**Project**

**on**

**QR Code-Based Metro Ticketing System**

**Introduction**

We want to develop a QR code-based metro ticketing system that generates a unique QR code for users upon each trip. The system records the entry station when the user scans the QR code at the entry gate and tracks the exit station when the QR code is scanned again at the destination. Fares are calculated based on predefined rates set by the admin, and the charges are automatically deducted from the user’s account balance. Users can top up their account balance via a mobile app or website.

**User Requirements**

1. **User Authentication:** Users must log in with their credentials (e.g., email & password or OTP) to access the system.
2. **Account Top-Up:** Users can add funds to their account through an integrated payment gateway (e.g., credit/debit cards, digital wallets, or mobile banking).
3. **QR Code Generation:** Users can generate a unique QR code for each trip, which is valid for a single journey.
4. **Entry and Exit:** Users can use the generated QR code to enter and exit the station by scanning it at the respective gates.
5. **Travel History:** Users can view detailed travel records, including entry/exit stations, travel time, fare deductions, and transaction history.
6. **Offline QR Code:** Users can generate a QR code in advance in case of network issues. The system validates it upon scanning at the gates.
7. **Low Balance Notifications:** Users receive notifications about low balances and prompts to recharge before their next trip.
8. **Fare Calculation:** Users can calculate the fare for a journey before starting their trip by selecting the entry and exit stations.
9. **Family/Group Travel:** Users can manage multiple tickets (e.g., for family or group travel).
10. **Family/Group Travel:** Users can manage multiple tickets in a single trip.
11. **Lost Item Reporting:** Users can report lost items through the system, and the system notifies station admins for assistance.
12. **Real-Time Updates:** Users receive real-time updates about station closures, maintenance, delays, or other service disruptions.

**Admin Requirements**

1. **Station Management:** Admins can add, update, or remove stations from the system as needed.
2. **Fare Management:** Admins can set or adjust fare rates based on travel distances.
3. **System Monitoring:** Admins can monitor overall system activity, including income reports, passenger statistics, and trip records.
4. **Blacklisting & Penalties:** Admins can block users involved in fraudulent activities and impose penalties for rule violations.
5. **Real-Time Dashboard:** Admins can access a live dashboard to view station occupancy, active users, and ongoing trips in real-time.
6. **Refund & Dispute Management:** Admins can handle refund requests and resolve transaction disputes efficiently.
7. **Emergency Alerts:** Admins can send real-time alerts to passengers in case of emergencies or unexpected disruptions.
8. **Custom Reports & Analytics:** Admins can generate detailed reports on revenue, user visits, and other key metrics.

**Database Design**

**Users Table:** Stores user account details.

CREATE TABLE Users (

UserId BIGINT IDENTITY(1,1) PRIMARY KEY,

FullName NVARCHAR(100) NOT NULL,

PhoneNumber NVARCHAR(15) NOT NULL UNIQUE,

EncryptedNID VARBINARY(MAX) NOT NULL,

PasswordHash NVARCHAR(256) NOT NULL,

CreatedAt DATETIME DEFAULT GETDATE(),

UpdatedAt DATETIME DEFAULT GETDATE()

);

**UserBalances Table:** Manages user balances and top-ups.

CREATE TABLE UserBalances (

BalanceID INT IDENTITY(1,1) PRIMARY KEY,

UserID INT UNIQUE NOT NULL FOREIGN KEY REFERENCES Users(UserId) ON DELETE CASCADE,

Balance DECIMAL(10, 2) DEFAULT 0.00,

LastUpdated DATETIME DEFAULT GETDATE()

);

**BalanceTransactions Table:** To log all balance updates

CREATE TABLE BalanceTransactions (

TransactionId INT PRIMARY KEY IDENTITY,

UserID INT FOREIGN KEY REFERENCES Users(UserId),

Amount DECIMAL(18, 2),

Type NVARCHAR(20), -- Values: 'top-up', 'fare-deduction', ' refund'

TransactionTime DATETIME DEFAULT GETDATE()

);

**Stations Table:** Stores metro stations.

CREATE TABLE Stations (

StationID INT IDENTITY(1,1) PRIMARY KEY,

StationName NVARCHAR(100) UNIQUE NOT NULL,

Location NVARCHAR(255) NOT NULL

);

