American International University-Bangladesh (AIUB)

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

PROJECT TITLE
DIGITAL RATION DISTRIBUTION SYSTEM

A Software Engineering Project Submitted
By

Semester: Fall_24_25		**Section:**	**Group Number:**	
SN	Student Name	Student ID	Contribution (CO3+CO4)	Individual Marks
1.	Ritu Rani Das	22-49459-3	20%	
2.	Abir Ahmed Antor	22-49460-3	20%	
3.	Anirban Sarkar	22-49680-3	20%	
4.	A.M. Fozlul Karim Sabbir	22-49677-3	20%	
5.	M.D. Fahim Montasir	22-49400-3	20%	

The project will be evaluated for the following Course Outcomes

	Total Marks

CO3: *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		
Appropriate Process Model Selection and Argumentation with Evidence	[5 Marks]	
Evidence of Argumentation Regarding Process Model Selection	[5Marks]	
Analysis of the impact of societal, health, safety, legal, and cultural issues	[5Marks]	
Submission, Defense, Completeness, Spelling, grammar, and Organization of the Project report	[5Marks]	
CO4: *Develop* a project management plan to manage software engineering projects following the principles of engineering management and economic decision process	Total Marks	
Develop the project plan, its components of the proposed software products	[5Marks]	
Identify all the activities/tasks related to project management and categorize them within the WBS structure. Perform detailed effort estimation correspond with the WBS and schedule the activities with resources	[5Marks]	
Identify all the potential risks in your project and prioritize them to overcome these risk factors.	[5Marks]	
CO5: Perform as an effective team member or leader in diverse team settings and solve multi-disciplinary problems in the computer science and engineering domain	Total Marks	
Taking project responsibility: perform assigned tasks on time independently	[5 Marks]	
Contribution to project group meetings, sharing fruitful ideas	[5Marks]	
Positive attitude towards group work, collaboration, compromise, helping others to understand their project work responsibility	[5Marks]	
Showing respect and value towards other team member's opinion	[5Marks]	

Description of Student's Contribution in the Project work

Student Name: Ritu Rani Das
Student ID: 22-49459-3
Contribution in Percentage (%): 20%
Contribution in the Project:

1. Collects ideas and needs from the government and people &Decides what features to build first
2. Makes sure everyone follows the process

________Ritu__________
Signature of the Student

Student Name: Abir Ahmed Antor

Student ID: 22-49460-3
Contribution in Percentage (%): 20%
Contribution in the Project:

1. Write the code for the system & Build features like login, stock tracking, reports
2. Fixing all the bugs that is detected and helping the Ui/Ux designer to make key function of the product outlook.

________Abir________
Signature of the Student

Student Name: Anirban Sarkar
Student ID: 22-49680-3
Contribution in Percentage (%): 20%
Contribution in the Project:

1. Test the system to find problem & fix all the bugs before launch the final prodect.
2. Making all the ui/ux desing(login page, home button etc.)

________Anirban_____
Signature of the Student

Student Name: A.M. Fozlul Karim Sabbir
Student ID: 22-49677-3
Contribution in Percentage (%): 20%
Contribution in the Project:

1. Decide how the system works from the behind.
2. Taking feedback from the government and the consumer also.

______Sabbir________
Signature of the Student

Student Name: M.D. Fahim Montasir
Student ID: 22-49400-3
Contribution in Percentage (%): 20%
Contribution in the Project:

1. Handle the launching.
2. Taking further feedback of final product from the consumer and making change in product with its essential needs.

_______Fahim_______
Signature of the Student

1. PROJECT PROPOSAL

1.1 Background to the Problem

In many regions, the traditional ration distribution system suffers from inefficiencies such as manual record-keeping, corruption, and lack of transparency. Same things are going to happen with the people of Bangladesh every year. The Public Food distribution System(PFDS) in Bangladesh is a network of programs designed to support vulnerable populations through food assistance. The survey says that there are 23.6 million people are facing food insecurity by Bangladesh government. On the contrary only 5 million of them have got the ration. What is the reason behind of it? Lack of digitalization and negligence are the main behind of it. The Beneficiaries often face long queues, delays, and irregular supply of rationed items. Additionally, tracking eligibility and consumption records is difficult due to poor digitization, which leads to leakages and unfair distribution of subsidized goods.

1.2 Solution to the Problem (Expanded Version)

The **Smart Ration System** presents a modern, technology-driven approach to resolving the inefficiencies of traditional ration distribution systems. It is designed as a comprehensive, digital platform that enhances transparency, security, and accessibility. Below are the major components and features of the proposed solution:

1. **Digitalization of Ration Distribution**
   The system will transition all manual record-keeping and physical documentation into a digital format. This eliminates paperwork, minimizes human error, and improves data accuracy and long-term accessibility.

