American International University-Bangladesh (AIUB)

Department of Computer Science Faculty of Science & Technology (FST)

MetroSheba

A Software Engineering Project Submitted
By

Semester: Summer 24-25		Section:	Group Number:	
SL	**Student Name**	**Student ID**	**Contribution (CO3 + CO4)**	**Individual Marks**
1	Mukaddas Mahdi Ullas	23-50873-1	25%	
2	Md Mostafa Hamid Yeasib	23-50897-1	25%	
3	Repha Tasneya	23-50944-1	25%	
4	Tanvir Ahmed	23-51316-1	25%	

The project will be evaluated for the following Course Outcomes

CO3 (PO-g-1) ***Select appropriate software engineering models, project management roles and their associated skills for the complex software engineering project and evaluate the sustainability of developed software, taking into consideration the societal and environmental aspects***	Total Marks	
Selection of Software Engineering Models: Process model selection and presents sufficient evidence to support argument for the model selection	[5 Marks]	
Role identification and Responsibility Allocation: Well-planned project with proper role identification and responsibility allocation in the project management activities	[5Marks]	
Formatting and Submission: Submission, Defense, Completeness, Spelling, grammar, and Organization of the Project report	[5Marks]	
CO4 (PO-k-1) ***Apply engineering management principles and economic decision making to develop software engineering project management plan.***	Total Marks	
Project WBS and Testcases: Relevant WBS (project task list) and testcases for the proposed project are stated properly.	[5Marks]	
Effort Estimation and Scheduling: Project estimation was described using proper effort estimation or schedules based on available project resources	[5Marks]	
Risk Management: Sufficient and appropriate risks are identified, analyzed, and properly categorized or prioritized.	[5Marks]	

1.1 PROJECT PROPOSAL

Background to the Problem

Contextual Background: Dhaka is experiencing rapid urban growth, which has increased the demand for efficient and reliable public transport. The Dhaka Metro Rail has been introduced to reduce traffic congestion and improve mobility. However, the current ticketing process relies heavily on manual counters, which often results in long queues, time delays, and inconvenience for daily commuters. With the growing adoption of smartphones and digital payments in Bangladesh, commuters expect a faster, contactless, and user-friendly ticketing system.

1.2 Solution to the Problem and Process Model Selection

1. **Project Scope**

   MetroSheba is a mobile app for Dhaka Metro Rail that allows passengers to buy tickets online, generate QR-based e-tickets, and enter stations quickly without standing in queues. The app lets users select boarding and destination stations, check fare rates, and make secure digital payments for a faster and more convenient travel experience.

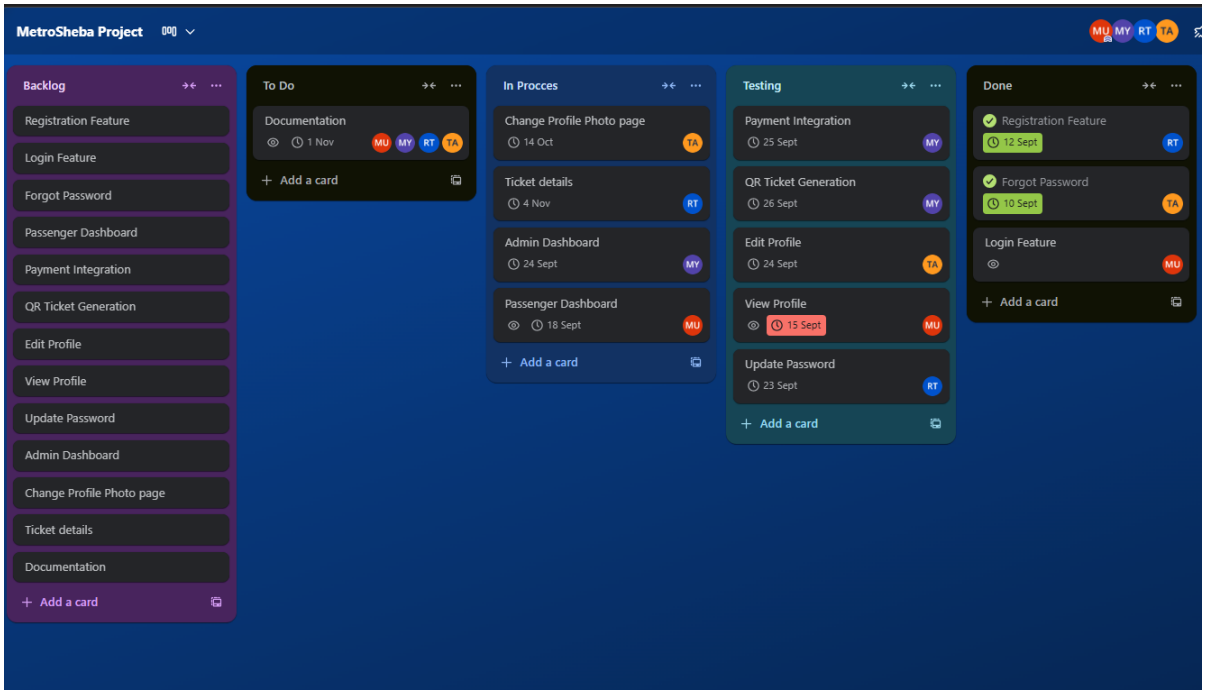
   Key Features

   - Online ticket booking with instant QR code generation.
   - View all the metro stations and select boarding & destination points.
   - Automatic fare calculation based on selected stations.
   - Multiple payment options – bKash, Nagad, Rocket, and cards.
   - Digital QR ticket access for smooth and paperless entry.

2. User table

User Table	As a
As a user, I want to see login and register buttons so that I can either log in or create an account.	Passenger
As a user, I want to register with my personal details (name, email, gender, mobile, password, NID, user type, verification code) so that I can create a secure account.	Passenger
I want to log in with my email and password so that I can access my dashboard.	Passenger, Admin
I want to select tickets (quantity, from-station, to-station) and pay securely so that I can buy metro tickets online.	Passenger
I want to view a dashboard with customer info, revenue, tickets sold, and reports so that I can manage and monitor the metro system	Admin
I want to pay using multiple options (card, bKash, Rocket, Nagad) so that I can complete my purchase easily.	Passenger
I want to see my ticket details and get a QR code so that I can quickly enter the metro without a paper ticket.	Passenger
I want to view my profile details so that I can confirm my information.	Passenger, Admin
I want to change my profile photo so that my account looks personalized.	Passenger, Admin
I want to change my password so that I can keep my account secure.	Passenger, Admin
I want to edit my profile information (name, email, phone, DOB, gender) so that my details remain up to date.	Passenger, Admin

3. Trello Story Board



4. Proposed Solution

To address the existing challenges in the Dhaka Metro Rail ticketing system, we propose the development of a mobile application called Metro Sheba**.** This application will serve as a complete end-to-end solution for ticket purchasing, digital payment, and QR-based entry.

Through the Metro Sheba app, users will be able to:

- View all available metro station names within the system.
- Select their boarding and destination stations.
- Instantly view ticket prices based on the selected route.
- Choose the number of tickets to purchase in a single transaction.
- Make payments securely using multiple options such as mobile banking (bKash, Nagad, Rocket) and debit/credit cards.
- Receive a QR code-based e-ticket inside the app immediately after successful payment.
- Scan the QR code at metro gates for quick, contactless, and paperless entry, reducing dependency on manual counters.

We chose the Agile model because it lets us build the app step by step and adjust easily if changes are needed. Agile helps us get regular feedback, work closely as a team, and stay open about progress. We are using Scrum, a part of Agile, because it gives us a clear plan with specific roles

and short work periods (called sprints). This helps the team stay focused, track their work, and keep improving, making development faster and more organized.

5. Project Environment Analysis

The Metro Sheba app will work in Bangladesh's metro system, where people need faster and easier ticketing.

- The app will use QR codes, smartphones, and mobile payments (bKash, Nagad, cards). These are common and reliable in Bangladesh.
- It must connect with Dhaka Metro authorities (DMTCL) and payment services.
- Daily passengers need a simple, Bangla-friendly app that also works for students, women, seniors, and people with disabilities.
- The app must follow government laws on money transactions and data safety.

Requirements:

- Some needs are fixed, like station selection, ticket price, payment, and QR tickets.
- Some needs may change, like adding student discounts, women-only coach info, or new metro stations.

6. Process Model Support and Feasibility

For the **Metro Sheba** project, we selected the **Scrum Agile Model**.

- **Team Size**: Scrum works well for small teams. Each member has a clear role (e.g., developer, tester, designer), and tasks are divided into sprints.
- **Communication**: Daily stand-up meetings in Scrum help the team share progress and problems quickly. This improves teamwork and avoids delays.
- **Coordination**: Scrum uses a product backlog and sprint backlog to organize tasks. This makes it easy to track progress and adjust when requirements change.

Feasibility for business studies

Yes, the solution is feasible. Metro Sheba supports Dhaka Metro's business goals by reducing ticket counter queues, encouraging digital payments, and making travel faster. Since Scrum is flexible, it allows the app to improve step by step, matching both commuter needs and the Digital Bangladesh vision**.**

7. Flexibility of the Model

The **Scrum Agile Model** is highly flexible and can easily adapt to changes in project scope, technology, or user requirements.

- **Scope Changes**: If new features are needed (e.g., student discounts, women-only coach info), they can be added to the product backlog and included in future sprints without affecting the whole project.
- **Technology Changes**: If new payment systems (like Tap-to-Pay or NFC) become popular, Scrum allows the team to integrate them step by step during later iterations.
- **User Requirements**: Scrum collects user feedback at the end of each sprint. This makes it easy to adjust designs, improve the interface, or add new options based on real commuter needs.

8. Deep Insight & Creative Solution

The real problem is not just traffic, but the time lost in queues and inconvenience at stations. Metro Sheba solves this by turning every smartphone into a digital metro pass. With end-to-end mobile ticketing, it removes queues, reduces crowding, and ensures safe, contactless travel. Metro Sheba provides a creative, real-life solution by turning every phone into a smart metro pass**:**

- **Queue-Free Access:** Digital tickets replace counters, saving commuters 15–20 mins daily.
- **Smart Payment Integration**: Local wallets (bKash, Nagad, Rocket) + cards → high adoption.
- **Offline Support**: QR tickets stored in app work even without internet.
- **Passenger Empowerment**: Fare shown upfront, multiple ticket purchase, family travel made easy.
- **Eco-Friendly**: Paperless ticketing aligns with smart city & sustainability goals.

9. Target Group of Users

- **Daily commuters** – students, office workers, service staff who travel regularly.
- **University & college students** – looking for student passes/discounts and easy ticketing.
- **Garments workers** – need cheap, reliable transport.
- **Women commuters** – want women-only coach info and safety features.
- **Occasional travellers** – shoppers, hospital visitors, exam candidates, etc.
- **People with disabilities** – require step-free routing, audio guidance, and platform accessibility.

How Users Will Benefit:

- Save time by buying tickets on the phone from anywhere.
- See the ticket price before buying.
- Use a digital ticket with a QR code to enter the metro fast.
- Pay safely with easy payment methods.
- Make traveling by metro easier and faster.

10. Contribution to Scientific Results

The Metro Sheba project contributes by:

- Shows how end-to-end mobile ticketing reduces queues and saves time.
- Validates local wallet adoption (bKash, Nagad, Rocket) in public transport.
- Supports studies on paperless ticketing, smart mobility, and Digital Bangladesh initiatives

11. Evidence for Selecting Scrum

The Scrum model is the best fit for developing Metro Sheba because:

- Metro Sheba has multiple features (ticketing, payment, QR validation, dashboards). Scrum allows breaking them into manageable sprints.
- User needs (e.g., new wallets, accessibility options) may evolve. Scrum supports iterative updates without disrupting the whole project.
- Regular sprint reviews let stakeholders (commuters, metro authority) test features early and give feedback.
- High-risk components (like secure payments and QR validation) can be developed and tested first.
- Instead of waiting for full project completion, core features (e.g., ticket purchase + QR entry) can go live earlier.
- Agile Scrum has been widely used in successful mobile app and transport solutions worldwide.

12. Risk & Uncertainty Management in Scrum Model

1. **Product Backlog (Planning)**
    - Risk: Missing/unclear requirements.
    - Action: Collect commuter stories (students, workers, women, etc.), prioritize features like QR tickets, payments, schedules.
2. **Sprint Planning**
    - Risk: Over-commitment or wrong priorities.
    - Action: Break work into small sprints, only pick achievable items.
3. **Sprint Execution (Development + Daily Scrum)**
    - Risk: Technical failures (QR not scanning, payment API errors).
    - Action: Daily stand-ups detect blockers early; small increments reduce failure impact.
4. **Sprint Review**
    - Risk: Features not meeting user needs.
    - Action: Demo app to stakeholders (metro staff, commuters) → immediate feedback.
5. **Sprint Retrospective**
    - Risk: Process inefficiencies or delays.
    - Action: Team reflects, improves methods for next sprint.
6. **Continuous Delivery**
    - Risk: App not scalable, bugs in live use.
    - Action: Frequent releases, pilot at few stations before city-wide rollout.

