................................ 6

2.9 Target Delivery Dates ............................................................................................................... 6

**3 Process Information ................................................................................................................7**

3.1 Current Process ........................................................................................................................ 7

3.1.1 Current Process Flow ........................................................................................................ 7

3.1.2 Current Process Description .............................................................................................. 7

3.2 New Processes or Future Enhancements ................................................................................. 8

3.2.1 New Process Flow ............................................................................................................. 8

3.2.2 New Process Description ................................................................................................... 8

**4 Requirements Information ......................................................................................................9**

4.1 Functional Requirements .......................................................................................................... 9

4.2 Infrastructure Requirements.................................................................................................... 10

4.3 Other Requirements................................................................................................................ 10

4.4 Non-Functional Requirements................................................................................................. 11

**5 Interfaces................................................................................................................................12**

5.1 System Interfaces ................................................................................................................... 12

5.2 Hardware Interfaces................................................................................................................ 12

5.3 Software Interfaces ................................................................................................................. 12

5.4 Communication Interfaces ...................................................................................................... 12

**6 Glossary .................................................................................................................................13**

**1 Purpose**

The purpose of this Functional Requirements Document (FRD) is to define the functional requirements for the development of a mobile application that enables users to book online tickets for local city transport. The application will also display real-time bus locations along travel routes, allowing users to track their journeys effectively. Furthermore, the app will generate a live, valid ticket for each user upon booking, with a green validation symbol to assist bus conductors in identifying active tickets. This document aims to provide a clear and comprehensive set of functional specifications to guide the design, development, and implementation of the software solution, ensuring that it meets the needs of both the users and the service providers.

**2 Project Information**

**Objectives:**

1. To provide a seamless, easy-to-use platform for booking tickets for city transport.
2. To implement a live tracking feature that shows the real-time location of buses along their routes.
3. To ensure that the app generates a valid, live ticket with a green symbol for easy ticket verification by bus conductors.
4. To reduce dependency on physical tickets, contributing to a more sustainable and efficient system.
5. To improve passenger convenience and satisfaction by providing accurate, up-to-date travel information.

**Major System Functionality:**

* **Ticket Booking:** Users will be able to select their desired route, choose a bus, and book tickets online.
* **Real-Time Bus Location Tracking:** The app will provide real-time updates on the locations of buses on selected routes.
* **Live Ticket Generation:** Upon booking, a valid, live ticket will be generated within the app with a green valid symbol to indicate its authenticity.
* **Route and Schedule Information:** The app will display available routes, bus schedules, and estimated travel times.
* **Payment Integration:** The app will support secure payment methods for ticket purchases, such as credit/debit cards or mobile wallets.

**Problems or Issues to Resolve:**

1. **Real-Time Data Accuracy:** Ensuring that the real-time tracking of buses is accurate and reflects the current location of buses in real-time, considering delays or unforeseen issues.
2. **User Accessibility:** Ensuring that the app is easy to use for a wide range of users, including those who may not be tech-savvy.
3. **Integration with Transport Systems:** The app needs to integrate seamlessly with the existing city transport infrastructure, including bus tracking systems, payment gateways, and ticketing systems.
4. **Security and Privacy:** Safeguarding users’ personal and payment information by implementing robust security measures.
5. **Reliability of Ticket Validation:** Ensuring that the live ticket validation system is effective, allowing bus conductors to accurately verify the authenticity of tickets in all circumstances.
6. **Scalability:** Ensuring that the system can handle a high number of concurrent users, especially during peak travel hours, without compromising performance or reliability.

**2.1 Project Description**

**Background:**  
As urban populations continue to grow, city transport systems are facing increased demand for efficient and user-friendly solutions. Many cities are moving towards digital transformation to streamline public transport services, enhance commuter convenience, and improve operational efficiency. Our project aims to address these needs by developing a mobile application that simplifies ticket booking for local city transport while providing users with real-time bus location updates. This app will improve the overall travel experience for commuters by making public transportation more accessible, efficient, and environmentally friendly.

**Vision:**  
The vision of this project is to create an intuitive and reliable mobile application that empowers passengers to book tickets for local buses, track their routes in real-time, and validate their tickets with ease. The goal is to contribute to a smarter, more sustainable transport system by reducing paper ticketing, improving ticket verification processes, and offering a seamless user experience. The app will also be designed with scalability in mind, ensuring it can support large volumes of users and city-wide transport networks as the service expands.

**Approach:**  
Our approach involves developing an intuitive, user-centric mobile application that integrates several key functionalities:

1. **Online Ticket Booking:** Users can select routes, book tickets, and pay securely through the app.
2. **Real-Time Bus Location Tracking:** The app will utilize GPS technology to display live bus locations on user-selected routes, providing accurate arrival time predictions.
3. **Live Ticketing:** After booking, users will receive a digital ticket that includes a green valid symbol to help conductors verify its authenticity in real time.
4. **Backend Integration:** We will ensure seamless integration with the existing city transport infrastructure, including bus tracking systems, ticketing, and payment platforms.
5. **User-Centric Design:** The app will be designed with a focus on simplicity and ease of use, ensuring accessibility for all commuters, regardless of technical proficiency.

The development will follow agile methodology to ensure that the product evolves through iterative cycles, incorporating user feedback and responding to any challenges that arise during implementation.

**Timeframe:**  
The project is expected to be completed in the following phases:

* **Phase 1 (Requirement Gathering & Design):** 4 weeks  
  During this phase, we will gather all functional and technical requirements and design the user interface and experience of the app.
* **Phase 2 (Development & Integration):** 12 weeks  
  In this phase, the app will be developed, integrated with city transport systems, and tested for real-time bus tracking, ticket booking, and validation functionalities.
* **Phase 3 (Testing & QA):** 4 weeks  
  Rigorous testing will be conducted to ensure the app’s functionality, performance, and security are up to standard.
* **Phase 4 (Deployment & User Training):** 2 weeks  
  The app will be deployed in a live environment, and users will be provided with training materials or onboarding sessions to ensure they can use the app effectively.
* **Phase 5 (Post-Launch Support):** Ongoing  
  After deployment, we will offer ongoing maintenance, user support, and periodic updates to improve functionality and incorporate user feedback.

**Key Deliverables:**

* A fully functional mobile app with ticket booking and real-time tracking features.
* A live ticket validation system for bus conductors.
* Integration with the city’s existing transport infrastructure.
* A user-friendly interface accessible on both Android and iOS platforms.
* Documentation for system maintenance and user guides.

**2.2 Project Approach**

The project will be delivered through a structured, phased approach that ensures clarity, focus, and quality throughout the entire development process. The project will follow an **Agile** methodology, enabling iterative development and continuous feedback from stakeholders to refine and improve the solution. The key phases of the project and their associated high-level activities are outlined below:

**Phase 1: Requirement Gathering & Design (Duration: 4 weeks)**

**Objectives:**

* Understand the detailed requirements from stakeholders (including transport authorities, users, and conductors).
* Define the functional and non-functional requirements for the application.
* Design the user interface (UI) and user experience (UX) to ensure the app is intuitive and user-friendly.

**Key Activities:**

* **Stakeholder Meetings:** Engage with transport authorities, potential users, and bus conductors to gather their expectations and pain points.
* **Define Functional Requirements:** Finalize all features for ticket booking, real-time bus tracking, live ticket validation, and payment integration.
* **Create Design Mockups and Prototypes:** Develop wireframes, UI/UX design prototypes, and user flows.
* **Approval:** Obtain stakeholder sign-off on the requirements and design.

**Importance to the Overall Project:** This phase sets the foundation for the entire project by establishing clear expectations and a shared vision among stakeholders. The design and functional requirements laid out here will guide all subsequent phases.

**Phase 2: Development & Integration (Duration: 12 weeks)**

**Objectives:**

* Develop the core functionalities of the mobile application.
* Integrate the app with real-time bus tracking systems and payment gateways.
* Ensure the application works on both Android and iOS platforms.

**Key Activities:**

* **Mobile App Development:**
  + Build the core ticket booking system, real-time tracking functionality, and live ticket generation system.
  + Implement payment gateways and integration with city transport systems.
* **API Development:** Create and integrate APIs to retrieve real-time bus locations and manage bookings.
* **Backend Development:** Build a secure backend system for ticket management, user accounts, and transaction handling.
* **Cross-Platform Testing:** Ensure that the app works seamlessly on both Android and iOS platforms.
* **Integration Testing:** Integrate the app with external systems (bus tracking and payment platforms) and conduct thorough testing.

