SmartSociety: Community-Centric Mobile Banking Solution for Financial Services

Submitted By

Md. Nazmus Shakib Khan

Student Id: 221-35-998

Department of Software Engineering

Daffodil International University

Supervised By

Most. Munira Tabassum

Designation: Lecturer

Faculty of Science and Information Technology

Department of Software Engineering

Daffodil International University

This project report has been submitted in fulfilment of the requirements for the degree a **Bachelor of Science in Software Engineering**

Department of Software Engineering

Faculty of Science and Information Technology

Supervisor Approval Form

Fall 2025	B.Sc. In SWE	Campus: DSC

Student Name	Student ID
Md. Nazmus Shakib Khan	221-35-998

Project/Thesis Information	
Project/Thesis Title	**SmartSociety: Community-Centric Mobile Banking Solution for Financial Services**
Type of work	Project

Supervisor information	
Supervisor Name	Most. Munira Tabassum
Supervisor Initial	MTM
Completed Credit till now	133
How many credits in this semester	12
Supervisor Consent	☑ Yes ☐ No

26.11.2025

Supervisor Signature

APPROVAL

This project titled on "**SmartSociety: Community-Centric Mobile Banking Solution for Financial Services**", submitted by **Md. Nazmus Shakib Khan (ID: 221-35-998)** to the Department of Software Engineering, Daffodil International University has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of Bachelor of Science in Software Engineering and approval as to its style and contents.

BOARD OF EXAMINERS

Chairman

Dr. S. M. Hasan Mahmud
Associate Professor
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

Internal Examiner 1

A.H.M Shahariar Parvez
Associate Professor
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

Internal Examiner 2

Tapushe Rabaya Toma
Assistant Professor
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

Internal Examiner 3

Khalid Been md. Badruzzaman Biplob
Lecturer (Senior Scale)
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

External Examiner

Dr. Md Sazzadur Rahman
Professor
Institute of Information technology
Jahangirnagar University, Bangladesh

DAFFODIL INTERNATIONAL UNIVERSITY

DECLARATION OF THESIS AND COPYRIGHT

Author's Full Name	:	Md. Nazmus Shakib Khan
Date of Birth	:	01-02--2002
Title	:	SmartSociety: Community-Centric Mobile Banking Solution for Financial Services
Academic Session	:	Spring 2022(221)

I declare that this thesis is classified as:

□	CONFIDENTIAL	(Contains confidential information under the Official Secret Act 1997)*
□	RESTRICTED	(Contains restricted information as specified by the organization where research was done)*
☑	OPEN ACCESS	I agree that my project to be published as online open access (Full Text)

I acknowledge that Daffodil International University reserves the following rights:

1. The Project is the Property of Daffodil International University.
2. The Library of Daffodil International University has the right to make copies of the Project for the purpose of research only.
3. The Library of Daffodil International University has the right to make copies of the Project for academic exchange.

Certified by:

Nazmus Shakib
_____
(Student's Signature)

26.11.2025
_____
(Supervisor's Signature)

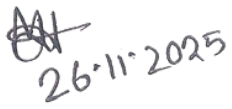
_____
Student ID : 221-35-998
Date: 26-11-2025

_____
Name of Supervisor :Most. Munira Tabassum
Date: 26-11-2025

NOTE: * If the Project is CONFIDENTIAL or RESTRICTED, please attach a thesis declaration letter.

SUPERVISOR'S DECLARATION

I hereby declare that I have checked this project and in my opinion, this project is adequate in terms of scope and quality for the award of the degree of Bachelor of Science.



_____

(Supervisor's Signature)

Full Name : Most. Munira Tabassum

Position : Lecturer

Date : 26-11-2025

STUDENT'S DECLARATION

I hereby declare that the work in this project is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at Daffodil International University or any other institution.

Nazmus Shakib

(Student's Signature)

Full Name : Md. Nazmus Shakib Khan
ID Number : 221-35-998
Date : 26-11-2025

SmartSociety: Community-Centric Mobile Banking Solution for Financial Services

Submitted By

Md. Nazmus Shakib Khan

Student Id: 221-35-998

Department of Software Engineering

Daffodil International University

This project report has been submitted in fulfilment of the requirements for the degree a **Bachelor of Science in Software Engineering**

Department of Software Engineering

Daffodil International University

NOVEMBER 2025

ACKNOWLEDGEMENT

In this walk of gratitude, my sincerest thanks go to Almighty Allah, whose endless compassion and strength enabled me to undertake this task with determination and resolve. Every step forward remained an act of Allah's grace.

From the bottom of my heart, I want to thank my supervisor, Most. Munira Tabassum, who teaches in the Department of Software Engineering at Daffodil International University. I would not have been able to do that without her continuous support, good advice, and encouragement. Her support and advice, not only assisted in creating the tone of this work, but it even took me to perform things I was not certain that I would be able to perform.

I would like to mention that I want to thank all the people that assisted us in our project, by sharing their knowledge, time, and support at each and every stage of the project. I would like to give my thanks to all people who assisted me to complete this research.

My other gratitude goes to all the instructors in Software engineering department, Daffodil international university. I always had you as a source of support, encouragement, and belief in my abilities in school.

This achievement demonstrates to me that I have made a long way and how much assistance, support, and wisdom I have received with the help of numerous individuals. Thanks.

DEDICATION

I swear that this project was done with the help of Most. Munira Tabassum, a teacher in the Daffodil International University's Department of Software Engineering. I really appreciate all of her help, support, and encouragement while this work was being done.

I also say that this project is completely original and that I haven't submitted any parts or pieces of it to any program or institution for academic credit or anyone else's benefit.I would like to thank my supervisor, Most. Munira Tabassum, who is a lecturer in the Department of Software Engineering at Daffodil International University, from the bottom of my heart. This achievement would not have been possible without her constant encouragement, wise advice, and support. Not only did her advice and support set the tone for this work, but they also pushed me to go beyond what I was used to.

We thank everyone who helped us out by giving us their time, knowledge, and support throughout the whole project. I want to thank everyone who helped me finish this project.

I also want to thank all of the teachers in the Software Engineering Department at Daffodil International University. Your constant support, encouragement, and faith in my abilities have kept me going through my academic journey.

This achievement shows how far I've come and how much I've learnt from and been helped by a wide range of people. Thanks.

ABSTRACT

This project is a simple E-Transaction system that makes it easier for everyone to pay for things online. People can buy things online, send money, pay stores, and use agents to handle cash transactions. Agents are in charge of cash services, which let businesses accept payments in person and online.

Get your commissions. Administrators are in charge of the users, the transactions, and how the system works. The platform does a good job of handling cash-out costs, recording transactions right away, and keeping balances up to date in real time.
This system's goal is to make buying and paying for things online easy, safe, and up-to-date by having a simple design and secure operation. This project is an E-Transaction system that is simple to use and makes it easier for everyone to pay online. People can buy things online, send money, pay stores, and use agents to get cash or send cash. Agents can handle cash transactions, but stores can also take payments in person or online.

Get paid commissions. Administrators are in charge of the system's setup, users, and transactions. The program takes care of cash-out fees automatically, records each transaction right away, and updates balances in real time.
The goal of this system is to make online shopping and paying easy, reliable, and up-to-date. Its design is simple, and it's safe to use.

Table ofContents

APPROVAL	**ii-iii**
DECLARATION	**iv-vi**
TITLE	**vii**
ACKNOWLEDGEMENT	**viii**
DEDICATION	**ix**
ABSTRACT	**x**
Chapter 1	**1-19**
Introduction	1
1.1 About the Project	1
1.2 Project Objectives	1
1.3 Project Scope Definition	2
1.4 Market Research	3
1.4.1 Market Analysis	3
1.4.2 Market Feasibility	4
1.5 System Feasibility Analysis	5
1.5.1 Feasibility Study Overview	5
1.5.2 Operational Feasibility	6
1.5.3 Technical Feasibility	7
1.5.4 Development Environment & Technology Stack	7
1.5.5 Financial Feasibility Analysis	7
A. Development Cost (One-Time)	8
B. Operational Cost (Annual)	8
C. Revenue Model	
1.6 Target Users	9
1.6.1 Target User Roles	9
1.6.2 USER PROFILES	10
Customers	10
Agents	11
Merchants	12
1.6.3 Elicitation Technique	13
1.7 System Requirements	13
1.8 Project Schedule	14
1.8.1 Project Schedule: May 2025 – November 2025	15
1.8.2 Backend Setup & Database Integration (June 2025)	16
1.8.3 Web Development – Core Features (July 2025)	17
1.8.4 Web Development – Advanced Features (August 2025)	18
1.8.5 Web Development (September 2025)	18
1.8.6 Testing & Debugging (October 2025)	18
1.8.7 Documentation & Final Submission (November 2025)	19
1.9 SmartSoceity Grant Chart	19

Chapter 2 **20-54**
Design and Implementation
2.1 FunctionalRequirements 20-23
2.2 Non-Functional Requirements 24-25
2.3 UML DIAGRAM 26
2.4 Use Case Diagram 28
2.5 CASE DESCRIPTIONS 29-439
2.6 ACTIVITY DIAGRAMS 48-53
2.7 SEQUENCE DIAGRAMS 53-54
2.8 ERDIAGRAM 54

Chapter 3 **55-61**
Software Testing
3.1 Testing Features 55
3.2 Testing Strategies 55
3.3 System Testing 76-61
3.3.1 System Test Specification (STS)
1. Test Objectives
2. Scope
3. Test Environment
4. Test Data
4.1 Sample User Accounts
5. Pass/ Fail Criteria
6. Detailed Test Cases

Chapter 4 **62-67**
Deployment and Maintenance
SWE EventBoard: Deployment and Maintenance Plan 62
1. Deployment Strategy 62
1.1 Web Application 62
2. Maintenance Plan 63-65
2.1 Routine Monitoring
2.2 Scheduled Maintenance
2.3 Incident Management
2.4 User Feedback Loop
2.5 CI/CD Automation
3. Versioning & Release Policy 66
3. Deployment Flow Diagram 66
4. Maintenance Workflow Diagram 67

Chapter 5 **68-73**
User Manual 68-73

Chapter 6 **74-75**
Project Summery
6.1 Project Overview
6.2 Purpose
6.3 Key Features
6.4 Technology Stack
6.5 ExpectedImpact
APPENDIX A: Source Code **76**
Reference **77**
Accounts Clearance
Originality Report

Chapter 1

Introduction

1.1 About the Project

SmartSociety System is an e-wallet platform used by individuals to send and receive funds through an easy and secure process. The platform is similar to other mobile payment systems, for example, bKash or Nagad, through which customers can control all their finances in one place.

The E-Transaction System consists of various types of users, including Admin, Customer, Merchant, and Agent. Each type of user is assigned a unique role. The Customer is allowed to send funds, make payments, and use agents for cash-in and cash-out processes, earning a small commission in the process. The Merchant is assigned the task of receiving payments and withdrawing funds, while agents assist customers with cash-in and cash-out. When registering, they are immediately granted a wallet and receive an account number. Users are then able to check how much is in their wallet, look at their history, and send various types of transfers. Each transaction is recorded in order to ensure all is secure and clear. The site also offers secure login features so as to safeguard any information.

The purpose of this program is to provide an understanding of how online money systems actually work. Some crucial concepts, such as the types of user roles, safe access, and the commission-sharing process, are also explored. In general, this represents an excellent example of how financial applications can benefit consumers as regards managing their finances.

1.2 Project Objectives

The most important things it does are:

- Create a digital wallet system that makes it simple and safe to keep track of your cash.
- You should be able to "Send Money," "Receive Money," "Cash In," and "Cash Out."

- Each user has their own account with a 6-digit number that is only theirs. To make it easier to keep track, show the wallet's balance and the full history of all transactions.
- Allow customers to talk to agents and merchants about money matters.
- Secure login procedures keep data safe by using JWT authentication.
- Create a simple mobile payment systems like Bkash and Nagad so that people can learn and practice.

1.3 Project Scope Definition

The E-Transaction System is an easy digital wallet that facilitates the ease of making people to make safe and smooth financial transactions. It supports various types of users with Admin, Customer, Merchant and Agent being the key ones, and each user has their tasks to complete to ensure that the system is not only running correctly but also open.

- Admin :
  - Supervises and oversees user and system activities.
  - Sets commission arrangements and grants agent funds requests.
  - Verifies transactions and has ability to suspend and reinstate accounts.
- Customer
  - Transfers money to other users and buys stuff.
  - Uses agents to handle cash-in and cash-out transactions.
  - Views wallet balance and transaction history
- Merchant
  - Accepts payments from customers
  - Performs cash-out through agents
  - Tracks revenue and transaction records

- Agent
  - Provides cash-in and cash-out services
  - Requests funds from Admin
  - Earns commission from cash-out transactions
  - Manages personal wallet and transaction logs

1.4Market Research

1.4.1 Market Analysis

The SmartSociety System is developed with the requirements of society in mind, and especially those communities in which access to conventional banking is difficult. In Bangladesh, mobile wallet transaction systems like bKash and Nagad have already shown how communities can be empowered by money transferring, payment, and saving systems being within reach through digital wallets. But some societies are still being faced with issues related to lack of financial literacy, infrastructure, and cash-based transaction systems. The proposed project will fill this void by creating an easy-to-understand, secure, and easy-to-use digital wallet system for communities. The system will enable customers to handle their financial affairs, communicate with agents and merchants, and carry out essential financial activities without any need to possess bank accounts. The proposed E-Transaction System, whether used in farm co-ops, village markets, and schools, holds the potential to lift and develop societies and help countries like Bangladesh move forward into becoming a completely connected and cashless society.

Comparing Current Systems

Feature	Manual Ledger & Cash	Mobile Wallets (bKash/Nagad)	SmartSociety Web Platform
Centralized Role-Based Management	No	Limited	Yes
Automated Wallet & Transaction Flow	No	Partial	Yes
Real-Time Notifications	No	Some	Yes
Feedback & Analytics	No	Basic	Advanced
Multi-Language Support	No	Limited	Yes
Admin & Agent Oversight	No	No	Yes

Fig: 1.1

1.4.2 Market Feasibility

Market Need and Relevance

In Bangladesh, it is also witnessed that a large portion of the populace, mainly from villages and semi-urban areas, prefers to conduct business in cash, as they lack easy access to bank facilities. Even though mobile financial service provider companies, including BKash and Nagad, have brought tremendous improvements to the picture for electronic payments, still, there is still scope for applications within smaller communities. The E-Transaction System fills this void by introducing an easy, secure, and role-based electronic wallet system with flexibility tailored according to the financial processes within the communities.

Target Market

The target beneficiaries of this system are members of local societies, which may include village cooperatives, housing communities, educational institutions, and small business organizations. Members of these groups may, at times, need to perform financial transactions within their organizations, for example, collecting fees, yet they may lack access to financial systems. The system, therefore, creates efficiency in the management of finances by introducing the following positions within the system: Admin, Customer, Merchant, and Agent.

Competitive Advantage

In contrast to large-scale MFS platforms, which emphasize national services, SmartSociety is intended for localized implementation. SmartSociety provides:

- Customizable user roles and permissions
- Transparent transaction tracking
- Commission Agents' Operations
- Offline-friendly workflows (using agents)
- Educational value for training and awareness

The simplicity and flexibility of BeFMQ are essential in communities where there is a need for a light, secure, and easy-to-manage financial platform without the complexities of business platforms.

Market Trends and Growth Potential

The trend is moving very fast toward becoming a completely cashless economy, with the use of digital wallets on the rise among all classes of people. The initiatives being carried out by the government to make everybody digitally enabled, as well as the increase in smartphone penetration, are some of the reasons why popular platforms, such as the E-Transaction System, are likely to gain popularity as people become more digitally literate.

1.5 System Feasibility Analysis

1.5.1 Feasibility Study Overview

The SmartSociety project will undergo a feasibility study to confirm its viability and assess its practicability. Operational, technical, and financial feasibility are among the important factors that are examined in order to determine whether the system can succeed.

1.5.2 Operational Feasibility

The E-Transaction System is operationally viable for use within communities. The system's role-based architecture promotes favorable financial operations, such as remitting funds, cash-in and cash-out, and payments. The system is developed with secure and scalable backend infrastructure and is easy to operate without training and can be controlled by local admins and agents. The system's well-defined workflows and easy-to-use interfaces make it viable to run in communities with elementary ICT infrastructure

1.5.3 Technical Feasibility

Section	Details
Technology Availability	Django REST Framework is what the system is created with, and it makes sure development that is safe, flexible, and made up of pieces. It can run on local servers or cloud platforms that don't need a lot of extra hardware.
System Capabilities	You may set up access depending on roles with the software, and it keeps your wallet up to current in real time.

