| **Software Requirement Specifications**  **FastTransit**  **Version: [1.0]**   | Code | CS3009 | | --- | --- | | Supervisor | Rubab Jaffer | | Co Supervisor | - | | Project Team | Syed Uzair Hussain Zaidi - 22K4212  Basil Yaqoob - 22K4634 | | Submission Date | 6/5/2025 | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

**Document History**

[Revision history will be maintained to keep a track of changes done by anyone in the document.]

| **Version** | **Name of Person** | **Date** | **Description of change** |
| --- | --- | --- | --- |
| 1.0 | Basil | 6/5/2025 | Document Created |
| 2.0 | Syed Uzair Hussain Zaidi | 6/5/2025 | Refined Document |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Distribution List**

[Following table will contain list of people whom the document will be distributed after every sign-off]

| **Name** | **Role** | |
| --- | --- | --- |
| Rubab Jaffer | | Supervisor |
| - | | Co- Supervisor |
|  | |  |

**Document Sign-Off**

[Following table will contain sign-off details of document. Once the document is prepared and revised, this should be signed-off by the sign-off authority.

Any subsequent changes in the document after the first sign-off should again get a formal sign-off by the authorities.]

| **Version** | **Sign-off Authority** | **Sign-off Date** |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Table of Contents**

[*1.*](#_mvu00rna9fck) *Introduction 8*

[1.1.](#_xsrnlh7kk354) Purpose of Document 8

[1.2.](#_yvedx1uvhufj) Intended Audience 8

[1.4](#_nc7izb117zyc) Document Convention 9

[*2.*](#_3358yg4s7eb6) *Overall System Description 10*

[2.1.](#_55yie5obtohn) Project Background 10

[2.2.](#_pro87b2iutxv) Project Scope 10

[2.3.](#_1z36pi1ncvqy) Not In Scope 10

[2.4.](#_fis4ox2ws4rz) Project Objectives 10

[2.5.](#_fkmvp587uvsi) Stakeholders 11

[2.6.](#_ua4e4lj21ie6) Operating Environment 11

[2.7.](#_afqp8zt303aq) System Constraints 11

[2.8.](#_3bxcjvngbcvv) Assumptions & Dependencies 12

[*3.*](#_2eqnfq7ix2w8) *External Interface Requirements 13*

[3.1.](#_xd4y91ss6n0r) Hardware Interfaces 13

[3.2.](#_4j86s8exdwv6) Software Interfaces 13

[3.3.](#_3tmtouanz56m) Communications Interfaces 13

[*4.*](#_c75zkqms1las) *Functional Requirements 14*

[4.2.](#_hw1vj3slyuo1) Use Cases 16

[16](#_b9iwqd8pjjbv)

[4.2.1.](#_ob1xehmp88pb) [Fare Payment via Stripe] 17

[*5.*](#_hf41b9yl6urc) *Non-functional Requirements 18*

[5.2.](#_gto3uzkc3wsp) Performance Requirements 18

[5.3.](#_qxgw4ufhzdqr) Safety Requirements 18

[5.4.](#_ivk9iknikgox) Security Requirements 18

[5.5.](#_fv5qhlwkmjwu) User Documentation 18

[*6.*](#_47zcpiik7foc) *References 19*

[*7. Appendices 20*](#_adfcblfjdhg8)

1. **Introduction**
   1. **Purpose of Document**

The purpose of this document is to define the functional and non-functional requirements for the FastTransit Point Management System. It provides a detailed description of the system’s capabilities, constraints, and behavior. This document will serve as a foundation for system design, development, testing, and validation. It ensures all stakeholders have a common understanding of what the system will do.

* 1. **Intended Audience**

This document is intended for the following audiences:

* **Project Developers:** To understand the system requirements clearly.
* **Designers & Architects:** To refer for designing the system architecture.
* **Testers:** To design and execute tests based on the documented requirements.
* **Project Manager:** To track project progress and ensure deliverables align with specifications.
* **Stakeholders (University Faculty or Transport Authority):** To validate if the system meets their expectations and needs.

