

**ADDIS ABABA UNIVERISITY**

**INSTITUTE OF TECHNOLOGY**

**DEPARTMENT OF SOFTWARE ENGINEERING**

**Addis Ababa Transport Management System Software Design Specification**

***Team Members:***

**ERMIAS DEREJE:ATE/4952/15**

**YASSIN RAHMETO:ATE/6776/15**

**JILO SHULE:ATE/2317/15**

**YONATHAN BIRHANU:ATE/8681/15**

***Date:APR11/2025***

**Table of Contents**

List of Tables ................................................. ii

List of Figures ................................................ iii

Definitions, Acronyms, Abbreviations ............ iv

1. Introduction

1.1 Purpose

1.2 General Overview

1.3 Development Methods & Contingencies

2. System Design Model

2.1 Subsystem Decomposition

2.2 Hardware/Software Mapping

2.3 Access Control

3. Object Model

3.1 Class Diagram

3.2 Sequence Diagram

3.3 State Chart Diagram

4. Detailed Design

References

* **1. Introduction**
  + **1.1 Purpose**
  + **1.2 General Overview**
  + **1.3 Development Methods & Contingencies**
* **2. System Design Model**
  + **2.1 Subsystem Decomposition**
  + **2.2 Hardware/Software Mapping**
  + **2.3 Access Control**
* **3. Object Model**
  + **3.1 Class Diagram**
  + **3.2 Sequence Diagram**
  + **3.3 State Chart Diagram (Optional)**
* **4. Detailed Design**
* **References**

**Definitions, Acronyms, Abbreviations**

* **AATB - Addis Ababa Transport Bureau**
* **API - Application Programming Interface**
* **AWS - Amazon Web Services**
* **UI - User Interface**
* **UX - User Experience**
* **DB - Database**

1. ***Introduction***
   1. **Purpose**

**This document defines the software design of the Addis Ababa Transport Management System. It translates the requirements in the SRS into an architectural and detailed technical design to guide development.**

* 1. **General Overview**

**The system consists of web and mobile platforms integrated into a cloud-based infrastructure. Major functionalities include vehicle registration, traffic monitoring, ticketing, and data analytics. The architecture supports modular design and scalable deployment.**

* 1. **Development Methods & Contingencies**
     + **Method: Object-oriented design using UML, Agile methodology**
     + **Tools: Django, Node.js, React, PostgreSQL**
     + **Contingencies:**
       - **Delayed third-party API integration (Google Maps, Payment Gateway)**
       - **Government policy changes**
       - **Low-bandwidth fallback mechanisms**

1. **System Design Model**
   1. **Subsystem Decomposition Subsystems:**
      * **Vehicle Registration**
      * **Public Transport Monitoring**
      * **Ticketing & Commuter App**
      * **Analytics Dashboard**
      * **Authentication & Authorization**
   2. **Hardware/Software Mapping**

**Deployment on AWS or Azure. Components mapped:**

* + - **Web Server (Node.js/React)**
    - **Backend Server (Django)**
    - **Database (PostgreSQL)**
    - **Mobile Apps (Android/iOS)**
  1. **Access Control**
     + **Role-based access: Admin, Staff, Commuter**
     + **OAuth 2.0 for authentication**
     + **Encrypted session management**

### ****Access Control Matrix****

| **Role** | **Register Vehicle** | **Purchase Ticket** | **Generate Report** | **View Analytics** |
| --- | --- | --- | --- | --- |
| Admin | ✅ | ❌ | ✅ | ✅ |
| Staff | ✅ | ❌ | ❌ | ✅ |
| Commuter | ❌ | ✅ | ❌ | ❌ |