CREATE TABLE Stations (

StationID INT IDENTITY(1,1) PRIMARY KEY,

StationName NVARCHAR(100) UNIQUE NOT NULL,

Latitude FLOAT NOT NULL,

Longitude FLOAT NOT NULL,

CreatedAt DATETIME DEFAULT GETDATE()

);

Table Name: Stations

Attributes:

StationId (Primary Key, INT, Identity)

StationName (NVARCHAR(100))

StationCode (NVARCHAR(10), Unique)

Location (NVARCHAR(255))

CreatedAt (DATETIME, Default: GETDATE())

UpdatedAt (DATETIME, Default: GETDATE())

**Fares Table:** Stores fare rates between stations.

CREATE TABLE Fares (

FareId INT IDENTITY(1,1) PRIMARY KEY,

EntryStation INT NOT NULL FOREIGN KEY REFERENCES Stations(StationId) ON DELETE CASCADE,

ExitStation INT NOT NULL FOREIGN KEY REFERENCES Stations(StationId) ON DELETE CASCADE,

FareAmount DECIMAL(6,2) NOT NULL

);

Table Name: Fares

Attributes:

FareId (Primary Key, INT, Identity)

StartStationId (Foreign Key referencing Stations.StationId, INT)

EndStationId (Foreign Key referencing Stations.StationId, INT)

FareAmount (DECIMAL(18, 2))

CreatedAt (DATETIME, Default: GETDATE())

UpdatedAt (DATETIME, Default: GETDATE())

7. Fare Rules Table (FareRules)

Defines fare rates based on distance.

CREATE TABLE FareRules (

RuleID INT IDENTITY(1,1) PRIMARY KEY,

DistanceRange NVARCHAR(50) NOT NULL,

FareAmount DECIMAL(10,2) NOT NULL,

CreatedAt DATETIME DEFAULT GETDATE()

);

**QRCodes Table:** Stores QR codes generated for trips.

CREATE TABLE QRCodes (

QRId INT IDENTITY(1,1) PRIMARY KEY,

UserId INT NOT NULL FOREIGN KEY REFERENCES Users(UserId) ON DELETE CASCADE,

TripId INT UNIQUE FOREIGN KEY REFERENCES Trips(TripId) ON DELETE CASCADE,

QRCode NVARCHAR(MAX) UNIQUE NOT NULL,

IssuedAt DATETIME DEFAULT GETDATE(),

Status NVARCHAR(20) CHECK (Status IN ('Unused', 'Used', 'Expired')) DEFAULT 'Unused'

);

CREATE TABLE QRCodes (

QRCodeID UNIQUEIDENTIFIER PRIMARY KEY DEFAULT NEWID(),

UserID UNIQUEIDENTIFIER FOREIGN KEY REFERENCES Users(UserID) ON DELETE CASCADE,

TripID UNIQUEIDENTIFIER FOREIGN KEY REFERENCES Trips(TripID) ON DELETE CASCADE,

QRData NVARCHAR(MAX) NOT NULL,

IsUsed BIT DEFAULT 0,

ExpirationTime DATETIME NOT NULL,

CreatedAt DATETIME DEFAULT GETDATE()

);

**Trips Table:** Tracks users' trips.

CREATE TABLE Trips (

TripId INT IDENTITY(1,1) PRIMARY KEY,

UserId INT NOT NULL FOREIGN KEY REFERENCES Users(UserId) ON DELETE CASCADE,

EntryStation INT NOT NULL FOREIGN KEY REFERENCES Stations(StationId),

ExitStation INT FOREIGN KEY REFERENCES Stations(StationId),

FareAmount DECIMAL(6,2) NOT NULL,

TripStatus NVARCHAR(20) CHECK (TripStatus IN ('Ongoing', 'Completed')) DEFAULT 'Ongoing',

EntryTime DATETIME DEFAULT GETDATE(),

ExitTime DATETIME NULL

);

CREATE TABLE Trips (

TripID UNIQUEIDENTIFIER PRIMARY KEY DEFAULT NEWID(),

UserID UNIQUEIDENTIFIER FOREIGN KEY REFERENCES Users(UserID) ON DELETE CASCADE,

EntryStationID INT FOREIGN KEY REFERENCES Stations(StationID),

ExitStationID INT FOREIGN KEY REFERENCES Stations(StationID) NULL,

StartTime DATETIME DEFAULT GETDATE(),

EndTime DATETIME NULL,

FareAmount DECIMAL(10,2) NULL,

Status NVARCHAR(20) CHECK (Status IN ('Ongoing', 'Completed', 'Cancelled')),

CreatedAt DATETIME DEFAULT GETDATE()

);

4. Trips Table

Stores trip details for each user.