1.5.4 Development Environment & Technology Stack

Category	Technology / Tools	Purpose
Frontend	React	Responsive UI design
Backend	Django REST Framework	API development, business logic.
Authentication	JWT	Secure login and role-based access control
Cloud Functions	Django admin	Triggering automatic actions
Database	PostgreSQL	Storing user data, wallet balances, and transactions
Hosting	Vercel/Netlify/Railway/Localhost	Deploying the backend and APIs

Catagory	Free	Post-Free / Paid
Cost Efficiency	No hosting fees; runs on local server or localhost	Hosting costs apply (e.g., Railway, Vercel); scalable with usage
Development Costs	Minimal setup; ideal for academic or prototype use.	Requires cloud configuration and monitoring; suitable for production
Storage Limits	Local database (e.g., PostgreSQL) with manual backups	Cloud storage with automated backups, higher capacity, and redundancy
Support & Maintenance	Self-managed; community support	Professional support options; monitoring tools and alerts
Expansion Potential	Limited scalability; manual upgrades	Easily scalable with modular architecture and cloud resources

1.5.6 Financial Feasibility Analysis

We were able to show that the SamartSociety System would work by figuring out the costs of development and operation and looking at possible sources of income.

A. Development Cost (One-Time)

Item	Details	Estimated Cost(USD)
Backend Development (Django)	600 hours × $6/hour	$3,600
Frontend Development	300 hours × $6/hour	$1,800
UI/UX Design	Role-based interface for web users	$500
Testing & QA	Manual + automated testing	$400
Documentation & Deployment	Setup, user guides, and hosting configuratio	$300
Total Development Cost = $6,600		

B. Operational Cost (Annual)

Item	Details	Estimated Cost (USD)
Hosting (Cloud or VPS)	Basic server for backend and database	$20/month → $240/year
Domain & SSL Certificate	Secure access and branding	Secure access and branding
Maintenance & Support	Part-time admin or developer	$100/month → $1,200/year
Backup & Recovery	Manual or automated backups	$10/month → $120/year
Total Operational Cost = $1,575/year		

C. Revenue Model

Revenue Stream	Details	Estimated Monthly Revenue (USD)
Agent Commissions	2–5% per cash-out transaction; incentivizes agent activity	$50–$150
Merchant Service Fees	Flat or tiered fees for accepting payments	$100–$150
Premium Features	Optional analytics, messaging, or priority support	$50–$100
Community Subscription	Monthly fee from societies for platform access	$100–$250
Total Estimated Revenue = $300-650/month **Total Estimated Revenue = $3600-7,800/year**		

1.6 Target Users

1.6.1 Target User Roles

User Role	Description	Key Responsibilities / Features
Customers	Users who manage personal wallets and perform transactions.	• Send money to other users • Make payments to merchants • View transaction history
Merchants	Vendors who accept payments and manage earnings.	• Receive payments from customers • Cash out wallet balance • Track sales and transactions
Agents	Community facilitators who handle cash-in/cash-out services.	• Perform cash-in and cash-out for users • Earn commission per transaction • Maintain transaction logs
Admins	System managers who oversee operations and user roles.	• Approve or reject fund requests • Manage user roles and permissions • Monitor system activity and generate reports

1.6.2USER PROFILES

Customers

User Class	Notes on characteristics	Requirement implied
Type of user	Customers	User interface, verification
Age range	19-80	verification
Mandatory	Yes	
Frequency of use	Several times a day	Performance,operation and acceptance,safety.
Computer Experience	Not required	User interface
Education	HSC	
Goals	Cashless Society	Scalability.
Language skills	Bangla,English	
Number of users	Many	Performance,operation and acceptance.
Training	Not required.	User interface
Other system used	No	
Ways of working	Sendmoney,CashIn,CashOut,Payment, View Transaction HIstory	Performance,operation and acceptance,security.

Merchants

User Class	Notes on characteristics	Requirement implied
Type of user	Customer	User interface, verification
Age range	19-50	verification
Mandatory	Yes	
Frequency of use	Several Times a day	Performance,operation and acceptance,safety.
Computer Experience	Not required	User interface
Education	SSC	
Goals	Accepting Online Payment	Performance,resource,security,maintenance, acceptance, scalability.
Language skills	Bangla,English	
Number of users	10	Performance,operation and acceptance.
Training	Not required.	User interface
Other system used	No Add	
Ways of working	Check notification, View Transaction,CashOut	Performance,operation and acceptance,security.

Agents

User Class	Notes on characteristics	Requirement implied
Type of user	Agents	User interface, verification
Age range	25-50	verification
Mandatory	Yes	
Frequency of use	Several times a day	Performance,operation and safety.
Computer Experience	Not required	User interface
Education	HSC	
Goals	Convert online currency in hand cash	Performance,operation and safety.
Language skills	Bangla,English	
Number of users	5	Performance,operation and acceptance.
Training	Required	User interface
Other system used	No	
Ways of working	Cashin ,Cashout,request money	Performance, operation, resource, security.

Admin

User Class	Notes on characteristics	Requirement implied
Type of user	Admin	User interface, verification
Age range	25-50	verification
Mandatory	Yes	
Frequency of use	Not often.	Performance,operation and safety.
Computer Experience	Required	User interface
Education	BSc	
Goals	Monitor Everything	Performance,operation and safety.
Language skills	Bangla, English	
Number of users	2	Performance,operation and acceptance.
Training	Required.	User interface
Other system used		
Ways of working	Monitor, Review, Take necessary steps	Performance, operation, resource, security.

1.6.3 Elicitation Technique

Source	Key Findings
User Interviews	Need for simple, mobile-friendly interface and clear roles
Community Feedback	Demand for secure, transparent transactions
Agent Discussions	Importance of commission tracking and cash-in/out clarity
Admin Observations	Need for role management and system monitoring tools

1.7 System Requirements

Category	Customer & Agent Panel (Web)	Admin Panel (Web Dashboard)
Hardware Requirements	Any device with 2 GB+ RAM and modern browser suppor	Desktop/laptop with 4 GB+ RAM recommended
Internet Connection	Stable internet (3G/4G or Wi-Fi)	Reliable broadband or Wi-Fi connection
Operating System	Windows, macOS, Linux, Android (browser-based access)	Windows, macOS, Linux
Web Browser	Chrome, Firefox, Edge (latest versions)	Chrome, Firefox, Edge (latest versions)
Frontend	React	React
Backend	Django REST Framework	Django REST Framework

1.8 Project Schedule

1.8.1 Project Schedule: May 2025 – November 2025

1. Project Planning & Research (May 2025)

- Tasks:
    - Complete the project's objectives and scope.
    - Examine frameworks Django, DRF, React
    - Explain the requirements, including functional and non-functional.
    - Create the frontend and design the admin panel, the users' interface, and user experience.
    - Configure the development environment by setting up React and an IDE.
- Deliverables:
    - Proposal for the undertaking.
    - Developed user interfaces
    - The development environment is prepared.

1.8.2 Backend Setup & Database Integration (June 2025)

- Tasks:
    - Create a Django project and app
    - Connect PostgreSQL in settings.py
    - Set up user authentication (email/password)
    - Design models: Users, Wallets, Transactions, Roles
    - Apply migrations and create a superuser
    - Set up Django REST Framework for APIs
    - Implement JWT-based access control

- Deliverables:
    - Django project initialized
    - PostgreSQL connected
    - Core models created
    - JWT authentication implemented
    - REST APIs built
    - Role-based access control added
    - API tested and documented
    - Local deployment ready

1.8.3Web Development – Core Features (July 2025)

- Tasks:

  The app for user is:
    - Register User
    - User Role
    - Login and Logout
    - Account Number create
    - View Profile
    - Edit Profile

- Deliverables:
    - Users created in the db and registration, login, and logout APIs are working
    - Fronted can receive data from backend

- Tasks:

  The app for customer is:
    - Send Money
    - Cashout
    - Check transaction history
    - Make Payment

- Deliverables:
    - User can check their balance

- Tasks:

  The app for wallet is:

  - add wallet for every registered user
  - onetoone relation with user db table

- Deliverables:
  - User can check their balance

- Tasks:

  The app for merchant is:

  - Receive payment notification
  - Cashout

- Deliverables:
  - Seamless transaction
  - No cash
  - easy to mantain

- Tasks:

  The app for agent is:

  - Cashin
  - Cashout
  - Check transaction history
  - Request Money

- Deliverables:
  - Convert to cash
  - Earn Commission
  - Cashin - cash to e-currency

1.8.4 Web Development – Advanced Features (August 2025)

- Tasks:

  Add advanced features:

  - Transaction statement
  - Finance Agent Integration
- Deliverables:
  - Get a PDF as a transaction statement
  - Get financial advice

1.8.5 Admin Panel Development (August 2025)

- Tasks:

  Develop the admin panel:

  ○ Dashboard for users, events, and registration tracking.

  ○ Analytics and reporting features.

  ○ Connect the database to the admin panel.

  ○ View the Android phones' admin panel.
- Deliverables:

  ○ A fully working admin panel.

  ○ You can run a database and an admin panel at the same time.

1.8.6 Testing & Debugging (September 2025)

- Tasks:
  - Conduct comprehensive testing of the admin panel and web application.
  - Resolve issues and enhance efficiency.
  - Do security checks.
  - Put together user feedback and make changes.
- Deliverables:
  - Web app with no bugs
  - Report on the tests.

1.8.7 Documentation & Final Submission (Oct 2025)

- Tasks:

  Write project documentation:

  - This serves as both the admin and website user handbook.
  - Technical documentation (code structure, system architecture).

- Deliverables:

  - full project records.
  - Presentation.
  - Final submission of the project.

1.9 SmartSociety Grant Chart

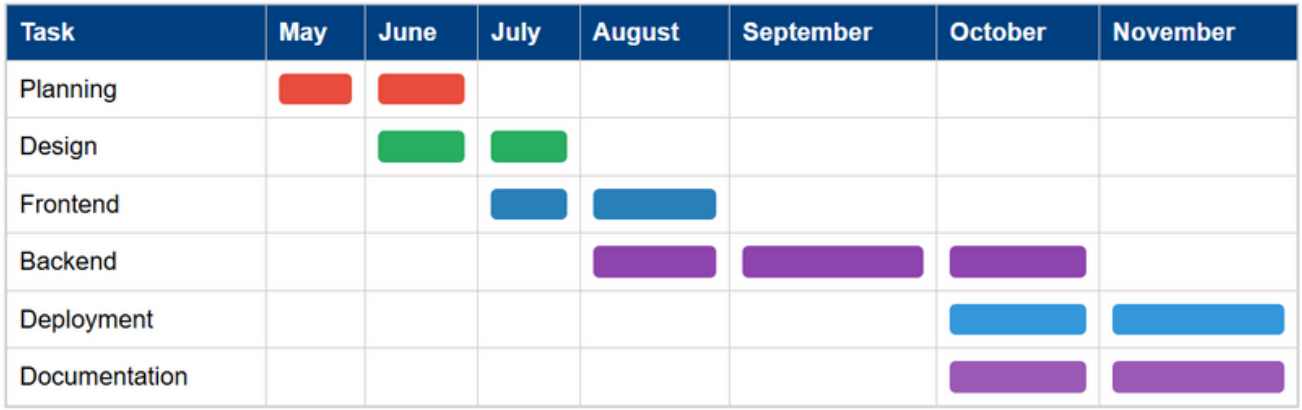


Fig: 1.2

Chapter 2

Design and Implementation

2.1 Functional Requirements

FR01	User Registration
Description	To use this system, each user must create an account using their name, phone number, or university email address. After that, they can access this system.
Stakeholder	Society Members, Merchant, Agent

FR02	Login
Description	Users need to enter their username and password in order to log in after registration. The user cannot utilize it otherwise.
Stakeholder	Society Members, Merchant, Agent

FR03	Send money
Description	Only Customers can send money between them
Stakeholder	Customer

FR04	Cashout
Description	Customer can cashout from agent
Stakeholder	Customer, Agent

FR05	Cashin
Description	Customer can topup from any agent
Stakeholder	Customer, Agent

FR06	Payment
Description	Customers can make payments through this application to any registered merchant shop
Stakeholder	Customer, Merchant

FR07	Check transaction history
Description	All user can check their transaction history and admins can monitor transaction
Stakeholder	Customer, Merchant, Agent, Admin

FR08	View Profile
Description	All user can view their profile
Stakeholder	Customer, Merchant, Agent

FR09	Edit Profile
Description	All user can edit their profile
Stakeholder	Customer, Merchant, Agent

FR10	Change Password
Description	All user can change their password
Stakeholder	Customer, Merchant, Agent

FR11	Request Money
Description	Agent can request money to the admin for cashin transaction
Stakeholder	Agent, Admin

FR12	Merchant Cashout
Description	Merchant users cash out their received money from the agent and admin by cutting a percentage
Stakeholder	Merchant, Admin, Agent

FR13	Customer Support
Description	Users get customer support from the admin to solve any technical problem
Stakeholder	Admin,User

FR14	Add, Remove, Freeze account
Description	Admin can add, remove, also freeze any account
Stakeholder	Admin

FR15	Notifications
Description	All users get notifications for sending money, receiving money, cashing in, cash out
Stakeholder	Customer, Merchant, Agent

FR16	Get Transactions report
Description	All users get a transaction statement PDF on request
Stakeholder	Customer, Merchant, Agent

FR17	Online Buy sell
Description	Users can buy and sell products here
Stakeholder	Customer, Merchant, Agent

FR18	Online Order
Description	Users can order products online from the registered merchant shop
Stakeholder	Customer, Merchant, Agent

FR19	Society Update
Description	Users can get any updated news from the society committee
Stakeholder	Customer, Merchant, Agent

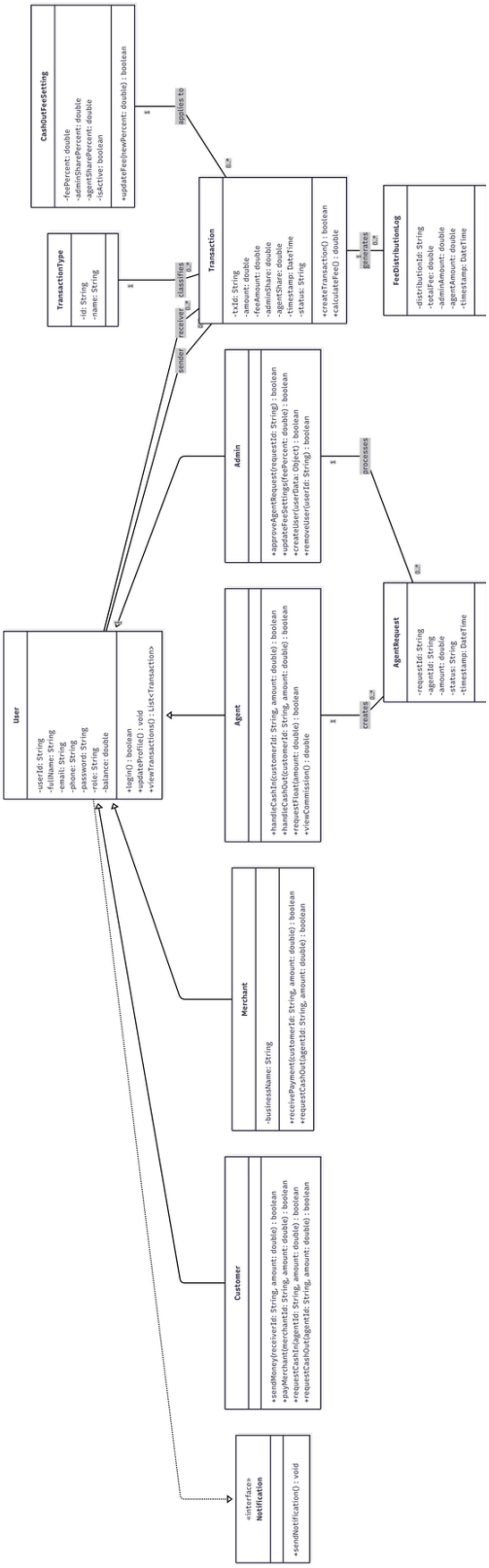
FR20	Log Out
Description	Users can logout from the application
Stakeholder	Customer, Merchant, Agent, Admin