Linking Scrum to Project Schedule

1. **Sprint 1**
    - Activities: UI homepage, booking API, unit/integration testing, demo, retrospective
    - How it supports deadlines: Early delivery of a working prototype builds a foundation and confirms direction before adding complex features.
2. **Sprint 2**
    - Activities: Booking page, payment integration, feature testing, sprint review
    - Benefit: Payment risk handled early, ensuring critical functionality is ready on time.
3. **Sprint 3**
    - Activities: Notifications, dashboard/analytics, feature testing, stakeholder feedback
    - Benefit: Enhancements delivered in manageable steps; deadlines met because each sprint has fixed duration.
    -

4. **Sprint 4**
    - Activities: Deployment, bug fixes, new feature release, final demo
    - Benefit: Ensures smooth launch, with time reserved for stabilization before full deployment.

Scrum Ensures Timely Delivery

- **Time-boxed sprints** (fixed 2–3-week durations) - prevents schedule slippage.
- **Sprint reviews & demos** - continuous stakeholder alignment, avoiding last-minute surprises.
- **Incremental delivery** - working features are ready at each milestone, reducing end-project pressure.
- **Retrospectives** - improve process each sprint, making future work faster and more predictable.

1.3 Reasons for excluding other models

- **Waterfall model:** We didn't choose the Waterfall model because it is too strict and there is no backtracking once a step is done. Also, testing happens at the end, so problems are found late. We chose Agile Scrum because it is flexible, lets us test and fix issues early, and is better for our small team and changing project.
- **V-Model:** The V-Model is strict and disciplined, requiring detailed planning at the start and no backtracking, which makes it hard to change anything once development begins. In contrast, Agile Scrum is flexible and allows us to make changes, test, and fix issues during each sprint. This makes Scrum better for small teams and projects where requirements may change.
- **Incremental model:** In the Incremental Model, software is built and delivered in parts, with each increment adding features until the full system is complete. It's structured and good for larger projects with mostly clear requirements. Agile Scrum, however, is faster and more flexible—using short sprints and continuous feedback to handle changing needs better, especially in small to medium projects.
- **DSDM:** DSDM follows strictly timeboxing. This model is often chosen when time and cost are fixed but scope is negotiable. Our project plan adds new features in every sprint, and Scrum is better for this because it can easily handle changes and updates.
- **FDD:** FDD usually requires a large development team and follows short iterations of about 1 to 5 days. Since our team is small and does not have enough members to manage such a

process, we decided to use the Scrum model, which is better suited for smaller teams and easier to manage.

- **XP:** We are choosing Scrum over XP because MetroSheba needs organized project management, clear team roles, and the ability to grow with bigger teams. Scrum's sprint system helps deliver features step by step while also handling changing needs smoothly. Unlike XP, Scrum does not require strict methods like pair programming or always having the customer on-site, which is difficult in a government project. So, Scrum is better for MetroSheba's teamwork and large project needs
- **Prototyping model:** In the Prototyping model, requirements are not clear at the start, so a prototype is built and updated many times based on customer feedback. This helps in identifying needs but can get unstructured for large systems. Scrum, on the other hand, manages changing requirements through short sprints, where working features are delivered step by step and improved with customer feedback. For MetroSheba, Scrum is better than simple prototyping because it provides regular updates, clear roles, and structured teamwork for a large public project.
- **RAD:** In the RAD model, different parts of the system are built in parallel like mini projects with fixed requirements, and then combined into a working system, usually within 90 ± 30 days. This gives the customer a quick functional system but works best when requirements are stable. Scrum, on the other hand, handles changing requirements better with short sprints, continuous feedback, and flexible scope. For MetroSheba, where requirements may evolve (ticketing, payments, metro schedule updates), Scrum is more suitable than RAD because it supports adaptability, teamwork, and long-term scalability

1.3 Project Role Identification and Responsibilities

- **Product Owner:** Decides what features the app needs like ticket booking, station selection, payments, and QR code. Make sure it matches the user's and business needs.
- **Scrum Master:** Helps the team follow Agile steps, runs daily meetings, and solves issues so work isn't delayed.
- **Scrum Team:** Designs develop, and tests the app in short sprints, based on feedback from users.
- **Management:** Sets goals, timelines, and ensures the app meets all requirements and is delivered on time.
- **Customer:** Passengers using the app to buy tickets, view routes, or scan QR codes. Their feedback helps improve the app

2. PRODUCT REQUIREMENTS DOCUMENT (PRD)

2.1 Functional Requirements

1. **Major Functionalities of the System (MetroSheba)**

    1. **Authentication & Access Control**

        - User and admin login, registration, and logout.
        - Password recovery (forgot/reset password).
    2. **Passenger Features**

        - Dashboard showing ticket options and travel details
        - Select ticket quantity and view related options.
        - View purchased ticket details and QR codes.
        - Access payment options (Card, bKash, Nagad, Rocket, etc.).

    3. **Admin Features**

        - Admin Dashboard to view customer information.
        - Manage tickets, payments, and passenger records.

    4. **Profile Management**

        - View profile (user and admin).
        - Edit profile details (name, email, phone, etc.).
        - Change profile photo.
        - Change password.

    5. **Payment & Ticketing**

        - Multiple payment methods (Card, Mobile Banking, etc.).
        - Secure ticket purchase flow.
        - QR-based ticket verification

2. Core Services, Operations, and Features of the MetroSheba System

Core Service	Operation	Feature
Authentication	Login	Users/Admins can log in with email and password.
	Register	Users/Admins can create an account with personal details.
	Forget Password	Users/Admins can reset their password securely
Passenger Dashboard	Ticket Selection	View and choose ticket quantity/details
Ticketing	Ticket Details & QR	Passengers can view ticket details and get QR codes for validation.
Payment	Multiple Payment Methods	Users can pay via Card, bKash, Nagad, Rocket, etc.
Admin Dashboard	Manage Records	Admins can view customer information, ticket usage, and system activity.
Profile Management	View Profile	Users/Admins can view their profile details.
	Edit Profile	Users/Admins can update name, email, phone, etc.
	Change Profile Photo	Users/Admins can upload or change their profile photo.
	Change Password	Users/Admins can update their password for security.

3. Functions and Their Contribution to Project Objectives

1. **Authentication (Login, Register, Forgot Password, Role Selection)**
   - Ensures secure access for both users and admins.
   - Protects passenger and transaction data, meeting the objective of a smart & secure system.
2. **Passenger Dashboard (Ticket Selection, Travel Menu)**
   - Provides passengers with an easy-to-use interface to browse and select tickets.
   - Supports the objective of improving user convenience and digital ticket booking.
3. **Ticketing (Ticket Details & QR Code)**
   - Generates digital tickets with QR codes for quick validation**.**
   - Reduces the need for paper tickets, making the system faster, modern, and eco-friendly**.**
4. **Admin Dashboard (Manage Records & Customer Data)**
   - Gives admins real-time oversight of passenger activities, ticket sales, and payments.
   - Supports objectives of system monitoring, fraud prevention, and better service management**.**
5. **Payment (Multiple Payment Methods)**
   - Allows users to pay via card or mobile banking (bKash, Nagad, Rocket).
   - Supports the goal of providing flexibility, security, and convenience in transactions.
6. **Profile Management (View, Edit, Change Photo, Change Password)**
   - Let's users and admins manage their information securely.
   - Supports personalization and ensures users have control over their accounts**.**

4. User Workflows in the MetroSheba System

Workflow 1: User Registration & Login

- User opens the MetroSheba web application.
- On the home page, the user selects Register.
- User fills in details (Name, Email, Phone, Password).
- System validates input, If valid, account is created.
- User receives confirmation message.
- Next, user selects Login, enters email & password.
- System verifies credentials and Grants access to the dashboard.

Workflow 2: Book and Pay for a Ticket

- Logged-in user navigates to Passenger Dashboard.
- User selects Ticket Menu then chooses travel route, ticket type, and quantity.
- System shows fare details.
- User proceeds to Payment.
- Chooses payment method (Card, bKash, Nagad, Rocket, etc.).

- Completes payment after that System verifies transaction.
- System generates Digital Ticket with QR Code.
- User can view and download ticket from Ticket Details section.

Workflow 3: Profile Update

- Logged-in user goes to Profile Section.
- Selects Edit Profile then updates name, phone, or email.
- Optionally, user selects Change Photo to upload a new profile picture.
- User can also choose Change Password.
- System verifies changes after that updates profile securely.
- Confirmation message is displayed.

5. Acceptance Criteria for MetroSheba System

1. Authentication (Login, Register, Forgot Password)

- Users/Admins must be able to register with valid inputs (name, email, phone, password).
- Login only works with correct credentials stored in the database.
- Wrong credentials show an error message.
- Password reset only works if the email exists in the system.
- System must send confirmation/notification after successful action.

2. Passenger Dashboard & Ticket Selection

- Dashboard loads within 3 seconds after login.
- Ticket selection requires mandatory inputs (route, ticket type, quantity).
- Invalid/missing inputs must trigger a validation message.

3. Ticketing (Ticket Details & QR Code)

- After successful payment, a digital ticket with a unique QR code must be generated.
- QR code must scan correctly and display ticket details (route, date, time, passenger info).
- Ticket details must always match the booking data.

4. Admin Dashboard

- Admin must see real-time data (tickets sold, passenger info, payments).
- Data must be updated automatically after each new transaction.
- Unauthorized users must not access the admin dashboard.

5. **Payment**

- Users must be able to choose from multiple payment methods (Card, bKash, Nagad, Rocket).
- Payment gateway must confirm success or failure.
- On success the ticket is generated. On failure there will be an error message, no ticket issued.
- Transaction logs must be stored securely.

6. **Profile Management**

- Profile page must display correct user/admin information.
- Edit Profile must update information in the database and reflect changes instantly.
- Change Photo must replace old photo with the new one.
- Change Password must only succeed if the current password matches the database record.

2.2 Non-Functional Requirements-

Quality Attributes of MetroSheba System

1. **Performance**

- The system should load the dashboard and core screens within 2–3 seconds.
- Ticket booking and payment processing must be completed in <5 seconds.
- Must support hundreds of concurrent users during peak hours without slowdown.

2. **Reliability**

- The system should maintain 99.5% uptime during metro service hours.
- Automatic recovery mechanisms for failed transactions (e.g., payment retries).
- Ticket and QR code details must always be available after booking.

3. **Integrity / Security**

- All sensitive data (passwords, payments) must be encrypted.
- Role-based authentication (User vs Admin) ensures proper access control.
- Protect against unauthorized access, data leaks, and fraud attempts.
- User privacy is maintained by storing only essential personal data.

4. **Usability**

- Provide a clean, simple interface for both passengers and admins.
- Mobile-friendly responsive design.

- Clear error messages and guidance for smooth user experience.
- Ensure accessibility for all users (easy navigation, readable text, icons).

5. **Maintainability**

- Modular codebase for easy bug fixing and updates.
- Well-documented system for developer support.
- Logging and monitoring tools to track errors and performance issues.

6. **Scalability**

- Should handle a growing number of passengers and tickets as metro usage increases.
- Support adding new routes, stations, or ticket types without redesigning the whole system.
- Allow integration of new payment gateways or transport services in the future.

3. PROJECT ESTIMATION AND SCHEDULING

3.1 Effort and Cost Estimation

1. **Define the scope of the project**
   The project is assumed to be a large-scale software system as it is a web-based service platform with an estimated size of 100,000 Source Lines of Code (SLOC). The scope includes core functionalities, user interfaces, and backend logic.

2. **Effort Estimation Using Lines of Code (LOC) and Productivity Rates**

$$E = [LOC \times B^{0.333}/P]^3 \times (1/t^4)$$

Where,

E = Effort in person-months or person-years (the amount of time, personnel devote to a specific project)

t (Project duration in months or years) =1.5 Months

B (Productivity Factors) = 1.2

P (Productivity Parameter) =4000 LOC/Person-month

$$E = [100000 \times 1.2^{0.333}/4000]^3 \times (1/1.5^4)$$

E= 3.703 person month

3. **Applying COCOMO Model**

PM: person-months needed for project (labor working hours)
SLOC: source lines of code
P: project complexity (1.04-1.24)
DM: duration time in months for project (weekdays)
T: SLOC-dependent coefficient (0.32-0.38)
ST: average staffing necessary

Coefficient$_{<\text{Effort Factor}>}$ = 3

Effort = PM = Coefficient$_{<\text{Effort Factor}>}$*(SLOC/1000) ^P = 3*(100,000/1000) ^1.1 = 475.468 person-days
Development time = DM = 2.50*(PM)^T = 2.50*(475.468) ^0.33 = 19.116 months

- **Required number of people** = ST = PM/DM = 475.468/19.116 = 24.87 = 25 person

3.2 Project Scheduling

1. **Project Work Breakdown and Task Assignment for Metrosheba**

Sprint 1 (Core setup)

- **Sprint 1 Planning**
  - Meet with team
  - Define sprint backlog
- **Sprint 1 Development**
  - Basic UI – Homepage
  - Backend API – Booking
- **Sprint 1 Testing**
  - Unit Testing
  - Integration Testing
- **Sprint 1 Review**
  - Stakeholder Demo
- **Sprint 1 Retrospective**
  - Process Improvement Notes
  - Sprint 1 Complete

Sprint 2 (Booking and Payment)

- **Sprint 2 Planning**
  - Refine backlog
- **Sprint 2 Development**

    - Design Booking Page
    - Payment Integration
- **Sprint 2 Testing**
    - Booking Feature Test
    - Payment Gateway Test
- **Sprint 2 Review**
    - Sprint Demo
- **Sprint 2 Retrospective**
    - Lessons Learned
    - Sprint 2 Complete

Sprint 3 (Final Feature)

- **Sprint** 3 Planning
    - Plan Final Features
- **Sprint 3 Development**
    - Notifications Feature
    - Dashboard + Analytics
- **Sprint 3 Testing**
    - Notifications Test
    - Dashboard Test
- **Sprint 3 Review**
    - Stakeholder Feedback
- **Sprint 3 Retrospective**
    - Continuous Improvement Notes
    - Sprint 3 Complete

Sprint 4 (Post-Launch & Maintenance)

- **Sprint 4 Planning**
    - Plan Deployment
- **Sprint 4 Execution**
    - Deployment
    - Fix Post-launch Bugs
    - Release New Features
- **Sprint 4 Review**
    - Final Demo
- **Sprint 4 Retrospective**
    - Post-Mortem Document
    - Sprint 4 Complete

2. **Effort Allocation for Metrosheba Project**

1. Analysis & Design (40%)

- Product Backlog Refinement: understanding requirements.
- Sprint Planning: designing features & defining acceptance criteria.
- Design Sessions: UI mockups, architecture discussions.