Table Name: Trips

Attributes:

TripId (Primary Key, INT, Identity)

UserId (Foreign Key referencing Users.UserId, INT)

StartStationId (Foreign Key referencing Stations.StationId, INT)

EndStationId (Foreign Key referencing Stations.StationId, INT)

QRCode (NVARCHAR(255), Unique

FareAmount (DECIMAL(18, 2))

StartTime (DATETIME)

EndTime (DATETIME)

Status (NVARCHAR(20)) -- Values: 'ongoing', 'completed', 'cancelled'

**Transactions Table:** Records payments, fare deductions, and refunds.

CREATE TABLE Transactions (

TransactionId INT IDENTITY(1,1) PRIMARY KEY,

UserId INT NOT NULL FOREIGN KEY REFERENCES Users(UserId) ON DELETE CASCADE,

TripId INT FOREIGN KEY REFERENCES Trips(TripId) ON DELETE SET NULL,

TransactionType NVARCHAR(20) CHECK (TransactionType IN ('Top-up', 'Fare Deduction', 'Refund')),

Amount DECIMAL(10,2) NOT NULL,

TransactionTime DATETIME DEFAULT GETDATE()

);

CREATE TABLE Transactions (

TransactionID UNIQUEIDENTIFIER PRIMARY KEY DEFAULT NEWID(),

WalletID UNIQUEIDENTIFIER FOREIGN KEY REFERENCES UserWallet(WalletID) ON DELETE CASCADE,

TransactionType NVARCHAR(20) CHECK (TransactionType IN ('Top-Up', 'Fare Deduction', 'Refund')),

Amount DECIMAL(10,2) NOT NULL,

PaymentMethod NVARCHAR(50) NULL,

ReferenceID NVARCHAR(100) NULL,

Status NVARCHAR(20) CHECK (Status IN ('Success', 'Pending', 'Failed')),

CreatedAt DATETIME DEFAULT GETDATE()

);

5. Transactions Table

Stores all financial transactions (top-ups, fare deductions).

Table Name: Transactions

Attributes:

TransactionId (Primary Key, INT, Identity)

UserId (Foreign Key referencing Users.UserId, INT)

Amount (DECIMAL(18, 2))

Type (NVARCHAR(20)) -- Values: 'top-up', 'fare-deduction'

TransactionTime (DATETIME, Default: GETDATE())

Description (NVARCHAR(255

**Blacklisted Users Table:** Stores users involved in fraud.

CREATE TABLE BlacklistedUsers (

BlacklistId INT IDENTITY(1,1) PRIMARY KEY,

UserId INT UNIQUE FOREIGN KEY REFERENCES Users(UserId) ON DELETE CASCADE,

Reason NVARCHAR(255) NOT NULL,

BlacklistedAt DATETIME DEFAULT GETDATE()

);

CREATE TABLE BlacklistedUsers (

BlacklistID UNIQUEIDENTIFIER PRIMARY KEY DEFAULT NEWID(),

UserID UNIQUEIDENTIFIER FOREIGN KEY REFERENCES Users(UserID) ON DELETE CASCADE,

Reason NVARCHAR(255) NOT NULL,

CreatedAt DATETIME DEFAULT GETDATE()

);

Blacklist Table

Stores blacklisted users and their penalties.

Table Name: Blacklist

Attributes:

BlacklistId (Primary Key, INT, Identity)

UserId (Foreign Key referencing Users.UserId, INT)

Reason (NVARCHAR(MAX))

PenaltyAmount (DECIMAL(18, 2))

BlacklistedAt (DATETIME, Default: GETDATE())

**Lost Items Reports Table:** Stores lost items reported by users.

CREATE TABLE LostItems (

ReportId INT IDENTITY(1,1) PRIMARY KEY,

UserId INT NOT NULL FOREIGN KEY REFERENCES Users(UserId) ON DELETE CASCADE,

StationId INT NOT NULL FOREIGN KEY REFERENCES Stations(StationId),

Description NVARCHAR(MAX) NOT NULL,

ReportTime DATETIME DEFAULT GETDATE(),

Status NVARCHAR(20) CHECK (Status IN ('Pending', 'Resolved')) DEFAULT 'Pending'

);

6. LostItems Table

Stores lost item reports by users.