**Importance to the Overall Project:** This is the development phase where the key features of the app are built and integrated into a working product. It is critical for delivering the app’s core functionalities and ensuring compatibility with the transport infrastructure.

**Phase 3: Testing & Quality Assurance (Duration: 4 weeks)**

**Objectives:**

* Ensure that the application functions as expected across all scenarios and platforms.
* Validate the performance, security, and usability of the app.

**Key Activities:**

* **Functional Testing:** Verify that all features—ticket booking, real-time bus tracking, and live ticket validation—work as intended.
* **Performance Testing:** Test the app under various load conditions to ensure scalability and reliability, especially during peak travel times.
* **Security Testing:** Ensure that user data, payment transactions, and app functionality are secure and comply with data protection regulations.
* **User Acceptance Testing (UAT):** Engage real users to test the app and provide feedback to ensure it meets their expectations and needs.
* **Bug Fixing and Refinements:** Resolve any issues found during testing and make necessary improvements based on user feedback.

**Importance to the Overall Project:** Quality assurance is crucial to ensure the app functions smoothly and securely under real-world conditions. Thorough testing prevents potential issues that could affect the user experience and system reliability upon launch.

**Phase 4: Deployment & User Training (Duration: 2 weeks)**

**Objectives:**

* Deploy the app to the production environment.
* Provide training materials and support to end-users (passengers and bus conductors).

**Key Activities:**

* **App Deployment:** Launch the mobile app on the App Store and Google Play Store.
* **Backend Deployment:** Set up the live backend infrastructure to handle real-time data, bookings, and transactions.
* **User Training:** Create user guides, onboarding tutorials, and conduct training sessions for both passengers and bus conductors.
* **Support Setup:** Set up a support team to assist users with any post-launch issues or questions.

**Importance to the Overall Project:** Deployment marks the official release of the app, making it available to the public. Proper training and support are essential to ensure users can easily adopt and use the system effectively.

**Phase 5: Post-Launch Support & Maintenance (Ongoing)**

**Objectives:**

* Ensure smooth operation of the app and address any emerging issues.
* Provide ongoing updates and improvements based on user feedback and system performance.

**Key Activities:**

* **Monitor System Performance:** Continuously monitor the app’s performance and real-time bus tracking functionality to ensure smooth operation.
* **User Feedback Collection:** Gather feedback from users and bus conductors to identify areas for improvement.
* **Bug Fixes & Updates:** Address any bugs or issues that arise after launch and release regular updates to enhance app features and security.
* **Enhancements & Feature Updates:** Implement new features or enhancements based on user needs and future technology advancements.

**2.3 Goals, Objectives, and Scope**

**Goals:**

* Develop a mobile app for seamless online ticket booking for local city transport.
* Provide real-time bus location tracking for improved user experience.
* Implement a live ticket validation system for easy conductor verification.
* Enhance sustainability by reducing paper ticketing.
* Ensure secure and reliable payment processing within the app.

**Objectives:**

* Create a user-friendly mobile application compatible with Android and iOS.
* Integrate real-time bus tracking and route information for accurate travel predictions.
* Generate live, green-valid tickets for conductors to verify active tickets.
* Provide a secure, scalable payment system for ticket purchases.
* Complete the development, testing, and deployment within a 6-month timeline.

**Scope:**

* **In-Scope:**
  + Development of the mobile app for booking tickets and real-time bus tracking.
  + Integration with the city’s existing transport infrastructure and payment gateways.
  + Real-time bus location tracking and live ticket validation features.
  + User accounts for ticket management and payment transactions.
* **Out-of-Scope:**
  + Development of the physical transport infrastructure (e.g., buses or bus tracking hardware).
  + Features unrelated to bus transport services (e.g., non-local transport or intercity transport features).
  + Long-term system maintenance beyond the first year of deployment.

**High-Level Project Deliverables:**

| **ID** | **Description** | **Included/Excluded** |
| --- | --- | --- |
| 1 | Mobile app for ticket booking and real-time bus tracking | Included |
| 2 | Real-time bus location tracking integration with city transport systems | Included |
| 3 | Live ticket generation system with green valid symbol for conductors | Included |
| 4 | Secure payment gateway integration for ticket purchases | Included |
| 5 | User account management and ticket history features | Included |
| 6 | Cross-platform (Android & iOS) app compatibility | Included |
| 7 | Backend system for data management and transaction processing | Included |
| 8 | Testing and quality assurance (functional, performance, security) | Included |
| 9 | User training materials and onboarding guides | Included |
| 10 | Post-launch support and maintenance for 6 months | Included |
| 11 | Development of physical infrastructure (e.g., GPS tracking hardware for buses) | Excluded |
| 12 | Development of non-local or intercity transport features | Excluded |
| 13 | Long-term maintenance and support beyond 6 months | Excluded |

**2.4 Business Drivers**

1. **Improved Operational Efficiency:**  
   The implementation of a digital ticketing and real-time tracking system will streamline the ticket booking process and reduce manual intervention, allowing for faster and more efficient operations for both passengers and conductors.
2. **Cost Reduction:**  
   By moving to digital ticketing, the need for paper tickets and manual validation is minimized, leading to lower operational costs and a reduction in waste production. Additionally, real-time tracking will optimize bus routes and schedules, improving overall operational cost-effectiveness.
3. **Enhanced Customer Experience:**  
   The app will provide passengers with real-time information about bus locations, helping them plan their travel better. The ease of digital ticket booking and live ticket validation will enhance convenience and user satisfaction.
4. **Increased Revenue:**  
   The app will make ticket purchasing more accessible, potentially increasing ticket sales through a more convenient and user-friendly platform. A secure and easy payment system will encourage more people to use public transport, leading to higher revenue generation.
5. **Sustainability:**  
   Moving away from paper-based ticketing contributes to sustainability efforts by reducing paper waste, aligning with environmental goals and appealing to eco-conscious users.
6. **Data-Driven Insights:**  
   The collection of data from ticket sales and real-time bus tracking will provide valuable insights into travel patterns, which can help improve route planning, capacity management, and future business decision-making.
7. **Scalability and Future Expansion:**  
   The digital platform can be scaled to accommodate future transport services or integrate with additional city transport options, allowing the solution to evolve and expand as urban transport needs grow.

**2.5 Stakeholders**

| **Name** | **Department** | **Role** |
| --- | --- | --- |
| **XYZ** | Product Management | Project Owner, oversees overall project progress and alignment with business objectives. |
| **ABC** | Development | Lead Developer, responsible for the design and development of the mobile application. |
| **DEF** | Transport Operations | Transport Coordinator, responsible for integrating the app with city transport systems and ensuring accurate bus data. |
| **GHI** | Quality Assurance | QA Lead, responsible for testing the app and ensuring its functionality and security. |
| **JKL** | Marketing | Marketing Manager, in charge of promoting the app and ensuring user adoption. |
| **MNO** | Finance | Financial Analyst, responsible for managing budgets, costs, and ensuring financial feasibility. |
| **PQR** | User Experience (UX) | UX/UI Designer, responsible for the app's interface and ensuring a smooth user experience. |
| **STU** | Customer Support | Support Manager, responsible for post-launch support and addressing customer issues or inquiries. |
| **VWX** | IT Infrastructure | IT Systems Architect, responsible for backend architecture, database management, and security. |
| **YZA** | Legal & Compliance | Legal Advisor, ensures compliance with data privacy laws and oversees contracts and agreements. |

**2.6 Assumptions, Dependencies, and Constraints**

**Assumptions:**

1. The city transport system will provide accurate real-time bus location data.
2. Users will have access to smartphones (Android or iOS) with internet connectivity.
3. Payment gateways will support secure transactions without major disruptions.
4. Bus conductors will be trained to validate digital tickets effectively.
5. The city transport infrastructure will support seamless integration with the app.

**Dependencies:**

1. Availability and reliability of GPS and bus tracking data from city transport systems.
2. Integration with third-party payment gateways for ticket purchases.
3. Availability of backend infrastructure to handle real-time data and user traffic.
4. Timely feedback and cooperation from stakeholders (e.g., transport authorities and conductors).
5. Regulatory approvals for data privacy and security compliance.