2. Coding / Development (20%)

- Sprint Execution: developers implement user stories.
- Daily Scrums: track coding progress, remove blockers.

3. Testing & Quality Assurance (40%):

- Unit & Integration Testing: as developers code.
- Sprint Review & Demo: validate with stakeholders.
- Sprint Retrospective: identify improvements.
- Continuous Testing: automated & manual checks.

3. **Earn Value Analysis**

BCWS (Budget Cost of Work Schedule) = 146.7
BCWP (Budget Cost of Work Performed) = 133.7
BAC (Budget at Completion) = 672
ACWS (Actual Cost of Work Schedule) = 137.7

- **SPI** = BCWP/ BCWS = 133.7/ 146.7 = 0.911
- **SV** = BCWP - BCWS = 133.7 - 146.7 = - 13person-day
- **CPI** = BCWP/ ACWP = 133.7/137.7 = 0.97
- **CV** = BCWP – ACWP = 133.7/137.7 = - 4 person-day
- **Percentage schedule for completion** = BCWS/BAC = 146.7/ 672
  = 0.21
  =21%
- **Percentage complete** = BCWP/ BAC = 133.7/ 672 = 0.199
  =20%

Schedule Performance

- **SPI** = 0.911; project is a little behind schedule.
- **SV** = –13; only 13 person-days behind.

Cost Performance

- **CPI** = 0.97 (the project is a bit cost-inefficiency)

- **CV** = –4 (overspent by only 4 person-days)

Completion

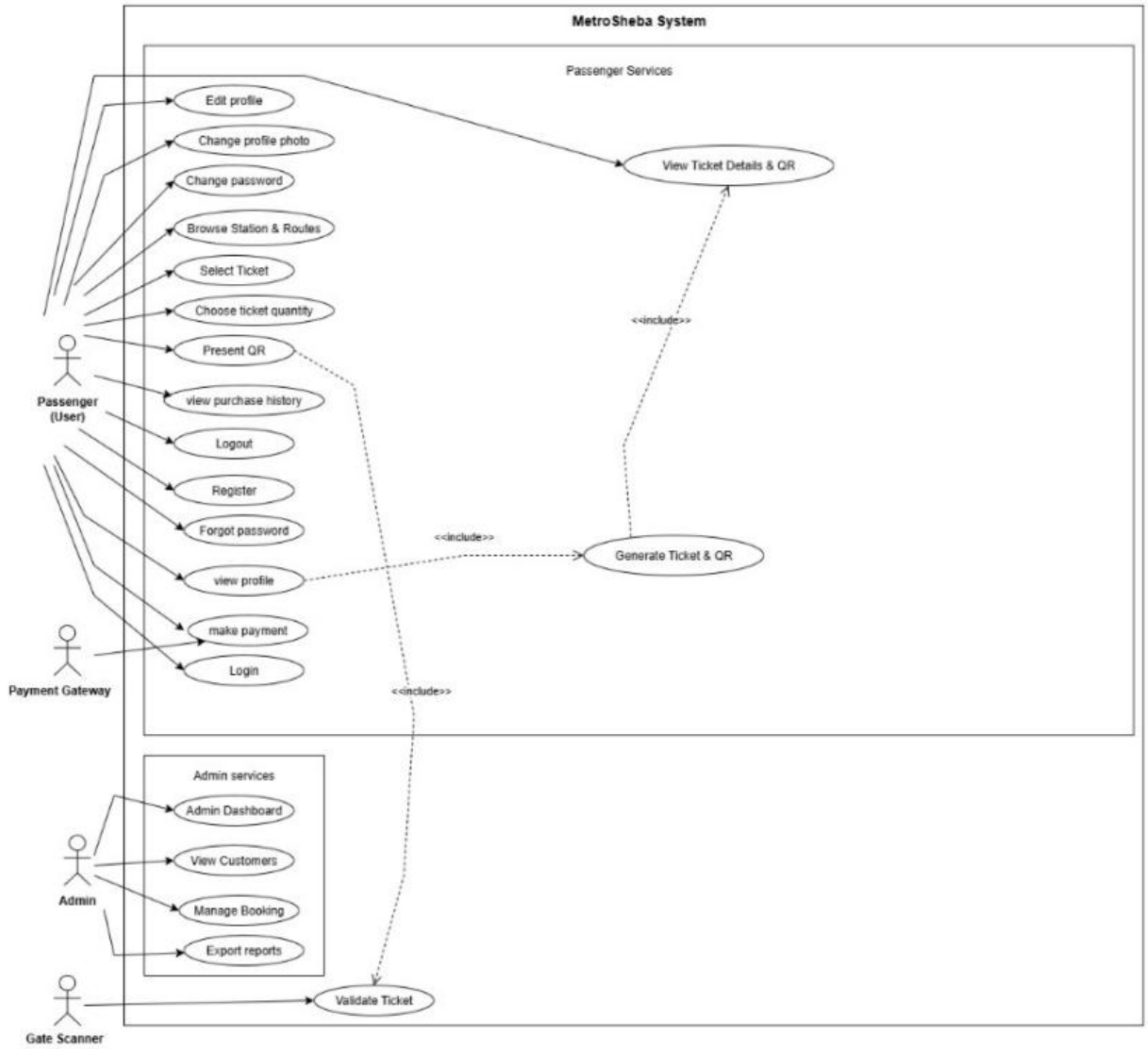
- Planned progress = 21%
- Actual progress = 20%

So, the progress is almost as planned (only ~1% behind).