2.2 Non-Functional Requirements

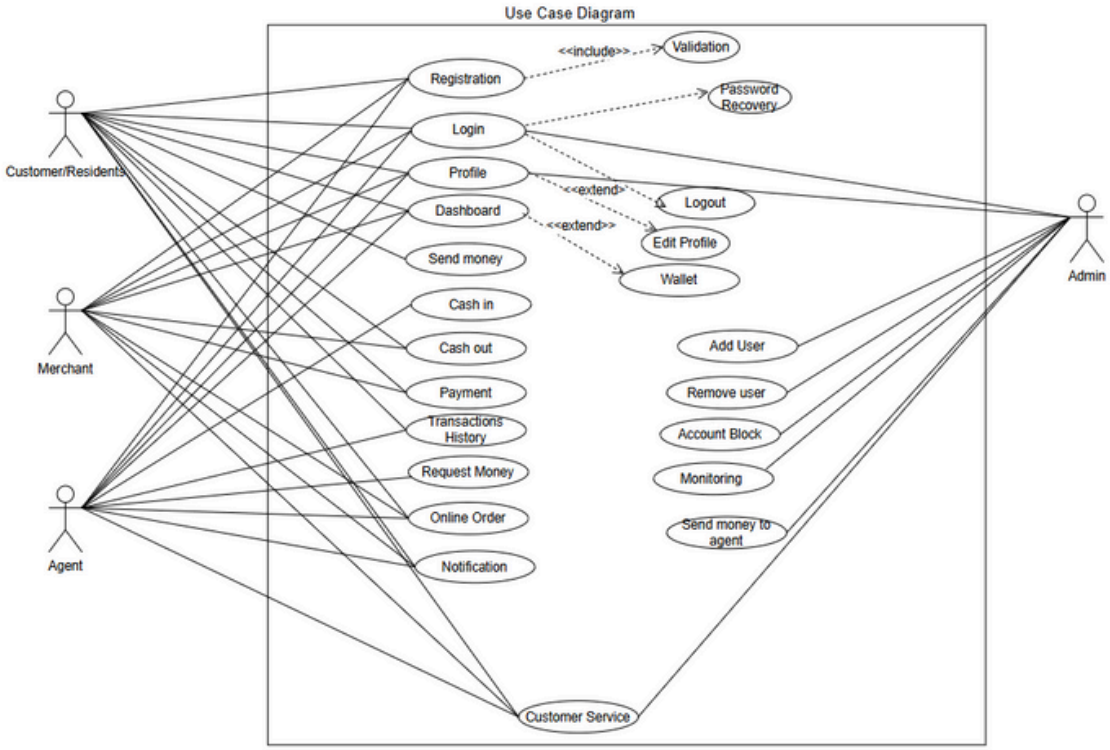
ID	Requirement	Description	Reason/Justification
NFR-1	Performance & Efficienc	The system should respond to user actions (e.g., login, transaction) within 2 seconds.	Ensures smooth user experience and trust in financial operations
NFR-2	Scalability	Must support 1000+ concurrent users without degradation in performance.	To accommodate growing user base and community expansion.
NFR-3	Security & Access Control	Use role-based access control (RBAC) and JWT for secure authentication and data protection.	Protects sensitive financial data and prevents unauthorized access.
NFR-4	Availability & Reliability	Maintain 99.9% uptime with auto-recovery and failover mechanisms.	Critical for uninterrupted financial services and user confidence
NFR-5	Usability & Localization	Interface should be intuitive and support both Bengali and English languages.	Enhances accessibility for diverse community users in Bangladesh.
NFR-6	Auditability	All transactions and user actions must be logged and traceable.	Required for financial transparency and regulatory compliance.

NFR-7	Maintainability	Codebase should follow modular design and be easy to update or extend.	Facilitates long-term sustainability and feature upgrades.
NFR-8	Compliance	Align with Bangladesh Bank's digital finance and data protection guidelines.	Ensures legal and regulatory adherence for financial operations.

2.3 UML DIAGRAM



2.4 Use Case Diagram



2.5 CASE DESCRIPTIONS

<table>
<tr><td>Use Case</td><td>Registration</td></tr>
<tr><td>Goal</td><td>Complete account registration.</td></tr>
<tr><td>Precondition</td><td>Customer, Agent, and Merchant must register .</td></tr>
<tr><td>Success End Condition</td><td>. The message "You are registered successfully" appears.</td></tr>
<tr><td>Failed End Condition</td><td>“Registration Failed” notification.</td></tr>
<tr><td>Primary Actors

Secondary Actors</td><td>Customer,
Agent, and Merchant</td></tr>
<tr><td>Trigger</td><td>Actors will do registration to run the system.</td></tr>
<tr><td>Description/Main success scenario</td><td><table><tr><td>1.</td><td>Performers will enter the system.</td></tr><tr><td>2.</td><td>They'll go with the choice to register.</td></tr><tr><td>3.</td><td>They will complete the information.</td></tr><tr><td>4.</td><td>Registration done.</td></tr></table></td></tr>
<tr><td>Alternative Flows</td><td><table><tr><td>4.1</td><td>Registration Failed.</td></tr><tr><td></td><td>4.1.a All fields are required to fill.</td></tr></table></td></tr>
<tr><td>Quality Requirements</td><td>Registration should be done within 30 minutes.</td></tr>
</table>

Use Case	Log in
Goal	Complete account login.
Precondition	Customer, Agent, Merchant, Admin can log in only.
Success End Condition	“Successful Login” notification.

<table>
<tr><td>Failed End Condition</td><td>“Something went wrong. Please give it another go.” notification.</td></tr>
<tr><td>Primary Actors

Secondary Actors</td><td>Customer,
Agent, and Merchant</td></tr>
<tr><td>Trigger</td><td>Actors will login to run the system.</td></tr>
<tr><td>Description/Main success scenario</td><td><table>
<tr><td>1.</td><td>Actors will go to the system.</td></tr>
<tr><td>2.</td><td>They will select the login option.</td></tr>
<tr><td>3.</td><td>They will fill up details.</td></tr>
<tr><td>4.</td><td>Log in done.</td></tr>
</table></td></tr>
<tr><td>Alternative Flows</td><td><table>
<tr><td>4.1</td><td>Login failed.</td></tr>
<tr><td></td><td>message “Invalid account number or password”.</td></tr>
</table></td></tr>
<tr><td>Quality Requirements</td><td>Login should be done within 10 minutes.</td></tr>
</table>

Use Case	Send Money
Goal	Transfer Money to others
Precondition	Sender and receiver must be a user and role customer
Success End Condition	“Transaction Successful” notification
Failed End Condition	“Transaction Failed” notification
Primary Actors **Secondary**	Customer

<table>
<tr><td>Actors</td><td></td></tr>
<tr><td>Trigger</td><td>Update wallet</td></tr>
<tr><td>Description/Main success scenario</td><td>
<table>
<tr><td>1.</td><td>Sender give receiver account number</td></tr>
<tr><td>2.</td><td>Sender give amount</td></tr>
<tr><td>3.</td><td>Vaildate user account number</td></tr>
<tr><td>4.</td><td>Check Balance</td></tr>
<tr><td>5.</td><td>Create a transaction</td></tr>
<tr><td>6.</td><td>Update both db</td></tr>
<tr><td>7.</td><td>Successful</td></tr>
</table>
</td></tr>
<tr><td>Alternative Flows</td><td>
<table>
<tr><td>3.1</td><td>If the user is not logged in</td></tr>
<tr><td></td><td>If the receiver account number is wrong</td></tr>
<tr><td></td><td>If the current balance is smaller than the amount</td></tr>
<tr><td>4.1</td><td>if internal slow</td></tr>
<tr><td>5.1</td><td>if system stuck</td></tr>
<tr><td></td><td></td></tr>
<tr><td>6.1</td><td></td></tr>
<tr><td>7.1</td><td></td></tr>
</table>
</td></tr>
</table>

Quality Requirements	The transaction will be successful within 5 min

Use Case	Cashout
Goal	Transfer money to the agent
Precondition	User must be Customer, Agent
Success End Condition	. The message "Successfully Cashout" appears.
Failed End Condition	“Cashout Failed” notification.
Primary Actors **Secondary Actors**	Customer, Agent
Trigger	Actors transfer money to convert it in cash
Description/Main success scenario	1. Transfer money to the agent 2. Agent receives notification. 3. Give money 4. Cash out done
Alternative Flows	4.1 Cashout failed Internet and insufficient balance.
Quality Requirements	Cashout should be done within 10 minutes.

<table>
<tr><td>Use Case</td><td>Payment</td></tr>
<tr><td>Goal</td><td>Make a payment after shopping</td></tr>
<tr><td>Precondition</td><td>User must be Customer, Merchant</td></tr>
<tr><td>Success End Condition</td><td>. The message "Payment Successful" appears.</td></tr>
<tr><td>Failed End Condition</td><td>“Payment Failed” notification.</td></tr>
<tr><td>Primary Actors

Secondary Actors</td><td>Customer,
Merchant</td></tr>
<tr><td>Trigger</td><td>Actors transfer money to the merchant user</td></tr>
<tr><td>Description/Main success scenario</td><td><table>
<tr><td>1.</td><td>enter amount</td></tr>
<tr><td>2.</td><td>Enter merchant account number</td></tr>
<tr><td>3.</td><td>Payment</td></tr>
<tr><td>4.</td><td>Payment Done</td></tr>
</table></td></tr>
<tr><td>Alternative Flows</td><td><table>
<tr><td>4.1</td><td>Payment failed</td></tr>
<tr><td></td><td>Internet, insufficient balance, and wrong account number</td></tr>
</table></td></tr>
<tr><td>Quality Requirements</td><td>Payment should be made within 10 minutes.</td></tr>
</table>

<table>
<tr><td>Use Case</td><td>Cash in</td></tr>
<tr><td>Goal</td><td>Top up to the customer account</td></tr>
<tr><td>Precondition</td><td>User must be Customer, Agent</td></tr>
<tr><td>Success End Condition</td><td>. The message "Cashin Successful" appears.</td></tr>
<tr><td>Failed End Condition</td><td>“Cashin Failed” notification.</td></tr>
<tr><td>Primary Actors

Secondary Actors</td><td>Customer,
Agent</td></tr>
<tr><td>Trigger</td><td>Agent transfers money to the customer</td></tr>
<tr><td>Description/Main success scenario</td><td><table>
<tr><td>1.</td><td>enter amount</td></tr>
<tr><td>2.</td><td>Enter customer account number</td></tr>
<tr><td>3.</td><td>cashin</td></tr>
<tr><td>4.</td><td>Cashin Done</td></tr>
</table></td></tr>
<tr><td>Alternative Flows</td><td><table>
<tr><td>4.1</td><td>Cashin failed</td></tr>
<tr><td></td><td>Internet, insufficient balance, and wrong account number</td></tr>
</table></td></tr>
<tr><td>Quality Requirements</td><td>Cashin should be made within 10 minutes.</td></tr>
</table>

<table>
<tr><td>Use Case</td><td>Request money</td></tr>
<tr><td>Goal</td><td>Add money to the agent account</td></tr>
<tr><td>Precondition</td><td>User must be an agent and admin</td></tr>
<tr><td>Success End Condition</td><td>. The message " Successful" appears.</td></tr>
<tr><td>Failed End Condition</td><td>“ Failed” notification.</td></tr>
<tr><td>Primary Actors

Secondary Actors</td><td>

Agent, Admin</td></tr>
<tr><td>Trigger</td><td>Admin transfers money to agent</td></tr>
<tr><td>Description/Main success scenario</td><td><table>
<tr><td>1.</td><td>enter amount</td></tr>
<tr><td>2.</td><td>Enter agent account number</td></tr>
<tr><td>3.</td><td>Successful</td></tr>
<tr><td>4.</td><td>Done</td></tr>
</table></td></tr>
<tr><td>Alternative Flows</td><td><table>
<tr><td>4.1</td><td>Failed</td></tr>
<tr><td></td><td>Internet, balance limit cross, and wrong account number</td></tr>
</table></td></tr>
<tr><td>Quality Requirements</td><td>Transfer money should be made within 60 minutes.</td></tr>
</table>

<table>
<tr><td>Use Case</td><td>Change Password</td></tr>
<tr><td>Goal</td><td>change password</td></tr>
<tr><td>Precondition</td><td>User must be a Customer, Merchant, or agent</td></tr>
<tr><td>Success End Condition</td><td>. The message "Password changed Successfuly" appears.</td></tr>
<tr><td>Failed End Condition</td><td>“ Failed” notification.</td></tr>
<tr><td>Primary Actors

Secondary Actors</td><td>Customer,
Merchant,
agent</td></tr>
<tr><td>Trigger</td><td>Change Password</td></tr>
<tr><td>Description/Main success scenario</td><td><table>
<tr><td>1.</td><td>Enter old password</td></tr>
<tr><td>2.</td><td>Enter new password</td></tr>
<tr><td>3.</td><td>Change Password</td></tr>
<tr><td>4.</td><td>Done</td></tr>
</table></td></tr>
<tr><td>Alternative Flows</td><td><table>
<tr><td>4.1</td><td>failed</td></tr>
<tr><td></td><td>Wrong password, unauthenticated user</td></tr>
</table></td></tr>
<tr><td>Quality Requirements</td><td>The password should be changed within 1 minute.</td></tr>
</table>

<table>
<tr><td>Use Case</td><td>Transaction History</td></tr>
<tr><td>Goal</td><td>Check transactions</td></tr>
<tr><td>Precondition</td><td>User must be Customer, Merchant, Agent</td></tr>
<tr><td>Success End Condition</td><td>. The message " Successful" appears.</td></tr>
<tr><td>Failed End Condition</td><td>“ Failed” notification.</td></tr>
<tr><td>Primary Actors

Secondary Actors</td><td>Customer,
Merchant,
Agent</td></tr>
<tr><td>Trigger</td><td>Check Transactions</td></tr>
<tr><td>Description/Main success scenario</td><td><table>
<tr><td>1.</td><td>Transaction History</td></tr>
<tr><td>2.</td><td>Retrive data for loged in user</td></tr>
<tr><td>3.</td><td>Check</td></tr>
<tr><td>4.</td><td>Done</td></tr>
</table></td></tr>
<tr><td>Alternative Flows</td><td><table>
<tr><td>4.1</td><td>Payment failed</td></tr>
<tr><td></td><td>Internet, Traffic</td></tr>
</table></td></tr>
<tr><td>Quality Requirements</td><td>Payment should be made within 2 minutes.</td></tr>
</table>

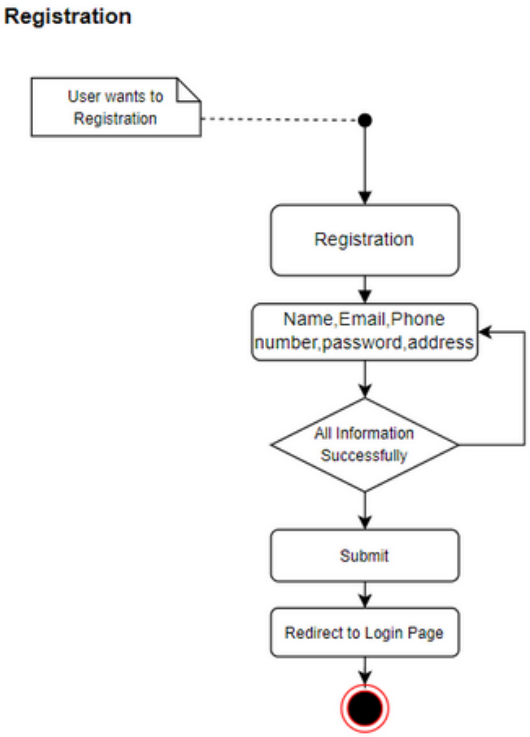
<table>
<tr><td>Use Case</td><td>Online order</td></tr>
<tr><td>Goal</td><td>Make an order online from a registered shop</td></tr>
<tr><td>Precondition</td><td>User must be Customer, Merchant, or Agent</td></tr>
<tr><td>Success End Condition</td><td>.The message "Order Successful" appears.</td></tr>
<tr><td>Failed End Condition</td><td>“Order Failed” notification.</td></tr>
<tr><td>Primary Actors