**Constraints:**

1. Limited budget for development and marketing of the app.
2. Time constraints for development and testing within the set timeline (6 months).
3. Limited availability of technical resources or expertise in certain areas (e.g., real-time tracking integration).
4. Compatibility issues with older versions of Android or iOS devices.
5. Potential delays in integration with existing transport systems and APIs.

**2.7 Risks**

| **Risk** | **Description** | **Mitigation/Work-around** |
| --- | --- | --- |
| **Real-Time Data Accuracy** | There is a risk that real-time bus location data may be inaccurate or delayed, affecting the reliability of the tracking feature. | Ensure robust integration with transport system APIs, and establish service-level agreements (SLAs) with transport authorities for data accuracy. Conduct regular monitoring and issue resolution. |
| **Integration Challenges** | Difficulty in integrating with existing city transport infrastructure or third-party systems (payment gateways, bus tracking). | Early and thorough system analysis, engage with experienced third-party vendors, and allocate extra time for testing integrations. |
| **Security Breaches** | The app could be vulnerable to data breaches, compromising user payment and personal information. | Implement strong encryption, multi-factor authentication, and comply with industry security standards (e.g., PCI-DSS for payment data). Regular security audits should be conducted. |
| **User Adoption** | Users may be resistant to adopting the new app, preferring traditional ticketing methods. | Implement a user education campaign with clear instructions, promotional offers, and incentives for early adoption. Provide easy-to-use features and a smooth onboarding process. |
| **Technical Delays** | Delays in app development, testing, or system integration could push back the launch date. | Use agile development cycles to quickly identify issues, allocate additional resources if necessary, and prioritize critical features for early release. |
| **Device Compatibility** | The app may not function as expected on older smartphones or certain devices, limiting its user base. | Ensure the app is compatible with a wide range of devices (at least the last two versions of Android and iOS). Conduct thorough compatibility testing and optimize for performance on older devices. |
| **Budget Overruns** | The project may exceed the allocated budget due to unforeseen development challenges. | Maintain a contingency budget, closely track project expenses, and regularly reassess resource needs to avoid unnecessary costs. |
| **Regulatory Compliance** | Changes in data privacy or transportation regulations could require changes to the app. | Stay informed about relevant regulations (e.g., GDPR, data protection laws) and build flexibility into the app's design to accommodate changes. Regularly consult with legal advisors. |
| **Post-Launch Support** | The app may encounter issues after launch, leading to customer dissatisfaction. | Set up a dedicated support team, create detailed FAQs, and provide timely updates to fix bugs and improve the app based on user feedback. |

**2.8 Costs**

Below is an estimated breakdown of the costs for implementing the proposed solution, categorized into **Capital Costs** (initial setup costs) and **Expense Costs** (ongoing operational costs). These estimates are based on general industry pricing and assumptions about the project scope.

### ****Capital Costs****

| **Cost Category** | **Estimated Cost** | **Assumptions** |
| --- | --- | --- |
| **Mobile App Development (Android & iOS)** | $120,000 | Assumes a team of 4-5 developers, 6-month development period. |
| **Backend Infrastructure & Server Setup** | $30,000 | Includes cloud hosting services for database and real-time data management. |
| **Third-Party Integrations (Payment Gateway, GPS)** | $25,000 | Includes licensing and setup for third-party services. |
| **UI/UX Design & Prototyping** | $15,000 | Cost for design resources, wireframes, and user flow prototypes. |
| **Quality Assurance & Testing** | $20,000 | Includes manual and automated testing efforts. |
| **Project Management** | $30,000 | Project management costs for the 6-month development timeline. |
| **Legal & Compliance** | $10,000 | Legal consultations for regulatory compliance, data privacy, and contracts. |
| **Marketing & Launch Campaign** | $40,000 | Marketing efforts including app store optimization, ads, and promotional campaigns. |
| **Training & Onboarding Materials** | $5,000 | Training sessions and materials for users and conductors. |

**Total Capital Costs**: **$275,000**

### ****Expense Costs****

| **Cost Category** | **Estimated Annual Cost** | **Assumptions** |
| --- | --- | --- |
| **Cloud Hosting & Data Management** | $15,000 | Ongoing server, storage, and real-time data services for app. |
| **App Maintenance & Updates** | $50,000 | Includes bug fixes, new features, and platform updates. |
| **Customer Support** | $30,000 | Ongoing customer support for app users. Includes staffing and tools. |
| **Marketing & User Retention** | $25,000 | Continuous marketing efforts to retain and grow user base. |
| **Security & Compliance Monitoring** | $10,000 | Ongoing security audits, legal consultations, and compliance checks. |
| **Payment Gateway Fees** | $12,000 | Transaction fees based on ticket sales. Estimated at 2% of revenue. |
| **Post-launch User Feedback Collection** | $8,000 | Tools for collecting feedback and surveys from users. |

**Total Annual Expense Costs**: **$140,000**

### ****Assumptions Used to Estimate Costs****:

1. **Development Team**: A typical team of 4-5 developers (mobile and backend) will be needed for the 6-month development period.
2. **Cloud Hosting**: Costs are estimated based on mid-range cloud services (AWS, Google Cloud, etc.) with sufficient resources for a moderately scaled app.
3. **Third-Party Integrations**: Estimated costs for integrating services like payment gateways (Stripe, PayPal) and GPS/bus tracking services (such as Google Maps API).
4. **Marketing**: Budget assumes digital marketing strategies including Google Ads, social media ads, and app store optimization for initial launch and early user acquisition.
5. **Post-launch Maintenance**: Includes ongoing improvements, bug fixes, platform updates, and user support for both passengers and bus conductors.
6. **Security & Compliance**: Includes periodic reviews and ongoing efforts to ensure the app complies with relevant regulations, particularly data privacy laws.

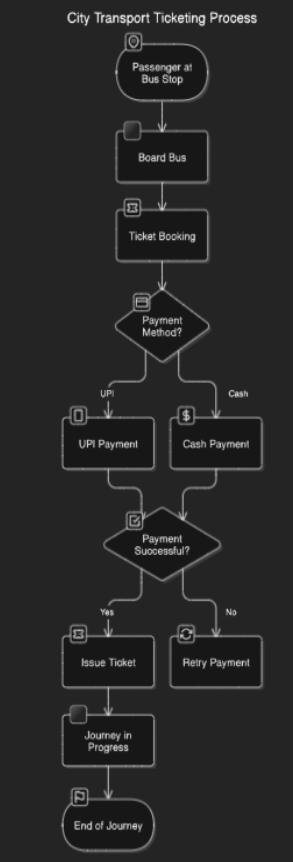
**2.9 Target Delivery Dates**

Here are the high-level deliverables and milestones for the project with associated target dates:

| **Milestone/Deliverable** | **Target Date** |
| --- | --- |
| **Project Kickoff** | February 1, 2025 |
| **Requirement Gathering and Design Approval** | February 28, 2025 |
| **Completion of UI/UX Design and Prototypes** | March 15, 2025 |
| **App Development - Alpha Version** | April 30, 2025 |
| **Integration with Real-Time Bus Tracking System** | May 15, 2025 |
| **Integration with Payment Gateway** | May 31, 2025 |
| **Beta Testing & User Acceptance Testing (UAT)** | June 15, 2025 |
| **Completion of App Testing and Bug Fixing** | June 30, 2025 |
| **App Deployment to App Stores (iOS & Android)** | July 15, 2025 |
| **Post-launch User Training and Support Setup** | July 31, 2025 |
| **Marketing & Launch Campaign Start** | August 1, 2025 |
| **Project Closure and Final Report** | August 15, 2025 |
| **Post-launch Support and Maintenance (Ongoing)** | August 16, 2025 |

**3 Process Information**

**3.1 Current Process**



**3.1.1 Current Process Flow**

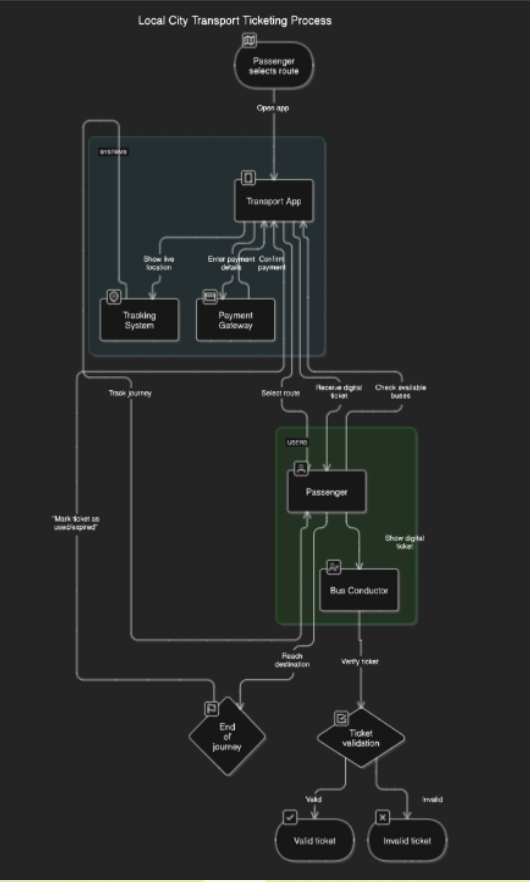
Passenger stands at the bus stop and takes the bus.