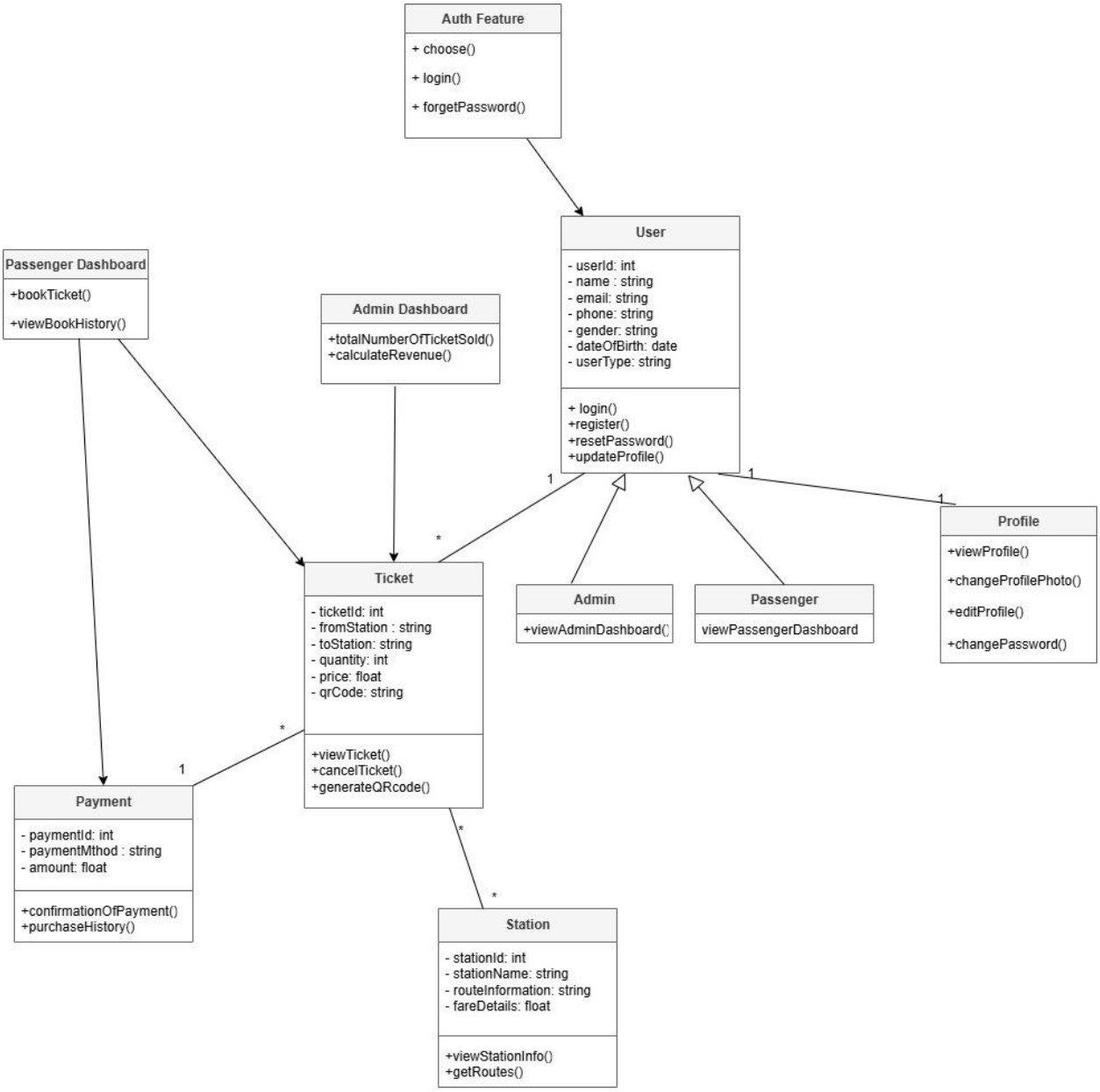
4. SOFTWARE DESIGN

4.1 System Design:

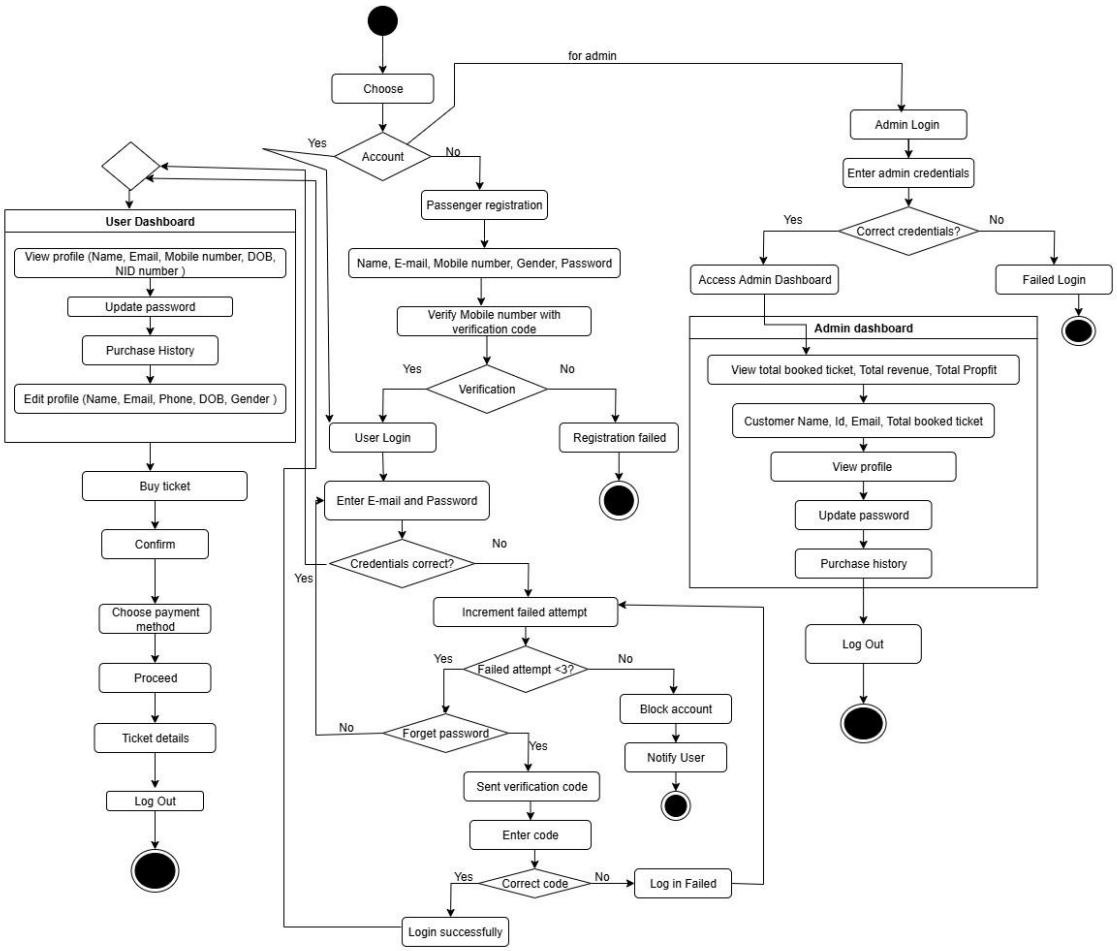
- **Use Class Diagram**



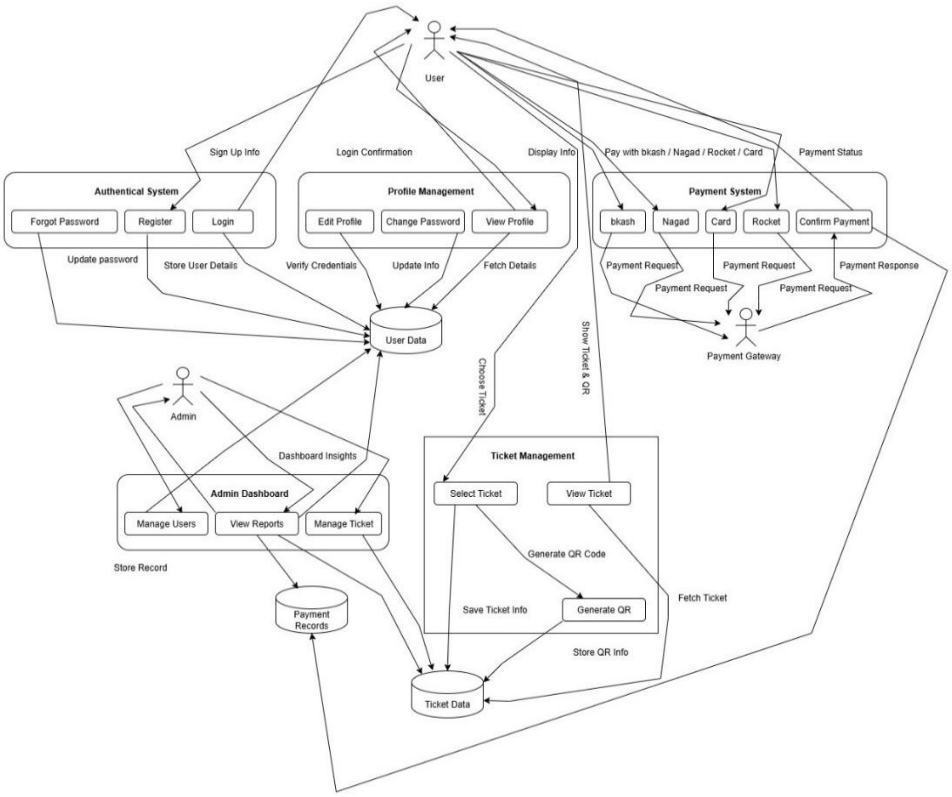
- **Class Diagram**



- **Activity Diagram**



- **Data Flow Diagram (DFD)**



4.2 UI / Wireframe Design using Figma

Figure 1: Choose page

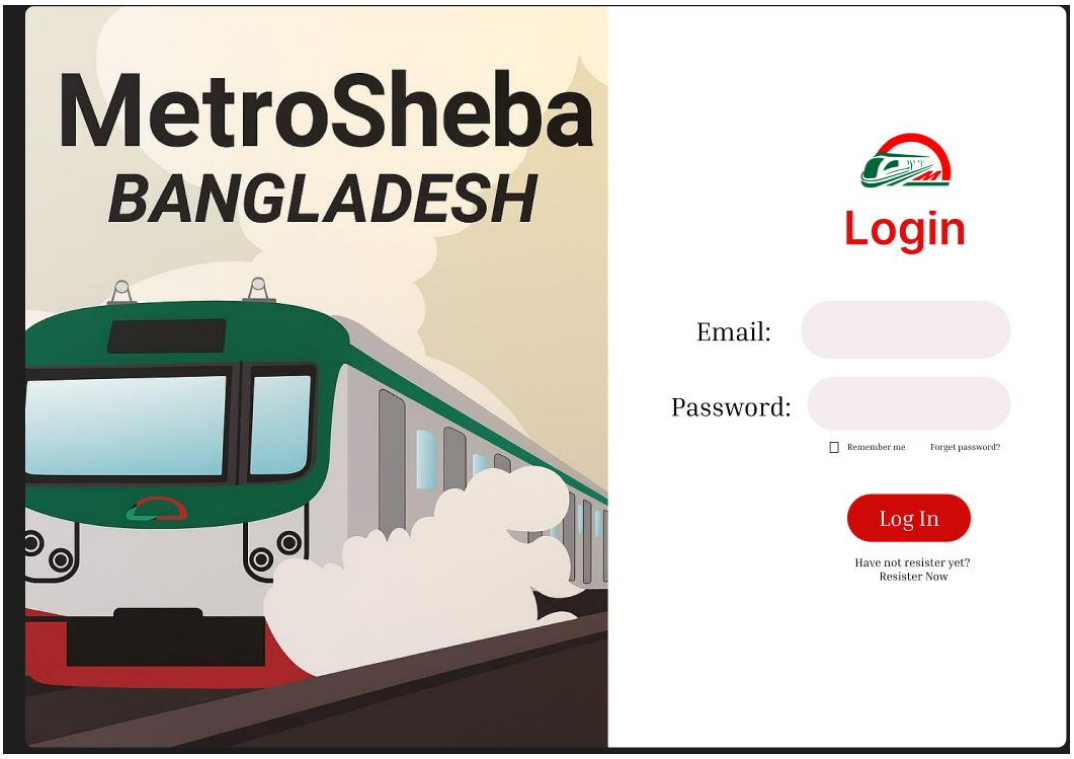


Figure 2: Login page

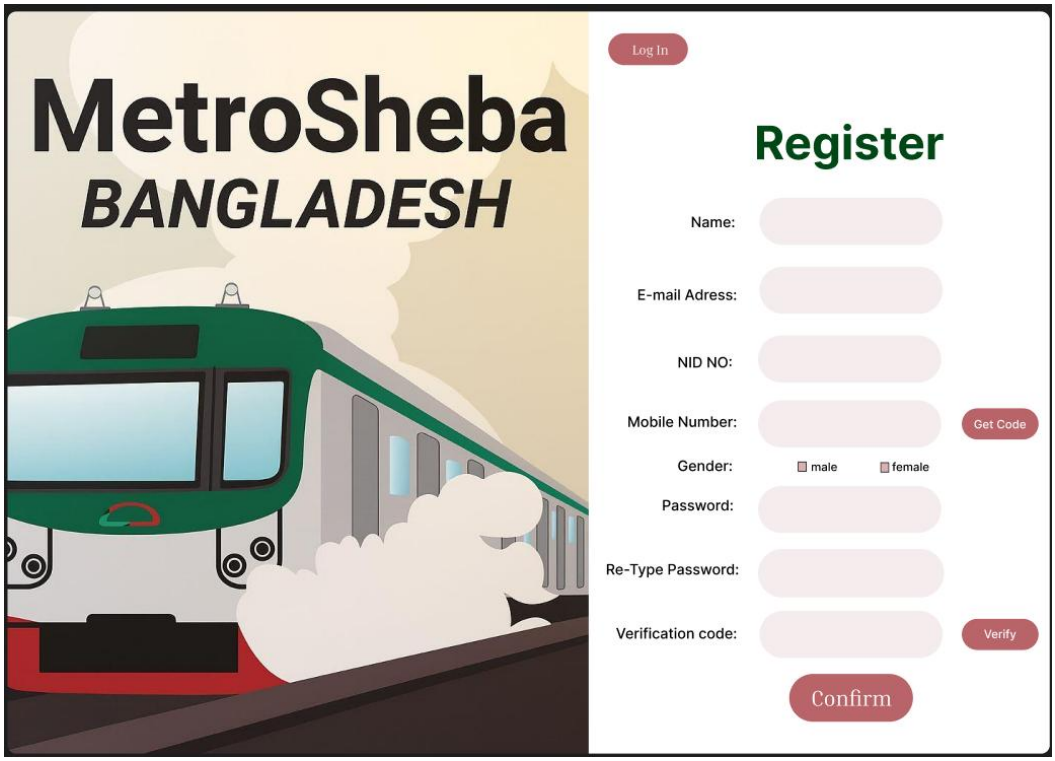


Figure 3: Register page

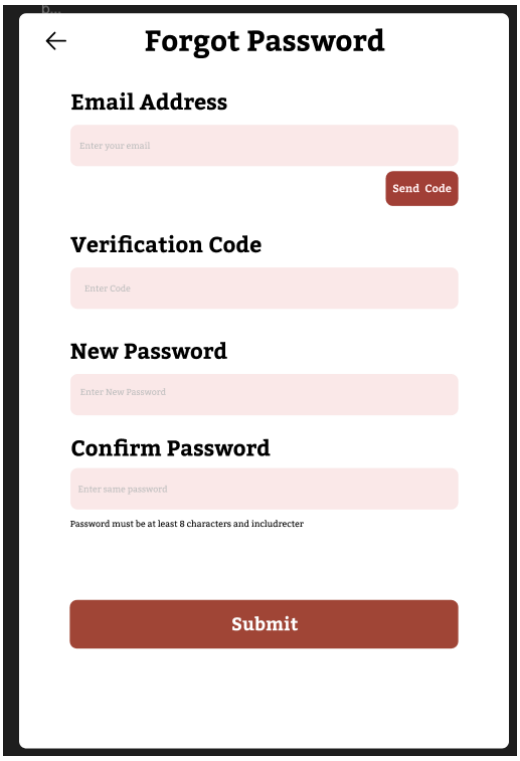


Figure 4: Forgot Password Page

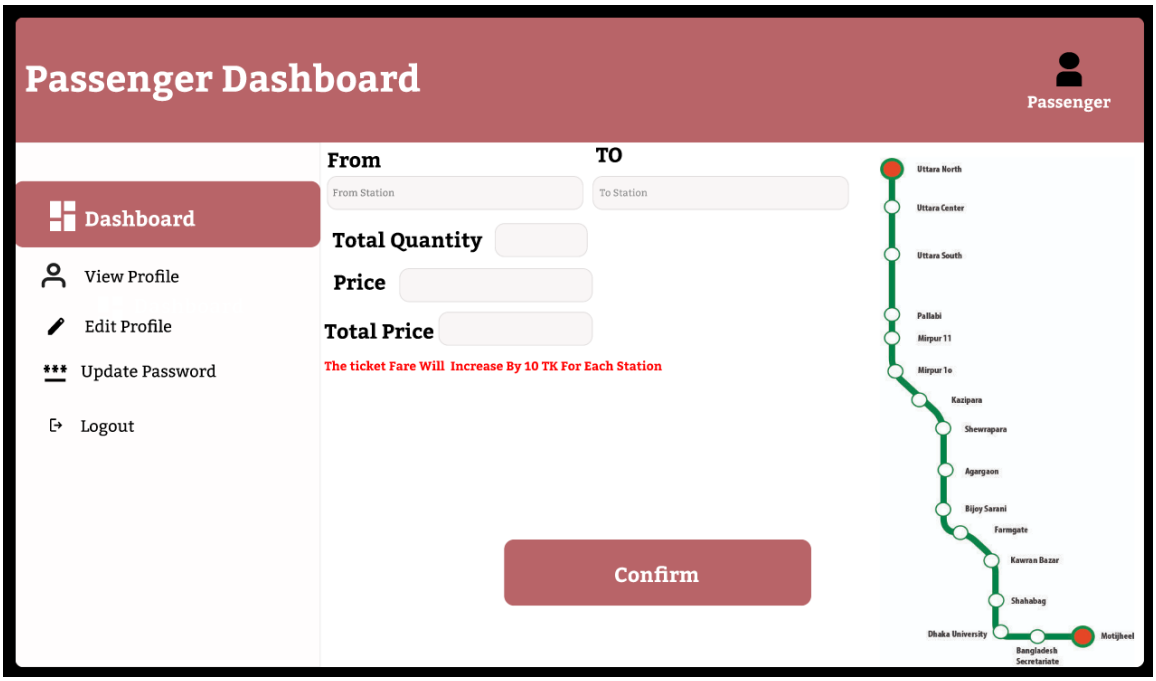