Table Name: LostItems

Attributes:

LostItemId (Primary Key, INT, Identity)

UserId (Foreign Key referencing Users.UserId, INT)

StationId (Foreign Key referencing Stations.StationId, INT)

Description (NVARCHAR(MAX))

ReportTime (DATETIME, Default: GETDATE())

Status (NVARCHAR(20)) -- Values: 'pending', 'resolved'

**Admin Table:** Stores admin users.

CREATE TABLE Admins (

AdminId INT IDENTITY(1,1) PRIMARY KEY,

FullName NVARCHAR(100) NOT NULL,

Email NVARCHAR(100) UNIQUE NOT NULL,

PasswordHash NVARCHAR(255) NOT NULL,

Role NVARCHAR(50) CHECK (Role IN ('SuperAdmin', 'StationManager', 'Finance')) DEFAULT 'StationManager',

CreatedAt DATETIME DEFAULT GETDATE()

);

Admins Table

Stores admin credentials and roles.

Table Name: Admins

Attributes:

AdminId (Primary Key, INT, Identity)

Name (NVARCHAR(100))

Email (NVARCHAR(255), Unique

PasswordHash (NVARCHAR(255))

Role (NVARCHAR(20)) -- Values: 'super-admin', 'station-admin'

CreatedAt (DATETIME, Default: GETDATE())

UpdatedAt (DATETIME, Default: GETDATE())

**Notifications Table:** Stores system alerts for users. Stores system alerts (low balance, travel updates, etc.).

CREATE TABLE Notifications (

NotificationId INT IDENTITY(1,1) PRIMARY KEY,

UserId INT NOT NULL FOREIGN KEY REFERENCES Users(UserId) ON DELETE CASCADE,

Message NVARCHAR(MAX) NOT NULL,

NotificationTime DATETIME DEFAULT GETDATE(),

Status NVARCHAR(20) CHECK (Status IN ('Unread', 'Read')) DEFAULT 'Unread'

);

CREATE TABLE Notifications (

NotificationID UNIQUEIDENTIFIER PRIMARY KEY DEFAULT NEWID(),

UserID UNIQUEIDENTIFIER FOREIGN KEY REFERENCES Users(UserID) ON DELETE CASCADE,

Message NVARCHAR(255) NOT NULL,

IsRead BIT DEFAULT 0,

CreatedAt DATETIME DEFAULT GETDATE()

);

7. Notifications Table

Stores notifications sent to users (e.g., low balance, service disruptions).

Table Name: Notifications

Attributes:

NotificationId (Primary Key, INT, Identity)

UserId (Foreign Key referencing Users.UserId, INT)

Message (NVARCHAR(MAX))

Type (NVARCHAR(20)) -- Values: 'low-balance', 'emergency', 'service-update'

SentAt (DATETIME, Default: GETDATE())

10. Refunds Table

Stores refund requests and their status.

Table Name: Refunds

Attributes:

RefundId (Primary Key, INT, Identity)

UserId (Foreign Key referencing Users.UserId, INT)

TripId (Foreign Key referencing Trips.TripId, INT)

Amount (DECIMAL(18, 2))

Status (NVARCHAR(20)) -- Values: 'pending', 'approved', 'rejected'

RequestedAt (DATETIME, Default: GETDATE())

ResolvedAt (DATETIME)

CREATE INDEX idx\_trips\_userid ON Trips(UserId);

CREATE INDEX idx\_qrcodes\_userid ON QRCodes(UserId);

CREATE INDEX idx\_fares\_entry\_exit ON Fares(EntryStation, ExitStation);

Database Relationships

Users → Trips: One-to-Many (A user can have multiple trips).

Stations → Fares: Many-to-Many (Fares are defined between pairs of stations).

Trips → Transactions: One-to-Many (Each trip may have a fare deduction transaction).

Users → LostItems: One-to-Many (A user can report multiple lost items).

Users → Notifications: One-to-Many (A user can receive multiple notifications).

Users → Blacklist: One-to-One (A user can be blacklisted once).