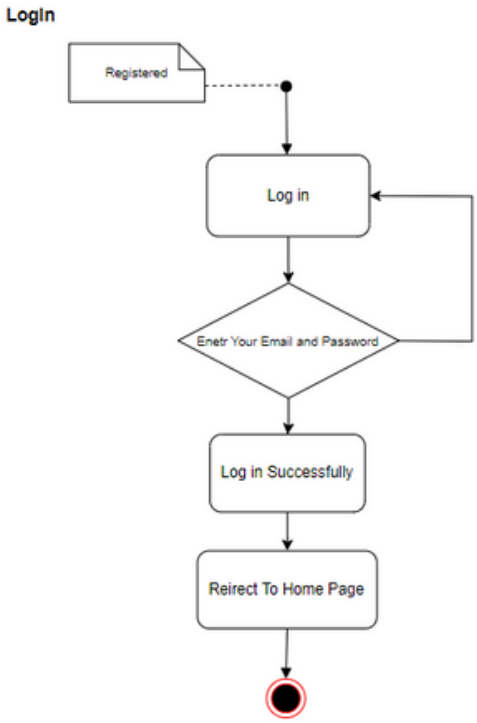
Secondary Actors</td><td>Customer,
Merchant,
Agent</td></tr>
<tr><td>Trigger</td><td>Actors order online from a registered shop</td></tr>
<tr><td>Description/Main success scenario</td><td><table>
<tr><td>1.</td><td>select shop</td></tr>
<tr><td>2.</td><td>Select Product</td></tr>
<tr><td>3.</td><td>Order</td></tr>
<tr><td>4.</td><td>Order Done</td></tr>
</table></td></tr>
<tr><td>Alternative Flows</td><td><table>
<tr><td>4.1</td><td>Order failed</td></tr>
<tr><td></td><td>Internet, unavailability, and traffic</td></tr>
</table></td></tr>
<tr><td>Quality Requirements</td><td>Order should be made within 5 minutes.</td></tr>
</table>

<table>
<tr><td>Use Case</td><td>Logout</td></tr>
<tr><td>Goal</td><td>Log out from the application</td></tr>
<tr><td>Precondition</td><td>User must be Customer, Merchant, Agent, Admin</td></tr>
<tr><td>Success End Condition</td><td>. The message "Logout Successful" appears.</td></tr>
<tr><td>Failed End Condition</td><td>“Logout Failed” notification.</td></tr>
<tr><td>Primary Actors

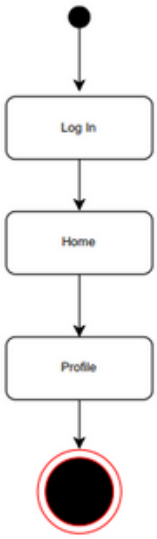
Secondary Actors</td><td>Customer,
Merchant,
Agent</td></tr>
<tr><td>Trigger</td><td>Actors log out from the site</td></tr>
<tr><td>Description/Main success scenario</td><td><table><tr><td>1.</td><td>Logout</td></tr><tr><td>2.</td><td>Logout done</td></tr><tr><td>3.</td><td></td></tr><tr><td>4.</td><td></td></tr></table></td></tr>
<tr><td>Alternative Flows</td><td><table><tr><td>4.1</td><td>Logout failed</td></tr><tr><td></td><td>Internet, refresh token expires, traffic</td></tr></table></td></tr>
<tr><td>Quality Requirements</td><td>Payment should be made within 10 minutes.</td></tr>
</table>

2.6 ACTIVITY DIAGRAMS

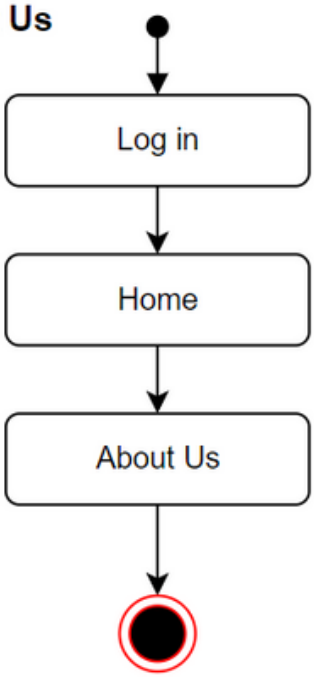




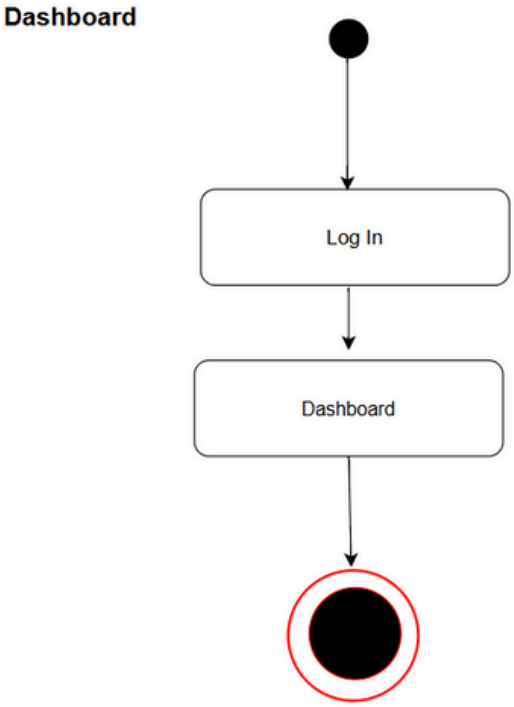
Profile

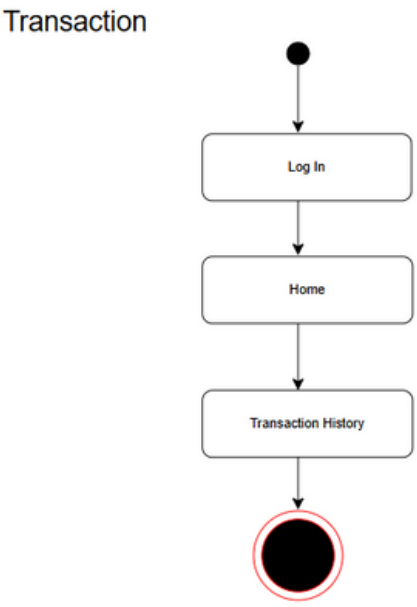


About Us

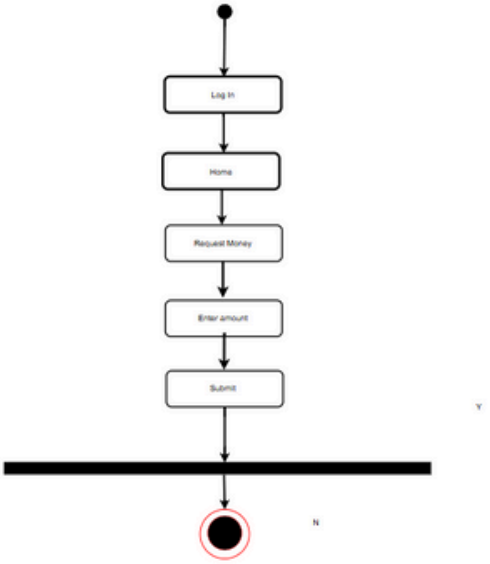


Dashboard

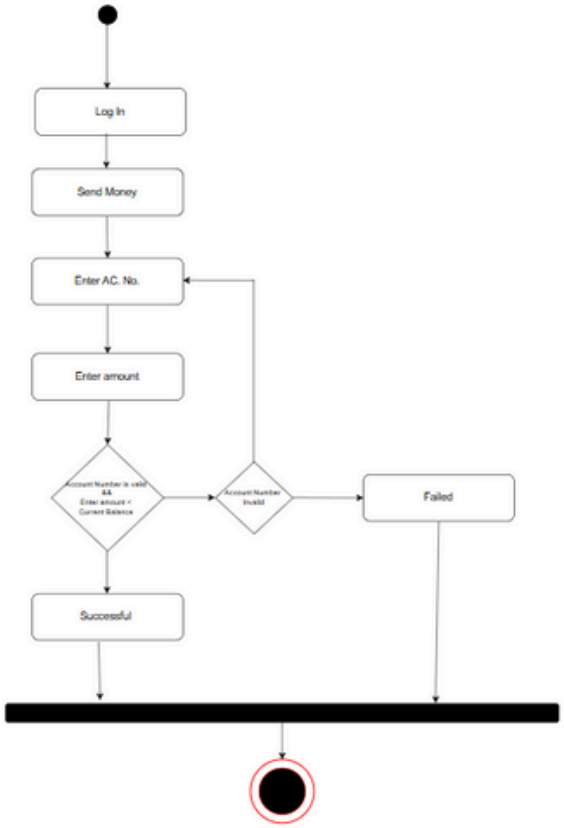

Dashboard
Log In
Dashboard


Transaction
Log In
Home
Transaction History

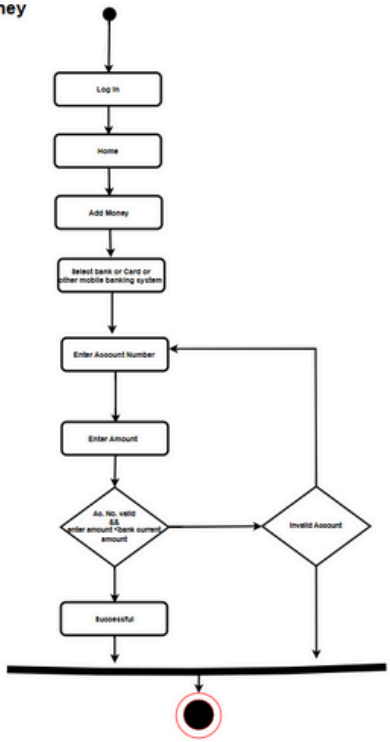
Request Money



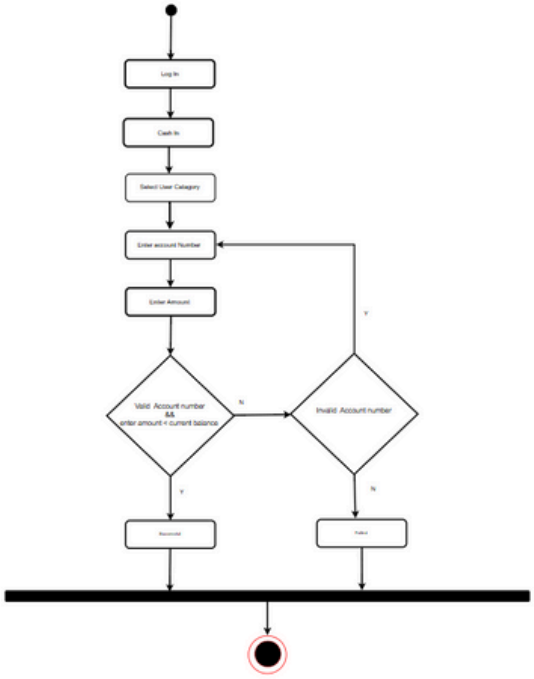
Send Money



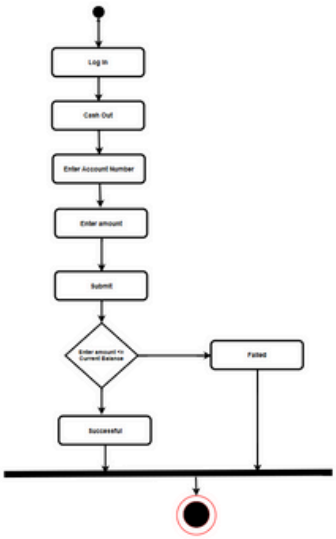
Add Money

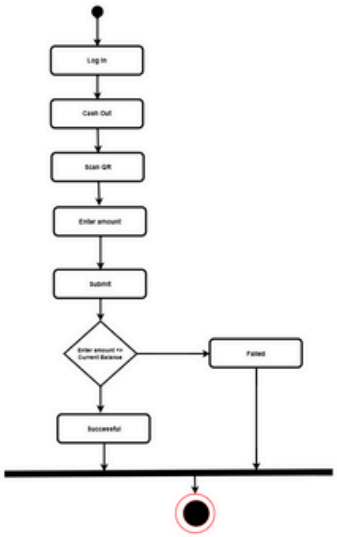


Cash In

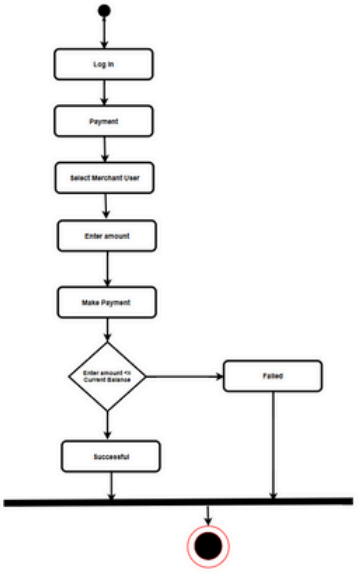


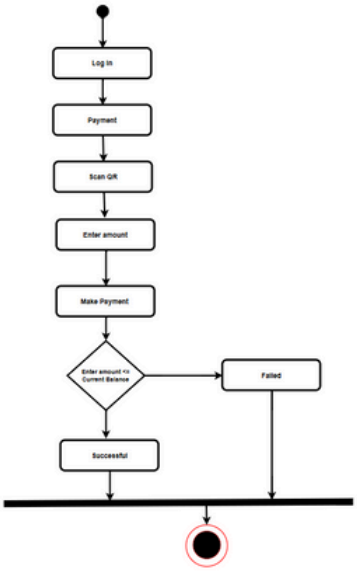
Cash Out

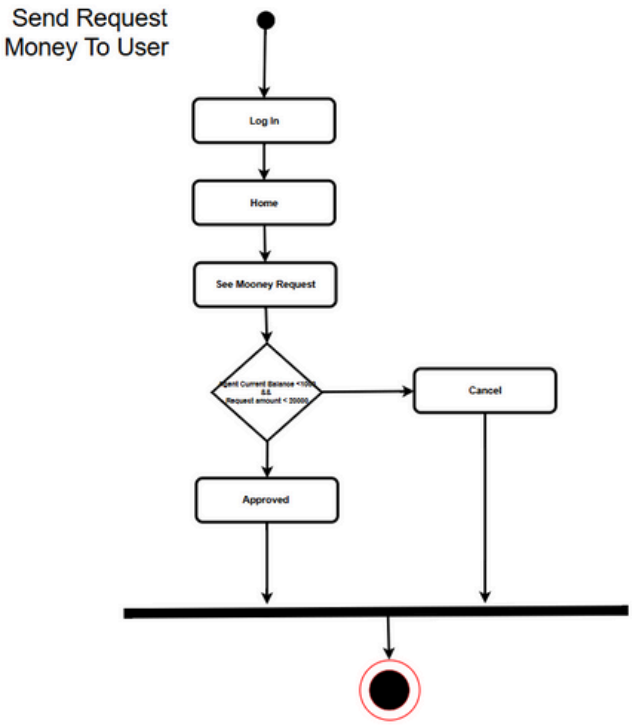


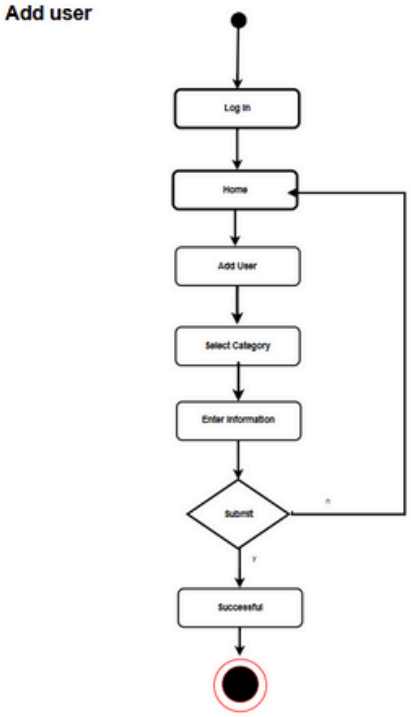


Payment

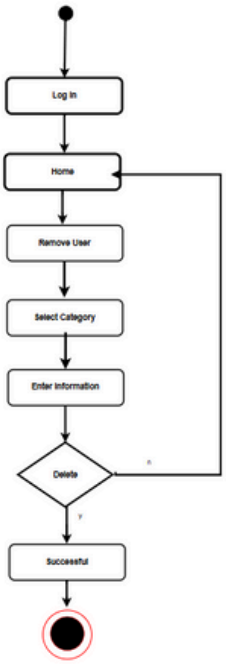




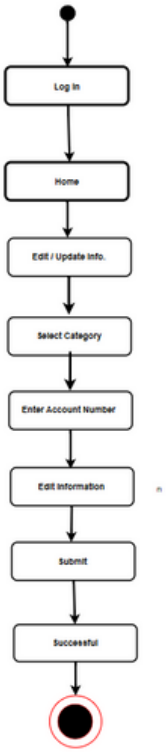

Send Request
Money To User
Log In
Home
See Mooney Request
Cancel
Approved

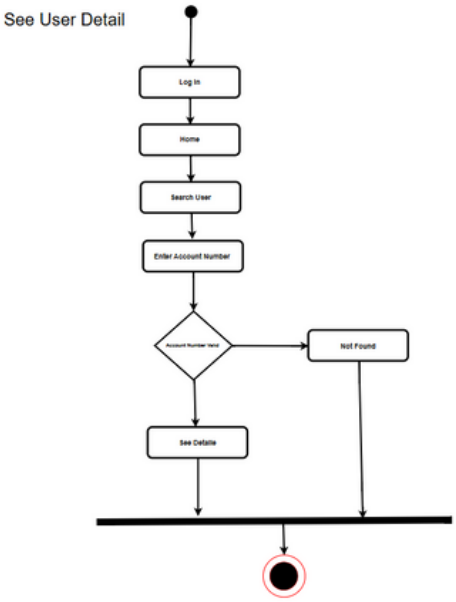

Add user
Log In
Home
Add User
Select Category
Enter Information
Submit
Successful