1. **Object Model**

**1, user (Abstract Class)**

**.SubClass:- Admin,Staff,Commuter**

**2, Vehicle**

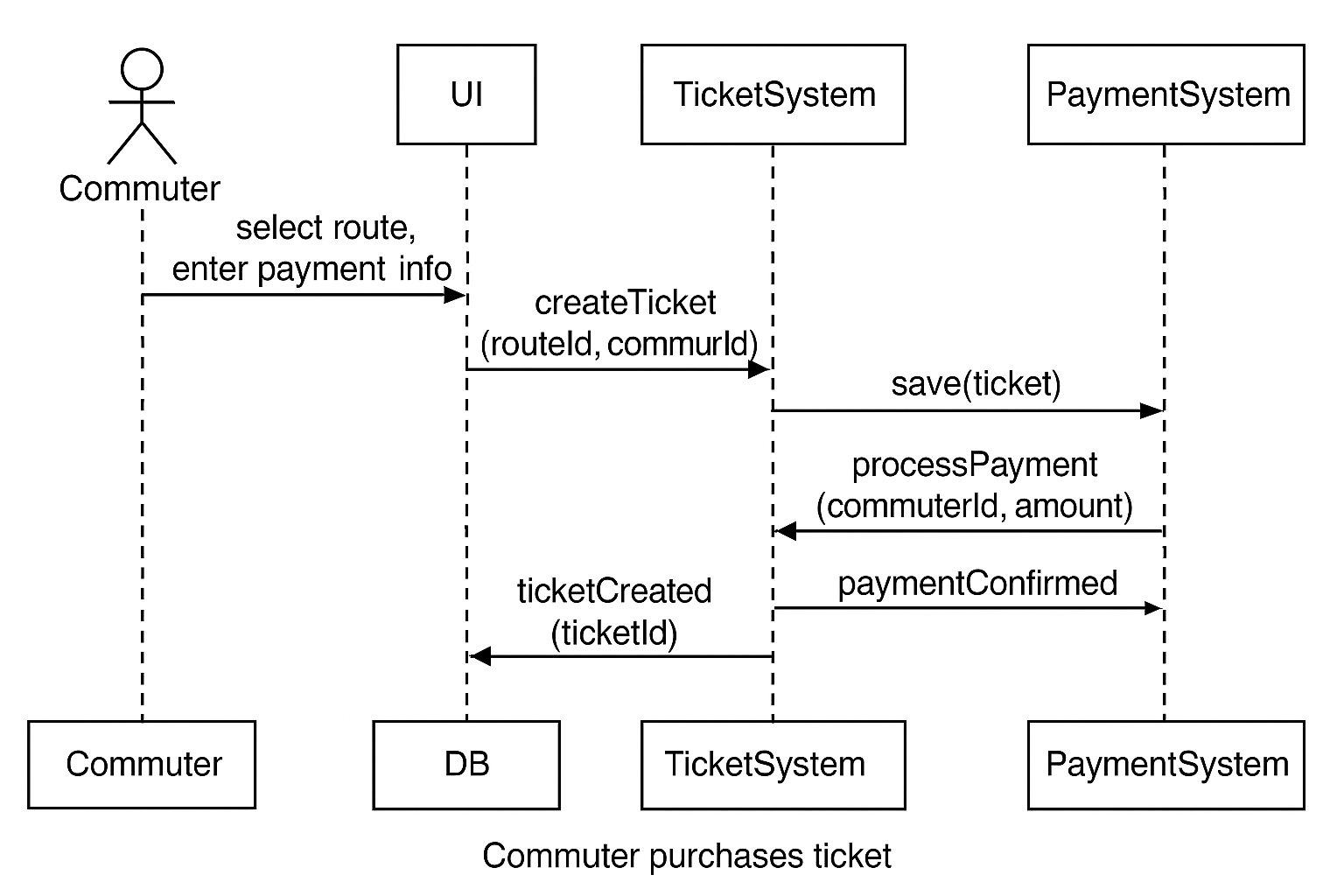
**3, TransportRoute**

**4, Ticket**

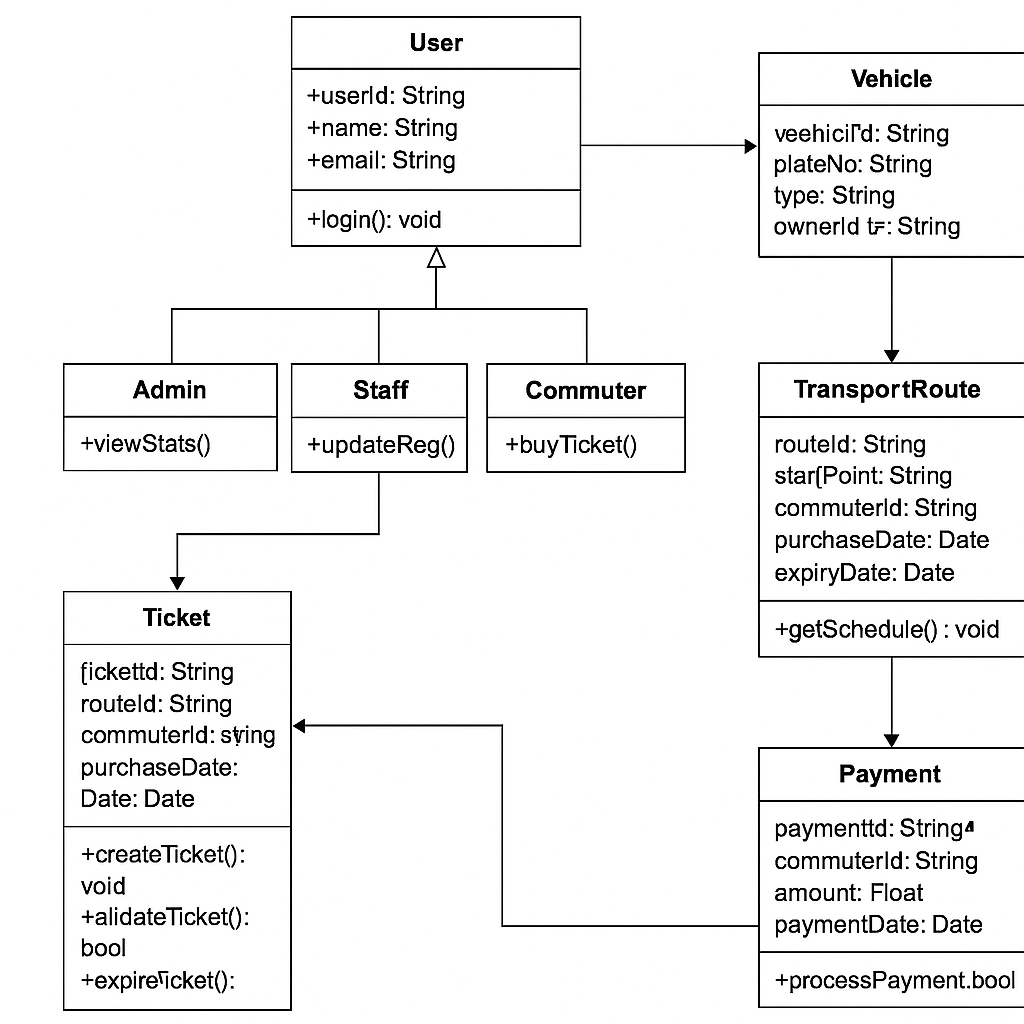
**5, AnalyticsReport**

**6, Payment**

**SEQUENTIAL DIAGRAM**

****

* 1. **Class Diagram**

****

***3.2 Classes:***

* + - **User: Admin, Staff, Commuter (abstract)**
    - **Vehicle**
    - **TransportRoute**
    - **Ticket**
    - **AnalyticsReport**
    - **Payment**

**3.3 Sequence Diagram Scenarios:**

* + - **Commuter purchases ticket**
    - **Staff updates vehicle registration**
    - **Admin views analytics**

**3.4 State Chart Diagram (optional) For objects like Ticket:**

**States = Created -> Paid -> Validated -> Expired**

1. **Detailed Design**

**CREATE TABLE users (**

**user\_id UUID PRIMARY KEY,**

**name TEXT NOT NULL,**

**email TEXT UNIQUE NOT NULL,**

**phone TEXT,**

**password\_hash TEXT,**

**role TEXT CHECK(role IN ('Admin', 'Staff', 'Commuter'))**

**);**

**CREATE TABLE vehicles (**

**vehicle\_id UUID PRIMARY KEY,**

**plate\_number TEXT UNIQUE,**

**type TEXT CHECK(type IN ('Bus', 'Taxi', 'Minibus')),**

**owner\_id UUID REFERENCES users(user\_id),**

**registration\_date DATE**

**);**

**CREATE TABLE tickets (**

**ticket\_id UUID PRIMARY KEY,**

**route\_id UUID REFERENCES transport\_routes(route\_id),**

**commuter\_id UUID REFERENCES users(user\_id),**

**purchase\_date DATE,**

**expiry\_date DATE,**

**status TEXT CHECK(status IN ('Created', 'Paid', 'Validated', 'Expired'))**

**);**

### 🔧 ****4. Detailed Design – Extended****

#### ✅ 4.1 ****Class Details****

##### ****User (Abstract Class)****

**Attributes**

userId: String — Unique identifier

name: String

email: String

phone: String

role: Enum {Admin, Staff, Commuter}

passwordHash: String

*****Operations*****

login()

logout()

updateProfile()

