United States International University-Africa

Education to take you places

AGABA MARK

MERCY WANJIRU

COURSE NAME: MID TERM PROJECT

COURSE CODE: APT 3065-VB

LECTURER: PROF LAWRENCE NDERU

ACTIVITY : SYSTEM DESIGN DOCUMENT | MY MATATU APP

My Matatu Application – System Design Document (SDD)

1. Introduction

1.1 Purpose of the Document

This System Design Document (SDD) provides a comprehensive and structured outline for the architecture, components, and operational mechanisms of the My Matatu Application. It is designed to translate business and functional requirements into a detailed and technically feasible implementation plan. This document ensures that developers, testers, stakeholders, and administrators share a unified understanding of the system's design, guiding its development, deployment, and long-term maintenance.

1.2 Scope of the System

The My Matatu Application is a hybrid Android and web-based platform aimed at streamlining the experience of commuting within Kenya's informal public transportation network, especially focusing on the matatu system. The application supports three main user groups:

For Commuters, the app enhances the public transport experience by enabling them to view nearby matatu stations using GPS, search for optimal routes to their destinations, and access real-time information regarding matatu ETAs, availability, and occupancy. When direct transport is unavailable, the system intelligently suggests alternative multi-leg journey routes, thereby reducing waiting time and confusion.

For Conductors, the mobile application offers a simplified interface where they can initiate and terminate trips with a single tap. The conductor's device shares live location data using GPS, and the app allows conductors to mark their matatus as either "available" or "full" depending on passenger capacity. This real-time data feeds directly into the commuter-facing application to maintain transparency and efficiency.

For SACCO Administrators and Matatu Owners, a browser-based dashboard provides robust administrative tools. These include the ability to monitor live route and trip activity, access historical usage analytics for operational insights, and manage various user roles and permissions. This administrative backend plays a crucial role in enforcing accountability and regulatory compliance.

1.3 Intended Audience

This document is intended for:

- Software developers responsible for mobile, backend, and web development.
- Quality Assurance (QA) teams managing validation and testing.
- Project stakeholders including SACCO representatives, regulatory bodies, and funding partners.
- System administrators and DevOps personnel managing deployment and infrastructure.

1.4 Glossary

- Matatu: Privately owned minibus used in Kenya's public transport sector.
- SACCO: Savings and Credit Cooperative Organization that manages matatu routes and compliance.
- ETA: Estimated Time of Arrival.
- AVD: Android Virtual Device for emulator-based testing.
- Firebase: A cloud-based Backend-as-a-Service (BaaS) by Google.
- Realtime Database: Firebase's live synchronization cloud database.

2. System Overview and Design Philosophy

2.1 High-Level Description

The My Matatu System is built around three interconnected components:

1. **Commuter App**: Native Android application that enables route search, real-time matatu tracking, and ETA estimations.
2. **Conductor App**: Native Android interface that allows trip initiation, GPS location sharing, and passenger capacity status updates.
3. **Web Admin Dashboard**: Browser-based panel for SACCOs and matatu owners, providing oversight, analytics, and user management tools.

All components are integrated through Firebase services, including Firebase Authentication, Realtime Database, and Hosting. Google Maps SDK powers the location and routing features.

