Software Requirements Specification (SRS)

Name: Nikil G S
For: City Transportation Portal

1. Introduction

1.1 Purpose

The purpose of this project is to develop a web-based City Transportation Portal that allows citizens, commuters, and administrators to access real-time public transport information. The system provides live bus, metro, and bike tracking, trip planning, feedback collection, and transport-related updates to ensure a transparent and efficient urban transportation experience.

1.2 Scope

The City Transportation Portal is a role-based system with the following primary features:

- **Users (Civilians/Commuters):** View real-time transport data, plan trips, browse schedules, and submit feedback.
- **Administrators (Future scope):** Monitor traffic projects, update alerts, manage system data.

The portal provides authentication, dashboards, responsive navigation, real-time transport tracking, trip recommendations, and interactive UI elements.

1.3 Definitions, Acronyms, and Abbreviations

- **SRS:** Software Requirement Specification
- **UI:** User Interface
- **SPA:** Single Page Application
- **CRUD:** Create, Read, Update, Delete
- **Portal:** Web-based application accessible via browser

1.4 References

- IEEE 830-1998 SRS Guidelines
- ReactJS Official Documentation
- React Router v6 Documentation

2. Overall Description

2.1 Product Perspective

The City Transportation Portal is a new, standalone product built using ReactJS. It uses Context API for authentication and data management and client-side routing to provide a seamless SPA experience. The portal integrates real-time vehicle simulation, dashboards, and feedback mechanisms.

2.2 Product Functions

- **Authentication Module:** Login, role-based redirection, logout
- **Real-Time Transport Tracker:** Live positions of buses, metros, and bikes
- **Trip Planner:** Suggests optimal routes based on user inputs and traffic data
- **Traffic Projects & Alerts:** Displays ongoing projects and updates
- **Feedback Module:** Collect user feedback and display recent submissions
- **Navigation:** Navbar with links to all portal sections
- **Responsive UI:** Works on desktop and mobile devices

2.3 User Characteristics

- **Civilians/Commuters:** General users, need simple browsing experience
- **Administrators (Future):** Manage alerts, traffic projects, and system data
- **Technical Expertise:** Basic computer skills, mobile-friendly UI

2.4 Constraints

- Built only with **ReactJS (Frontend)**
- Data is currently **static/hardcoded**
- Works only in **modern browsers**

2.5 Assumptions and Dependencies

- Users will access via **desktop or mobile browser**
- Backend API may be integrated in the future
- Internet connection required for full functionality

3. Specific Requirements

3.1 Functional Requirements

- **FR1:** Authentication System (Login, role-based redirection, logout)
- **FR2:** Real-Time Transport Tracker (Display buses, metros, bikes)
- **FR3:** Trip Planner (Get route recommendations)
- **FR4:** Traffic Projects & Alerts (View updates and progress)
- **FR5:** Feedback System (Submit and view feedback)
- **FR6:** Navigation (Navbar links for all sections)

3.2 Non-Functional Requirements

- **Performance:** Load within 3 seconds
- **Usability:** Mobile-friendly, simple UI
- **Security:** Role-based access (future admin module)
- **Maintainability:** Modular React components
- **Scalability:** Backend integration possible

3.3 UI Requirements

- Modern, clean city-themed UI
- Login page with background image
- Tables/cards with hover effects
- Real-time vehicle icons and maps
- Dark/light mode toggle