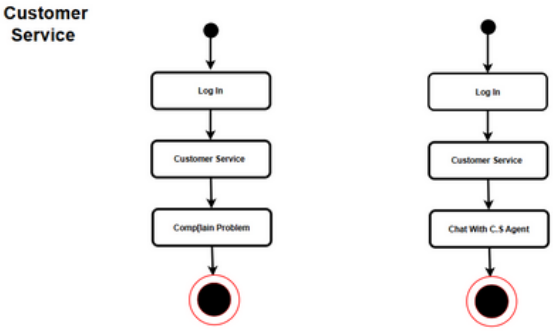
Remove user

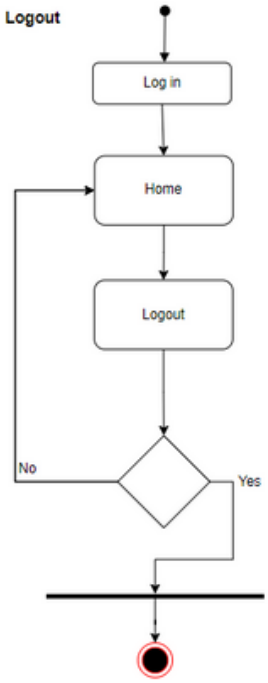


Edit/Update Information



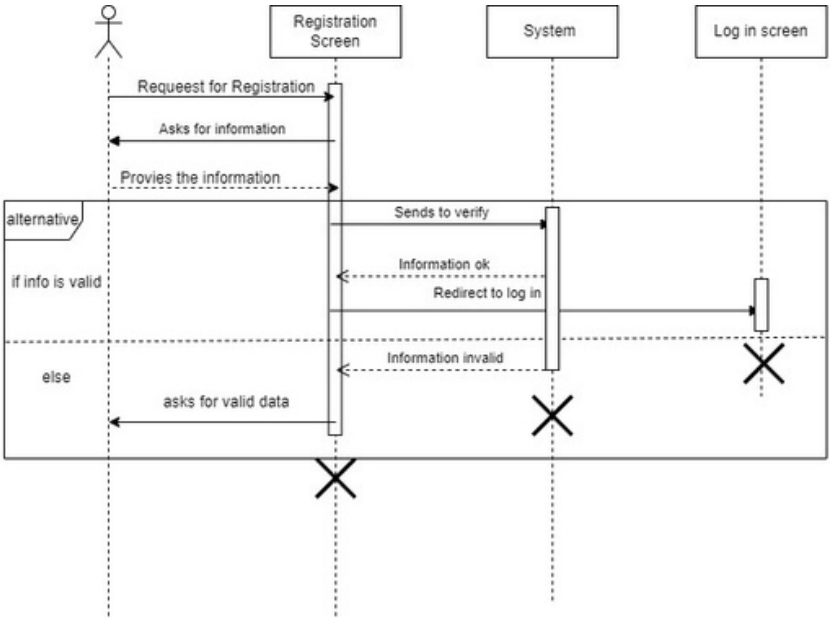

See User Detail
Log In
Home
Search User
Enter Account Number
Not Found
See Details


Customer Service
Log In
Customer Service
Complain Problem
Log In
Customer Service
Chat With C.S Agent