2. **Automation of Processes**
   Important tasks like checking who gets rations, managing stock, and giving out supplies will be done automatically. This means less need for manual work, faster service, and better efficiency.
3. **Biometric Authentication**
   To ensure that only eligible users can access subsidized goods, the system will incorporate biometric authentication mechanisms. These may include:
   - **Voter Id card / National Id card**
   - **Fingerprint / facial recognition systems**
   - This step guarantees identity verification, minimizes duplication and strengthens the security of the system.

4. **Real-Time Inventory Management**
   The system will track ration stock levels in real time at every ration shop or distribution center. This will:
   - Prevent shortages and overstocking
   - Provide administrators with instant visibility of stock flow
   - Enable proactive restocking based on predictive analytics

5. **Automated Beneficiary Verification**
   The system will be connected to a centralized government or welfare database to automatically verify the eligibility of ration recipients. This ensures that:
   - Only qualified individuals receive subsidized items
   - The system remains fair and corruption-free
   - Fraudulent claims are reduced significantly

6. **Digital Ration Cards and Receipts**
   Beneficiaries will receive **digital ration cards** which will store user data, transaction history, and eligibility status. After each transaction, the system will generate **digital receipts**, which can be:
   - Emailed or sent via SMS
   - Downloaded through the web or mobile app
   - Used for tracking and dispute resolution

7. **User-Friendly Interface**
   The platform will be designed to cater to both tech-savvy and less-experienced users, including:
   - Simple navigation for rural users
   - Multiple language support
   - Voice assistance (optional) for visually impaired users
   - Accessible via smartphones, tablets, or desktop computers

8. **Enhanced Transparency and Accountability**
   Every transaction, inventory change, and user interaction will be logged and monitored. This promotes:
   - Accountability among ration shop operators
   - Auditability for government oversight
   - Public trust in the system

9. **Fraud Reduction and Corruption Prevention**
   With biometric access, digital records, and real-time checks, the system will drastically minimize:
   - Identity fraud
   - Duplicate beneficiaries
   - Unauthorized diversion of goods

10. **Integration with Notification Systems**
    Beneficiaries will receive:
    - **SMS and email alerts** about ration availability, schedule changes, or new policies
    - **Reminders** for collection dates
    - **Emergency alerts** in times of shortage or crisis

The Smart Ration System is strong and can grow easily. It makes sure the ration process is fair, fast, and clear, while also making things better for both the people who give the rations and those who receive them.

2. Software Development Life Cycle

2.1 Process Model

We choose Scrum from an agile process model to build our software.

Why Scrum is best for the Smart Ration System?

- Supports complexity: Handles components like biometric, inventory systems, government database integrationet and others.

- Engages stakeholders: Regular sprint reviews and feedback loops help involve both government and end users.

- Transparency: Ideal for a project where accountability and documentation of progress are important.

- Iterative by nature: Breaks down the full system into manageable releases (exmple - one sprint for digital cards, one for biometric).

- Scalability: Can scale with Scrum of Scrums if multiple teams work on different modules.

Why we are going to use iterative instead of linear process model like waterfall model?

The Waterfall model is a linear and rigid software development approach. It works best when:

- Requirements are well-known and unlikely to change
- The project is simple and short-term
- There's little need for feedback from end-users during development

However, the Smart Ration System doesn't fit these conditions. Here's why:

1. **Changing Requirements**

- Government regulations, user feedback, and technology (like biometric tools) may change over time.
- Waterfall doesn't handle changes well once a phase is completed — going back is difficult and expensive.

Agile (Scrum) handles this much better through iterative sprints and constant feedback.

2. **User Involvement Needed**

- Beneficiaries, ration shop operators, and government staff must be involved to ensure usability.

- Waterfall limits user involvement to early stages, leading to late discovery of usability issues.

Scrum includes users throughout the process via sprint reviews, testing, and feedback.

3. **High System Complexity**

- This system integrates **biometric auth, real-time stock, notifications, fraud detection**—all with many modules.
- Waterfall doesn't cope well with large, interdependent systems unless everything is planned perfectly up front.

Agile allows for **incremental building and testing**, reducing risk in complex systems.

4. **Long Time to See Results**

- In Waterfall, you only see the working system at the **end** of the project.
- For a critical public service, early feedback and visible progress are essential.

In Scrum, working features are delivered **every sprint**, so progress is visible and usable sooner.

5. **Maintenance & Upgrades**

- Ration systems must evolve (e.g., new welfare policies, disaster response, mobile support).
- Waterfall is not ideal for continuous updates — you'd need to restart the whole process.

Agile is **naturally iterative**, making updates and new features easy to plan and deliver.

Waterfall is too rigid and risky for a modern, evolving system like the Smart Ration System. **Scrum (Agile)** offers the flexibility, feedback, and speed needed to make this project successful.

Why not other iterative process model?

Let's now compare **Scrum** with **other iterative models** like **XP**, **DSDM**, **FDD**, and **Kanban**— and explain **why they may not be the best primary choice** for the **Smart Ration System**.