**Ticket booking:** The passenger tells conductor where the go.

Conductor enters journey details and gives a ticket to the passenger after successful payment, either UPI or Cash.

**End of journey**: After reaching the destination, the journey is completed for the user.

**3.2 New Processes or Future Enhancements**



**3.2.1 New Process Flow**

 **Passenger selects a route**: The passenger opens the transport app, selects the route they want to travel, and checks the available buses.

 **Ticket booking**: The passenger enters payment details, books the ticket, and receives a digital ticket.

 **Ticket validation**: Upon boarding, the bus conductor verifies the ticket using a physical or manual system.

 **Real-time tracking**: The app shows the current live location of buses, allowing passengers to track their journey.

 **End of journey**: After reaching the destination, the journey is completed, and the ticket is marked as used or expired in the system.

**4 Requirements Information**

**4.1 Functional Requirements**

### ****Mandatory (Must Have) Requirements:****

| **ID** | **Functional Requirement** | **Description** |
| --- | --- | --- |
| FR1 | **Ticket Booking** | The app must allow passengers to book tickets for their selected route and travel date, providing options to select from available buses. |
| FR2 | **Real-Time Bus Location Tracking** | The app must display the current live location of buses in real-time, enabling passengers to track the progress of their chosen bus. |
| FR3 | **Live Ticket Validation** | The app must generate a live, valid ticket after purchase, which can be shown to the bus conductor for verification, indicated with a green valid symbol. |
| FR4 | **Payment Integration** | The app must integrate with secure payment gateways (e.g., Stripe, PayPal) to process ticket payments securely. |
| FR5 | **User Accounts and History** | Users must be able to create accounts, save their ticketing history, and manage payments and personal information. |
| FR6 | **Cross-Platform Compatibility** | The app must be compatible with both Android and iOS devices, offering a seamless experience across platforms. |
| FR7 | **Push Notifications** | The app must send notifications to passengers about ticket confirmations, bus arrivals, delays, and other critical updates. |
| FR8 | **Admin/Backend System** | A secure backend must be developed to handle ticket management, user accounts, and real-time data updates. |

### ****Desirable (Nice to Have) Requirements:****

| **ID** | **Functional Requirement** | **Description** |
| --- | --- | --- |
| FR9 | **Multi-language Support** | The app can support multiple languages to cater to users from diverse linguistic backgrounds. |
| FR10 | **User Feedback and Ratings** | Passengers should be able to rate their journey and provide feedback on the app or the transport service. |
| FR11 | **Route Optimization Suggestions** | The app can offer suggestions for optimal routes based on real-time traffic data, providing users with faster travel options. |
| FR12 | **Multiple Payment Options** | The app can offer multiple payment options (e.g., credit/debit cards, mobile wallets, etc.) for added convenience. |
| FR13 | **Discounts and Offers** | The app can display special offers or discounts on tickets, such as for regular commuters or off-peak hours. |
| FR14 | **Advanced Reporting for Admins** | The admin panel should have advanced reporting and analytics features for transport authorities to track usage, revenue, and other metrics. |
| FR15 | **Social Media Integration** | The app can allow users to share their trip details or experiences on social media platforms (Facebook, Twitter, etc.). |
| FR16 | **Customizable User Interface** | The app can offer a customizable interface (e.g., theme changes, layout adjustments) for user preferences. |

**4.2 Infrastructure Requirements**

### ****Infrastructure Information for Project Implementation:****

#### **1. Servers and Cloud Infrastructure:**

* **Cloud Hosting Platform:**
  + The app will be hosted on a cloud platform (e.g., AWS, Google Cloud, or Microsoft Azure) for scalability and reliability.
  + Services like **AWS EC2** or **Google Cloud Compute** will be used for app hosting, ensuring high availability and low latency.
  + Cloud **storage solutions** (e.g., AWS S3, Google Cloud Storage) for storing ticket data, user data, and other app assets.
* **Database Servers:**
  + A **relational database** (e.g., PostgreSQL, MySQL) or **NoSQL database** (e.g., MongoDB) will be set up for storing user data, transaction logs, ticket history, and real-time bus data.
  + Database servers will be set up to ensure high availability with **replication** and **backup** mechanisms for data redundancy and disaster recovery.
* **Load Balancers:**
  + **Load balancers** (e.g., AWS Elastic Load Balancing or Nginx) will distribute traffic across app servers to ensure optimal performance and minimize server overloads.

#### **2. Network Devices and Configuration:**

* **Virtual Private Network (VPN):**
  + For secure communication between the app’s backend and transport data systems, a **VPN** or **Private Subnets** will be established for data security.
* **Firewall and Security Configuration:**
  + **Firewalls** will be implemented for both cloud-based and on-premises systems to restrict unauthorized access.
  + Network security policies will be put in place to monitor and control incoming and outgoing traffic, ensuring only trusted sources can access the servers.
* **API Gateway:**
  + An **API Gateway** (e.g., AWS API Gateway) will be used to handle traffic between the mobile app, payment systems, and backend servers, ensuring that all APIs are secure and monitored.

#### **3. Middleware and Operating System Software:**

* **Middleware:**
  + The app will use **middleware** for tasks like authentication, session management, and routing between the app and the backend. Examples include **Node.js**, **Spring Boot**, or **Express.js**.
  + The integration of **real-time bus tracking** and ticket validation features will rely on middleware to handle communication between the app and third-party services.
* **Operating System (OS):**
  + For cloud-based servers, the operating systems used will include **Linux** (Ubuntu or CentOS) for backend services and **Windows** for specific software needs, where applicable.
  + App servers and databases will run on the latest stable versions of **Ubuntu** or **Red Hat Enterprise Linux (RHEL)** to ensure performance and security.

#### **4. Network Management:**

* **Network Monitoring:**
  + Tools like **AWS CloudWatch**, **Google Cloud Monitoring**, or **Nagios** will be used to monitor server health, performance, and network traffic.
  + Continuous monitoring for **uptime**, **response times**, and **latency** will be critical for ensuring the real-time tracking and ticket booking functionality remains operational.
* **Traffic Routing and Content Delivery:**
  + **Content Delivery Networks (CDNs)** like **Cloudflare** or **AWS CloudFront** will be used to speed up access for users by caching static assets (images, ticket data) at edge locations, reducing latency.
  + **API rate limiting** and **traffic routing optimization** will be implemented to ensure smooth handling of high user traffic, especially during peak hours.

#### **5. Data Center Changes/Upgrades:**

* **Data Center Location:**
  + The backend systems and databases will be hosted in geographically distributed data centers to ensure redundancy and low-latency access for users across the region.
  + These data centers will be compliant with data protection regulations such as **GDPR** (General Data Protection Regulation) or **CCPA** (California Consumer Privacy Act).
* **Disaster Recovery Planning:**
  + A **disaster recovery plan** will be in place, with data backed up to offsite locations and high availability configurations.
  + **Failover mechanisms** will be set up to ensure uninterrupted service in case of server or data center outages.