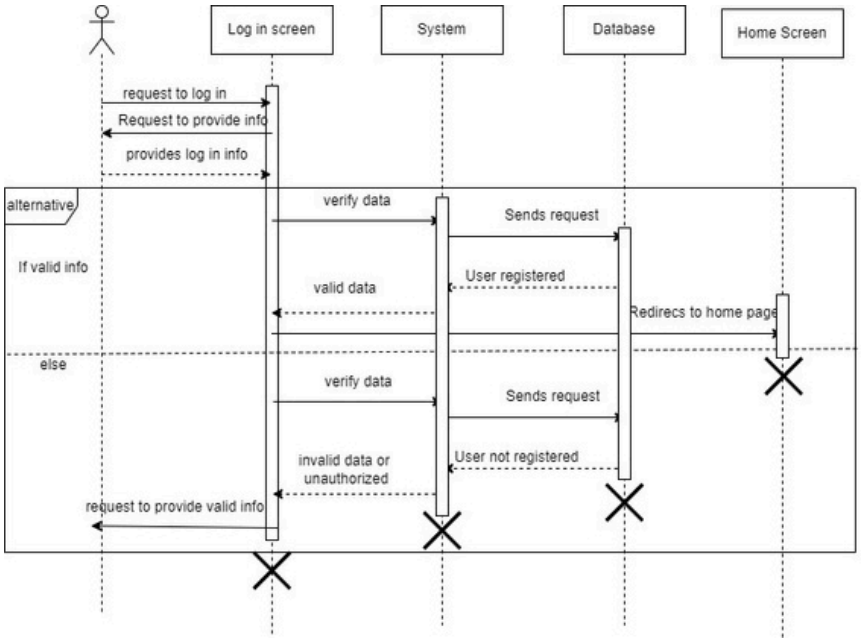

Logout
Log in
Home
Logout
No
Yes

2.7 SEQUENCE DIAGRAMS

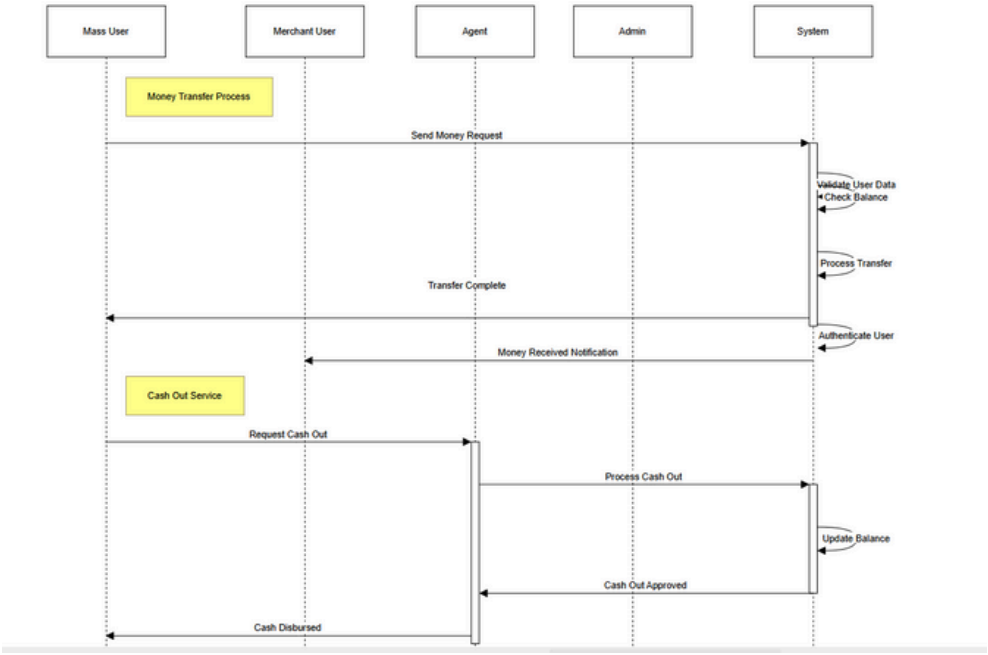
Registration



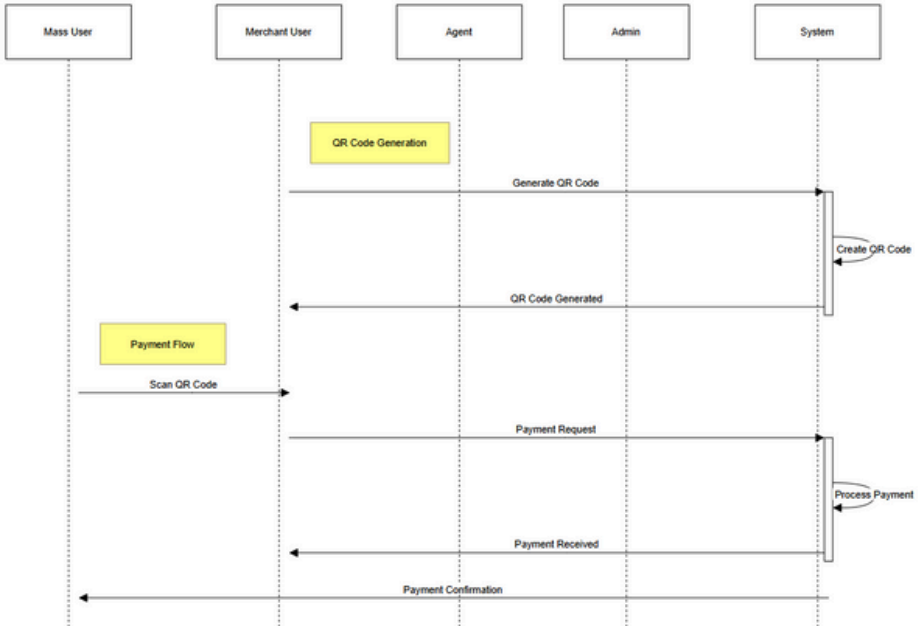
Login



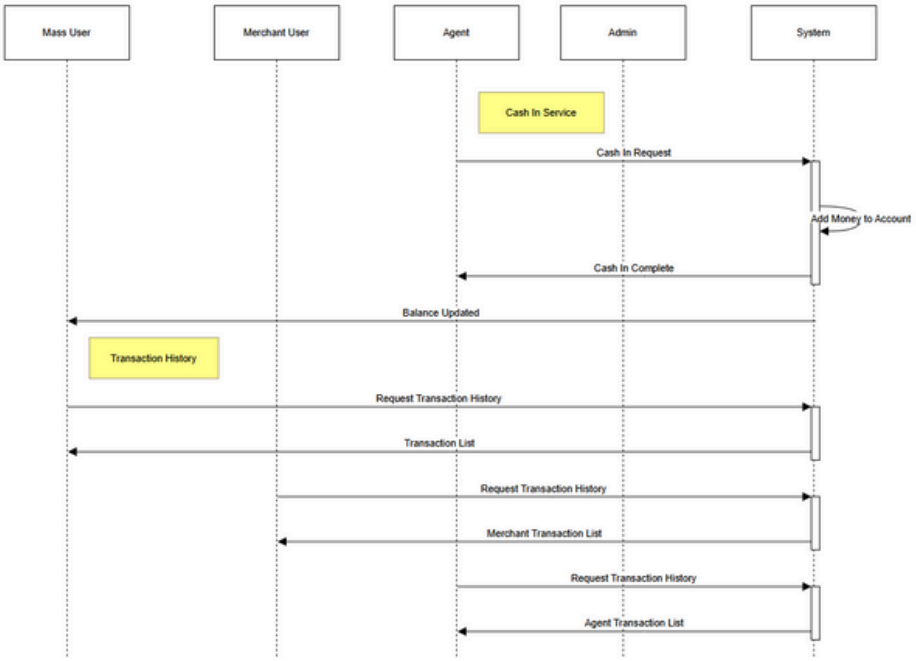
Send Money and Cash Out



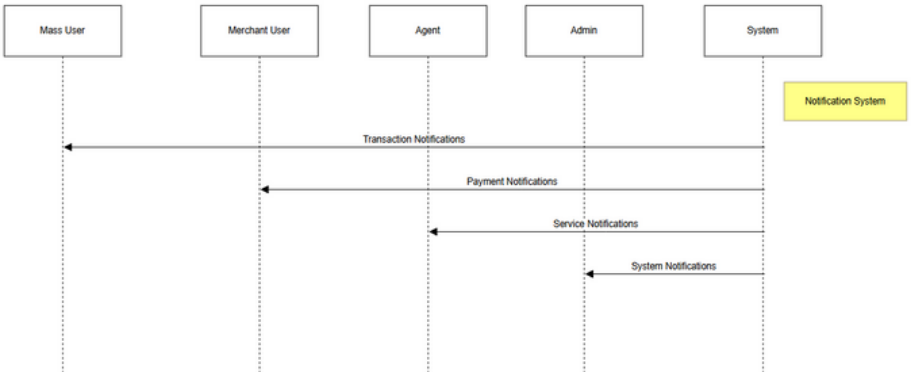
Payment



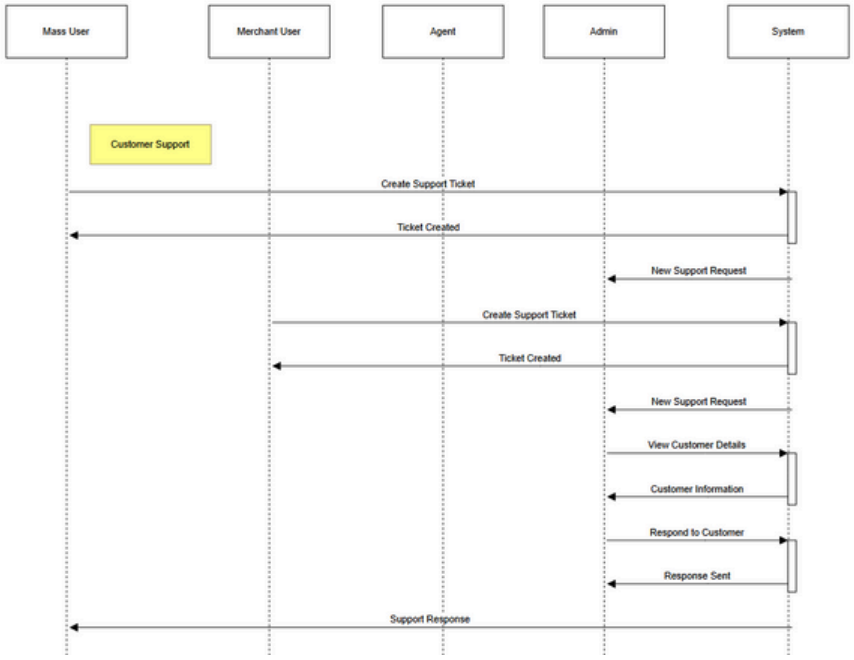
Cash In and Transaction History



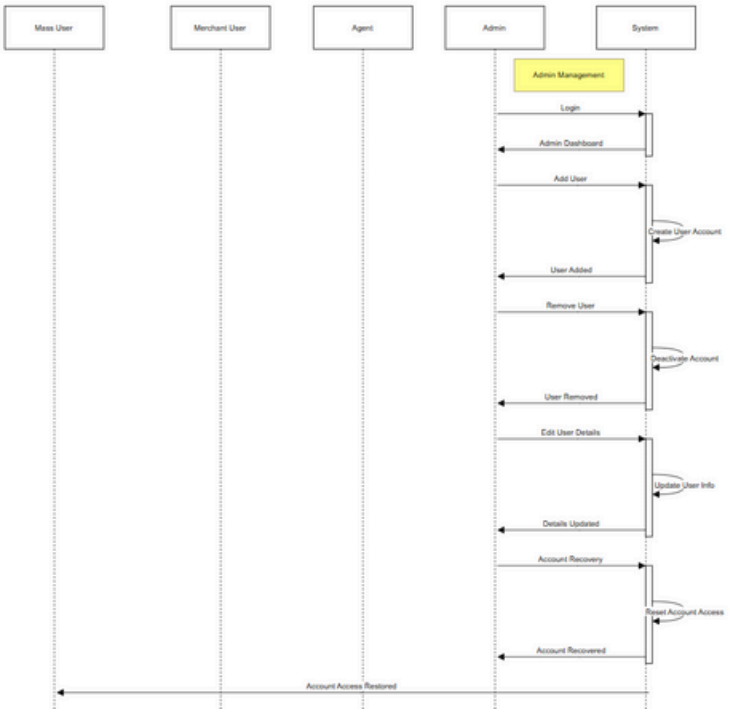
Notification



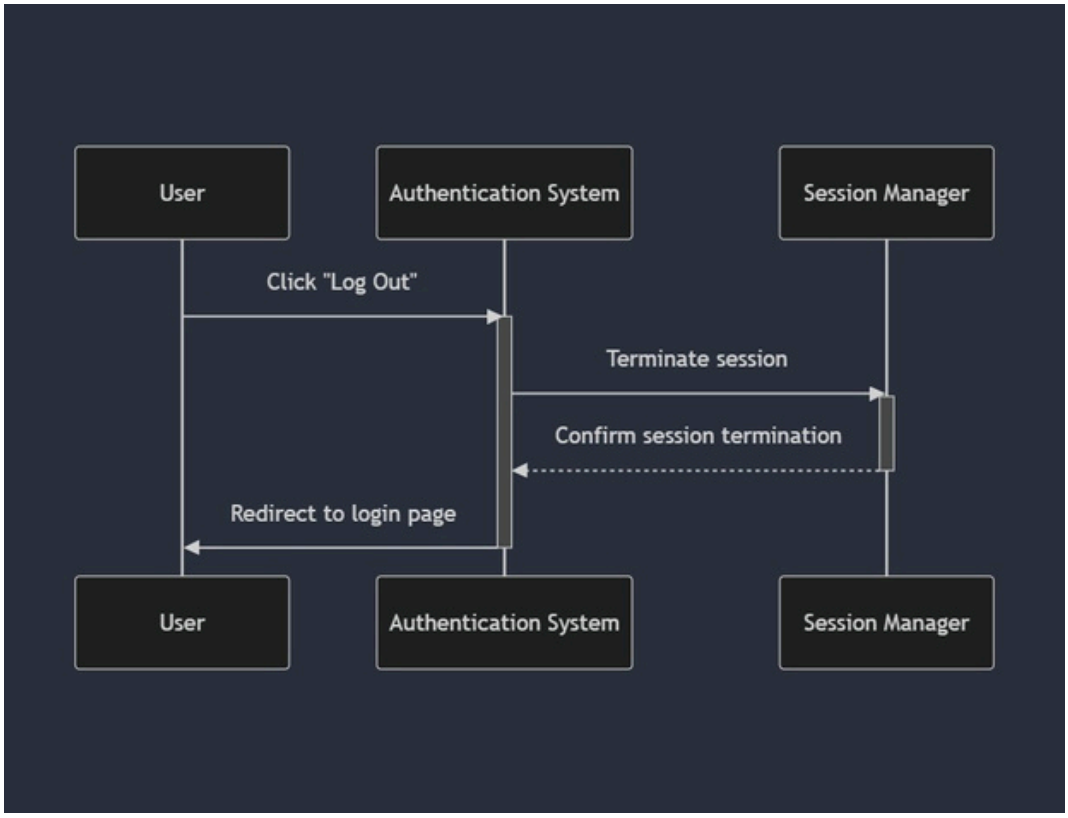
Customer Support



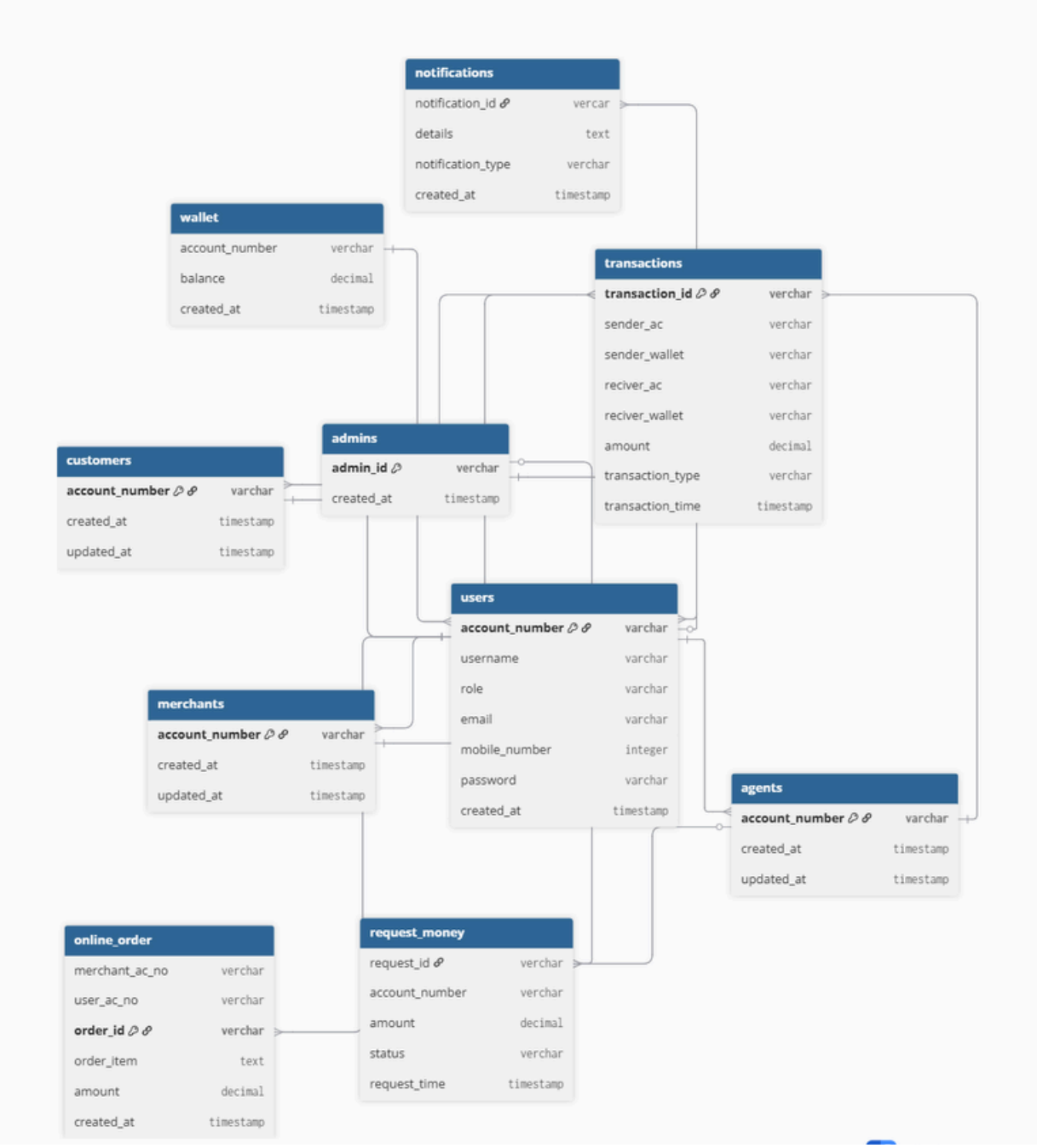
Admin



LogOut


User
Authentication System
Session Manager
Click "Log Out"
Terminate session
Confirm session termination
Redirect to login page
User
Authentication System
Session Manager

2.8 ER DIAGRAM



Chapter 3

Software Testing

3.1 Testing Features

Feature Area	Specific Functionality Tested
User Authentication	Account creation, login/logout, JWT validation, role-based access control, unauthorized access checks
Wallet & Transactions	Deposit, withdrawal, transfer flows; balance updates; transaction status tracking; error handling
Merchant & Agent Roles	Merchant onboarding, agent assignment, role switching, access restrictions
Dashboard Functionality	Admin views, agent dashboards, merchant transaction history, user activity logs
Request Handling	Submitting cash-out/help requests, status updates, resolution tracking

3.2 Testing Strategies

In order to make sure that the SmartSociety platform was robust, reliable, and efficient, the project involved a multi-layered approach to testing. The platform first involved Unit Testing, where each module in the backend API was individually tested for correctness in terms of functionality, data, and error handling before it was accepted as valid. This made sure that every piece of code worked when it was needed. Then there was integration testing, which checked that the platform's modules, such as authentication, wallet, and transacting, worked together without any problems. This made sure that. The platform was built in a way that made sure that data flowed correctly between the different services it offered. Lastly, the platform had Manual End-to-End testing, which meant that it acted like users like Admins, Agents, Merchants, and Customers. This made sure that every step of the procedure on the platform worked seamlessly together from beginning to end.

3.3 System Testing

This part gives you more test cases for important parts of the system. Check that they work well in a variety of situations.The main goal of system testing is to make sure that the SmartSociety platform works as a whole.

3.3.1 System Test Specification (STS)

Project Name: SmartSociety E-transaction syste,
Document Type: IEEE 829 System Test Specification
Prepared By: Md. Nazmus Shakib Khan
Date: 20/10/2025

1. Test Objectives

The goals are:

- Check that the system meets all of the functional requirements that were given.
- Check how well the system works in both normal and hard situations by testing it.
- Make sure that problems with connections, mistakes, and unauthorised access are handled correctly.

2.Scope

This testing includes functional tests like user authentication, sending money, cashing in, cashing out, making a payment, asking for money, and placing an order online.

● **Non-Functional testing:** Performance, scalability, usability, security, and dependability are all examples of non-functional testing.

●Taking care of Negative and Exception. Unauthorised access, incorrect inputs, issues with the backend, and issues with the network. This section gives you more test cases for important system features. Make sure they work well in a range of situations. The main goal of system testing is to check that the SmartSociety platform works as a whole.

3. Test Environment

Component	Details
Client Devices	PC, Mobile
Operating Systems	Windows, Mac OS, Android
Network	Stable broadband connection (5 Mbps) + simulated 3G
Backend	Postman
Frontend	local host
Testing Tools	Manual Testing.

4. Test Data

4.1 Sample User Accounts

Role	Account Number	Password
Customer	C475061	123
Agent	A595376	123
Merchant	M457893	123
Admin	batman(username)	123

5. Pass/ Fail Criteria

Condition	Pass Criteria	Fail Criteria
Functional Test	Output matches expected result exactly	Output deviates from expected result
Non-Functional Test	Meets performance/security benchmarks	Fails benchmarks or causes errors
Negative Test	System rejects invalid input or unauthorized actions correctly	System accepts invalid input or grants unauthorized access
Reliability Test	Systemrecovers from error conditions	System crashes or becomes unresponsive

6. Detailed Test Cases

Test Case ID	Feature Area	User role	Scenario Type	Pre-Conditions	Test Steps	Expected Result
Authentication & Access Control						
TC-AUTH-001	User Registration	Customer	Positive	Not logged in	1. Go to the registration page. 2. Enter accurate information. 3. Send in.	Email delivered, confirmation displayed, account setup.
TC-AUTH-002	User Registration	Customer	Negative – Invalid Email	Not logged in	1. Enter an incorrect email format. 2. Send it in.	"Invalid email address" is the error.
TC-AUTH-003	User Login	All	Positive	There is a legitimate account.	1.Put in your password and email address. 2. Login.	Redirect to correct dashboard.
TC-AUTH-004	User Login	All	Negative – Wrong Password	Valid account exists	1. You entered the incorrect password. 2. Submit.	Error: “Invalid credentials.”
TC-AUTH-005	User Login	All	ConnectivityIssue	Internet disconnected	1. Enter valid credentials. 2. Submit.	Error: “No internet connection.”
TC-AUTH-006	Role-Based Access	Customer	Unauthorized	Customr logged in	1. Access admin dashboard.	Error: “Unauthorized Access.”
TC-AUTH-007	Role-Based Access	Agent	Unauthorized	Agent	1. Access system settings.	Error: “Unauthorized

				logged in		Access.”
TC-AUTH-008	Session Timeout	All	Negative	Logged in, idle for > session limit	1. Stay inactive for 30 min. 2. Try an action.	Forced logout, redirected to login page.

Test Case ID	Feature Area	User role	Scenario Type	Pre-Conditions	Test Steps	Expected Result
Customer - Send money, Cash out, & Profile view						
TC-STU-001	Send Money	Customer	Positive	Logged in	1. Navigate to “Send money.”	Send money success
TC-STU-002	View Profile	Customer	System Error	DB down	1. Navigate to “Profile view.”	Error: “Unable to load profile”
TC-STU-003	Payment	Customer	Positive	Logged in	Navigate to make payment	payment successful.
TC-STU-004	Cash out	Customer	Positive	Logged in	Navigate to Cash out	Cashout successful.
TC-STU-005	View other profile	Customer	negative	Logged in	Navigate to profile	Error: “unauthorized
TC-STU-006	Send Money	Customer	Negative	Internet Issue	Navigate to send money	Unstable internet
TC-STU-007	Password change	Customer	Postive	Loged in	1. Click	Successful

					“\	
TC-STU-008	View Notificatio ns	Custom er	Positive	Logged in	1. Open Notification s.	All recent notifications shown.
TC-STU-009	View Transacti on history	Custom er	Positive	Logged in	Navigate to transaction history	transaction history displayed
TC-STU-010	Cashout	Custom er	Positive	Logged in	Navigate tio cashout	Confirmation message shown, status updated
TC-STU-011	cash out	Custom er	Negative	Logged in	Navigate to cashout	Error Insufficen t balance
TC-STU-012	CashOut	Custom er	Negative	Logged in	Navigate to cashout	Bad connection
TC-STU-012	Payment	Custom er	Negative	Loged In	Navigate to make payment	Error Insufficent balance
TC-STU-013	Cashout	Custom er	Negativ e	Logged in	Navigate to cashout	Wrong A/C no

Test Case ID	Feature Area	User role	Scenario Type	Pre-Conditions	Test Steps	Expected Result
Agent and Merchant						
TC-ADM-01	Cashin	Agent	Positive	Loged in	Navigate to cashin	Succesful
TC-ADM-002	Cashin	Agent	Negative	Loged in.	Navigate to cashin	Wrong AC no
TC-ADM-003	Request Money	Agent	Positive	Loged In	Navigate to request money	Sucessfull
TC-ADM-004	Request Money	Agent	Negative	Loged in	Navigate to request money	bad Connection
TC-ADM-005	Recieve Payment notification	Merchant	Positive	Loged In	Navigate to payment notifiction	View notification
TC-ADM-006	Cheeck Transaction	Merchant	Positive	Logged in	Navigate to transaction	transaction history displayed
TC-ADM-0	Cheeck Transaction	Agent	Postive	Logged in	Navigate to transaction	transaction history displayed

Chapter 4

Deployment and Maintenance

SmartSociety: Deployment and Maintenance Plan

1.Deployment Strategy

1.1 Web Application

- Platform: Vercel/Railway/Netlify
- Technology Stack: HTML, CSS, React, Backend: DRF, Postgres
- Deployment Steps:

1. Build Optimization

- Use python manage.py collectstatic and migratefor clean builds
- Minify static files and enable Gzip/Brotli compression via Nginx
- Optimize database queries with indexing and select_related
- Set DEBUG=False and secure production settings
- Automate builds and tests via GitHub Actions for consistent deployment

2. Testing & QA

- Utilize both UI and unit tests across various screen sizes and devices.
- To make sure that none of the current functionality is broken, perform some regression testing.

3. Deployment Credentials & Environment Security

- Generate and securely store environment secrets in a versioned vault or CI/CD secret manager.
- When you build and deploy, be careful to include these secrets to your CI/CD pipelines.
- Sign and encrypt all web traffic via HTTPS and SSL certificates.
- Set up SSH keys and db restrictions to limit who can go to production systems.