1. **XP (Extreme Programming)**

Good For: Projects needing fast releases, high-quality code, and constant refactoring.

Pros:

- Emphasizes clean code and testing (TDD, pair programming).
- Great for rapidly changing requirements.

Why *not ideal*:

- Too focused on developers and coding practices, not much on stakeholder roles or planning.
- Lacks structured roles and sprint planning needed in a government/public service project.
- Doesn't emphasize project management or system-level planning like Scrum.

2. **DSDM (Dynamic Systems Development Method)**

Good For: Business-critical applications with strict timelines and budgets.

Pros:

- Well-structured with full lifecycle coverage.
- Prioritizes business needs using the MoSCoW method.

Why *not ideal*:

- Too heavy and formal — requires a lot of upfront documentation and role setup.
- Overkill for a system like Smart Ration unless it's a multi-agency, enterprise-scale rollout.
- Less flexible than Scrum in smaller or mid-sized team environments.

3. **FDD (Feature-Driven Development)**

Good For: Large, object-oriented software projects focused on delivering features.

Pros:

- Focuses on building features in short cycles.
- Scalable for large dev teams.

Why *not ideal*:

- Very developer-centric – doesn't focus on stakeholder engagement or feedback cycles.
- Doesn't emphasize collaboration, testing, or user experience, which are vital in public service systems.
- Not suited for systems requiring real-time monitoring, authentication, and integrations.

4. Kanban

Good For: Ongoing support projects, operations, or continuous delivery teams.

Pros:

- Great for visualizing and limiting work-in-progress.
- Easy to implement in support or maintenance phases.

Why *not ideal for development*:

- No structured sprints, planning, or clear project milestones.
- Lacks focus on initial feature development, roadmap planning, and iterative delivery.
- Better suited after the main system is built, to handle updates and fixes.

2.2 Project Role Identification and Responsibilities

In our project, many people work together to ensure success. Each person has a specific role with clear responsibilities. Here's a simple explanation of each role:

Role	Who They Are	What They Do	Contribution
Product Owner (PO)	The person who knows what the system should do	- Collects ideas and needs from the government and people - Decides what features to build first - Checks if the system is working as expected	**Ritu**
Scrum Master	The team guide or helper	- Helps the team stay organized - Solves problems that slow the team - Makes sure everyone follows the process	**Ritu**
Developers	The people who build the software	- Write the code for the system - Build features like login, stock tracking, reports - Fix bugs and improve performance	**Abir**
Testers (QA)	The people who check if everything works	- Test the system to find problems - Make sure it works correctly and safely - Help fix issues before users see them	**Abir** **Anirban**
UI/UX Designer	The person who designs how the system looks and feels	- Makes the system easy to use - Designs buttons, menus, screens - Supports different languages and simple layouts	**Anirban**
System Architect	The tech planner	- Decides how the system works behind the scene - Chooses tools and technologies - Makes sure it is secure and fast	**Sabbir**
Government Coordinator	The link between the team and the government	- Shares real -world rules and policies - Helps understand how ration shops work - Gets feedback from officials	**Sabbir**

DevOps Engineer	The person who handles launching and running the system	- Sets up the servers and software updates - Keeps the system running 24/7 - Manages backups and speed	**Fahim**
End Users / Stakeholders	The people who use the system	- Ration card holders, shopkeepers, officers - Use the system and give feedback - Report any problems or confusion	**Fahim**

By clearly defining each person's role, the team can:

- Work together efficiently.
- Adapt to changes quickly.
- Deliver high-quality features that meet user needs.
- Continuously improve the product and team process.

Using the Scrum model ensures that everyone knows what to do, how to collaborate, and how to handle challenges effectively.

3. Features and Functionality

Phase	Feature Set	Key Features
Phase 1: Foundation	Project Setup & Core Infrastructure	Project setup, database, API structure, UI framework, admin login
Phase 2: Digitalization	Ration Registration System	Digital ration card creation, beneficiary registration, admin panel
Phase 3: Authentication	Biometric Verification	Fingerprint/facial recognition integration, NID/Voter ID linking
Phase 4: Automation	Stock & Distribution Management	Real-time stock tracking, automated allocation, receipt generation
Phase 5: Communication	Notifications & Alerts	SMS/email alerts, collection reminders, crisis notifications
Phase 6: User Experience	Interface & Accessibility	Multi-language support, voice guidance, mobile-friendly layout
Phase 7: Governance	Transparency & Reporting	Audit logs, analytics dashboard, fraud detection features

1.0 Registration and User Onboarding

1.1 User Registration

- Users can sign up using name, mobile number, Aadhaar (or national ID), and ration card number.
- OTP verification ensures valid identity.
- Aadhaar and ration card are cross-verified with government databases.

1.2 Profile Completion

- Users can optionally add family member details.
- Select nearest fair price shop (FPS).