**1.3 Abbreviations**

| **Abbreviation** | **Description** |
| --- | --- |
| SRS | Software Requirements Specification |
| CRUD | Create, Read, Update, Delete |
| UI | User Interface |
| DB | Database |
| ID | Identifier |
| OTP | One-Time Password |
| API | Application Programming Interface |
| PK | Primary Key |
| FK | Foreign Key |

* 1. **Document Convention**

This document uses the following formatting conventions:

* **Font:** Times New Roman
* **Font Size:**
  + Headings: 14 pt, Bold
  + Body Text: 12 pt, Regular
* **Paragraph Spacing:** 1.15 line spacing
* **Page Margins:** 1 inch on all sides

1. **Overall System Description**
   1. **Project Background**

Public transport systems in urban areas often lack a streamlined and digitalized fare management and user tracking system. Manual processes result in long wait times, inefficient bus tracking, and a poor commuting experience. The FastTransit project was initiated to solve these problems by developing a digital Point Management System. The goal is to enhance commuter convenience, reduce manual operations, and offer a unified platform for users, transporters, and administrators to interact efficiently.

* 1. **Project Scope**

The FastTransit system aims to:

* Allow users to register and login securely.
* Enable users to select their preferred stop and route.
* Let users track buses in real time.
* Integrate online fare payment via Stripe.
* Generate a Point ID card after payment confirmation.
* Allow transporters to update bus location and availability.
* Give administrators the ability to manage all users, transporters, routes, and system settings.

**Project Boundaries:**

* Web-based system (desktop and mobile responsive)
* Focused on city/urban transportation network
* Only handles the backend logistics for travel point/fare systems—not the actual transport operation.

* 1. **Not In Scope**

The following features are not part of the current scope:

* Mobile App development (native Android/iOS)
* AI-based route prediction or dynamic rerouting
* Offline payment systems or cash handling
* Real-time voice or GPS tracking beyond static updates
* Integration with biometric ID systems
  1. **Project Objectives**
* Automate and digitize point fare systems.
* Offer secure user and transporter registration.
* Enable efficient bus assignment and tracking.
* Integrate a reliable payment system for fare collection.
* Ensure that passengers receive proof of payment and route eligibility via a generated point ID.
* Provide admin control for monitoring and managing system-wide activities.
  1. **Stakeholders**

| **Stakeholder** | **Role** |
| --- | --- |
| **Admin** | Controls and manages the entire system including users, transporters, and routes. |
| **User (Passenger)** | Uses the system to choose stops, pay fare, and get a travel ID. |
| **Transporter (Driver/Operator)** | Updates route, trip status, and interacts with assigned users. |
| **Developers** | Build and maintain the software system. |
| **University/Instructor** | Evaluate the project for academic purposes. |

* 1. **Operating Environment**

 **Platform:** Web-based application

 **Backend:** Django (Python)

 **Frontend:** HTML, CSS, JavaScript

 **Database:** MySQL

 **Payment Gateway:** Stripe

 **Browser Compatibility:** Chrome, Firefox, Edge

 **Network:** Requires internet connection for real-time updates and payments

* 1. **System Constraints**

 **Software Constraints:** Must use Django as backend framework; Stripe for payment

 **Hardware Constraints:** Web access required (PC or smartphone); transporters must have devices to update status

 **Cultural Constraints:** English will be the primary language (no multilingual support in this phase)

 **Legal Constraints:** Compliance with local transport payment and privacy laws

 **Environmental Constraints:** Should be operable in noisy or busy environments (UI must be intuitive and simple)

 **User Constraints:** Interface should be simple enough for non-technical users

 **Third-Party Component Constraints:** Stripe integration requires user to have a debit/credit card

* 1. **Assumptions & Dependencies**

 Assumes users have access to internet and valid email/phone for registration

 Assumes transporters will regularly update bus information

 Depends on third-party APIs (e.g., Stripe) remaining stable and available

 Depends on admin users for data verification and transporter assignment

 Assumes project will be evaluated and used in an academic or pilot setting first

1. **External Interface Requirements**

This section outlines how the FastTransit Point Management System interacts with various external components including hardware devices, other software systems, and communication protocols. These interfaces are essential to ensure that the system can operate within a real-world environment with seamless integration and reliable data exchange. The following subsections define the major interfaces at a high level of abstraction.

* 1. **Hardware Interfaces**

The FastTransit system is primarily web-based and thus depends on end-user hardware devices like personal computers, laptops, tablets, and smartphones to access the system through a standard web browser. There are no complex or specialized hardware dependencies. For the transporters (bus drivers or operators), any internet-enabled device like an Android smartphone or tablet can be used to log into the system and update trip status and location. The system requires only basic I/O interaction—keyboard, mouse or touchscreen—for user input and screen output. Additionally, hardware compatibility assumes access to a camera or printer in case the system is extended to generate printable or scannable point ID cards.