#### **6. Security Services and Oversight:**

* **Authentication and Authorization:**
  + User authentication will be handled via **OAuth** or **JWT** tokens, ensuring secure logins and data access.
  + Admin access to backend systems will be controlled using role-based access control (RBAC) to minimize unauthorized access.
* **Security Monitoring Tools:**
  + **Intrusion Detection Systems (IDS)** and **Intrusion Prevention Systems (IPS)** will be implemented to monitor for unauthorized access attempts.
  + **Security Information and Event Management (SIEM)** tools like **Splunk** will be used to collect and analyze logs for potential security threats.
* **Encryption:**
  + **TLS/SSL encryption** will be used for securing communication between the app and the backend.
  + **End-to-end encryption** will be applied to sensitive user data (e.g., payment information).

#### **7. Services and Oversight:**

* **Project Oversight and Management:**
  + Continuous oversight will be provided by the **project management team** to ensure timely and cost-effective execution of infrastructure-related tasks.
  + Regular **infrastructure reviews** will be conducted to ensure alignment with business objectives and to address any emerging issues.
* **Third-Party Service Providers:**
  + Services such as **payment gateways**, **real-time tracking data providers**, and **SMS/Email services** will be monitored and supported through vendor agreements and service level agreements (SLAs).

**4.3 Other Requirements**

| **Requirement Category** | **Requirement** | **Description** |
| --- | --- | --- |
| **Usability Requirements** | **User Interface (UI) Design** | The app must have a clean, intuitive interface with easy navigation and visual cues for users of all ages and expertise. |
|  | **User Onboarding and Support** | Easy onboarding, account creation, and tutorial guides. In-app support through FAQs and direct customer support. |
|  | **Personalization** | Allow users to save favorite routes, ticket preferences, and payment methods. |
| **Operational Requirements** | **Multi-language Support** | Support multiple languages based on geographic location for better accessibility. |
|  | **Cross-Platform Compatibility** | Must work seamlessly on both Android and iOS, supporting current and two previous versions. |
|  | **Payment Gateway Integration** | Support for multiple payment methods like credit cards, debit cards, and digital wallets (Google Pay, Apple Pay). |
|  | **Real-Time Data Update** | Live bus location data should update in real-time with minimal delay. |
| **Reliability & Availability** | **System Availability** | Minimum 99.9% uptime to ensure users can reliably book tickets and track buses anytime. |
|  | **Performance Requirements** | Response time for user actions (booking tickets, viewing bus locations) must be under 3 seconds. |
|  | **Load Handling** | The system must handle high volumes of concurrent users, especially during peak hours. |
|  | **Graceful Degradation** | In case of failure, essential features (e.g., bus tracking) should remain available, even if some features are down. |
| **Reliability & Recoverability** | **Data Recovery and Backup** | Daily backups of critical data and stored in geographically dispersed locations. |
|  | **Disaster Recovery (DR)** | A DR plan to restore services within 4 hours in case of major failure. |
|  | **Automated Failover Mechanisms** | Ensure uninterrupted service during server failures by using automated failover systems. |
| **System Maintenance** | **Scheduled Maintenance** | Notify users of scheduled downtime 24 hours in advance, preferably during off-peak hours. |
|  | **Continuous Updates** | Regular updates (e.g., monthly) for bug fixes, new features, and OS compatibility. |
|  | **Monitoring and Logging** | Comprehensive monitoring and logging for system usage, performance, and error tracking. |
| **Security & Compliance** | **Data Privacy** | Compliance with GDPR, CCPA, and other relevant privacy regulations. Sensitive data must be encrypted at rest and in transit. |
|  | **User Authentication** | Support for multi-factor authentication (MFA) for enhanced security. |
|  | **Role-Based Access Control (RBAC)** | Admin access to backend systems restricted based on specific roles. |
| **Scalability Requirements** | **Horizontal Scalability** | Backend system must scale horizontally to handle increased traffic during peak periods. |
|  | **Data Scalability** | Database should scale to handle growing user data and transaction records without compromising performance. |
| **User Feedback and Continuous Improvement** | **User Feedback Collection** | In-app features for collecting user feedback on their experience (ratings, surveys). |
|  | **Analysis of Usage Data** | Data analytics to identify trends, performance issues, and improvement areas in functionality and user experience. |

**4.4 Non-Functional Requirements**

| **Category** | **Non-Functional Requirement** | **Description** |
| --- | --- | --- |
| **Product Requirements** | **Usability** | The app must be user-friendly, with an intuitive interface that can be easily navigated by users of various technical expertise. |
|  | **Accessibility** | The app should be accessible to users with disabilities, including support for screen readers and color contrast adjustments. |
|  | **Cross-Platform Support** | The app must be compatible with both Android and iOS platforms and function smoothly across different device types. |
|  | **Internationalization** | The app should support multiple languages and adapt to the local formats for time, dates, and currencies. |
| **Organizational Requirements** | **Compliance with Industry Standards** | The app must comply with industry standards such as **ISO 27001** for information security and **GDPR** for data privacy. |
|  | **Availability of Support** | 24/7 customer support must be available to assist users with issues, especially during peak times or travel hours. |
|  | **Training and Documentation** | End-user and administrative training, along with clear and comprehensive documentation, should be provided. |
| **System Requirements** | **Performance** | The system must process and respond to user requests (e.g., booking tickets, checking bus locations) within 3 seconds. |
|  | **System Availability** | The app must ensure **99.9% uptime**, guaranteeing reliable service and availability during peak usage times. |
|  | **Scalability** | The system should be able to handle increased load and scale horizontally to accommodate high traffic during peak times. |
|  | **Fault Tolerance** | The system should continue to operate at a reduced capacity if a component fails, without affecting the user experience. |
|  | **Disaster Recovery** | The system should implement a **disaster recovery plan** that ensures service restoration within 4 hours after a catastrophic failure. |
|  | **Data Retention and Archiving** | Data retention policies should be in place to securely store user data for required periods and archive it afterward in compliance with data protection laws. |
| **External Requirements** | **Security** | The app must comply with **industry-standard encryption (TLS/SSL)** for all data transactions and **end-to-end encryption** for user data. |
|  | **Authentication and Authorization** | Implement **multi-factor authentication (MFA)** and **role-based access control (RBAC)** for secure access management. |
|  | **Regulatory Compliance** | The app must comply with **GDPR** (General Data Protection Regulation), **CCPA** (California Consumer Privacy Act), and other regional data protection laws. |
|  | **Payment Gateway Security** | Integrate with secure, PCI-DSS compliant payment gateways to ensure safe transactions and protection of payment data. |
|  | **Third-Party Integrations** | Any third-party services used (e.g., real-time tracking data providers, payment processors) should meet stringent security standards and compliance. |
|  | **Audit Logging and Monitoring** | All system interactions (e.g., ticket purchases, user logins) should be logged for audit purposes. The logs must be secure, encrypted, and stored for a predefined period. |
| **Capacity and Scalability** | **User Capacity** | The system must be able to handle up to **10,000 concurrent users** during peak usage times, with minimal performance degradation. |
|  | **Data Capacity** | The database must be capable of handling and scaling to store **millions of transactions** and user records without compromising performance. |
|  | **Elastic Scalability** | The system should automatically scale resources up or down based on traffic or load, ensuring resource efficiency and cost control. |
| **Legal and Regulatory** | **Data Privacy** | Ensure that user data is handled in compliance with local, national, and international data privacy regulations (GDPR, CCPA, etc.). |
|  | **Consumer Protection** | The app must meet consumer protection laws by ensuring secure transactions and providing clear terms of service and privacy policies. |
|  | **Tax Compliance** | Ensure compliance with local tax regulations for processing payments and issuing tickets in different regions. |

**5 Interfaces**

**5.1 System Interfaces**

### ****1. Communication Hardware, Software, and Their Requirements****

| **Component** | **Description** |
| --- | --- |
| **Communication Hardware** | The app will interface with bus location tracking systems and payment gateways, requiring **GPS devices** for live location data and **payment terminals** for ticket processing. |
| **GPS Tracking Systems** | **GPS tracking devices** on buses will transmit real-time location data to the backend system. The system must support **communication protocols** like **MQTT** or **WebSocket** for real-time data transfer. |
| **Mobile Devices (User Side)** | The solution will be deployed on mobile devices running **Android** and **iOS** operating systems, using **cellular data** or **Wi-Fi** for communication with the backend systems. |
| **Server Communication** | The backend will communicate with external systems (e.g., payment gateways) via **HTTPS** with SSL encryption for secure communication. **API gateways** will be used to route traffic securely between the app and the backend. |
| **Payment Gateway Hardware** | The app must interface with third-party **payment terminals** (e.g., POS systems) in cities where offline payments are processed. The terminals must support **card swiping** or **QR code scanning**. |
| **Network Infrastructure** | **Reliable Internet connections (Wi-Fi/4G/5G)** will be needed for continuous communication between mobile devices, buses, and backend servers. |