4. Web Application Deployment

- Have a privacy policy and terms of service page that is easily reachable by the users.
- Between user roles (Admin,) provide closed beta testing and internal QA testing before going live.

5. Post-Deployment Verification

- Verify core features (login, transactions, dashboards) and secure access controls post-deployment.
- Monitor performance, analytics, and uptime; confirm backups and error tracking are active.

2. Maintenance Plan

The maintenance plan will also ensure that the E-Transaction system runs smoothly even after. deployment. Conducting a regular back up of data, and performance. Periodic review of the system will be done. Accelerate the speed, address user reported problems and update features when there is need. All maintenance The workflow will be systematic to make sure that each task is reliable, reduces the downtime, and is long-term system stability.

2.1 Routine Monitoring

- Track the performance, errors, and logs with the help of such tools as Sentry, Prometheus, or New.
- Check database performance, authentication logs and server resource regularly.

2.2 Scheduled Maintenance

Monthly:

- Check the performance indicators and logs of errors.
- Erase all the logs, temporary files, or any other information that you do not require.
- Assist the users in queries and complaints.
- Conduct security audits and penetration tests after every three months.

Quarterly:

- Instantiate security audits and penetration testing.
- Logs of old transaction and activities.
- Revise and revise privacy policy and terms of service.
- Measure infrastructure utilisation and modify resources in case of necessity.

2.3 Incident Management

Bug Fix Protocol:

- Attach the problem to a tracker.
- Available in staging area and correct the root cause.
- Use the fix version inside.
- Try and apply it in the production.

Downtime Protocol:

- Show a maintenance page and show error message during downtime, while monitoring.

2.4 User Feedback Loop

- Customer can give feedback.
- Respond in 48 hours.

2.5 CI/CD Automation

Add both Github Actions or Bitrise for:

- Set up GitHub Actions to run automated tests on every commit and pull request.
- Deploy to staging or production automatically when changes are merged to the main branch.
- Securely inject environment variables and monitor build status for failures or rollback.

3. Versioning & Release Policy

VersionType: Patch, Major, Minor (e.g., 2.1.0)

Major: Significant new features or problems that cause something to break.

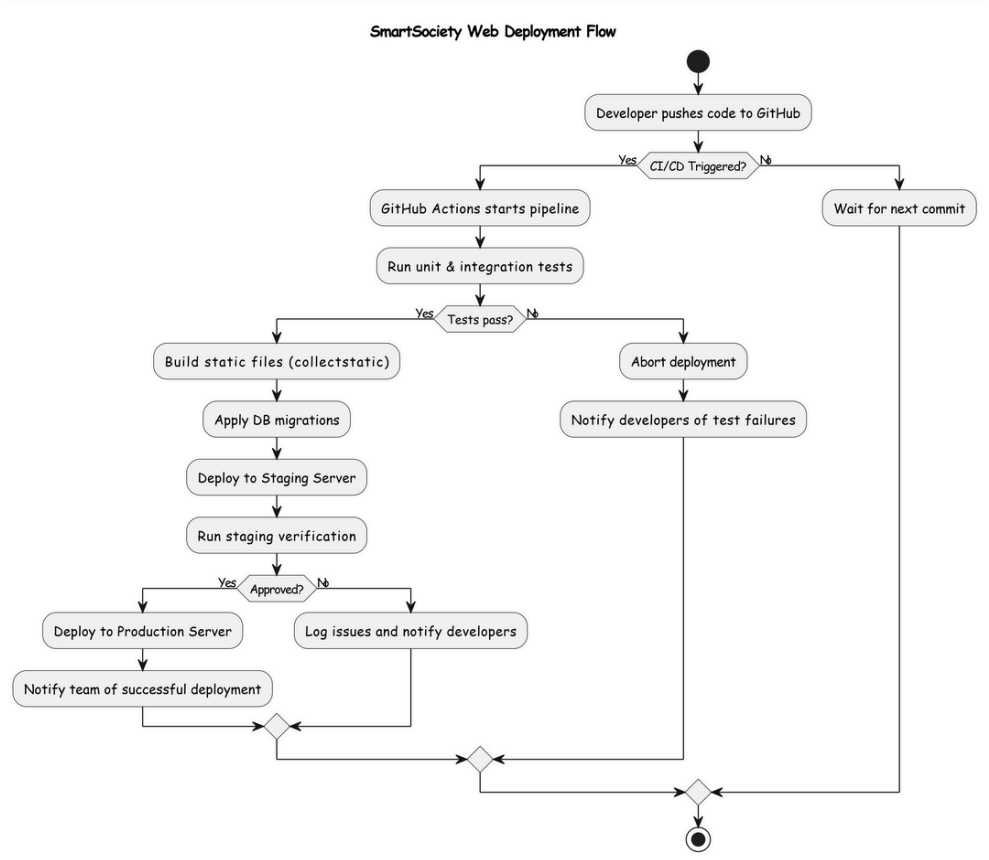
Minor: It incorporates functionality that works with previous software versions.

Patch: Performance enhancements and bug fixes.

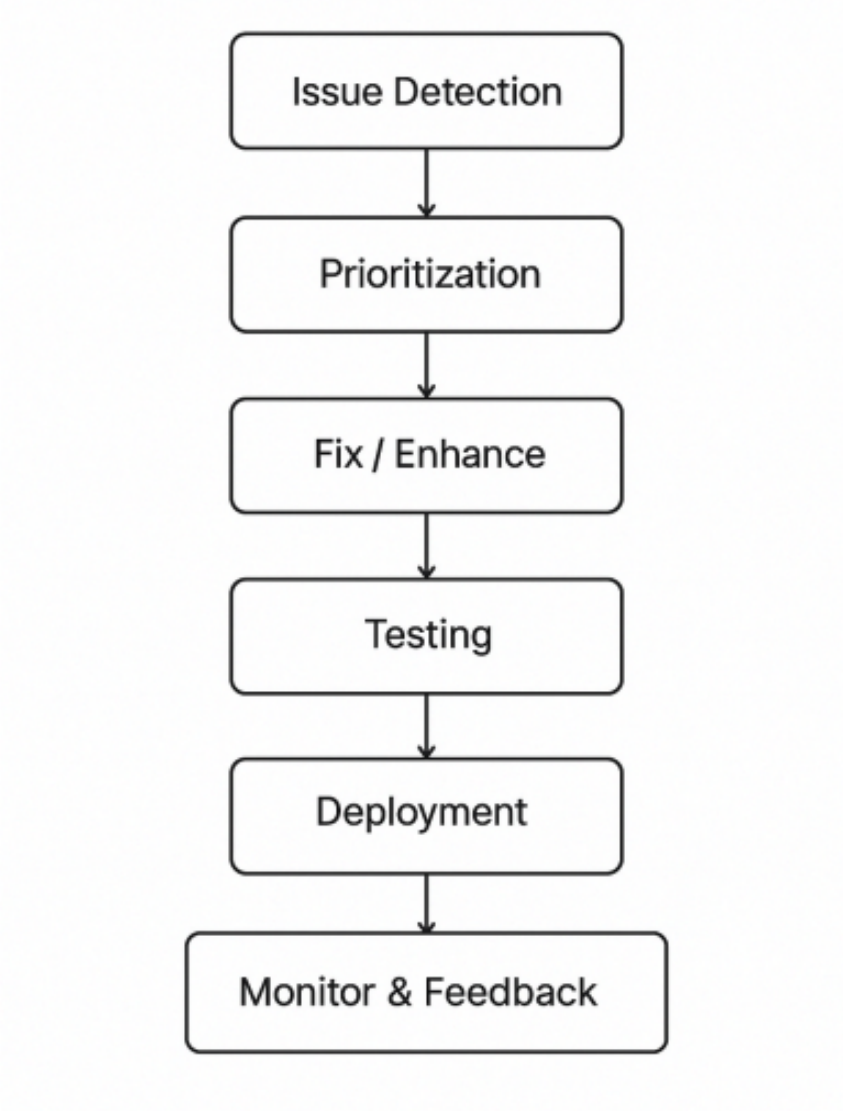
Release Cadence:

- Patches and minor updates: every two weeks or more frequently as needed.

3. Deployment Flow Diagram



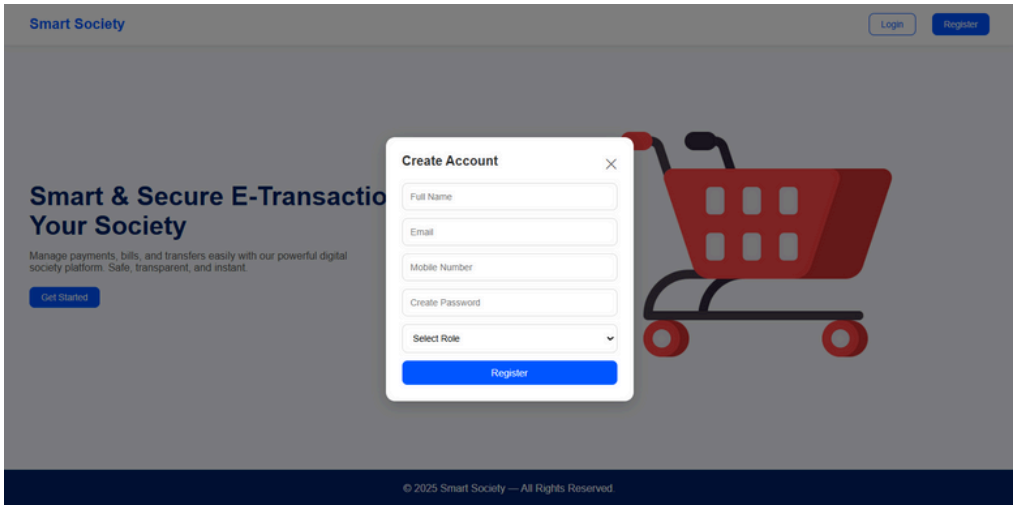
4. Maintenance Workflow Diagram



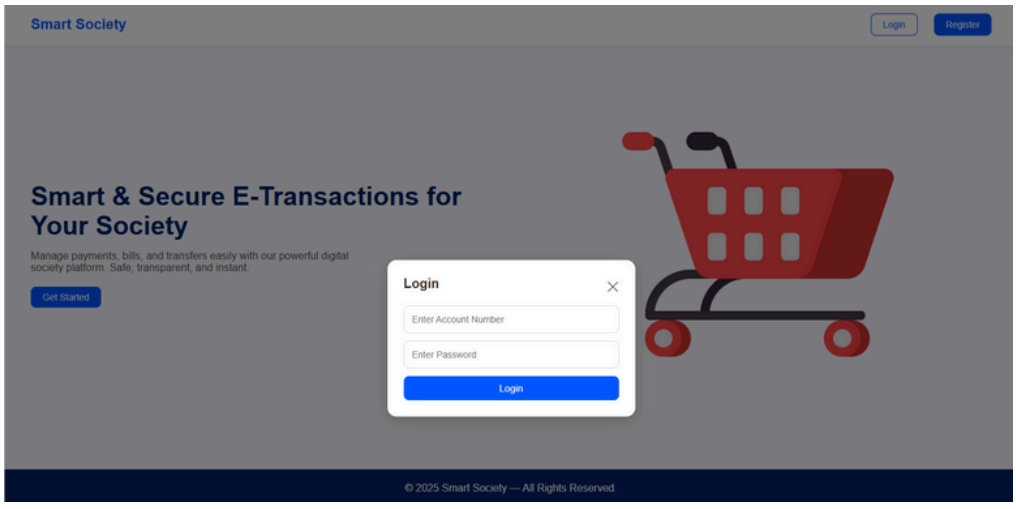
Chapter 5

User Manual

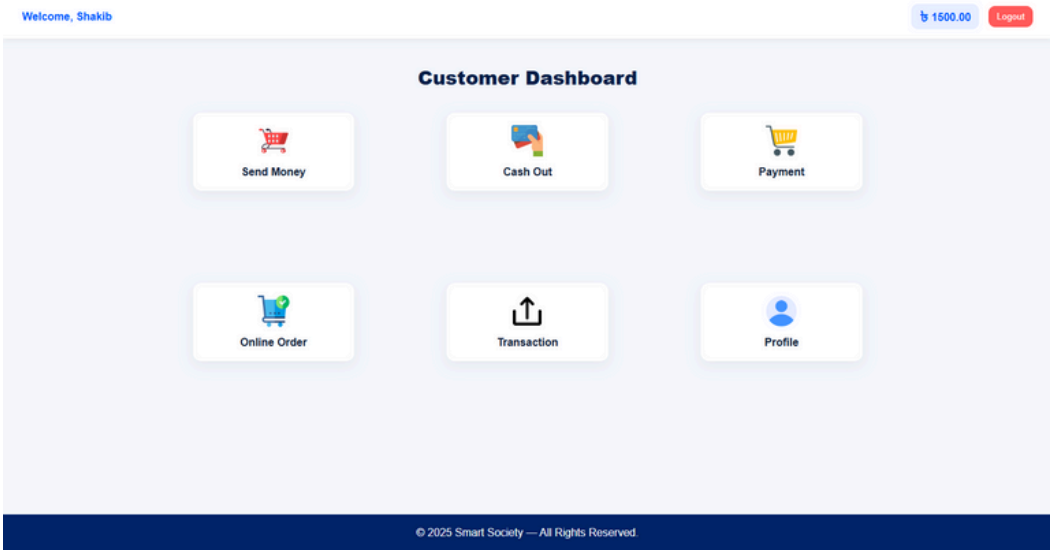
- **The user can register by providing valid credentials. When the user clicks the register button will view this form.**



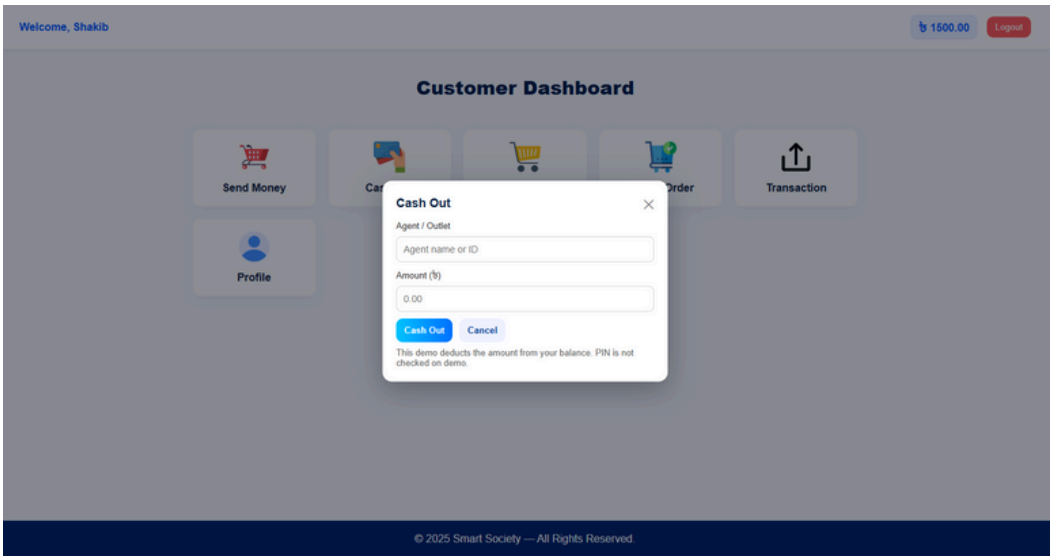
- **The registered user can log in by providing valid data. When the user clicks login button, it will show this form**