Figure 5: Passenger Dashboard page

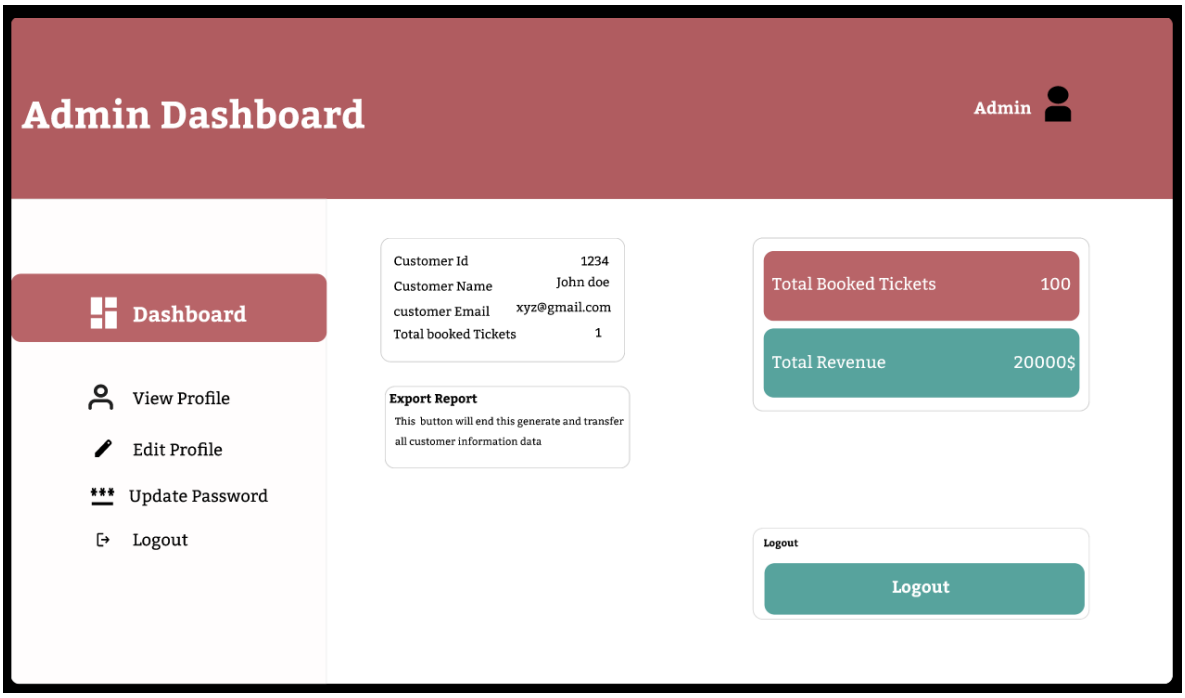


Figure 6: Admin Dashboard page

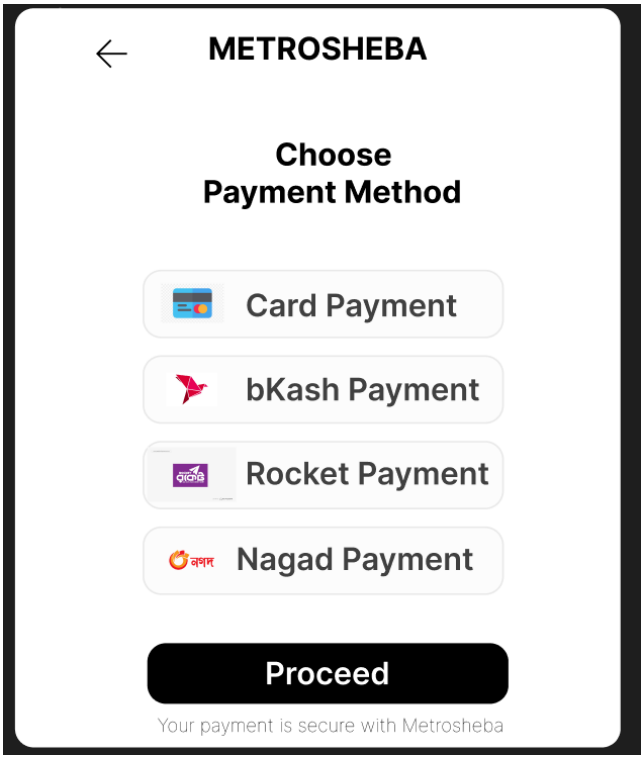


Figure 7: Payment page

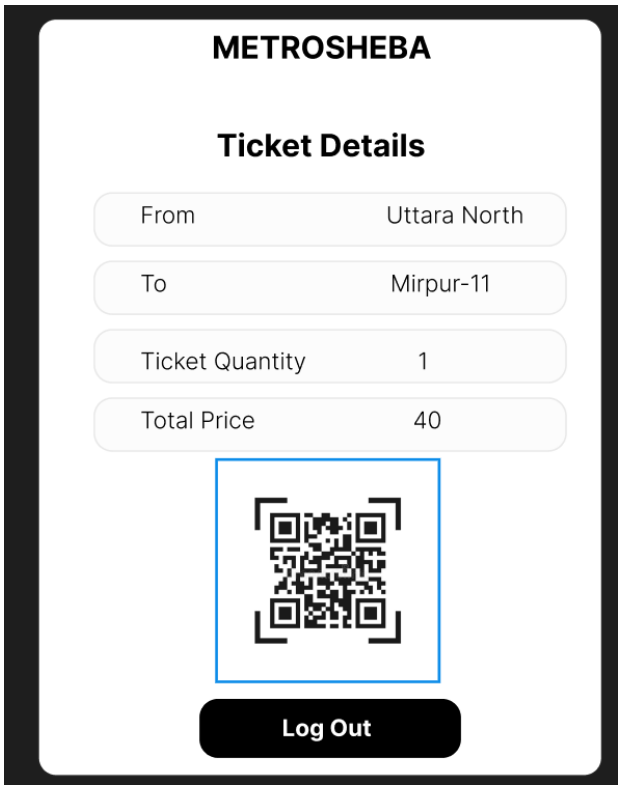


Figure 8: Ticket Details

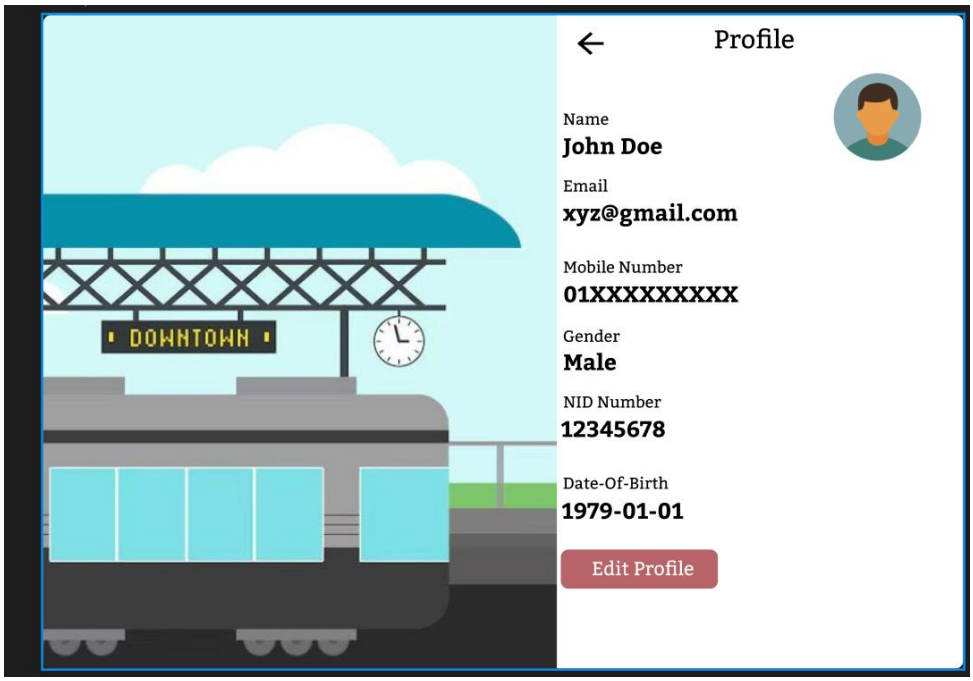


Figure 9: View Profile page

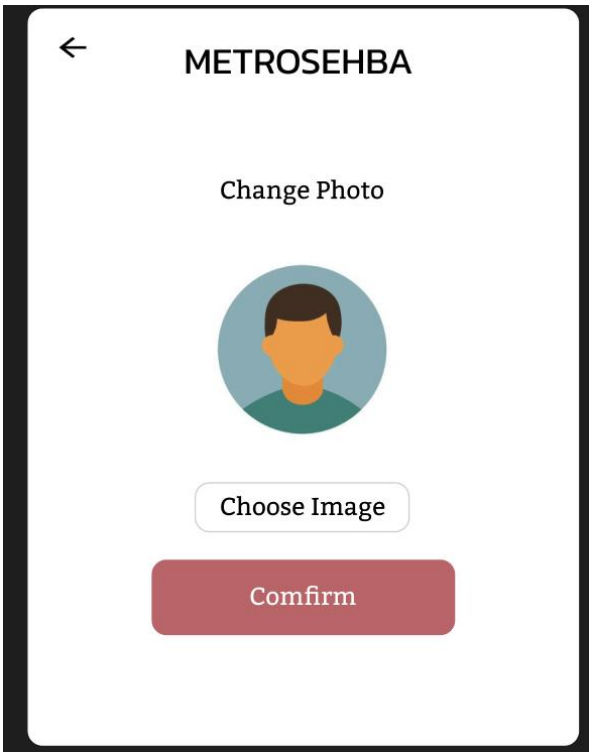


Figure 10: Change Photo page

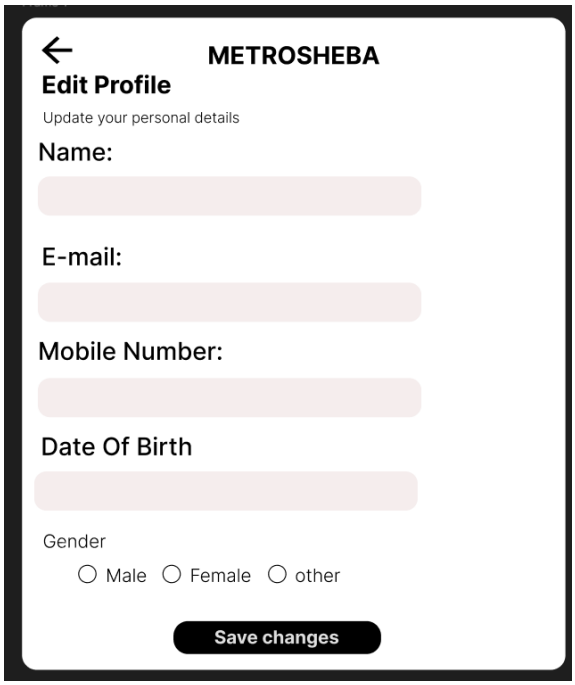


Figure 11: Edit Profile page

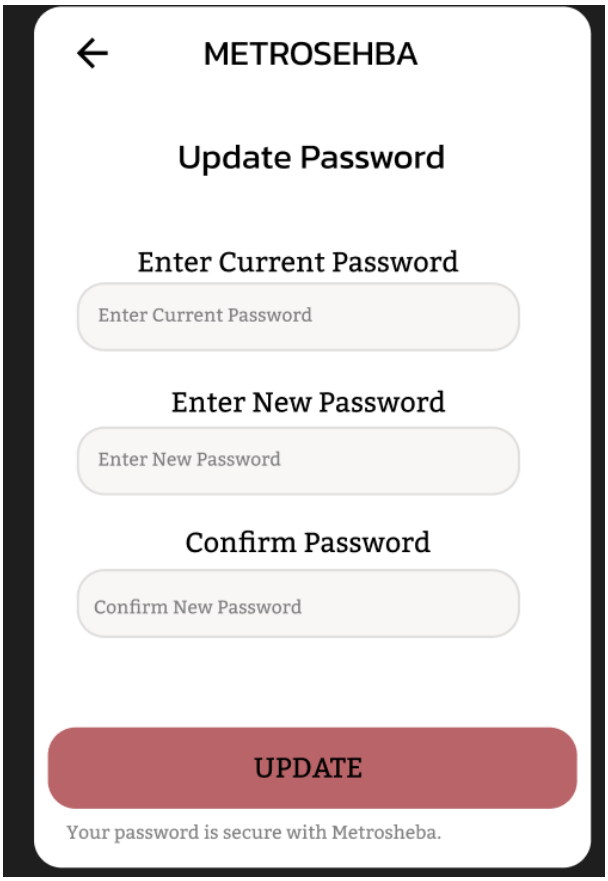
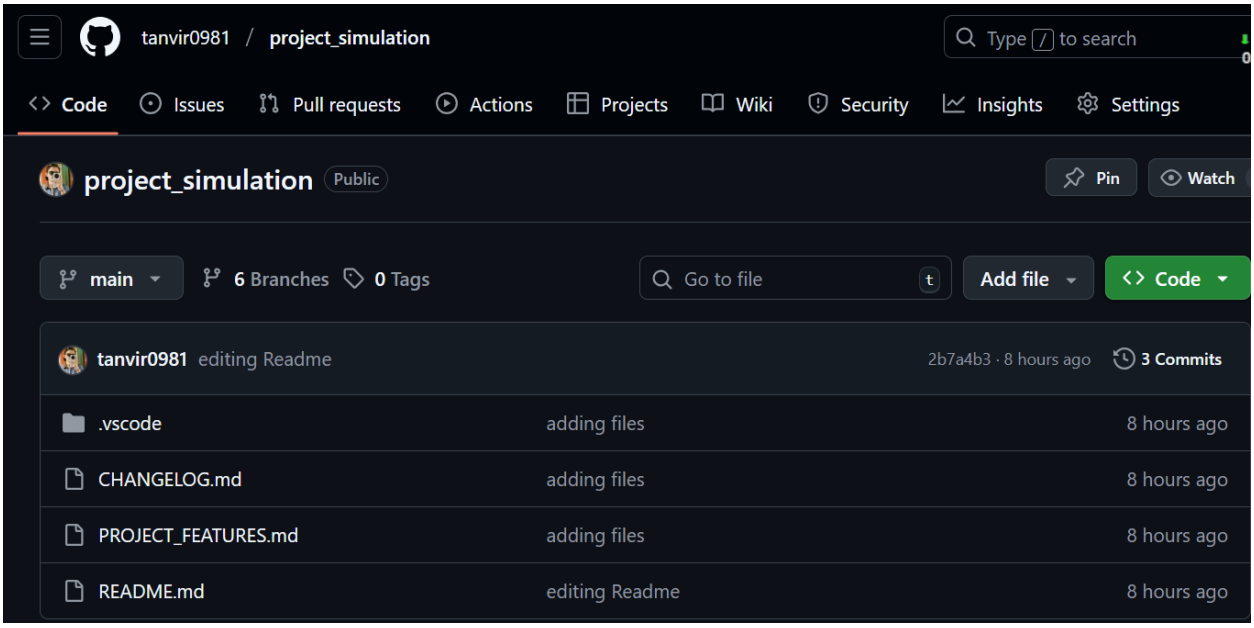