---

4. System Models

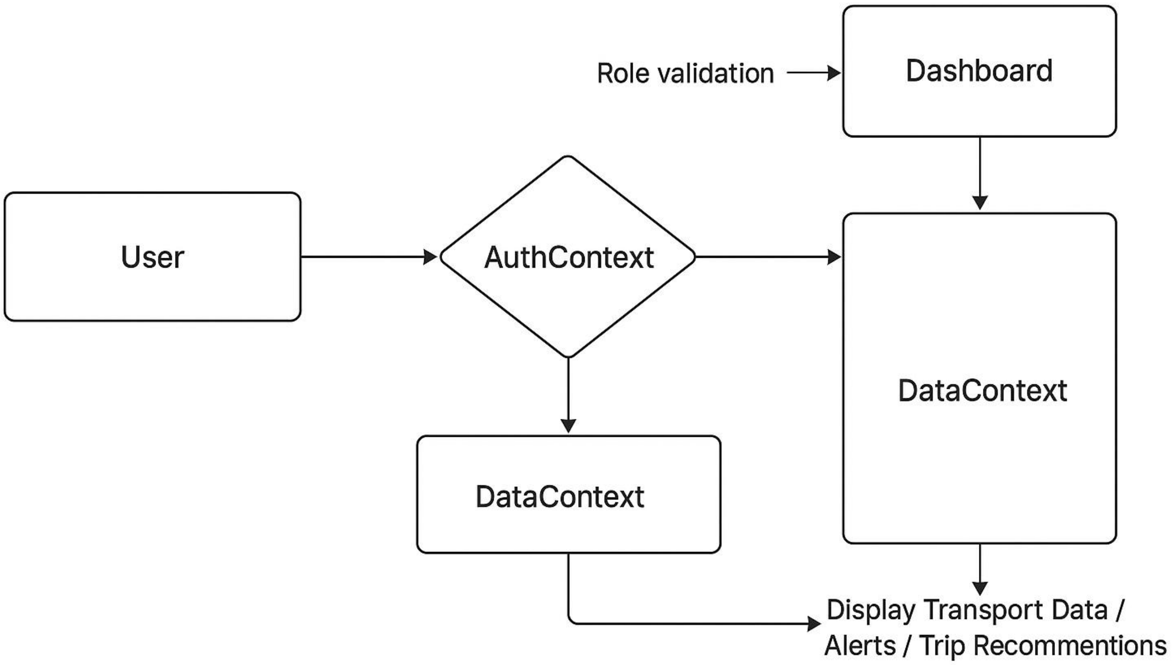
4.1 Use Case Diagram

Actors: Civilian, Administrator (future)
Use Cases: Login, View Transport, Plan Trip, Submit Feedback, View Alerts

Example Flow:

- Civilian → Login → Dashboard → Real-Time Tracker
- Civilian → Plan Trip → Recommendations
- Civilian → Submit Feedback → View Feedback