* 1. **Software Interfaces**

The backend of the system is developed using Django (Python), which communicates with a relational database management system such as PostgreSQL or MySQL to store and retrieve data. The frontend interfaces through a web browser using standard technologies such as HTML5, CSS3, and JavaScript. The system integrates with the Stripe API (version 2023 or later) to process online payments securely. Stripe handles sensitive payment information and communicates the transaction results back to the application using RESTful API responses in JSON format. The system is also dependent on Django’s built-in user authentication modules and third-party libraries for session handling and form validation. It may further utilize open-source packages such as django-rest-framework to expose certain API endpoints. The software interface must support secure and consistent data exchange between the frontend, backend, database, and external payment gateway.

* 1. **Communications Interfaces**

The FastTransit system will operate entirely through a client-server model over the Internet. Users, transporters, and admins will interact with the system through a web browser, requiring support for HTTP/HTTPS protocols. HTTPS is mandatory to ensure secure data transfer, especially during authentication and payment transactions. The system should support standard email protocols (SMTP) if email verification or notifications are integrated in the future. Communication with third-party services like Stripe will follow RESTful API architecture using JSON data format. The system should also be compatible with TCP/IP networking standards and must allow for scalability and cloud-based deployment if needed. Data synchronization between user actions and database updates should happen in real-time, and caching mechanisms may be used for performance optimization. Security considerations such as encryption, session tokens, and CSRF protection are enforced using Django's default settings and additional middleware.

1. **Functional Requirements**

**4.1. Functional Hierarchy**

The FastTransit system is divided into multiple core modules, each serving specific roles to ensure efficient point management and transit operations. Below is the hierarchical structure of the main functions and their sub-functions:

**4.1.1. Admin Module**

* **1.1. Manage Users**
  + View registered users
  + Delete or suspend accounts
* **1.2. Manage Transporters**
  + Approve or reject transporter registration
  + View transporter details
* **1.3. Manage Stops and Routes**
  + Add/Edit/Delete stops
  + Assign routes to buses
* **1.4. Fare Management**
  + Define fare for each route
  + Update fare rules
* **1.5. Voucher & Point ID Management**
  + Generate voucher templates
  + Issue point ID cards
* **1.6. Reporting**
  + Generate daily/weekly reports
  + Download payment and user activity logs