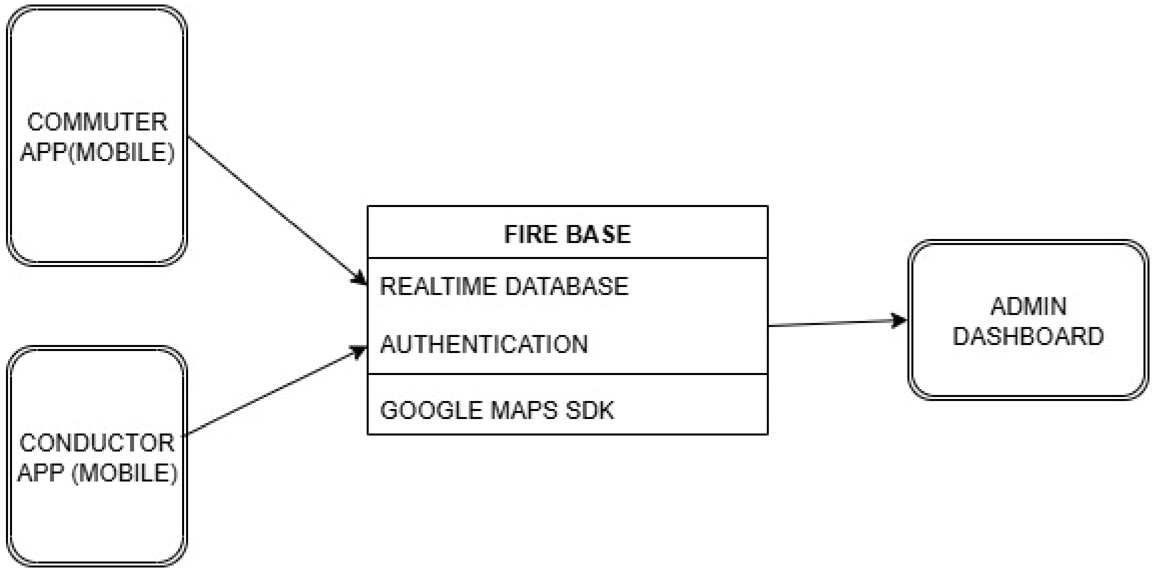
2.2 Design Principles

The application is developed with core software engineering principles in mind to ensure long-term sustainability and efficiency. These include:

- **Modularity**: The system is broken down into distinct modules—such as location services, routing, authentication, and data management—to allow for independent updates, easier debugging, and improved scalability.
- **Scalability**: Leveraging Firebase's infrastructure ensures seamless horizontal scaling. As more users join the platform, backend resources can be automatically provisioned to maintain performance.
- **Reusability**: Common code components, such as authentication workflows and map utilities, are designed as reusable modules to minimize duplication and simplify testing.
- **Security**: Role-based access control ensures that users only have access to the features relevant to their role. All data exchanges are encrypted using HTTPS, and Firebase's security rules provide an additional layer of role-bound data protection.
- **Performance Optimization**: Techniques such as GPS throttling, background services, and optimized rendering of map data ensure that the app is responsive while preserving battery and data usage.

3. Architectural Design

3.1 Architecture Diagram



3.2 Architecture Style

The application follows a client-server architecture where the clients (Android apps and the web dashboard) interact with the server-side services through RESTful APIs and Firebase SDKs. Firebase acts as the centralized backend for data storage, user authentication, and hosting, ensuring a unified and scalable architecture.

3.3 Component Descriptions

The My Matatu System is comprised of the following technical components:

Mobile Frontend: Built using native Android technologies (Java + XML), the mobile frontend includes two applications: one for commuters and another for conductors. The commuter app incorporates the Google Maps SDK to offer location-aware services, while the conductor app allows real-time trip control and status updates.

Web Frontend: The administrative dashboard is designed using ReactJS (or optionally vanilla JavaScript for lightweight deployments). It features data visualization tools, user management controls, and real-time monitoring capabilities for SACCOs.

Backend Services: Firebase services provide the backbone of the application. Firebase Authentication secures login workflows. The Realtime Database stores and synchronizes location and trip data, while Firebase Hosting serves the web dashboard. Cloud Functions may be introduced in future releases for backend logic.

Third-Party Integrations: Google Maps SDK is used for route rendering and location tracking. Google Places API enhances the search experience by providing intelligent location predictions.

4. Detailed Design

4.1 Module Descriptions
Each core module within the system is outlined below:

Login & Role Management Module: This module facilitates the onboarding process by allowing users to register or log in using their email and password. Once authenticated via Firebase Auth, users are redirected to their respective dashboards depending on their role (commuter, conductor, or admin). The system ensures that each user can only assume one active role at any given time.

Trip Management Module (Conductor App): This module enables conductors to create trip instances by marking the beginning and end of a journey. Conductors can also update their real-time GPS location and toggle the status of their matatu to reflect availability. All trip records are synced to the backend and made visible to commuters in real-time.

Matatu Tracking Module (Commuter App): The commuter app queries the current location and status of matatus operating on a specified route. The module displays pins on a map interface, colored to reflect availability (green for available, red for full), thereby empowering users with actionable and up-to-date travel information.

Route Suggestion Module: Using the Google Maps Directions API, this module generates optimal routes between selected start and end points. If no direct matatu is available, it computes alternative multi-leg trips involving transfers at strategic stations, aiming to minimize both cost and time.

4.2 Interface Design
The user interface (UI) has been carefully designed using Figma to ensure an intuitive and visually coherent experience across devices. Key interfaces include:

- A login screen that dynamically routes users based on role.

- A commuter home screen with map integration and destination search.
- A conductor panel with actionable trip controls.
- An administrative dashboard featuring user metrics, trip history, and role management tools.

5. Database Design

5.1 ER Diagram

[To be developed and attached using dbdiagram.io, MySQL Workbench, or similar tool.]