4.2 Data Flow Diagram (Level 1)



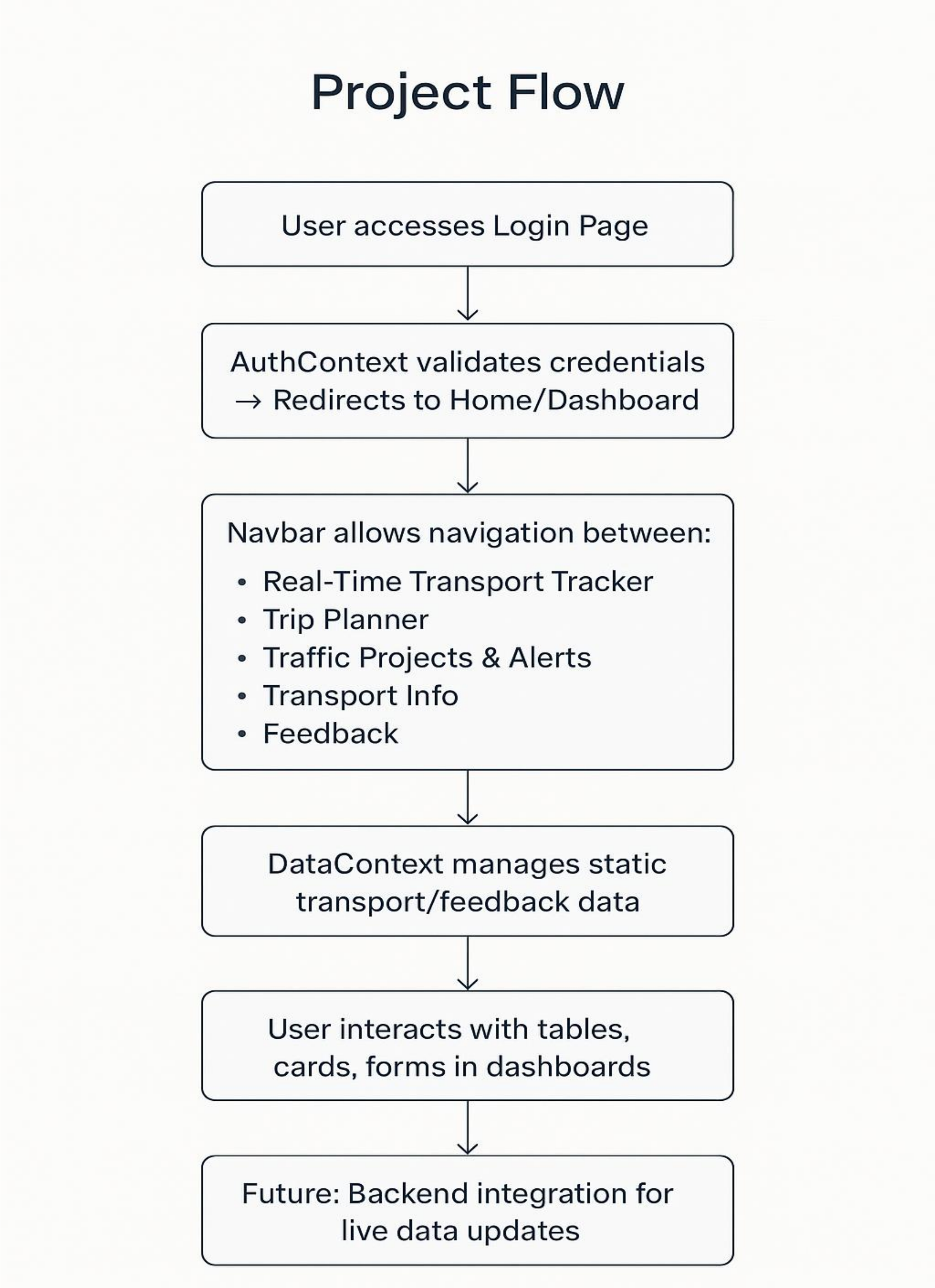
5. Future Enhancements

- Backend integration (Firebase, MongoDB)
- Admin Panel for managing vehicles and traffic data
- Live GPS integration for real buses/metros
- Price or fare display for transport
- Multi-language support
- Mobile app version

6. Project Structure and Flowchart

```
src/
├── assets/
|
├── AuthContext.jsx
├── DataContext.jsx      (Stores transport and feedback data globally)
├── Navbar.jsx           (Navigation bar component)
├── App.css              (Global styling)
├── App.jsx              (Main application component)
├── index.css            (Base styling)
├── main.jsx             (ReactDOM entry point)
├── pages/
|   ├── Home.jsx         (Landing page / dashboard)
|   ├── Login.jsx        (Login page UI + logic)
|   ├── RealTimeBus.jsx  (Real-time transport tracker)
|   ├── TripPlanner.jsx  (Trip planning tool)
|   ├── Feedback.jsx     (Feedback system)
|   ├── TrafficProjects.jsx (Traffic project tracker)
|   ├── TransportInfo.jsx (Transport options info)
|
└── components/
    ├── TrafficAlerts.jsx  (Displays live traffic alerts)
    └── ...              (Other reusable components)
```

7. Project Flow



Project Flow

User accesses Login Page

AuthContext validates credentials → Redirects to Home/Dashboard

Navbar allows navigation between:

- Real-Time Transport Tracker
- Trip Planner
- Traffic Projects & Alerts
- Transport Info
- Feedback

DataContext manages static transport/feedback data

User interacts with tables, cards, forms in dashboards

Future: Backend integration for live data updates