1.3 Terms & Consent

- Users agree to terms before account creation; consent is logged.

1.4 Error Handling

- Duplicate Aadhaar/ration card errors are flagged.
- Incomplete fields prompt specific error messages.

2.0 Login and Authentication

2.1 Secure Login

- Login using registered mobile number + OTP or password.

2.2 Forgot Password

- Option to reset password via OTP.

2.3 Two-Factor Authentication

- Optional OTP on each login for extra security.

3.0 Ration Allocation & Claims

3.1 Monthly Allocation

- The system displays individual/family quota based on government rules.
- Quotas refresh monthly.

3.2 Claim Ration

- Users visit FPS and authenticate via app or fingerprint (if integrated with POS).
- Claimed amount is updated in real time.

3.3 Ration History

- Users can view previously claimed and remaining ration for the month.

4.0 FPS Management

4.1 FPS Dashboard

- Fair Price Shop dealers have access to stock inventory, distribution logs, and user verification.

4.2 Daily Reporting

- Auto-generated daily reports on distributed vs. remaining stock.

4.3 Stock Refill Request

- FPS owners can request refills from the central depot via app.

5.0 Grievance and Support

5.1 Submit Complaints

- Users can submit ration-related issues (e.g., incorrect weight, denied service).
- Attach images or documents.

5.2 Resolution Tracking

- Complaint status tracking with resolution timelines.

6.0 Notification & Alerts

- Alerts for monthly allocation updates.
- Pickup reminders and shortage warnings.
- Notification on complaints, verification, and profile updates.

7.0 Analytics and Reporting (Admin Panel)

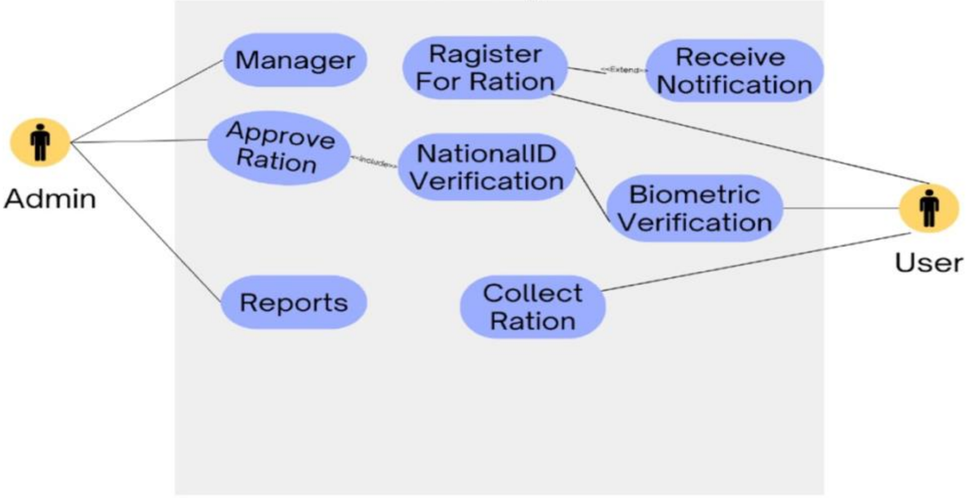
- District/state officials can monitor:
    - Total distributed rations
    - Shop-wise performance
    - Defaulters and anomalies
- Exportable reports (CSV/PDF).

8.0 Feedback and Improvement

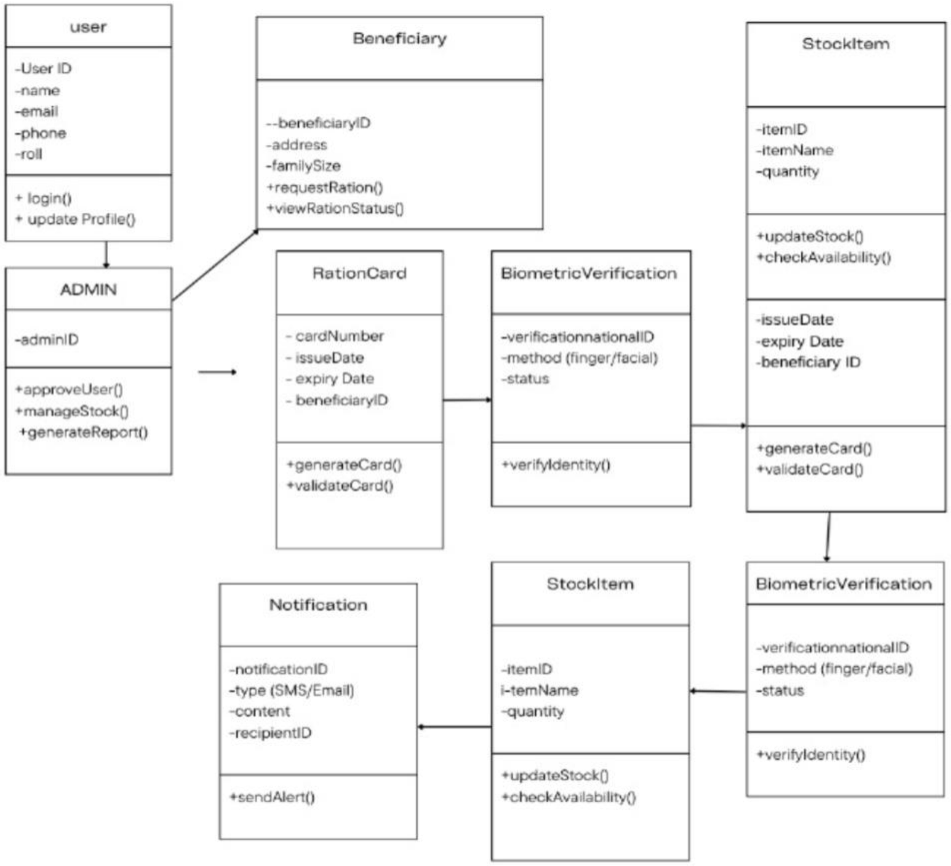
- In-app feedback for service quality and app performance.
- Periodic surveys for users and FPS dealers.
- Admins view reports to act on suggestions.