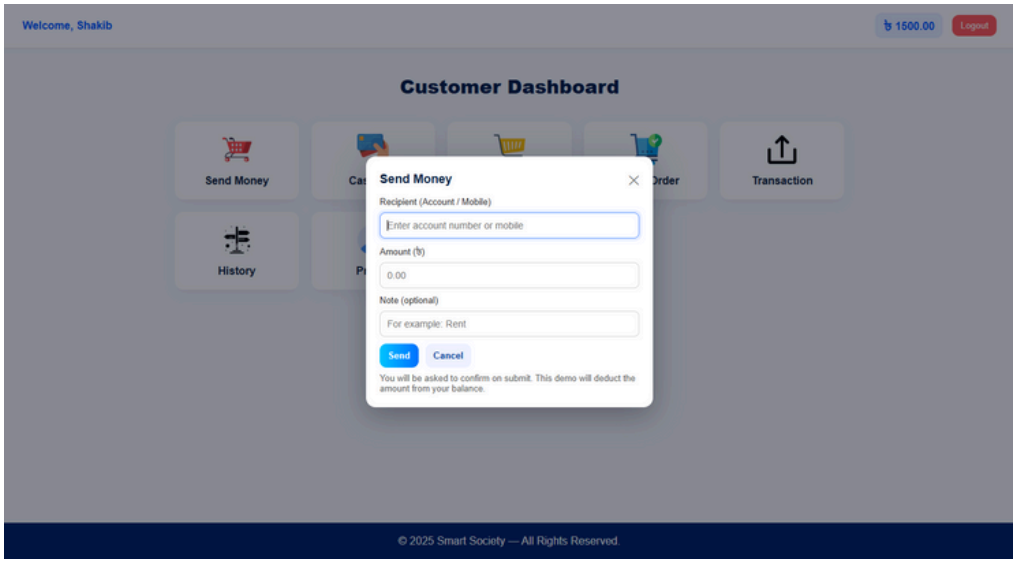
- **Customer view after login**



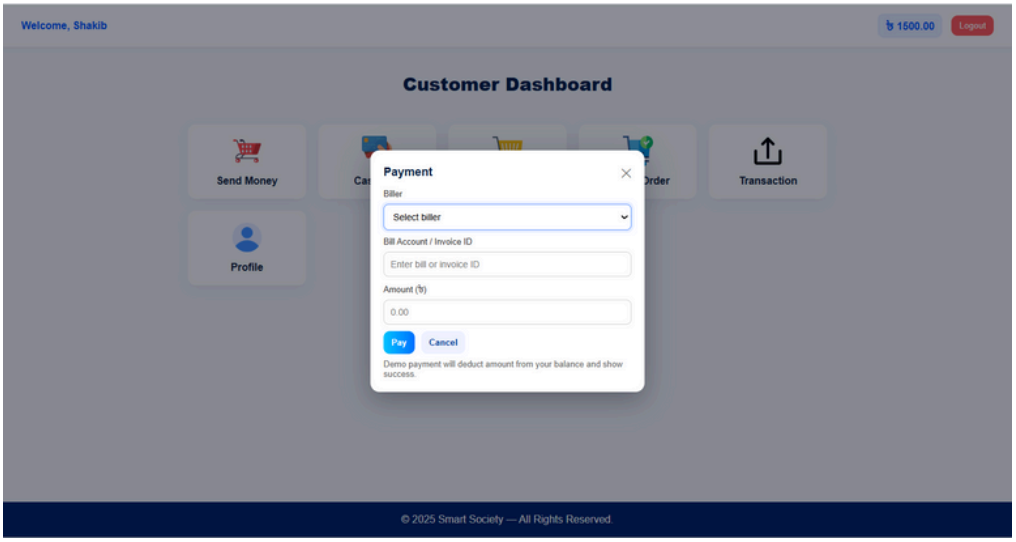
- **Cash out view after clicks cashout**



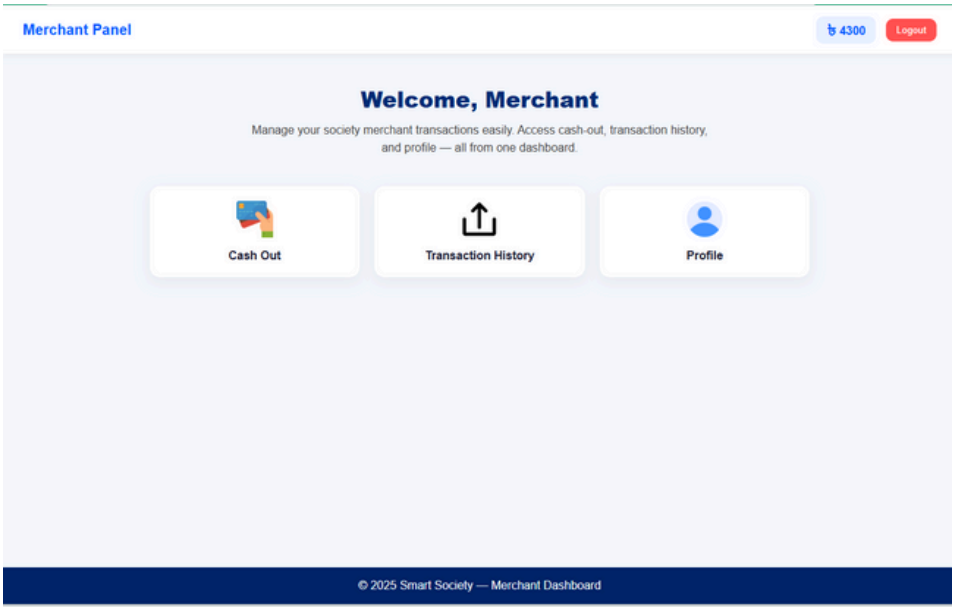
- **Send money view after clicking send money**



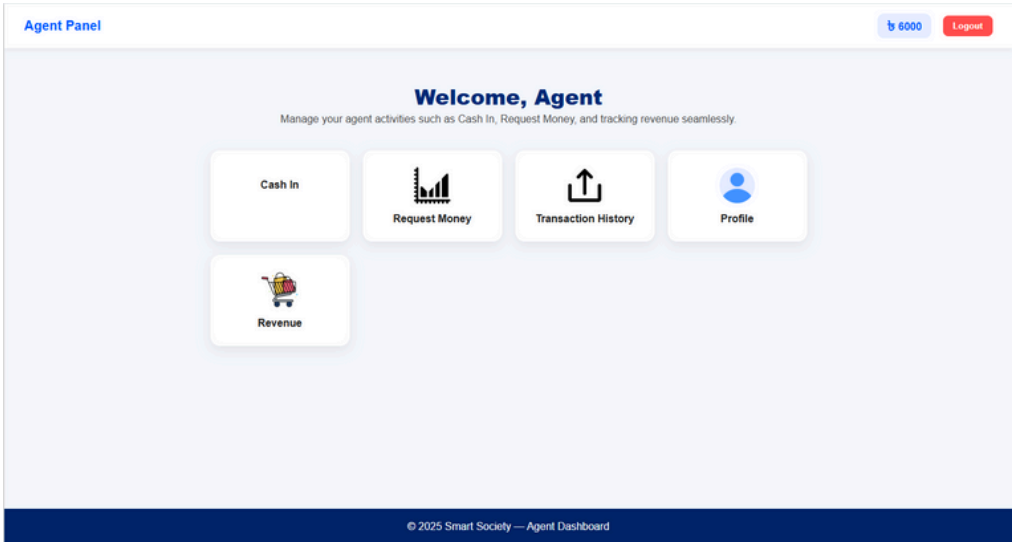
- **Payment view after clicking payment view**



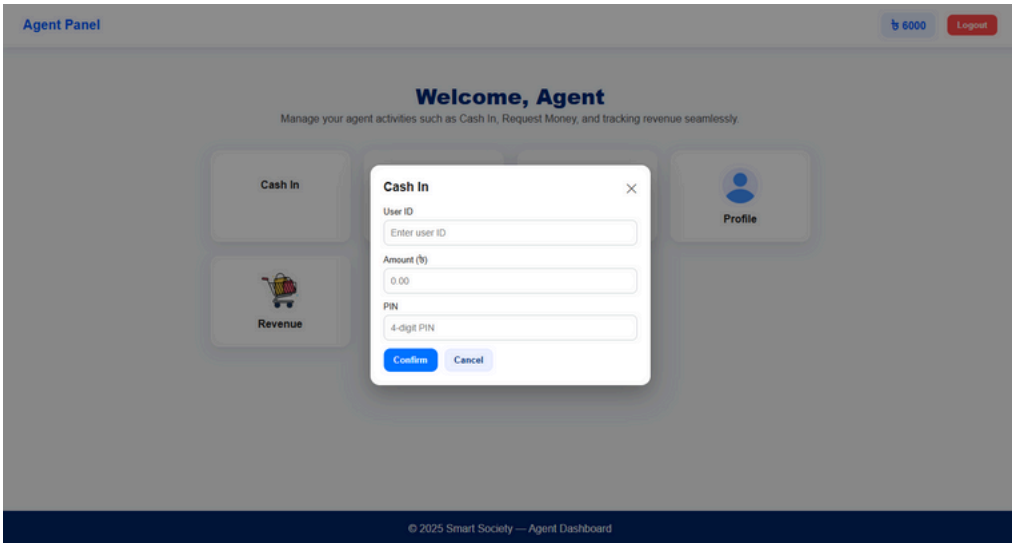
- **Merchant view after login**



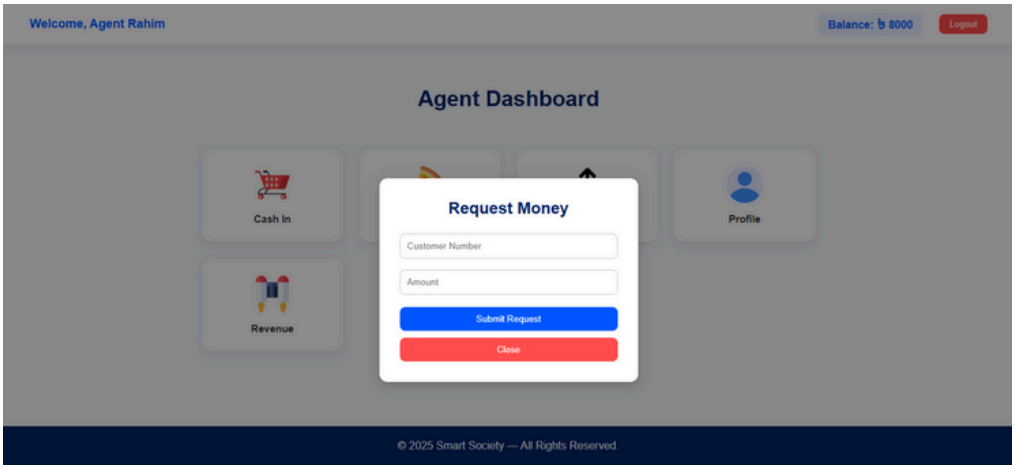
- **Agent view after login**



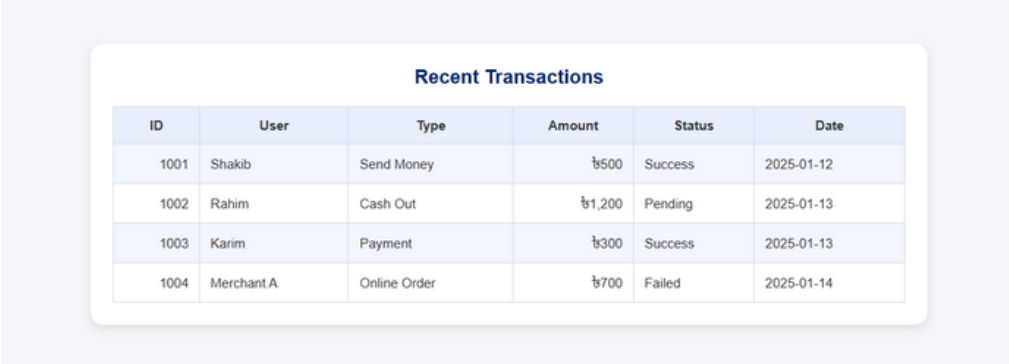
- **Cash in view after clicking cashin**



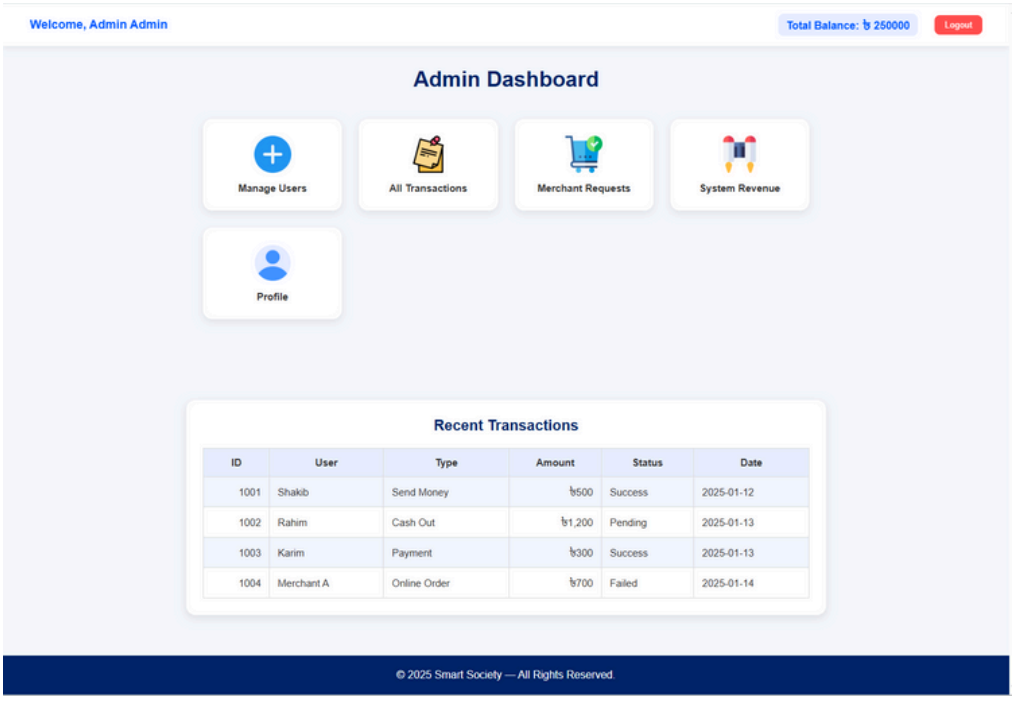
- **Request money view after clicking request money**



- **Transaction view after clicking transactions history**



- **Admin view**



Chapter 6

Project Summery

6.1 Project Overview

SmartSociety is a role-based e-transaction web platform designed to streamline digital financial services for local communities. It is a Django REST Framework and PostgreSQL based project. Supports Author, Agent, Merchant and Customer functions with secured and scalable operations.

6.2 Purpose

The project will facilitate financial inclusion, which facilitates a smooth digital transaction. Wallet management, and access to the service by various categories of users in Bangladesh.

6.3 Key Features

- Admin, Agent, Merchant, and Customer role based dashboards.
- Safe wallet system and history of transaction and balance.
- Analytics and stability error monitoring.

6.4 Technology Stack

- **Frontend:**HTML, CSS, Boostrap
- **Backend:** Django REST Framework
- **Hosting:** Vercel/Railway
- **Database:** Postgresql

6.5 Expected Impact

Through SmartSociety, underserved communities will be empowered by having easy access to fintech tools. Dependence on cash, and promote faith in digital financial systems. It facilitates a scaled up growth and conforms to national objectives of digital transformation.

6.6 Future Possibilities

There is a good prospect of the E-Transaction system as digital financial services keep on rising.to grow and be innovational. As the number of users who desire faster, safer and more increases. Easy payment options, the site can become a more hi-tech digital. ecosystem. Some of the future possibilities that can facilitate functionality are listed below.

- Mobile App Integration: It would have a specific mobile app that would make the transactions faster and easier. Easy to use by ordinary people.
- QR Code Payments: Fast in-store and online payment it involves scanning a merchant and making a payment.
- Enhanced Security Enhancements: Biometrics, MFA, and fraud detection would be added.
- Utility/Bill Payments: Utility assistance in the form of electricity, water, internet and mobile recharge services is available.
- Expand system usage. International Transactions: Global opening could be through international payments and conversion of currency.
- AI Chatbot Assistance: Instant automatic assistance can be offered to the user to address transaction problems promptly.

Github Link - https://github.com/Md-Nazmus-Shakib/SmartSociety-Community-Centric-Mobile-Banking-Solution-for-Financial-Services.git

References

1. https://en.wikipedia.org/wiki/Mobile_banking#:~:text=Mobile%20banking%20is%20a%20service,on%20a%2024%2Dhour%20basis.
2. https://www.bkash.com/products-services/add-money?gad_source=1&gad_campaignid=21439289226&gbraid=0AAAAA9iiHy45Eb0HcyJNQRidPS_uwT5gt&gclid=CjwKCAjwx8nCBhAwEiwA_z__07xHB7DADjN714mY54OobBGwhUfjfssKg_DWrQ8EFUZEnNkYUkI1vhoCh0UQAvD_BwE
3. https://www.paypal.com/bd/home
4. https://daffodilvarsity.edu.bd/article/1card-diu