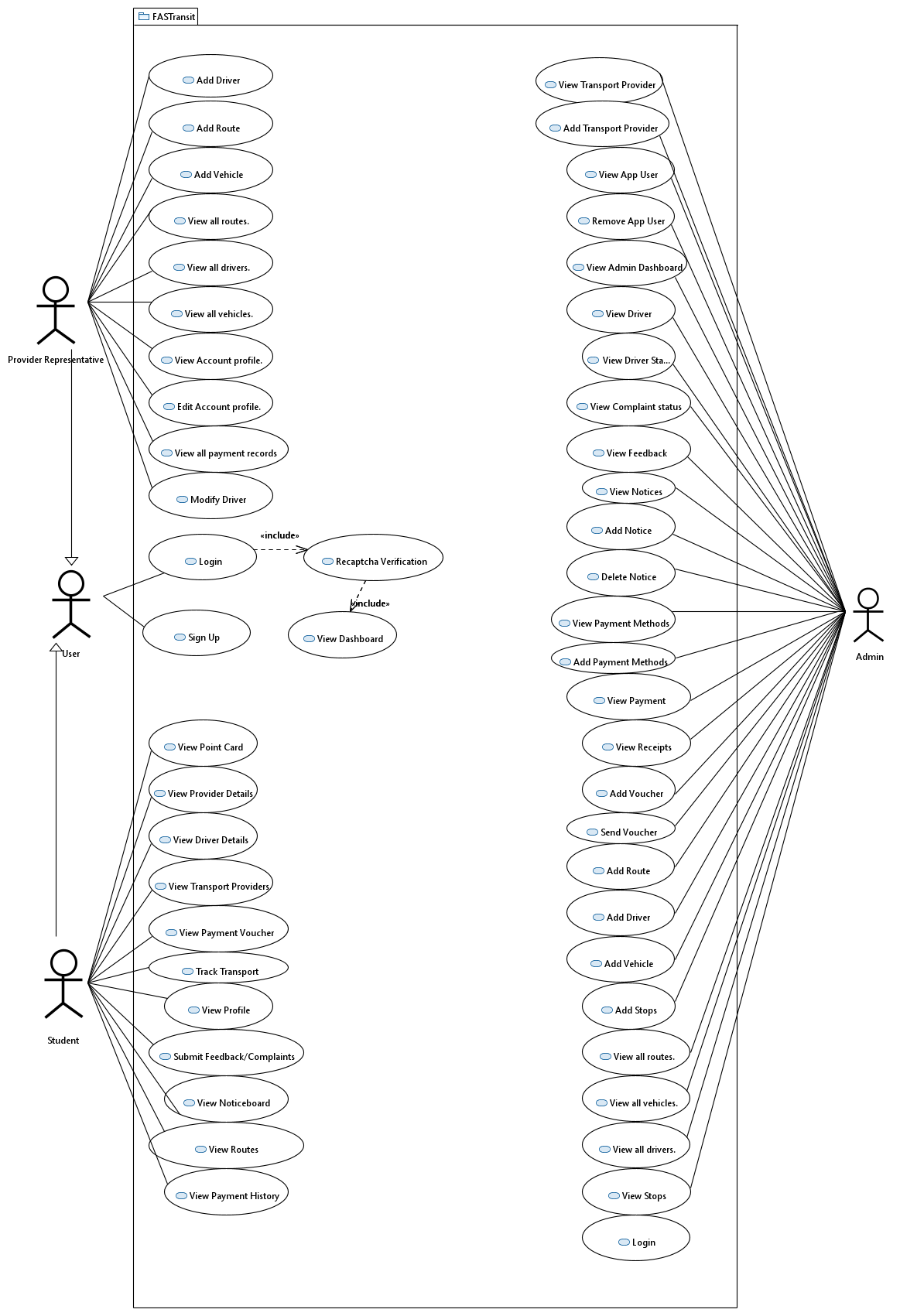
**4.1.2. Transporter Module**

* **2.1. Profile Management**
  + View and update personal and vehicle details
* **2.2. Trip Management**
  + Start or end a route trip
  + Mark route checkpoints
* **2.3. Bus Tracking**
  + Real-time location updates
  + Stop status updates
* **2.4. Payment & Fare Collection**
  + Scan voucher or receive fare
  + View earnings summary

**4.13. User Module**

* **3.1. Account Management**
  + Register/Login
  + Update profile details
* **3.2. Route & Bus Discovery**
  + Search for nearest stops
  + View available buses and estimated arrival time
* **3.3. Fare Payment**
  + Pay fare using Stripe
  + Generate and scan digital voucher
* **3.4. Point ID Card**
  + View and download digital point ID card
  + Show ID for validation
* **3.5. Trip History**
  + View past trips
  + Download trip receipts

* 1. **Use Cases**

****

* + 1. **[Fare Payment via Stripe]**

| **Use Case ID: UC01 - Fare Payment via Stripe** | | | | |
| --- | --- | --- | --- | --- |
| **Use case Id:** | | UC01 | | |
| **Actors:**  User, Stripe Payment Gateway | | | | |
| **Feature:** Fare Payment | | | | |
| **Pre-condition:** | | User must be logged in and have selected a route and bus | | |
| **Scenarios** | | | | |
| **Step#** | **Action** | | | **Software Reaction** |
| **1.** | User selects a bus and route. | | | System displays fare and asks for confirmation. |
| **2.** | User selects a bus and route. | | | System opens Stripe payment gateway. |
| **3.** | User enters card details and confirms. | | | System processes payment and shows confirmation. |
| **4.** | User receives a payment receipt. | | | System updates trip history and generates a voucher |
| **Alternate Scenarios:** Write additional, optional, branching or iterative steps. Refer to specific action number to ensure understandability. | | | | |
| **1a: If the user cancels before payment, the system navigates back to route selection.**  **2a: If the payment fails, the system displays an error and prompts a retry or change method.** | | | | |
| **Post Conditions** | | | | |
| **Step#** | **Description** | | | |
| **1.** | Fare is successfully paid. | | | |
| **2.** | User receives a digital receipt. | | | |
| **3.** | System generates a voucher for this trip. | | | |
| **4.** | Trip details are saved in user history. | | | |
|  | | |  | |

1. **Non-functional Requirements**
   1. **Performance Requirements**

The FastTransit system is designed to be responsive, reliable, and efficient under normal and peak usage. The application should load web pages within 2 seconds for users with a stable internet connection. It must support multiple concurrent users (at least 200 active sessions) without performance degradation. Critical operations such as login, route tracking, fare payments, and voucher generation should complete within 3 seconds. The backend server and database are expected to handle thousands of records efficiently, with appropriate indexing and optimized queries. System uptime should be at least 99.5%, ensuring availability for users, transporters, and admins during working hours. The platform should also be scalable so that performance does not degrade as user base or data volume increases.