4. SOFTWARE DIAGRAM

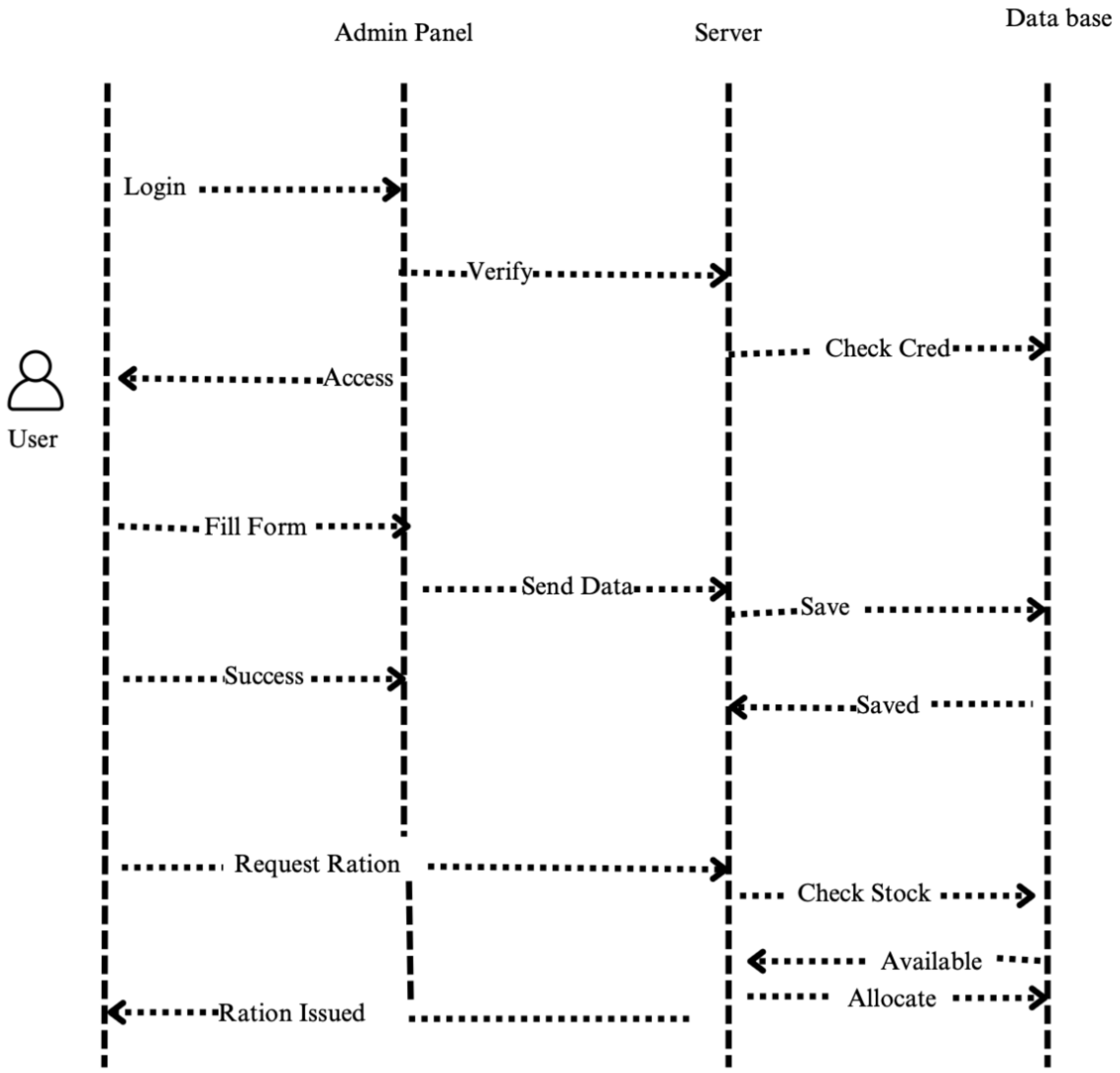
Use Case diagram



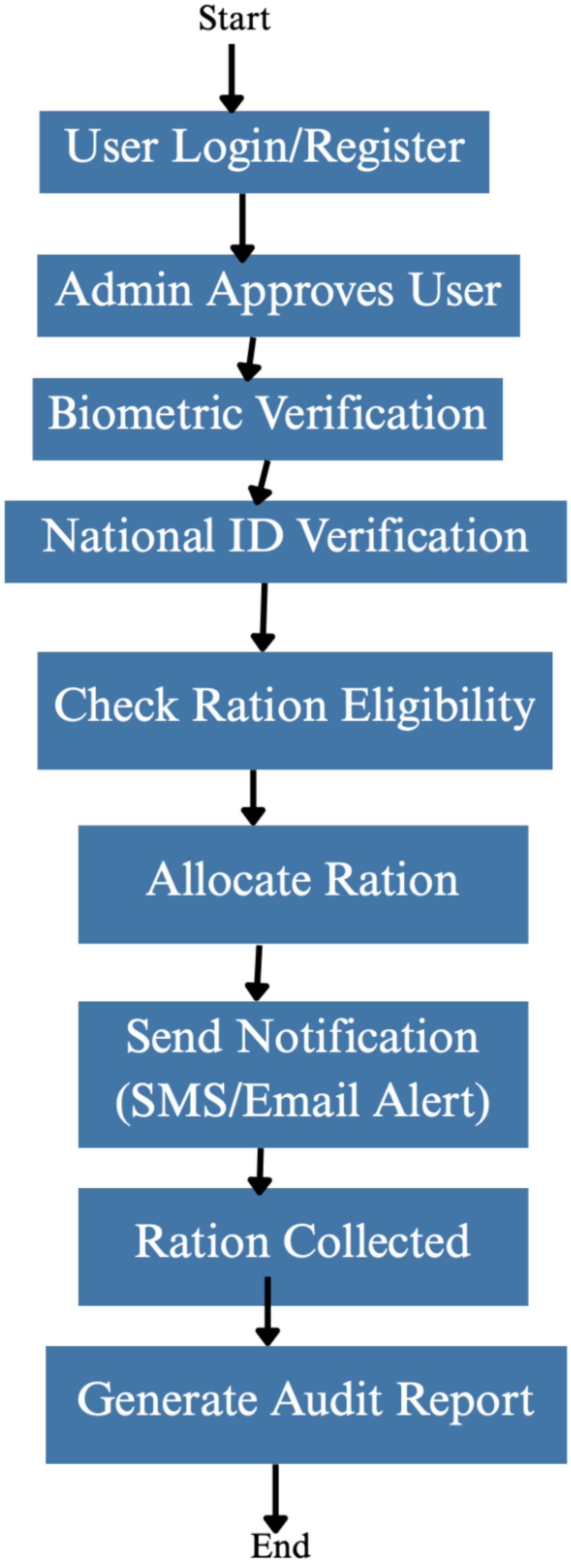