### ****2. Data, Formats, Messages, and Transfer Schedules****

| **Component** | **Description** |
| --- | --- |
| **Data Formats** | The system will utilize **JSON** for REST API communication and **XML** for any legacy integrations with third-party systems (e.g., city transport systems). |
| **Ticket Data** | Ticket information will be stored in **JSON** format with fields such as ticket ID, user ID, route details, payment status, etc. |
| **Real-Time Bus Data** | Bus location updates will be received in **JSON** or **GeoJSON** format, including GPS coordinates, speed, and timestamps for accurate tracking. |
| **Messages for Notifications** | The app will send **SMS** and **Email notifications** for ticket bookings, reminders, or disruptions using **REST APIs**. **Twilio** or **SendGrid** services may be used for message delivery. |
| **Transfer Schedules** | Data related to bus schedules, stops, and routes will be transferred at predefined intervals (e.g., **every 15 minutes** for schedule updates and **every 5 seconds** for live bus location data). |
| **Payment Data** | Payment transaction details will be transferred in **secure HTTPS format** to ensure the **integrity** and **confidentiality** of sensitive user information. |
| **Batch Data Transfers** | For large datasets, like historical ticketing data or bulk updates (e.g., route changes), data will be transferred in **batch mode** using **FTP/SFTP** with **compression (e.g., ZIP)** to optimize bandwidth usage. |

### ****3. Performance and Capacity****

| **Component** | **Description** |
| --- | --- |
| **Scalability** | The system should be able to scale horizontally to handle **up to 10,000 concurrent users** during peak traffic times, including real-time ticket booking and bus tracking. |
| **Data Storage Capacity** | The system should handle and store **millions of transaction records** and **real-time GPS data**, ensuring smooth performance as the number of users and buses increases. |
| **API Performance** | The system's **REST APIs** should respond within **2 seconds** for normal user requests (e.g., checking routes, booking tickets) and within **5 seconds** for more complex requests (e.g., transaction processing). |
| **Latency Requirements** | Real-time bus data updates should have a latency of less than **3 seconds** to provide accurate and timely information to users. |
| **Server Capacity** | The system should be able to handle **peak loads of up to 1000 simultaneous transactions per second** with minimal performance degradation. |
| **Real-Time Data Processing** | The backend must process **real-time GPS data** and update bus locations for thousands of buses, ensuring smooth tracking for users with **no more than 5 seconds delay**. |
| **Backup and Recovery** | The system should support **daily backups** with a **2-hour recovery time** in case of failure. |

### ****4. Security Designs and Considerations****

| **Security Component** | **Description** |
| --- | --- |
| **Encryption** | All sensitive data, including **user payment details** and **personal information**, must be encrypted using **TLS/SSL** for transmission and **AES-256** for data at rest. |
| **Authentication** | The app must implement **multi-factor authentication (MFA)** for users, ensuring secure access to sensitive functionalities. |
| **Authorization** | **Role-Based Access Control (RBAC)** must be implemented on the backend to ensure only authorized users can access administrative functions (e.g., ticket validation, route management). |
| **Data Privacy** | The system must comply with **GDPR** and **CCPA** standards for data protection, ensuring that user data is anonymized where necessary and that users can request data deletion. |
| **Payment Gateway Security** | Payment data should be handled according to **PCI-DSS** standards to ensure the secure handling of credit card information during transactions. |
| **Third-Party Services** | All third-party integrations, such as **payment gateways**, **real-time tracking services**, and **SMS/Email providers**, must adhere to security best practices and provide adequate documentation of their compliance. |
| **Logging and Monitoring** | The system must implement **security logs** for tracking user actions, system errors, and potential security threats. Tools like **Splunk** or **ELK stack** can be used for log management. |

### ****5. Names of Reference Manuals and Documentation****

| **Documentation Component** | **Description** | **Location** |
| --- | --- | --- |
| **API Documentation** | Comprehensive guide to the app’s public and internal APIs, data formats, and usage examples. | [API Docs Location URL or internal repository] |
| **User Manual** | Detailed instructions for end-users on how to book tickets, view bus locations, and manage accounts. | [Location URL or internal help desk] |
| **System Architecture Docs** | Overview of system architecture, including integration points with third-party services (payment gateways, tracking systems, etc.). | [Internal SharePoint/Confluence] |
| **Security Policy** | Detailed documentation on security protocols, encryption standards, and compliance with industry regulations. | [Internal SharePoint/Confluence] |
| **Operational Manuals** | Guide for system administrators covering setup, monitoring, and maintenance procedures. | [Internal SharePoint/Confluence] |
| **Disaster Recovery Plan** | Plan for handling major system failures, including backup procedures, failover mechanisms, and recovery time. | [Internal SharePoint/Confluence] |
| **Compliance and Privacy Docs** | Documentation outlining adherence to GDPR, CCPA, and other privacy regulations. | [Internal SharePoint/Confluence] |
| **Payment Gateway Integration Docs** | Documentation for integrating and securing payment gateway systems. | [External or internal payment gateway docs] |

**5.2 Hardware Interfaces**

### ****1. Communication Hardware Interface****

| **Component** | **Description** |
| --- | --- |
| **Structure** | **GPS Devices** installed on buses, **Payment Terminals**, and mobile devices (user-side) as communication interfaces. **GPS devices** transmit real-time bus location data to the backend system, while **payment terminals** interact with mobile devices for ticket purchases. |
| **Location** | - **GPS Devices**: Installed in buses or transport vehicles, typically placed in the driver's cabin or near the vehicle's control systems. |
|  | - **Payment Terminals**: Positioned in stations or on buses where tickets are purchased. Can also be mobile-based for ticket validation and payment processing. |
|  | - **Mobile Devices**: Installed on the user-side, on **Android** or **iOS** devices where the app is installed. |
| **Activity** | - **GPS Devices** transmit bus location data in real-time to the backend servers at regular intervals (e.g., every 5 seconds) via **MQTT** or **WebSocket** protocols. |
|  | - **Payment Terminals** process ticket payments, validate payment information, and interface with **payment gateways** using secure communication protocols (e.g., **HTTPS**, **PCI-DSS** compliant). |
|  | - **Mobile Devices** enable users to book tickets, track buses, and receive notifications in real-time. The app communicates with backend systems to send booking data, payment info, and receive live location updates. |

### ****2. Server Hardware Interface****

| **Component** | **Description** |
| --- | --- |
| **Structure** | Backend servers that host the application's databases, API services, user data, and manage real-time interactions (e.g., ticket booking and tracking). |
| **Location** | - Servers are typically hosted in **data centers** or using **cloud infrastructure (AWS, Azure, etc.)**. The backend may consist of **separate instances** for web servers, database servers, and real-time data processing. |
| **Activity** | - **Processing Requests**: Servers handle requests from mobile devices, such as user logins, ticket bookings, and bus location updates. |
|  | - **Data Storage**: Servers store user profiles, ticket transactions, payment data, bus schedules, and real-time GPS information in databases. |
|  | - **API Management**: Servers provide RESTful APIs to interact with mobile devices, handle **real-time bus data**, and manage ticket transactions. |
|  | - **Payment Gateway Interaction**: Servers also interact with third-party **payment gateways** for processing payments in real-time. |
|  | - **Security Management**: Servers implement security protocols (SSL/TLS) to ensure that user data and payment information are transmitted securely. |

### ****3. Database Hardware Interface****

| **Component** | **Description** |
| --- | --- |
| **Structure** | **Database Servers** used to store application data, such as ticket bookings, user details, bus routes, payment transactions, and historical records. |
| **Location** | - Databases are generally hosted on either **on-premises** or **cloud-based** servers (e.g., **AWS RDS**, **Google Cloud SQL**, or **Microsoft Azure SQL Database**). |
| **Activity** | - **Data Storage**: Database stores critical data such as user profiles, bus schedules, transaction logs, and real-time GPS data for tracking buses. |
|  | - **Querying & Reporting**: The database is queried by the app for real-time or historical data, such as bus routes, schedules, and payment transaction histories. |
|  | - **Backup & Recovery**: The system should ensure **daily backups** of critical data and implement **disaster recovery** processes in the event of hardware failure. |