* 1. **Safety Requirements**

The FastTransit system does not directly interface with life-critical components; however, it handles financial transactions and user information, which introduces the risk of financial loss or privacy violations if misused. The system must have built-in safeguards to prevent unauthorized actions such as data tampering, duplicate fare charging, or deletion of records without admin approval. Important operations such as payment processing must include confirmation prompts to avoid accidental transactions. In case of errors or unexpected shutdowns, the system must preserve data integrity through transaction rollbacks and error logging. Regular backups of the database should be scheduled to prevent data loss in the event of system failure.

* 1. **Security Requirements**

To maintain the security and privacy of user data, the system must implement role-based access control (RBAC). Users, transporters, and admins should each have distinct permissions. All user data, especially passwords, must be securely stored using hashing and salting techniques. Sessions must be managed securely using Django's built-in CSRF tokens and secure cookies. Payment information is never stored by the application and is instead processed directly via the Stripe payment gateway. All API interactions must be done over HTTPS to prevent man-in-the-middle attacks. The system should also implement input validation to prevent SQL injection, cross-site scripting (XSS), and other web-based threats. Logs of user activities should be maintained to ensure traceability.

* 1. **User Documentation**

The FastTransit system will be delivered with complete user documentation to ensure ease of use and onboarding. This includes a comprehensive User Manual describing how to use all modules (User, Admin, Transporter), system prerequisites, and frequently asked questions. Online Help will be accessible through tooltips or modals within the application for contextual guidance. A Quick Start Guide will be provided for first-time users to learn system basics like registration, stop selection, and fare payment. Additional tutorial videos or walkthroughs may be included on the system’s landing page or help section to support less tech-savvy users.

1. **References**

The following documents, tools, and standards were referred to during the creation of this Software Requirements Specification (SRS) for the FastTransit Point Management System:

* IEEE Std 830-1998 – IEEE Recommended Practice for Software Requirements Specifications. Institute of Electrical and Electronics Engineers, 1998.

Available at: <https://ieeexplore.ieee.org>

* Django Documentation (Version 5.0) – The Web framework for perfectionists with deadlines.

Available at: <https://docs.djangoproject.com/>

* Stripe API Reference (2025) – Online Payment Gateway Integration Guide. Stripe Inc.

Available at: <https://stripe.com/docs/api>

* SRS Guidelines – FAST University – Course CS3009 Software Engineering SRS Guidelines, FAST-NUCES Department of Computer Science, 2025.

Instructor-provided material.

* ISO/IEC/IEEE 29148:2018 – Systems and Software Engineering – Life Cycle Processes – Requirements Engineering.

Available at: <https://www.iso.org>

## **7. Appendices**

This section includes supplementary and supporting information that is useful for understanding the SRS document but is too detailed to be placed in the main body.

**7.1. Glossary**

* **CRUD** – Create, Read, Update, Delete operations in a database.
* **SRS** – Software Requirements Specification.
* **UI** – User Interface.
* **DAL** – Data Access Layer.
* **PK** – Primary Key.
* **FK** – Foreign Key.
* **API** – Application Programming Interface.
* **OTP** – One-Time Password.
* **POS** – Point of Sale.

**7.2. Diagrams and Models**

* **System Architecture Diagram** – Depicts the User Interface Layer, Middle Tier (Business Logic), and Data Access Layer and their interactions.
* **Context Diagram** – Shows the system's interaction with external entities like Admin, Users, and Transporters.
* **Use Case Diagram** – Illustrates the primary interactions of actors with the system.
* **ER Diagram** – Entity Relationship diagram representing database structure.
* **State Diagram** – Shows transitions between states in the Point Management workflow.
* **Activity Diagram** – Demonstrates the flow of operations (e.g., User registration, ticket booking).

**7.3. Tools and Technologies**

* **Backend**: Django (Python)
* **Frontend**: HTML5, CSS3, JavaScript (with Bootstrap or React)
* **Database**: PostgreSQL / SQLite (during development)
* **Version Control**: Git with GitHub
* **Project Management**: Jira Kanban Board
* **Deployment**: Pythonanywhere.com / AWS