Class Diagram



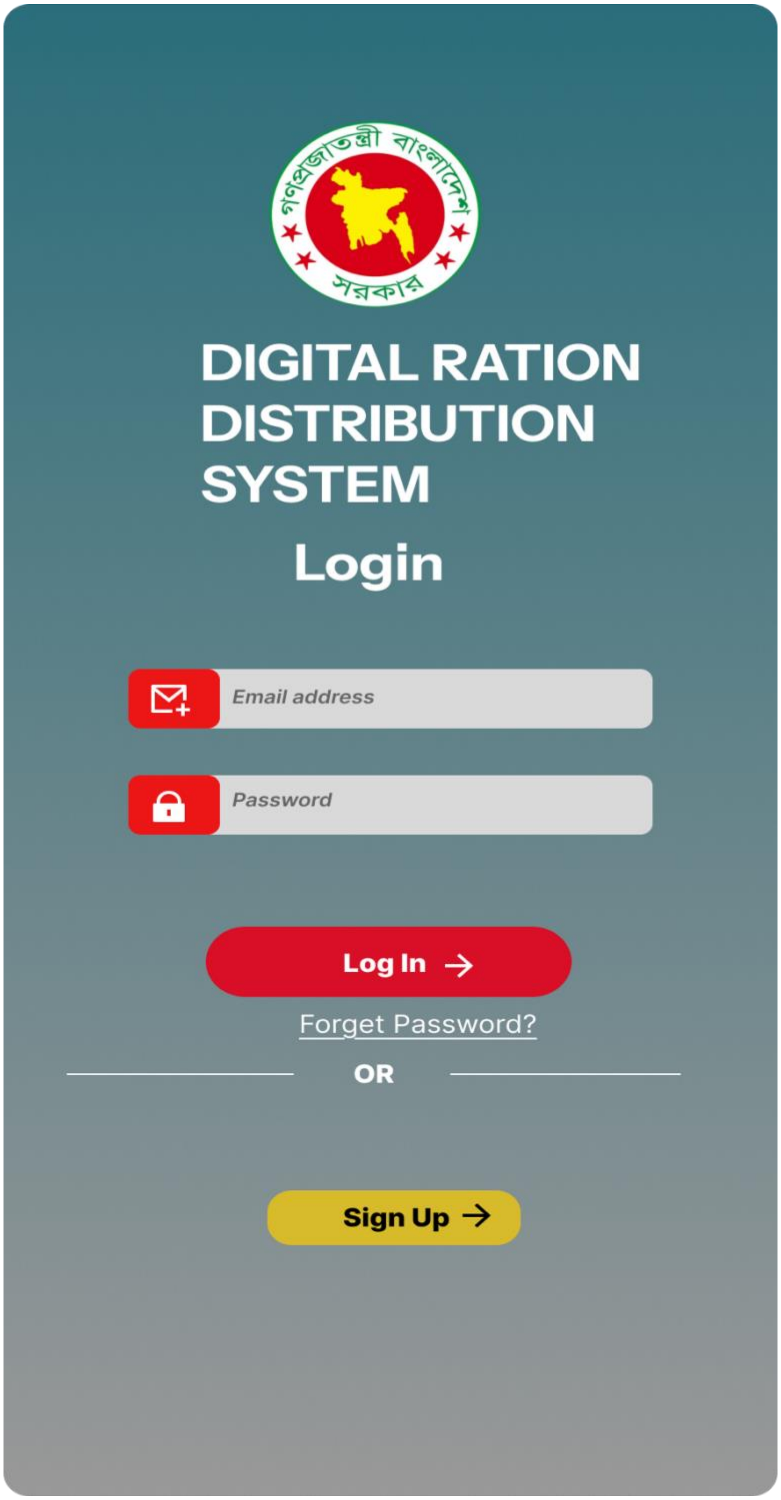
Sequence Diagram (User Registration & Ration Allocation)