### ****4. User Interface Hardware****

| **Component** | **Description** |
| --- | --- |
| **Structure** | **Mobile Devices** (smartphones, tablets) that users interact with to book tickets and track buses. These devices will run the mobile app. |
| **Location** | - Users will access the app from **smartphones**, typically located in pockets or bags while traveling in the city. |
| **Activity** | - **Interaction with the App**: Users will interact with the mobile app for searching bus routes, booking tickets, and viewing bus locations in real-time. |
|  | - **Data Exchange**: The app communicates with the backend system to send booking requests, user data, and receive live updates on bus locations. |
|  | - **Notifications**: The app will send notifications about ticket status, bus arrival times, and any route changes. |

### ****5. Payment Gateway Hardware Interface****

| **Component** | **Description** |
| --- | --- |
| **Structure** | Payment processing hardware, such as **payment terminals** for point-of-sale (POS) systems and **third-party payment gateways** integrated into the app for online payments. |
| **Location** | - **Payment Terminals**: Typically located at bus stations, transit centers, or onboard buses for ticket purchases. |
|  | - **Payment Gateway Servers**: Hosted by third-party payment processors (e.g., **Stripe**, **PayPal**, **Square**) in secure data centers or cloud environments. |
| **Activity** | - **Payment Processing**: Payment terminals handle the transactions for ticket purchases, either through **card payments** or **QR code scanning**. |
|  | - **Secure Payment Gateway Communication**: Payment gateways encrypt payment information and send it securely to third-party processors for validation. |
|  | - **Data Transfer**: After payment validation, the gateway sends a confirmation response to the app to issue the digital ticket. |

### ****6. GPS Tracking Interface****

| **Component** | **Description** |
| --- | --- |
| **Structure** | **GPS receivers** installed in buses or vehicles transmit real-time location data to the backend system for tracking purposes. |
| **Location** | - **Buses/Transport Vehicles**: The GPS hardware is typically installed on the bus itself, in the vehicle's internal systems or tracking device. |
| **Activity** | - **Live Bus Location Updates**: GPS devices transmit location data (latitude, longitude) to backend servers every few seconds, allowing real-time updates on the bus location. |
|  | - **Route Tracking**: GPS devices also transmit **speed**, **direction**, and **timestamp** data, which helps users monitor the bus journey. |
|  | - **Data Transmission**: GPS devices use **cellular networks** or **Wi-Fi** to send data in real-time using secure protocols such as **MQTT** or **WebSockets**. |

**5.3 Software Interfaces**

### ****1. Mobile Application (User-side)****

| **Application Name** | **Owner** | **Interface Information** |
| --- | --- | --- |
| **City Transport Ticketing App** | **XYZ Software Solutions** | - **User Interface (UI)**: Provides a user-friendly interface for users to book tickets, view bus routes, and track buses in real-time. The interface is developed for both **Android** and **iOS** platforms. |
|  |  | - **API Interface**: The app communicates with the backend via **RESTful APIs** to perform actions like ticket booking, retrieving bus locations, and user authentication. |
|  |  | - **Payment Gateway Interface**: The app interfaces with third-party payment processors (e.g., **Stripe**, **PayPal**) to handle secure payment transactions for ticket purchases. |
|  |  | - **Notification Service Interface**: The app integrates with external services like **Twilio** or **Firebase** for sending SMS and push notifications related to ticket status or bus updates. |

### ****2. Backend System (Server-side)****

| **Application Name** | **Owner** | **Interface Information** |
| --- | --- | --- |
| **City Transport Backend System** | **XYZ Software Solutions** | - **API Interface**: The backend exposes **REST APIs** to communicate with the mobile application and external systems (e.g., payment gateway, bus tracking systems). |
|  |  | - **GPS Data Interface**: The backend receives real-time bus location data from **GPS tracking systems** through secure protocols like **MQTT** or **WebSocket**. |
|  |  | - **Database Interface**: The backend interacts with a **relational database** (e.g., **MySQL**, **PostgreSQL**) to store user data, ticket bookings, and bus schedules. |
|  |  | - **Payment Gateway Interface**: The backend system integrates with third-party payment providers (e.g., **Stripe**, **Square**) via **APIs** to securely process ticket payments. |

### ****3. Payment Gateway System****

| **Application Name** | **Owner** | **Interface Information** |
| --- | --- | --- |
| **Payment Gateway (e.g., Stripe, PayPal)** | **Third-party provider** | - **API Interface**: The payment gateway provides **REST APIs** that allow the backend to securely handle credit card transactions and process payments for ticket purchases. |
|  |  | - **Transaction Data Interface**: The payment gateway communicates transaction statuses (success, failure) and payment confirmation data back to the backend via secure APIs. |
|  |  | - **Secure Communication**: Data between the payment gateway and backend is encrypted using **TLS/SSL** protocols to protect sensitive payment data. |

### ****4. Real-Time Bus Tracking System (GPS System)****

| **Application Name** | **Owner** | **Interface Information** |
| --- | --- | --- |
| **Bus GPS Tracking System** | **Third-party provider** | - **Data Interface**: The GPS tracking system sends real-time **bus location data** (e.g., latitude, longitude, speed, and timestamp) to the backend via **MQTT** or **WebSocket** protocols. |
|  |  | - **API Interface**: The GPS system communicates with the backend via **API** to provide live updates on the bus’s current location and status. |
|  |  | - **Data Protocols**: The data is transmitted in **GeoJSON** or **JSON** format, allowing easy parsing and integration with the backend system. |

### ****5. Notification Service (SMS/Push Notification)****

| **Application Name** | **Owner** | **Interface Information** |
| --- | --- | --- |
| **Twilio / Firebase Cloud Messaging** | **Twilio / Google** | - **API Interface**: The notification service (e.g., **Twilio** for SMS or **Firebase Cloud Messaging** for push notifications) integrates with the backend system via **REST APIs** to send messages to users regarding ticket status, bus arrival times, or route changes. |
|  |  | - **Push Notification Interface**: For mobile notifications, the service communicates directly with mobile applications, sending timely alerts and updates. |
|  |  | - **Message Templates**: The backend uses predefined templates for sending ticket-related notifications, which are then personalized and sent through these platforms. |

### ****6. Database Management System (DBMS)****

| **Application Name** | **Owner** | **Interface Information** |
| --- | --- | --- |
| **MySQL / PostgreSQL** | **XYZ Software Solutions** | - **Database Interface**: The backend system interfaces with **MySQL** or **PostgreSQL** to store and retrieve user data, ticket transactions, route schedules, and bus tracking information. |
|  |  | - **SQL Queries**: Data is fetched from the database using **SQL queries** for operations such as booking tickets, checking bus availability, and validating transactions. |
|  |  | - **Data Security**: Sensitive data, such as user personal and payment information, is stored in **encrypted fields** using **AES-256 encryption** to comply with data protection standards. |

### ****7. Cloud Infrastructure and Hosting Services****

| **Application Name** | **Owner** | **Interface Information** |
| --- | --- | --- |
| **AWS / Azure / Google Cloud** | **Third-party cloud provider** | - **Cloud Hosting Interface**: The system is hosted on **cloud services** such as **AWS**, **Microsoft Azure**, or **Google Cloud**. The backend communicates with cloud storage, networking, and compute services through **APIs**. |
|  |  | - **Scalability Interface**: The system integrates with cloud services to automatically scale compute resources (e.g., **AWS EC2**, **Azure VM**) during high demand, ensuring the system can handle peak loads. |
|  |  | - **Data Storage Interface**: Cloud storage services (e.g., **Amazon S3**, **Google Cloud Storage**) are used to store large files such as historical transaction records, backup data, and logs. |
|  |  | - **Cloud Monitoring**: The cloud environment provides interfaces for **monitoring** and **logging** system performance, uptime, and error tracking. |

### ****8. City Transport Scheduling System****

| **Application Name** | **Owner** | **Interface Information** |
| --- | --- | --- |
| **City Transport Scheduling System** | **City Transport Authority** | - **API Interface**: The city’s transport scheduling system communicates with the backend to provide **live schedule updates**, **route changes**, and **bus availability** data. |
|  |  | - **Data Transfer**: Bus schedule data is transferred in **JSON** or **XML** format for integration into the app, allowing real-time tracking and scheduling for users. |
|  |  | - **Updates**: The backend system periodically queries the scheduling system for updates on bus routes, schedule changes, or service interruptions to provide users with the latest information. |

