Project Report: Metro Card System

Your Name

October 26, 2025

1. Project Overview

- **Project Name:** Metro Card System
- **Description:** A Spring Boot application for managing metro card transactions, passenger details, and journey summaries. The application uses MySQL for data storage and Docker for containerization.

2. Technologies Used

Category	**Technologies/Tools**
Backend Framework	Spring Boot 3.5.6
Language	Java 21
Database	MySQL 8.0
Build Tool	Maven
Containerization	Docker, Docker Compose
CI/CD	GitHub Actions
Version Control	Git, GitHub
IDE	IntelliJ IDEA, VS Code

3. Project Structure

```
metro-card-system/
 src/
    main/
       java/com/example/demo/
          controller/
          model/
          repository/
          service/
          MetroCardSystemApplication.java
```

```
      resources/
         application.properties
         data.sql
         schema.sql
      webapp/
   test/
target/
pom.xml
docker-compose.yml
Dockerfile
.github/
    workflows/
        ci-cd.yml
```

4. Key Features Implemented

4.1. Core Functionality

- **Passenger Management:** Add and manage passenger details (ID, name, type).
- **Metro Card Management:** Track metro card balances and details.
- **Transaction Processing:** Record journey transactions (single/return journeys).
- **Summaries:** Generate passenger summaries and collection summaries.

4.2. Database Schema

- **Tables:**
  - `passenger`: Stores passenger details.
  - `metro_card`: Stores metro card details.
  - `transaction`: Records all transactions.

4.3. Dockerization

- **Dockerfile:** Containerizes the Spring Boot application.
- **Docker Compose:** Manages MySQL and Spring Boot containers.

4.4. CI/CD Pipeline

- **GitHub Actions:** Automates building, testing, and deploying the application.

- **Workflow:**
  - Builds the application with Maven.
  - Builds and pushes the Docker image to Docker Hub.
  - Deploys the application using Docker Compose.

5. Challenges Faced and Solutions

Challenge	**Solution**
MySQL connection issues in Docker	Used `depends_on` with health checks in `docker-compose.yml`.
SQL initialization errors (`data.sql` running before tables were created)	Used `spring.jpa.defer-datasource-initialization=true` and `spring.jpa.hibernate.ddl-auto=create-drop`.
Docker Hub authentication errors in CI/CD	Added `DOCKER_HUB_USERNAME` and `DOCKER_HUB_TOKEN` as GitHub secrets.
Sensitive data (passwords) exposed in GitHub	Used environment variables and `.env` files, added `.env` to `.gitignore`.

6. Current Status

- **Application:** Running successfully in Docker containers.
- **CI/CD Pipeline:** Configured and passing in GitHub Actions.
- **Database:** MySQL container running with persistent data.
- **Security:** Sensitive data removed from the repository and stored in GitHub Secrets.

7. How to Run the Project Locally

7.1. Prerequisites

- Java 21
- Maven
- Docker and Docker Compose
- Git

7.2. Steps

1. Clone the Repository:

```
git clone https://github.com/your-username/metro-card-system.git
cd metro-card-system
```

2. Build the Application:

```
mvn clean package -DskipTests
```

3. Build the Docker Image:

```
docker build -t your-dockerhub-username/metro-card-system:latest .
```

4. Run the Application:

```
docker-compose up
```

5. Access the Application: Open your browser and navigate to `http://localhost:8080/index.html`.

8. CI/CD Workflow

8.1. Workflow File: `.github/workflows/ci-cd.yml`

```
name: CI/CD Pipeline

on:
  push:
    branches: ["main"]
  pull_request:
    branches: ["main"]

jobs:
  build-and-test:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout code
        uses: actions/checkout@v4

      - name: Set up JDK 21
```

```
      uses: actions/setup-java@v3
      with:
        java-version: '21'
        distribution: 'temurin'

    - name: Build with Maven
      run: mvn clean package -DskipTests

    - name: Login to Docker Hub
      uses: docker/login-action@v3
      with:
        username: ${{ secrets.DOCKER_HUB_USERNAME }}
        password: ${{ secrets.DOCKER_HUB_TOKEN }}

    - name: Build and push Docker image
      run: |
        docker build -t your-dockerhub-username/metro-card-system:latest .
        docker push your-dockerhub-username/metro-card-system:latest

deploy:
  needs: build-and-test
  runs-on: ubuntu-latest
  steps:
    - name: Checkout code
      uses: actions/checkout@v4

    - name: Login to Docker Hub
      uses: docker/login-action@v3
      with:
        username: ${{ secrets.DOCKER_HUB_USERNAME }}
        password: ${{ secrets.DOCKER_HUB_TOKEN }}

    - name: Pull and run Docker containers
      run: |
        docker-compose down
        docker-compose pull
        docker-compose up -d
```

8.2. GitHub Secrets

- `DOCKER_HUB_USERNAME`: Your Docker Hub username.
- `DOCKER_HUB_TOKEN`: Your Docker Hub access token.
- `MYSQL_ROOT_PASSWORD`: MySQL root password.

9. Next Steps

9.1. Enhancements

- **Frontend:** Improve the UI with a modern framework like React or Angular.
- **Monitoring:** Add Prometheus and Grafana for monitoring application performance.
- **Security:** Implement authentication and authorization (e.g., Spring Security).
- **Testing:** Add unit and integration tests.

9.2. Deployment

- Deploy the application to a cloud provider (e.g., AWS, DigitalOcean).
- Set up a production-grade database (e.g., Amazon RDS for MySQL).

9.3. Documentation

- Add a `README.md` file with setup instructions.
- Document the API endpoints (e.g., using Swagger).

10. Lessons Learned

- **Docker and Docker Compose:** Essential for containerizing applications and managing dependencies.
- **CI/CD with GitHub Actions:** Automates the build, test, and deployment process.
- **Security:** Always avoid hardcoding sensitive information; use environment variables and secrets.
- **Debugging:** Logs and error messages are crucial for diagnosing issues in Docker and CI/CD pipelines.

11. Conclusion

You've successfully built a **Metro Card System** application with:

- A **Spring Boot backend**.
- A **MySQL database**.

- **Docker containers** for easy deployment.
- A **CI/CD pipeline** using GitHub Actions.

Your project is now **secure, containerized, and automated** for future updates. Great job!

Final Checklist

Application runs locally in Docker.

CI/CD pipeline is passing in GitHub Actions.

Sensitive data is secured using GitHub Secrets.

Project is well-documented and ready for further enhancements.