Figure 12: Update Password page

5. GIT WORKFLOW

Repository Setup




```
ADMIN@Tanvir MINGW64 ~/Downloads/Git WorkFlow (main)
$ git branch dev

ADMIN@Tanvir MINGW64 ~/Downloads/Git WorkFlow (main)
$ git branch stage

ADMIN@Tanvir MINGW64 ~/Downloads/Git WorkFlow (main)
$ git push origin dev
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'dev' on GitHub by visiting:
remote:      https://github.com/tanvir0981/project_simulation/pull/new/dev
remote:
To https://github.com/tanvir0981/project_simulation.git
$ git push -u origin stage
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'stage' on GitHub by visiting:
remote:      https://github.com/tanvir0981/project_simulation/pull/new/stage
remote:
To https://github.com/tanvir0981/project_simulation.git
 * [new branch]      stage -> stage
branch 'stage' set up to track 'origin/stage'.
```

```
ADMIN@Tanvir MINGW64 ~/Downloads/Git WorkFlow (master)
$ echo "# project_simulation" >> README.md

ADMIN@Tanvir MINGW64 ~/Downloads/Git WorkFlow (master)
$ git init
Initialized empty Git repository in C:/Users/ADMIN/Downloads/Git WorkFlow/.git/

ADMIN@Tanvir MINGW64 ~/Downloads/Git WorkFlow (master)
ADMIN@Tanvir MINGW64 ~/Downloads/Git WorkFlow (master)
$ git commit -m "first commit"
[master (root-commit) 40872f5] first commit
 1 file changed, 1 insertion(+)
 create mode 100644 README.md

ADMIN@Tanvir MINGW64 ~/Downloads/Git WorkFlow (master)
$ git branch -M main

ADMIN@Tanvir MINGW64 ~/Downloads/GADMIN@Tanvir MINGW64 ~/Downloads/Git WorkFlow (master)
$ git commit -m "first commit"
[master (root-commit) 40872f5] first commit
 1 file changed, 1 insertion(+)
 create mode 100644 README.md

ADMIN@Tanvir MINGW64 ~/Downloads/Git WorkFlow (master)
$ git branch -M main

ADMIN@Tanvir MINGW64 ~/Downloads/Git WorkFlow (main)
$ git remote add origin https://github.com/tanvir0981/project_simulation.git

ADMIN@Tanvir MINGW64 ~/Downloads/Git WorkFlow (main)
$ git push -u origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 234 bytes | 234.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/tanvir0981/project_simulation.git
```

```
ADMIN@Tanvir MINGW64 ~/Downloads/Git WorkFlow (main)
$ echo "# Changelog" > CHANGELOG.md

ADMIN@Tanvir MINGW64 ~/Downloads/Git WorkFlow (main)
$ echo "# Implemented Features" > PROJECT_FEATURES.md

ADMIN@Tanvir MINGW64 ~/Downloads/Git WorkFlow (main)
$ git commit -m"adding required files"
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .vscode/
        CHANGELOG.md
        PROJECT_FEATURES.md

nothing added to commit but untracked files present (use "git add" to track)

ADMIN@Tanvir MINGW64 ~/Downloads/Git WorkFlow (main)
$ git add .
warning: in the working copy of 'CHANGELOG.md', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'PROJECT_FEATURES.md', LF will be replaced by CRLF the next time Git touches it

ADMIN@Tanvir MINGW64 ~/Downloads/Git WorkFlow (main)
$ git commit -m"adding files"
[main 30e7781] adding files
 3 files changed, 5 insertions(+)
 create mode 100644 .vscode/settings.json
 create mode 100644 CHANGELOG.md
 create mode 100644 PROJECT_FEATURES.md

ADMIN@Tanvir MINGW64 ~/Downloads/Git WorkFlow (main)
$ git push origin main
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 12 threads
```

Student Setup - Clone and Configure

Other Person Task

```
MINGW64:/c/Users/Ullas/project_simulation
Ullas@Hungry_Beast MINGW64 ~/project_simulation (dev)
$ git pull origin dev
From https://github.com/tanvir0981/project_simulation
 * branch            dev        -> FETCH_HEAD
Already up to date.

Ullas@Hungry_Beast MINGW64 ~/project_simulation (dev)
$ git checkout -b feature/passenger_dashboard
Switched to a new branch 'feature/passenger_dashboard'

Ullas@Hungry_Beast MINGW64 ~/project_simulation (feature/passenger_dashboard)
$ echo "## Passenger Dashboard: Implement Passenger Dashboard Page" >> PROJECT_F
EATURES.md

Ullas@Hungry_Beast MINGW64 ~/project_simulation (feature/passenger_dashboard)
$ git add .
warning: in the working copy of 'PROJECT_FEATURES.md', LF will be replaced by CR
LF the next time Git touches it

Ullas@Hungry_Beast MINGW64 ~/project_simulation (feature/passenger_dashboard)
$ git commit -m"This task implementing Passenger Dashboard"
[feature/passenger_dashboard a2f7f4b] This task implementing Passenger Dashboard
 2 files changed, 1 insertion(+)
 create mode 100644 FETCH_HEAD

Ullas@Hungry_Beast MINGW64 ~/project_simulation (feature/passenger_dashboard)
$ git push -u origin feature
error: src refspec feature does not match any
error: failed to push some refs to 'https://github.com/tanvir0981/project_simula
tion.git'

Ullas@Hungry_Beast MINGW64 ~/project_simulation (feature/passenger_dashboard)
$

Ullas@Hungry_Beast MINGW64 ~/project_simulation (feature/passenger_dashboard)
$  git push -u origin feature/passenger_dashboard
remote: Permission to tanvir0981/project_simulation.git denied to Ullas50.
fatal: unable to access 'https://github.com/tanvir0981/project_simulation.git/':
 The requested URL returned error: 403

Ullas@Hungry_Beast MINGW64 ~/project_simulation (feature/passenger_dashboard)
$ git push -u origin feature/passenger_dashboard
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 484 bytes | 484.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'feature/passenger_dashboard' on GitHub by visiting:
remote:      https://github.com/tanvir0981/project_simulation/pull/new/feature/passenger_dashboard
remote:
To https://github.com/tanvir0981/project_simulation.git
 * [new branch]      feature/passenger_dashboard -> feature/passenger_dashboard
branch 'feature/passenger_dashboard' set up to track 'origin/feature/passenger_dashboard'.

Ullas@Hungry_Beast MINGW64 ~/project_simulation (feature/passenger_dashboard)
$
```

Other Person Task

```
MINGW64:/c/Users/User/project_simulation
The most similar commands are
        pull
        push

User@MHYEASIB28NOV MINGW64 ~/project_simulation (dev)
$ git pull origin dev
From https://github.com/tanvir0981/project_simulation
 * branch            dev        -> FETCH_HEAD
Already up to date.

User@MHYEASIB28NOV MINGW64 ~/project_simulation (dev)
$ git checkout -b feature/Admin_dashboard
Switched to a new branch 'feature/Admin_dashboard'

User@MHYEASIB28NOV MINGW64 ~/project_simulation (feature/Admin_dashboard)
$ echo "## Admin Dasboard: Implement Admin dashboard Page" >> PROJECT_FEATURES.m
d

User@MHYEASIB28NOV MINGW64 ~/project_simulation (feature/Admin_dashboard)
$ git add .
warning: in the working copy of 'PROJECT_FEATURES.md', LF will be replaced by CR
LF the next time Git touches it

User@MHYEASIB28NOV MINGW64 ~/project_simulation (feature/Admin_dashboard)
$ git commit -m"This task implementing Admin Dashboard"
[feature/Admin_dashboard 10e5084] This task implementing Admin Dashboard
 1 file changed, 1 insertion(+)

User@MHYEASIB28NOV MINGW64 ~/project_simulation (feature/Admin_dashboard)
$ git push -u origin feature/Admin_dashboard
remote: Permission to tanvir0981/project_simulation.git denied to MdMostafaHamid
Yeasib.
fatal: unable to access 'https://github.com/tanvir0981/project_simulation.git/':
 The requested URL returned error: 403

User@MHYEASIB28NOV MINGW64 ~/project_simulation (feature/Admin_dashboard)
$ git push -u origin feature/Admin_dashboard
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 332 bytes | 332.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'feature/Admin_dashboard' on GitHub by visitin
g:
remote:      https://github.com/tanvir0981/project_simulation/pull/new/feature/A
dmin_dashboard
remote:
To https://github.com/tanvir0981/project_simulation.git
 * [new branch]      feature/Admin_dashboard -> feature/Admin_dashboard
branch 'feature/Admin_dashboard' set up to track 'origin/feature/Admin_dashboard
'.

User@MHYEASIB28NOV MINGW64 ~/project_simulation (feature/Admin_dashboard)
$
```

Other Person Task

```
MINGW64:/c/Users/USER/project_simulation/project_simulation
$ ^C

USER@LAPTOP-K1B0TUAP MINGW64 ~/project_simulation/project_simulation (main)
$ git checkout dev && git pull origin dev
branch 'dev' set up to track 'origin/dev'.
Switched to a new branch 'dev'
From https://github.com/tanvir0981/project_simulation
 * branch            dev        -> FETCH_HEAD
Already up to date.

USER@LAPTOP-K1B0TUAP MINGW64 ~/project_simulation/project_simulation (dev)
$ git checkout stage && git pull origin stage
branch 'stage' set up to track 'origin/stage'.
Switched to a new branch 'stage'
From https://github.com/tanvir0981/project_simulation
 * branch            stage      -> FETCH_HEAD
Already up to date.

USER@LAPTOP-K1B0TUAP MINGW64 ~/project_simulation/project_simulation (stage)
$ git checkout dev
Switched to branch 'dev'
Your branch is up to date with 'origin/dev'.

USER@LAPTOP-K1B0TUAP MINGW64 ~/project_simulation/project_simulation (dev)
$ get pull origin dev
bash: get: command not found

USER@LAPTOP-K1B0TUAP MINGW64 ~/project_simulation/project_simulation (dev)
$ git pull origin dev
From https://github.com/tanvir0981/project_simulation
 * branch            dev        -> FETCH_HEAD
Already up to date.

USER@LAPTOP-K1B0TUAP MINGW64 ~/project_simulation/project_simulation (dev)
$ git checkout -b feature/Ticket_details
Switched to a new branch 'feature/Ticket_details'

USER@LAPTOP-K1B0TUAP MINGW64 ~/project_simulation/project_simulation (feature/Ti
cket_details)
$ echo "## Ticket Details:implement Ticket Details Page">>PROJECT_FEATURES.md

USER@LAPTOP-K1B0TUAP MINGW64 ~/project_simulation/project_simulation (feature/Ti
cket_details)
$ git push -u origin feature/Ticket_details
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'feature/Ticket_details' on GitHub by visiting
:
remote:      https://github.com/tanvir0981/project_simulation/pull/new/feature/T
icket_details
remote:
To https://github.com/tanvir0981/project_simulation.git
 * [new branch]      feature/Ticket_details -> feature/Ticket_details
branch 'feature/Ticket_details' set up to track 'origin/feature/Ticket_details'.

USER@LAPTOP-K1B0TUAP MINGW64 ~/project_simulation/project_simulation (feature/Ti
cket_details)
$
```

Code Review and Merge to Dev (Reviewer's Task)

PROJECT_FEATURES.md M ● CHANGELOG.md

PROJECT_FEATURES.md > # Implemented Features > ## Ticket Details:Implement Ticket Details Page

```md
# Implemented Features
## Passenger Dashboard: Implement Passenger Dashboard Page
## Admin Dashboard: Implement Admin dashboard Page
## Ticket Details:Implement Ticket Details Page
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS bash

```
$ git pull origin dev
From https://github.com/tanvir0981/project_simulation
 * branch            dev        -> FETCH_HEAD
Already up to date.

tanvi@Tanvir MINGW64 ~/project_simulation (dev)
$ git branch -d feature/Admin_dashboard
warning: deleting branch 'feature/Admin_dashboard' that has been merged to
         'refs/remotes/origin/feature/Admin_dashboard', but not yet merged to HEAD
Deleted branch feature/Admin_dashboard (was 10e5084).

tanvi@Tanvir MINGW64 ~/project_simulation (dev)
$ git branch -d feature/passenger_dashboard
warning: deleting branch 'feature/passenger_dashboard' that has been merged to
         'refs/remotes/origin/feature/passenger_dashboard', but not yet merged to HEAD
Deleted branch feature/passenger_dashboard (was a2f7f4b).

tanvi@Tanvir MINGW64 ~/project_simulation (dev)
$ git branch -d feature/Ticket_details
Deleted branch feature/Ticket_details (was 2b7a4b3).