### ****9. Cloud Backup and Disaster Recovery System****

| **Application Name** | **Owner** | **Interface Information** |
| --- | --- | --- |
| **AWS Backup / Google Cloud Backup** | **Cloud Provider (AWS/Google)** | - **Data Backup Interface**: The system interfaces with **cloud backup services** like **AWS Backup** or **Google Cloud Backup** for regular **data snapshots** and disaster recovery. |
|  |  | - **Automatic Backup**: Interfaces with the system to automatically back up critical data (e.g., user profiles, ticket transactions) on a scheduled basis. |
|  |  | - **Restore Interface**: In case of failure, the backup system interfaces with the cloud provider to **restore** data, minimizing downtime and service disruption. |

**5.4 Communication Interfaces**

### ****1. Communication with Mobile Application (User-side)****

| **System/Device** | **Type of Communication** | **Method** |
| --- | --- | --- |
| **Mobile Application** | User interaction with backend | **RESTful APIs**: The mobile app communicates with the backend system via secure **HTTP/HTTPS** requests to perform actions like ticket booking, tracking buses, and user login. |
| **Real-Time Updates** | Receiving live bus data and notifications | **WebSocket** or **MQTT**: The mobile app receives real-time bus location data, ticket status updates, and notifications from the backend using **WebSocket** or **MQTT** protocols. |

### ****2. Communication with Payment Gateway (External System)****

| **System/Device** | **Type of Communication** | **Method** |
| --- | --- | --- |
| **Payment Gateway** (e.g., Stripe, PayPal) | Payment transaction processing | **RESTful APIs**: The backend communicates with third-party payment gateways (e.g., **Stripe**, **PayPal**) using **HTTPS** for secure transaction processing. |
| **Payment Authorization** | Validating payment transactions | **PCI-DSS Compliance**: Payment information is securely transmitted through **SSL/TLS** encryption, and payment status (approved/declined) is communicated back to the backend system. |

### ****3. Communication with GPS Tracking Devices (Real-Time Bus Location)****

| **System/Device** | **Type of Communication** | **Method** |
| --- | --- | --- |
| **GPS Tracking Devices** | Sending real-time bus location data | **MQTT** or **WebSocket**: The GPS devices installed on buses send real-time data (e.g., bus location, speed, and direction) to the backend system using lightweight **MQTT** or **WebSocket** protocols. |
| **GPS Data Transmission** | Continuous updates for bus location | **Cellular Networks** or **Wi-Fi**: GPS devices use **cellular data** or **Wi-Fi** to transmit location data in real-time to the backend servers, ensuring up-to-date location tracking. |

### ****4. Communication with Database Management System (DBMS)****

| **System/Device** | **Type of Communication** | **Method** |
| --- | --- | --- |
| **Database Management System** | Storing and retrieving user data, transactions, and bus schedules | **SQL Queries**: The backend system communicates with **MySQL** or **PostgreSQL** database servers using **SQL queries** over **secure connections** (e.g., **SSL/TLS**). |
| **Data Retrieval and Updates** | Storing booking data, user profiles, and payment transaction history | **Database Connection Pooling**: The backend maintains a persistent connection to the database for fast retrieval and update of data. **ACID** principles are used for transaction integrity. |

### ****5. Communication with City Transport Scheduling System****

| **System/Device** | **Type of Communication** | **Method** |
| --- | --- | --- |
| **City Transport Scheduling System** | Providing bus schedules and routes | **RESTful APIs**: The backend communicates with the city transport scheduling system to obtain real-time bus schedules and updates on any route or timing changes. |
| **Data Syncing** | Syncing schedule data for users | **JSON or XML**: Bus schedule data is transferred in **JSON** or **XML** format for easy integration with the backend, ensuring users get up-to-date travel information. |

### ****6. Communication with Notification Services (SMS/Push Notification)****

| **System/Device** | **Type of Communication** | **Method** |
| --- | --- | --- |
| **Twilio/Firebase Cloud Messaging** | Sending SMS or push notifications | **RESTful APIs**: The backend uses **Twilio API** (for SMS) or **Firebase Cloud Messaging API** (for push notifications) to send ticket status, bus arrival times, and route changes to users. |
| **Notification Delivery** | Message delivery (SMS, push notifications) | **Push Notification Service (Firebase)** or **SMS Gateway (Twilio)**: Notifications are sent to users' devices using secure communication channels such as **HTTPS** or **SMS protocols**. |

### ****7. Communication with Cloud Infrastructure and Hosting Services****

| **System/Device** | **Type of Communication** | **Method** |
| --- | --- | --- |
| **Cloud Hosting Services** | Hosting the backend and databases | **Cloud APIs**: The backend communicates with cloud services (e.g., **AWS**, **Google Cloud**, **Azure**) using their **RESTful APIs** for **resource provisioning**, **data storage**, and **scalability management**. |
| **Cloud Backup & Recovery** | Backing up and recovering data | **Cloud Backup APIs**: Communication with **cloud backup** services for **data snapshots** and **restoration** in case of system failure or disaster. |
| **Cloud Storage (e.g., AWS S3)** | Storing large files (e.g., logs, backups) | **RESTful APIs**: Backend system communicates with **cloud storage** services to upload or retrieve data using **HTTP/HTTPS** protocols. |

### ****8. Communication with Real-Time Data Processing System (Live Updates)****

| **System/Device** | **Type of Communication** | **Method** |
| --- | --- | --- |
| **Real-Time Data Processing System** | Processing live bus data and transactions | **WebSocket** or **MQTT**: Real-time bus location data and live updates are processed and sent from the GPS system and backend to mobile apps using **WebSocket** or **MQTT**. |
| **Data Processing for Updates** | Broadcasting live updates to apps | **Pub/Sub Model**: The system uses the **publish/subscribe** model to send live updates to multiple users concurrently. |

### ****9. Communication with External Regulatory Systems (if applicable)****

| **System/Device** | **Type of Communication** | **Method** |
| --- | --- | --- |
| **Regulatory Compliance System** | Reporting and tracking data (e.g., financial or operational) | **Secure APIs**: The backend communicates with regulatory systems (if required) to submit **transaction logs**, **financial data**, and **operational reports** in a secure manner, ensuring compliance with industry regulations. |
| **Audit Logging** | Data transfer for auditing purposes | **Encrypted File Transfer**: Data may be transferred via **SFTP**, **HTTPS**, or other secure file transfer methods to ensure compliance with audit and legal standards. |

### ****10. Communication with External System Monitoring Tools****

| **System/Device** | **Type of Communication** | **Method** |
| --- | --- | --- |
| **System Monitoring Tools** | Monitoring system health and status | **REST APIs/Cloud Monitoring Services**: External monitoring tools (e.g., **AWS CloudWatch**, **Datadog**, or **Prometheus**) monitor the system’s performance, providing data to the backend via **REST APIs** or direct integration with cloud services. |
| **Incident Alerts** | Alerting administrators about system issues | **Webhooks** or **Push Notification Services**: Alerts are sent to system administrators in case of any downtime or anomalies detected in the system using **webhooks** or **cloud push notification services**. |

**6 Glossary**

|  |  |
| --- | --- |
| **Term Definition** | |
| Functional  Requirements | Functional requirements define the internal workings of the software, i.e., the calculations, technical details, data manipulations, and other specific functionality that show how the events are to be satisfied.  The core of the requirement is the description of the required behavior, which must be a clear and readable description of the required behavior. This behavior can come from organizational or business rules, or it can be discovered through working sessions with users, stakeholders, and other experts within the organization.  Functional requirements generally contain a unique name and number, a brief summary, and reason for it. This information is used to help the reader understand why the requirement is needed, and to track the requirement through the development of the system.  Functional requirements are supported by non-functional requirements, which impose constraints on the design or implementation. |
| Non-Functional  Requirements | Non-functional requirements specify criteria that can be used to judge the operation of a system, rather than specific behaviors. They impose constraints on the design or implementation.  Typical non-functional requirements are performance, reliability, security, scalability, and cost. Other terms for non-functional requirements are "constraints", "quality attributes" and "quality of service requirements”. |