5.2 Data Dictionary

The following schema definitions represent the core database tables:

Users Table: Contains basic user account information.

- uid: STRING (Primary key)
- email: STRING
- role: ENUM (commuter, conductor, admin)
- registered_on: TIMESTAMP

Trips Table: Tracks each matatu journey in real-time.

- trip_id: STRING (Primary key)
- conductor_id: STRING (Foreign key referencing Users)
- route_id: STRING
- start_time: TIMESTAMP
- status: ENUM (active, full, complete)

Stations Table: Contains metadata for pick-up and drop-off points.

- station_id: STRING
- name: STRING

- location: GEOPOINT

6. Data Flow and Control Flow

6.1 Data Flow Diagram (Level 1)
The system data flow can be summarized as follows:

- A commuter interacts with the mobile app, which communicates with Firebase to retrieve nearby station data and route suggestions. The Maps SDK renders the data visually.
- A conductor initiates a trip on the Conductor App, pushing location and status data to Firebase. This data is then displayed in real-time on the Commuter App.

6.2 Control Flow Overview
Control flow defines the application's response to user actions:

- Upon logging in, the user's role is identified, and they are routed to the appropriate dashboard.
- Conductors begin trips, and their location updates are streamed continuously to Firebase.
- Commuters request a destination, and the system computes and presents a suitable route.

7. Non-Functional Requirements

7.1 Performance Optimization
The system is optimized for resource-constrained environments. Location updates are throttled to 30-second intervals to conserve device battery and bandwidth. The UI is built with lightweight components to maintain responsiveness even on low-end devices.

7.2 Security and Compliance
All communication between client and server is encrypted using HTTPS. Firebase Authentication enforces secure user identity management, while Firebase Rules restrict data access based on authenticated user roles and unique identifiers.

7.3 Usability and Accessibility
A high-contrast theme using white, purple, and orange ensures visibility in outdoor conditions. The interface includes large touch targets and readable fonts for field usability, especially for conductors working under varying lighting conditions.

7.4 Availability & Fault Tolerance

Firebase’s multi-region support ensures redundancy and high availability. In the event of connectivity issues, local data caching enables offline operation, and automatic retries are triggered once the network is restored.

8. Deployment Strategy

8.1 Supported Platforms

The system will be deployed across the following platforms:

- Android OS for commuter and conductor apps.
- Web browsers for the administrative dashboard.

8.2 Infrastructure Overview

Development and deployment pipelines are managed using GitHub, with automated CI/CD for version control. Firebase serves as the hosting platform and provides the necessary backend services including authentication, data storage, and analytics.

8.3 Environment Segmentation

Three distinct environments will be maintained:

- **Development**: Used by developers for feature testing using Android emulators and Firebase's test database.
- **Staging**: Deployed to internal teams on real devices for usability and integration testing.
- **Production**: Public-facing release, distributed via the Google Play Store and hosted on the live Firebase environment.

9. Testing Strategy

9.1 Methodologies

A multi-layered testing strategy is implemented:

- **Unit Testing**: Focused on individual modules such as login logic and trip status toggles.
- **Integration Testing**: Ensures seamless interaction between Firebase services and application modules.

- **System Testing**: Covers the entire user journey from registration and login to trip completion and feedback.

9.2 Tools & Frameworks

Testing tools include:

- **JUnit** for validating backend logic and utilities.
- **Espresso** for automating UI interactions on Android.
- **Firebase Test Lab** for cross-device compatibility testing to ensure consistent behavior across varied hardware.

10. Risks and Mitigation

The following table outlines potential risks and their mitigation strategies:

Risk	**Likelihood**	**Impact**	**Mitigation Strategy**
Conductors neglect to update status	Medium	Medium	Implement in-app reminders every 10 minutes
Exceeding Firebase quotas	Low	High	Monitor usage metrics and upgrade to Blaze Plan as needed
Network downtime	Medium	High	Enable offline data caching and implement retry mechanisms
User onboarding difficulties	Medium	Medium	Provide a comprehensive in-app walkthrough for new users

11. Appendices

- **Appendix A**: UI Wireframes and Figma Prototypes
- **Appendix B**: Architecture Diagram (draw.io or Lucidchart)
- **Appendix C**: API Specifications (Swagger/OpenAPI or Postman Collection)
- **Appendix D**: Design Logs & Revision History

- **Appendix E**: Firebase Security Rules (JSON Export)

Prepared By: Mark Agaba, Mercy Wanjiru, Faith Gisemba

Date: July 2025