##### ****Vehicle****

**Attributes**

vehicleId: String — Unique, auto-generated

plateNumber: String

type: Enum {Bus, Taxi, Minibus}

ownerId: String — Links to User

registrationDate: Date

**Operations**

registerVehicle()

updateRegistration()

fetchDetails()

##### ****TransportRoute****

****Attributes****

routeId: String

origin: String

destination: String

stops: List<String>

fare: Float

****Operations****

createRoute()

updateRoute()

calculateDistance()

##### ****Ticket****

**Attributes**

ticketId: String

routeId: String

commuterId: String (Private, Encrypted)

purchaseDate: Date

expiryDate: Date

status: Enum {Created, Paid, Validated, Expired}

**Operations**

createTicket()

validateTicket()

expireTicket()

refundTicket()

##### ****AnalyticsReport****

**Attributes**

reportId: String

type: Enum {Usage, Revenue, Traffic}

generatedBy: Admin

dateGenerated: Date

data: JSON

**Operations**

generateReport()

exportCSV()

filterByDate()

##### ****Payment****

**Attributes**

paymentId: String

ticketId: String

amount: Float

status: Enum {Pending, Completed, Failed}

method: Enum {Card, MobileMoney, Wallet}

**Operations**

initiatePayment()

confirmPayment()

retryPayment()

### 🔄 ****4.2 Sequence Scenarios (Detailed)****

#### Scenario: Commuter Purchases Ticket

Commuter logs in

Views available routes

Selects route and ticket

Initiates payment

System creates ticket → updates status to Paid

Sends confirmation

#### Scenario: Staff Updates Vehicle Registration

Staff logs in

Navigates to registration module

Searches vehicle by plate

Updates information

System saves and logs changes

### 📶 ****4.3 Data Flow****

**Frontend (React):** Sends REST/GraphQL requests to the backend

**Backend (Django):** Processes logic, accesses PostgreSQL

**Mobile App:** Communicates with same backend APIs

**Cloud Services:** AWS RDS (PostgreSQL), EC2, S3, etc.