tanvi@Tanvir MINGW64 ~/project_simulation (dev)
$ git push origin --delete feature/Admin_dashboard
To https://github.com/tanvir0981/project_simulation.git
 - [deleted]         feature/Admin_dashboard

tanvi@Tanvir MINGW64 ~/project_simulation (dev)
$ Git push origin --delete feature/Ticket_details
To https://github.com/tanvir0981/project_simulation.git
 - [deleted]         feature/Ticket_details

tanvi@Tanvir MINGW64 ~/project_simulation (dev)
$ Git push origin --delete feature/passenger_dashboard
To https://github.com/tanvir0981/project_simulation.git
 - [deleted]         feature/passenger_dashboard
```

Perform a Hotfix

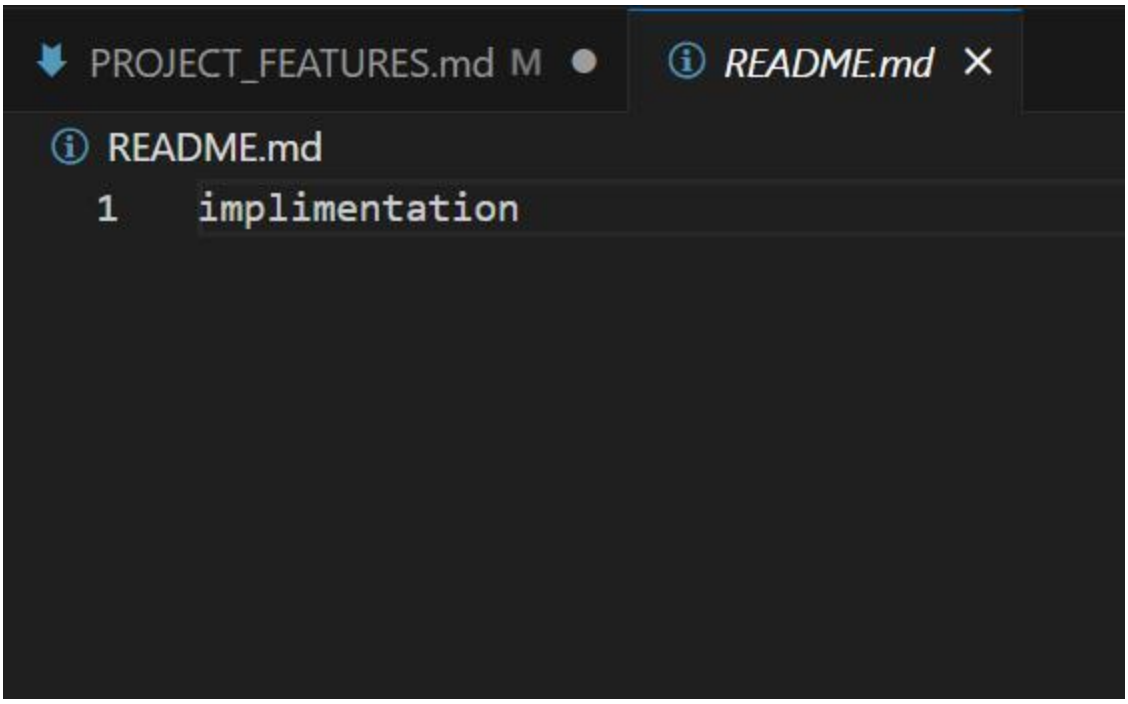


Figure: Discover the issue

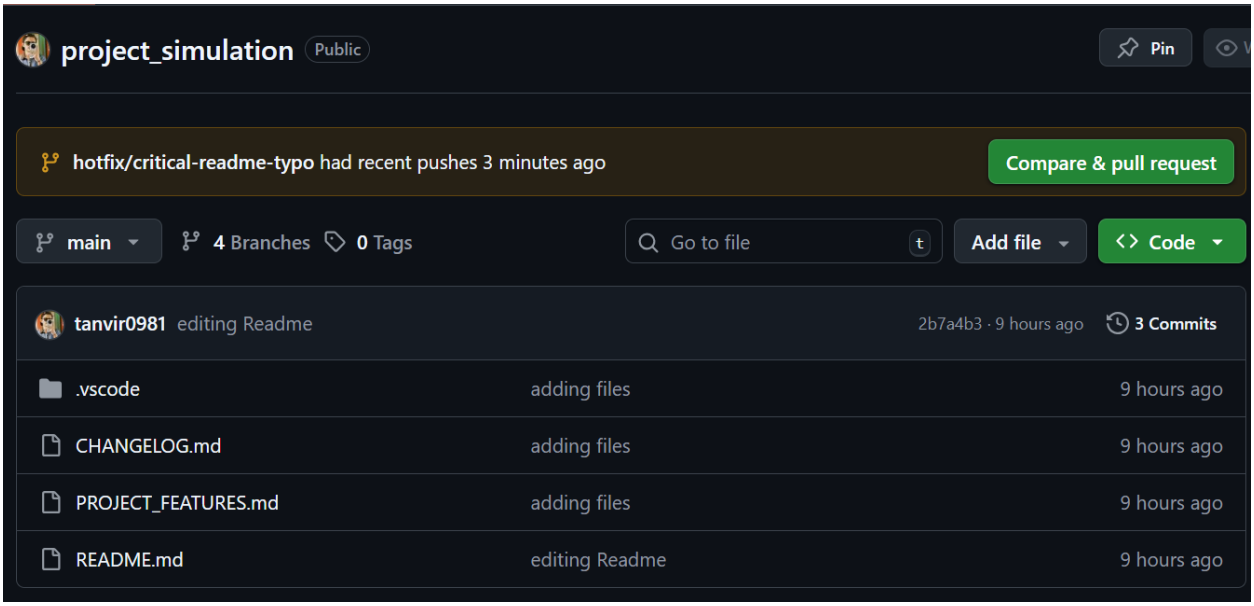


Figure: Create the branch and pull request

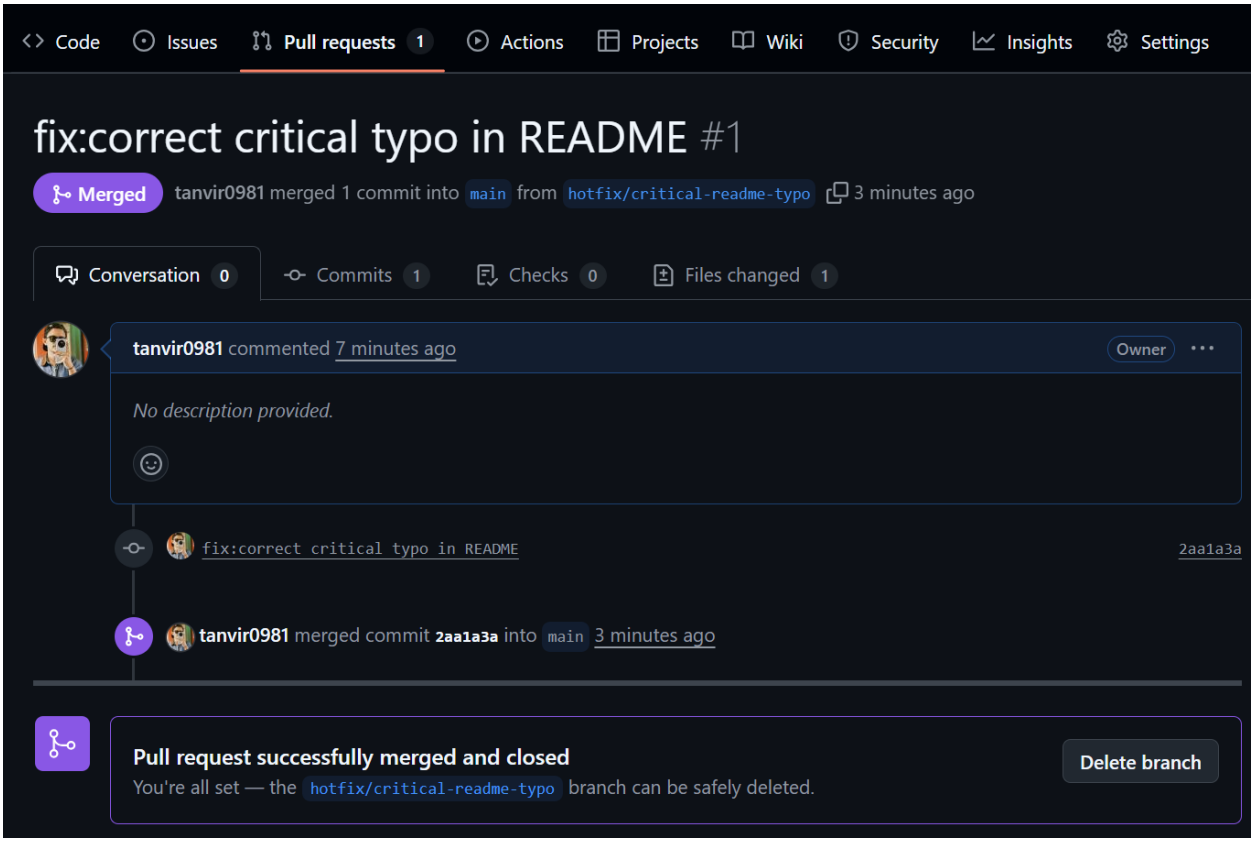


Figure: commit and marge the branch

```
tanvi@Tanvir MINGW64 ~/project_simulation (dev)
$ git checkout main
M       PROJECT_FEATURES.md
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

tanvi@Tanvir MINGW64 ~/project_simulation (main)
$ git pull origin main
From https://github.com/tanvir0981/project_simulation
 * branch            main       -> FETCH_HEAD
Already up to date.

tanvi@Tanvir MINGW64 ~/project_simulation (main)
$ git checkout -b hotfix/critical-readme-typo
Switched to a new branch 'hotfix/critical-readme-typo'

tanvi@Tanvir MINGW64 ~/project_simulation (hotfix/critical-readme-typo)
$ git add README.md

tanvi@Tanvir MINGW64 ~/project_simulation (hotfix/critical-readme-typo)
$ git commit -m"fix:correct critical typo in README"
[hotfix/critical-readme-typo 2aa1a3a] fix:correct critical typo in README
 1 file changed, 1 insertion(+), 1 deletion(-)

tanvi@Tanvir MINGW64 ~/project_simulation (hotfix/critical-readme-typo)
$ git push -u origin hotfix/critical-readme-typo
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 289 bytes | 289.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
```

Figure: Creating branch

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

branch 'hotfix/critical-readme-typo' set up to track 'origin/hotfix/critical-readme-typo'.

tanvi@Tanvir MINGW64 ~/project_simulation (hotfix/critical-readme-typo)
$ git pull origin main
From https://github.com/tanvir0981/project_simulation
 * branch            main       -> FETCH_HEAD
Already up to date.

tanvi@Tanvir MINGW64 ~/project_simulation (hotfix/critical-readme-typo)
$ git checkout origin main
error: pathspec 'main' did not match any file(s) known to git

tanvi@Tanvir MINGW64 ~/project_simulation (hotfix/critical-readme-typo)
$ git checkout main
M       PROJECT_FEATURES.md
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

tanvi@Tanvir MINGW64 ~/project_simulation (main)
$ git merge hotfix/critical-readme-typo
Updating 2b7a4b3..2aa1a3a
Fast-forward
 README.md | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)

tanvi@Tanvir MINGW64 ~/project_simulation (main)
$ git push -u origin main
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/tanvir0981/project_simulation.git
   2b7a4b3..2aa1a3a  main -> main
branch 'main' set up to track 'origin/main'.

tanvi@Tanvir MINGW64 ~/project_simulation (main)
$ git branch -d hotfix/critical-readme-typo
Deleted branch hotfix/critical-readme-typo (was 2aa1a3a).
```

Figure: Solve the problem and merge

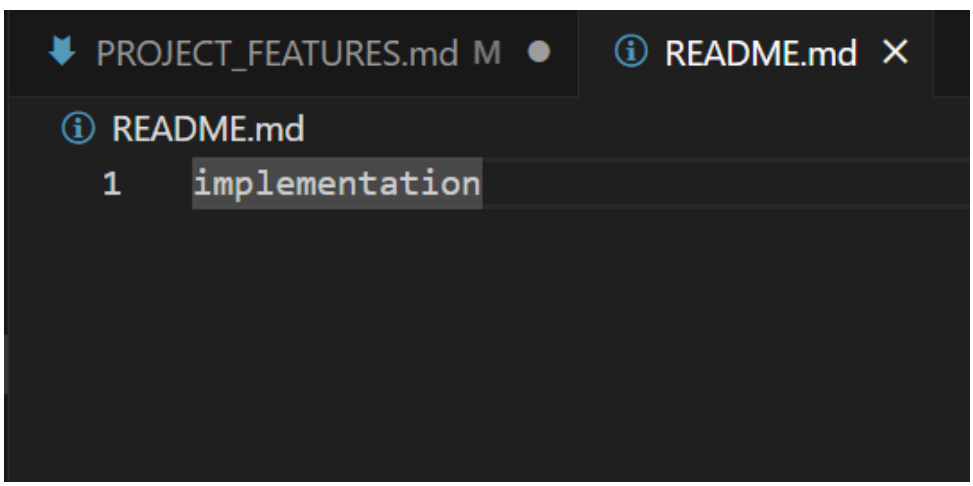


Figure: Problem Solve

6. TESTING

Project Name: MetroSheba			**Test Designed by: Repha Tasneya**	
Test Case ID: TC_01			Test Designed date: 09/12/2025	
Test Priority (Low, Medium, High): High			Test Executed by: Repha Tasneya	
Module Name: Registration			Test Execution date: 09/12/2025	
Test Title: Register with valid details				
Description: Verify registration with all valid details.				
Precondition: User is on registration page				
Dependences: Email not used before				
Test Steps	**Test Data**	**Expected Results**	**Actual Results**	**Status (Pass/Fail)**
1. Open registration page 2. Enter valid details 3. Click "Get Code" 4. Click "Confirm"	Name: Repha Tasneya Email: Repha@.com, Mobile: 01712345678 NID:1234312322, Gender: Female, Password: 1234, Re-Password: 1234 Verification Code: 9876	User is registered successfully, redirected to Dashboard, Mobile Number confirmation sent.	As expected	Pass

