Title: System Design for Public Transport Fare Collection and Management System

- **Metro Trans Investments Ltd**

- **Prepared by:** JoyAnn Wairimu Mwangi

- **Supervisor:** Charles Malungu

System Architecture & Scalability

- **Client-Server Architecture with Cloud-Based Integration**
  - **Frontend:** React.js (Web), Flutter (Mobile)
  - **Backend:** Python (Django/Flask)
  - **Database:** MySQL/PostgreSQL
  - **Payment Gateway:** MPesa API

- **Scalability Features:**
- Horizontal Scaling (Adding More Servers)
- Database Sharding for Large Datasets
- Load Balancing for High Traffic
- Cloud Hosting (AWS/GCP) for Reliability

Database & Data Handling

- **Key Tables & Relationships:**
  - **Users Table:** Passenger, Tout, Admin Information
  - **Transactions Table:** Logs All Payments
  - **Bus Fare Records:** Tracks Fare Collections
  - **Fraud Detection:** Flags Suspicious Transactions

- **Data Handling:**
  - **Encryption** for secure storage
  - **Data validation** for accuracy
  - **Logging & Auditing** for transparency
  - **Archiving Policies** for database performance

Security & Performance

- **Security Measures:**
  - **OAuth 2.0 & JWT Authentication** for Role-Based Access
  - **Fraud Detection AI** for Anomaly Detection
  - **Transaction Encryption** (AES-256)

- **Performance Optimization:**
  - **Asynchronous Processing** for Faster Transactions
  - **Database Indexing & Caching (Redis)** for Quick Access
  - **API Rate Limiting** to Prevent Overload

System Functionality & Interfaces

- **Core Features:**
  - **Passengers:** Pay fares via MPesa, NFC, QR Codes
  - **Touts:** Verify & Track Payments
  - **Admins:** Monitor Transactions, Detect Fraud, Generate Reports

- **API & Interfaces:**
  - **MPesa API** for Secure Payments
  - **QR/NFC API** for Digital Ticket Validation
  - **Admin API** for Reporting & Fraud Detection

System Design Considerations & Goals

- **System Considerations:**
    - **Redundancy & Failover** for Reliability
    - **Fault Tolerance & Auto Recovery**
    - **Load Balancing** for Efficient Resource Distribution

- **Design Goals:**
- **Efficiency:** Transactions **under 2 seconds**
- **Security:** Prevents fraud & unauthorized access
- **Scalability:** Supports **thousands of users**
- **Reliability: 99.9% uptime** with **automated backups**