Project Name: MetroSheba			**Test Designed by: Repha Tasneya**	
Test Case ID: TC_02			Test Designed date: 09/12/2025	
Test Priority (Low, Medium, High): High			Test Executed by: Repha Tasneya	
Module Name: Registration			Test Execution date: 09/12/2025	
Test Title: Register with invalid email				
Description: Verify registration fails with invalid email format.				
Precondition: User is on registration page				
Dependence: Email not used before				
Test Steps	**Test Data**	**Expected Results**	**Actual Results**	**Status (Pass/Fail)**
1. Open registration page 2. Enter invalid email 3. Enter other details 4. Click Confirm	Email: invalid Email	System shows errors 'Invalid email format.'	As expected	Pass

Project Name: MetroSheba			**Test Designed by: Repha Tasneya**	
Test Case ID: TC_03			Test Designed date: 09/12/2025	
Test Priority (Low, Medium, High): High			Test Executed by: Repha Tasneya	
Module Name: Registration			Test Execution date: 09/12/2025	
Test Title: Register with invalid email				
Description: Verify registration fails when the verification code do not match.				
Precondition: User is on registration page				
Dependence: Email not used before				
Test Steps	**Test Data**	**Expected Results**	**Actual Results**	**Status (Pass/Fail)**
1. Open registration page 2. Enter valid details but mismatched Verification Code 3. Click Confirm	Verification Code: 123456	System shows errors 'Verification code do not match.'	As expected	Pass

Project Name: MetroSheba			**Test Designed by: Repha Tasneya**	
Test Case ID: TC_F04			Test Designed date: 09/12/2025	
Test Priority (Low, Medium, High): High			Test Executed by: Repha Tasneya	
Module Name: Login			Test Execution date: 09/12/2025	
Test Title: Login with unregistered email				
Description: Verify login fails when user tries to log in with an unregistered email.				
Precondition: User not registered				
Dependence: Database has no record of email				
Test Steps	**Test Data**	**Expected Results**	**Actual Results**	**Status (Pass/Fail)**
1. Open login page 2. Enter unregistered email and password 3. Click Login	Email: tasneya@.com, Password: pass123	System shows error 'Email or password does not match'.	The system accepted the email or password	Fail

Project Name: MetroSheba			**Test Designed by: Mukaddas Mahdi Ullas**	
Test Case ID: TC_05			Test Designed date: 09/12/2025	
Test Priority (Low, Medium, High): High			Test Executed by: Mukaddas Mahdi Ullas	
Module Name: Login			Test Execution date: 09/12/2025	
Test Title: Login with valid credentials				
Description: Verify login succeeds with valid credentials.				
Precondition: User is already registered				
Dependence: Database has no record of email				
Test Steps	**Test Data**	**Expected Results**	**Actual Results**	**Status (Pass/Fail)**
1. Open login page 2. Enter valid email & password 3. Click 'Login'	Email: ullas@gmail.com, Password: 1234	User redirected to Dashboard.	As expected	Pass

Project Name: MetroSheba			**Test Designed by: Mukaddas Mahdi Ullas**	
Test Case ID: TC_06			Test Designed date: 09/12/2025	
Test Priority (Low, Medium, High): Medium			Test Executed by: Mukaddas Mahdi Ullas	
Module Name: Forgot Password			Test Execution date: 09/12/2025	
Test Title: Forgot Password with valid email				
Description: Verify login succeeds with valid credentials.				
Precondition: User has a registered account				
Dependence: Database email exists				
Test Steps	**Test Data**	**Expected Results**	**Actual Results**	**Status (Pass/Fail)**
1. Open forgot password page 2. Enter valid email 3. Click 'Send Code' 4. Enter New Password 5. Enter Same Password 6. Click 'Submit'	Email: ullas@gmail.com Verification Code: 1233 New Password: 2345 Confirm Password: 2345	Verification code sent to email.	As expected	Pass

Project Name: MetroSheba			**Test Designed by: Mukaddas Mahdi Ullas**	
Test Case ID: TC_07			Test Designed date: 09/12/2025	
Test Priority (Low, Medium, High): High			Test Executed by: Mukaddas Mahdi Ullas	
Module Name: Passenger Dashboard			Test Execution date: 09/12/2025	
Test Title: Ticket Booking route				
Description: Verify ticket booking with valid stations, quantity, and payment option.				
Precondition: User logged in				
Dependence: Stations and fares configured				
Test Steps	**Test Data**	**Expected Results**	**Actual Results**	**Status (Pass/Fail)**
1. Open Passenger dashboard page 2. Select from-station, to-station, ticket quantity 3. Click 'Confirm'	From: Uttara, To: Motijheel, Tickets: 2,	Booking successful, confirmation message displayed.	As expected,	Pass

Project Name: MetroSheba	**Test Designed by: Mukaddas Mahdi Ullas**
Test Case ID: TC_F08	Test Designed date: 09/12/2025
Test Priority (Low, Medium, High): High	Test Executed by: Mukaddas Mahdi Ullas
Module Name: Passenger Dashboard	Test Execution date: 09/12/2025
Test Title: Book ticket with zero quantity	
Description: Verify system does not allow booking with ticket quantity set to zero	
Precondition: N/A	
Dependence: Stations and fares configured	

Test Steps	**Test Data**	**Expected Results**	**Actual Results**	**Status (Pass/Fail)**
1. Open Passenger dashboard page 2. Select from-station, to-station, ticket quantity 3. Set ticket quantity = 0 4. Click ‘Confirm’	From: Uttara, To: Motijheel, Tickets: 0,	Error 'Please select a valid ticket quantity'.	The system accepted without ticket quantity	Fail

Project Name: MetroSheba			**Test Designed by: MD. Mostafa Hamid Yeasib**	
Test Case ID: TC_09			Test Designed date: 09/12/2025	
Test Priority (Low, Medium, High): High			Test Executed by: MD. Mostafa Hamid Yeasib	
Module Name: Passenger Payment			Test Execution date: 09/12/2025	
Test Title: Payment with bKash option				
Description: Verify payment succeeds using bKash.				
Precondition: User has active booking				
Dependence: bKash gateway available				
Test Steps	**Test Data**	**Expected Results**	**Actual Results**	**Status (Pass/Fail)**
1. Select bKash option 2. Enter valid bKash details 3. Click 'Payment'	Payment method: bKash, Account: 017XXXXXXXX	Payment processed successfully.	As expected	Pass

Project Name: MetroSheba	**Test Designed by: MD. Mostafa Hamid Yeasib**
Test Case ID: TC_10	Test Designed date: 09/12/2025
Test Priority (Low, Medium, High): High	Test Executed by: MD. Mostafa Hamid Yeasib
Module Name: Admin Dashboard	Test Execution date: 09/12/2025
Test Title: Verify auto-calculation of revenue and tickets.	
Description: Ensure revenue and ticket count are updated correctly.	
Precondition: Admin is logged in	
Dependence: Admin login successful	

Test Steps	**Test Data**	**Expected Results**	**Actual Results**	**Status (Pass/Fail)**
1. Go to Admin Dashboard 2. Check total tickets sold 3. Check total revenue	Sample booking records Sample booking fares	Matches total bookings Matches total payments collected	As expected	Pass

Project Name: MetroSheba			**Test Designed by: MD. Mostafa Hamid Yeasib**	
Test Case ID: TC_11			Test Designed date: 09/12/2025	
Test Priority (Low, Medium, High): High			Test Executed by: MD. Mostafa Hamid Yeasib	
Module Name: Change Password			Test Execution date: 09/12/2025	
Test Title: Change Password with mismatched new passwords				
Description: Verify system rejects mismatched new passwords.				
Precondition: User logged in				
Dependence: Database has valid current password				
Test Steps	**Test Data**	**Expected Results**	**Actual Results**	**Status (Pass/Fail)**
1. Go to Change password 2. Enter current password, mismatched new &Confirm Password 3. Click 'Save'	Current: 1234, New: NewPass123, Confirm: WrongPass	Error 'Passwords do not match'.	As expected	Pass

Project Name: MetroSheba			**Test Designed by: MD. Mostafa Hamid Yeasib**	
Test Case ID: TC_F12			Test Designed date: 09/12/2025	
Test Priority (Low, Medium, High): High			Test Executed by: MD. Mostafa Hamid Yeasib	
Module Name: Change Password			Test Execution date: 09/12/2025	
Test Title: Change Password with incorrect current password				
Description: Verify password change fails if the current password is incorrect				
Precondition: User logged in				
Dependence: Database has valid current password				
Test Steps	**Test Data**	**Expected Results**	**Actual Results**	**Status (Pass/Fail)**
1. Go to Change password 2. Enter incorrect current password, new Password 3. Click 'Save'	Current: 1234, New: NewPass123, Confirm: WrongPass	Error 'Current password is incorrect'.	The system accepted with the incorrect password	Fail

Project Name: MetroSheba			**Test Designed by: Tanvir Ahmed**	
Test Case ID: TC_13			Test Designed date: 09/12/2025	
Test Priority (Low, Medium, High): Medium			Test Executed by: Tanvir Ahmed	
Module Name: Edit Profile			Test Execution date: 09/12/2025	
Test Title: Edit Profile with invalid phone number				
Description: Verify profile update fails with invalid phone number.				
Precondition: User logged in				
Dependence: N/A				
Test Steps	**Test Data**	**Expected Results**	**Actual Results**	**Status (Pass/Fail)**
1. Go to edit profile 2. Enter invalid phone number 3. Click ‘Save’	Phone:12345	Error 'Enter valid 11-digit number'.	As expected	Pass

Project Name: MetroSheba			**Test Designed by: Tanvir Ahmed**	
Test Case ID: TC_14			Test Designed date: 09/12/2025	
Test Priority (Low, Medium, High): Low			Test Executed by: Tanvir Ahmed	
Module Name: View Profile			Test Execution date: 09/12/2025	
Test Title: View Profile after login				
Description: Verify profile details are displayed correctly.				
Precondition: User logged in				
Dependence: Profile info stored in Database				
Test Steps	**Test Data**	**Expected Results**	**Actual Results**	**Status (Pass/Fail)**
1. Go to profile page	N/A	Profile details (Name, Email, Phone) displayed.	As expected	Pass

Project Name: MetroSheba			**Test Designed by: Tanvir Ahmed**	
Test Case ID: TC_15			Test Designed date: 09/12/2025	
Test Priority (Low, Medium, High): Low			Test Executed by: Tanvir Ahmed	
Module Name: Change Photo			Test Execution date: 09/12/2025	
Test Title: Change Profile Photo with invalid file format				
Description: Verify system rejects invalid photo file format.				
Precondition: User logged in				
Dependence: N/A				
Test Steps	**Test Data**	**Expected Results**	**Actual Results**	**Status (Pass/Fail)**
i. Go to change profile photo ii. Upload invalid file (e.g. .pdf) iii. Click 'Confirm'	File: photo.pdf	Error 'Invalid file type. Only .jpg/.png allowed'.	The system accepted with the incorrect picture format	Fail

Project Name: MetroSheba			**Test Designed by: Tanvir Ahmed**	
Test Case ID: TC_F16			Test Designed date: 09/12/2025	
Test Priority (Low, Medium, High): Low			Test Executed by: Tanvir Ahmed	
Module Name: Profile Photo			Test Execution date: 09/12/2025	
Test Title: Change Profile Photo with invalid file format				
Description: Verify Change Profile Photo with invalid file format				
Precondition: User logged in				
Dependence: N/A				
Test Steps	**Test Data**	**Expected Results**	**Actual Results**	**Status (Pass/Fail)**
1. Go to change profile photo 2. Upload invalid file (e.g. .pdf) 3. Click ‘Confirm’	File: photo.pdf	Error 'Invalid file type. Only .jpg/.png allowed'.		Pass

7. SOFTWARE PRODUCT METRICS

The software maturity index is computed in the following manner:

Where, M T = Release Module = 50

Fa = Release Module =6

Fc= Change Module =15

Fd =Delete Module =6

SMI = [MT - (Fa +Fc +Fd)]/M T

= [50-(6+15+6)/50]

= 0.7

SMI<1

So, the project is stable.

8. CONCLUSION

The MetroSheba project successfully demonstrated how software testing ensures quality and requirement conformance. By designing and executing multiple test cases across modules such as Registration, Login, Passenger Dashboard, Payment, and Admin Dashboard, the system's functionality was validated against its requirements. Testing identified both strengths (successful user registration, booking, payment integration) and weaknesses (login with unregistered email, zero-ticket booking, invalid file uploads), providing valuable insights for improvement. Overall, the testing phase increased the project's reliability, user confidence, and readiness for deployment.

Future Work

1. **Passenger Packages & History Tracking**

    - Add a feature for passengers to view their past travel history (tickets booked, payments made, routes taken).
    - Introduce package/subscription models for frequent travelers.

2. **Smart Recommendations**

    - Provide personalized route suggestions based on travel history.
    - Notify passengers about peak/off-peak hours and alternative routes.

3. **Mobile App Optimization**

    - Improve user interface (UI) for better passenger experience.
    - Add offline booking history so passengers can check records without internet.

4. **Analytics & Reporting (for Admins)**

    - Enhance dashboards with real-time analytics on revenue, ticket demand, and busiest stations.