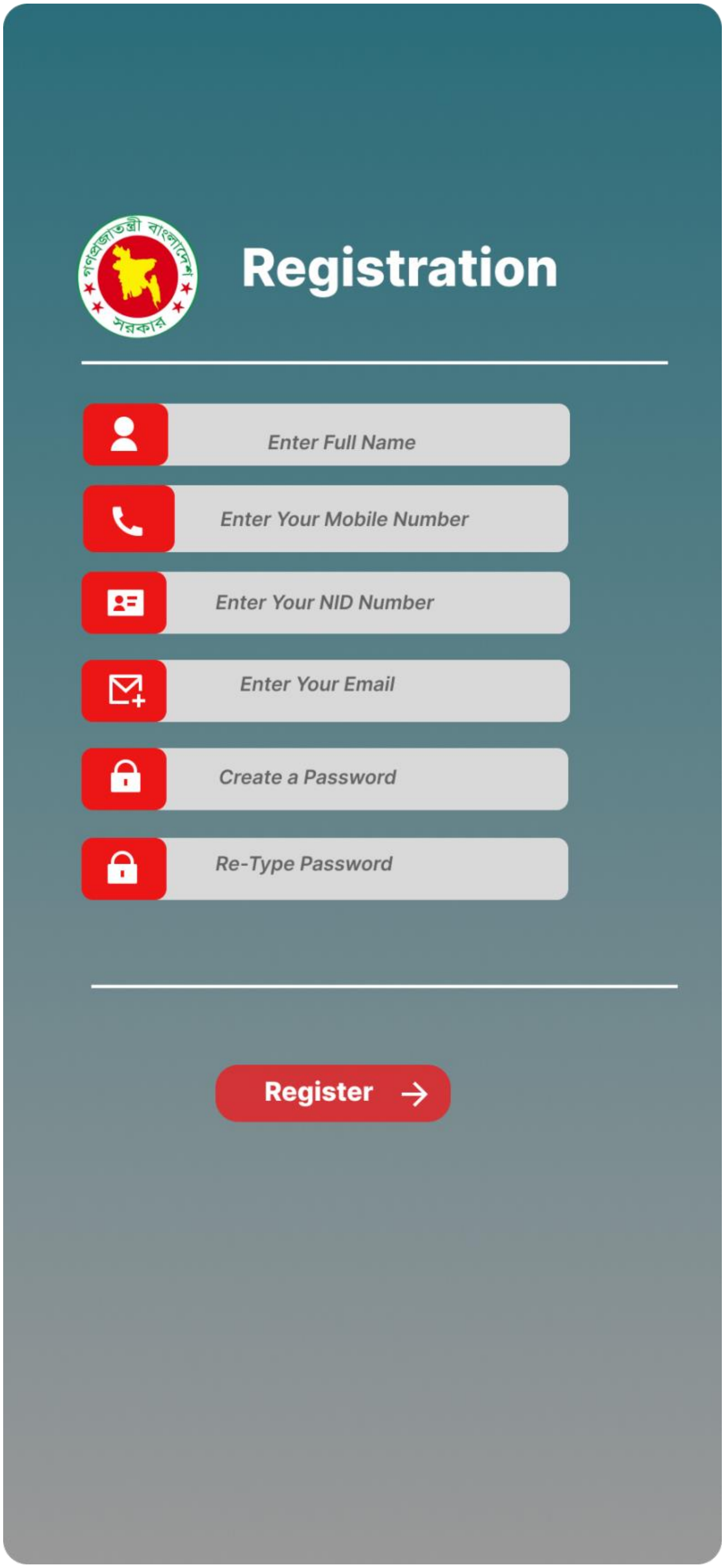


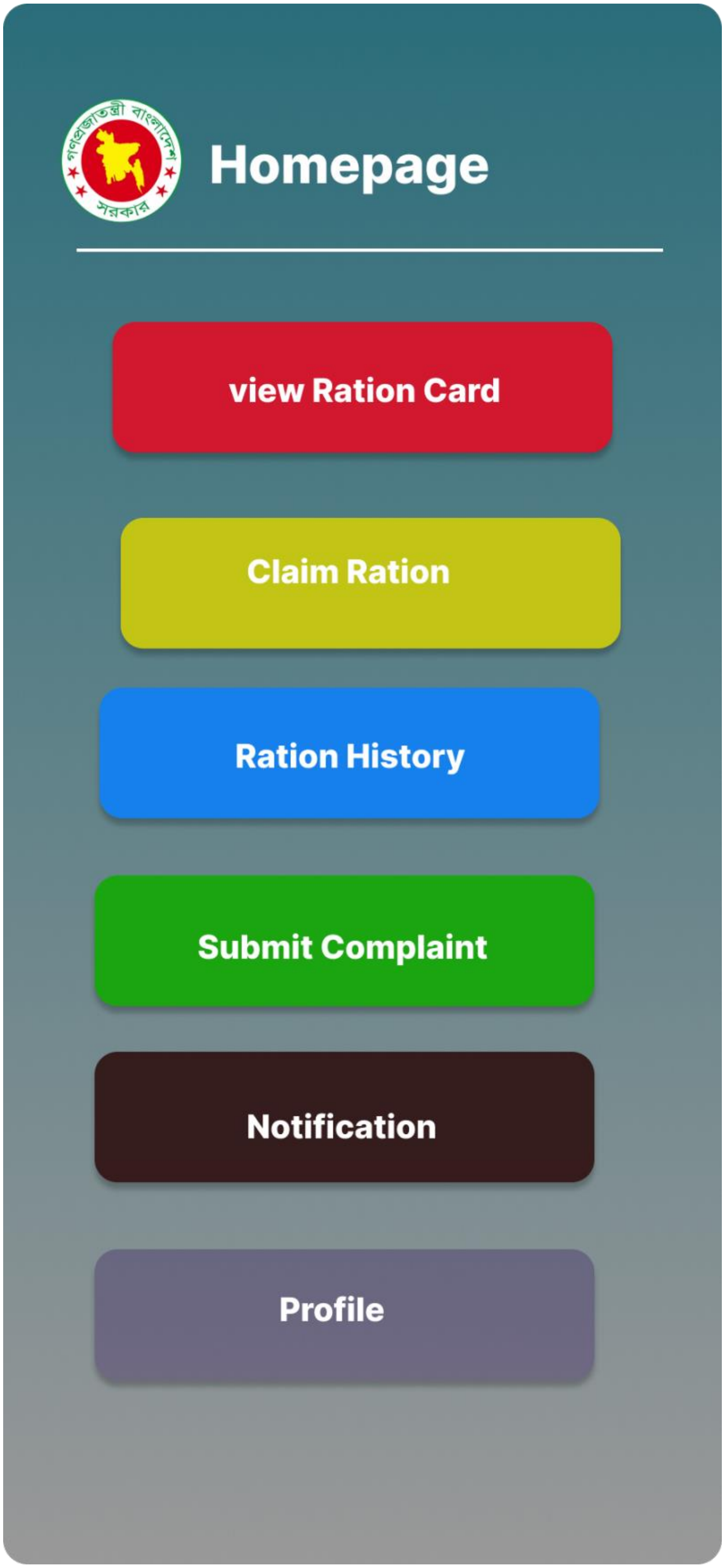
Activity Diagram(Digital Ration Workflow)

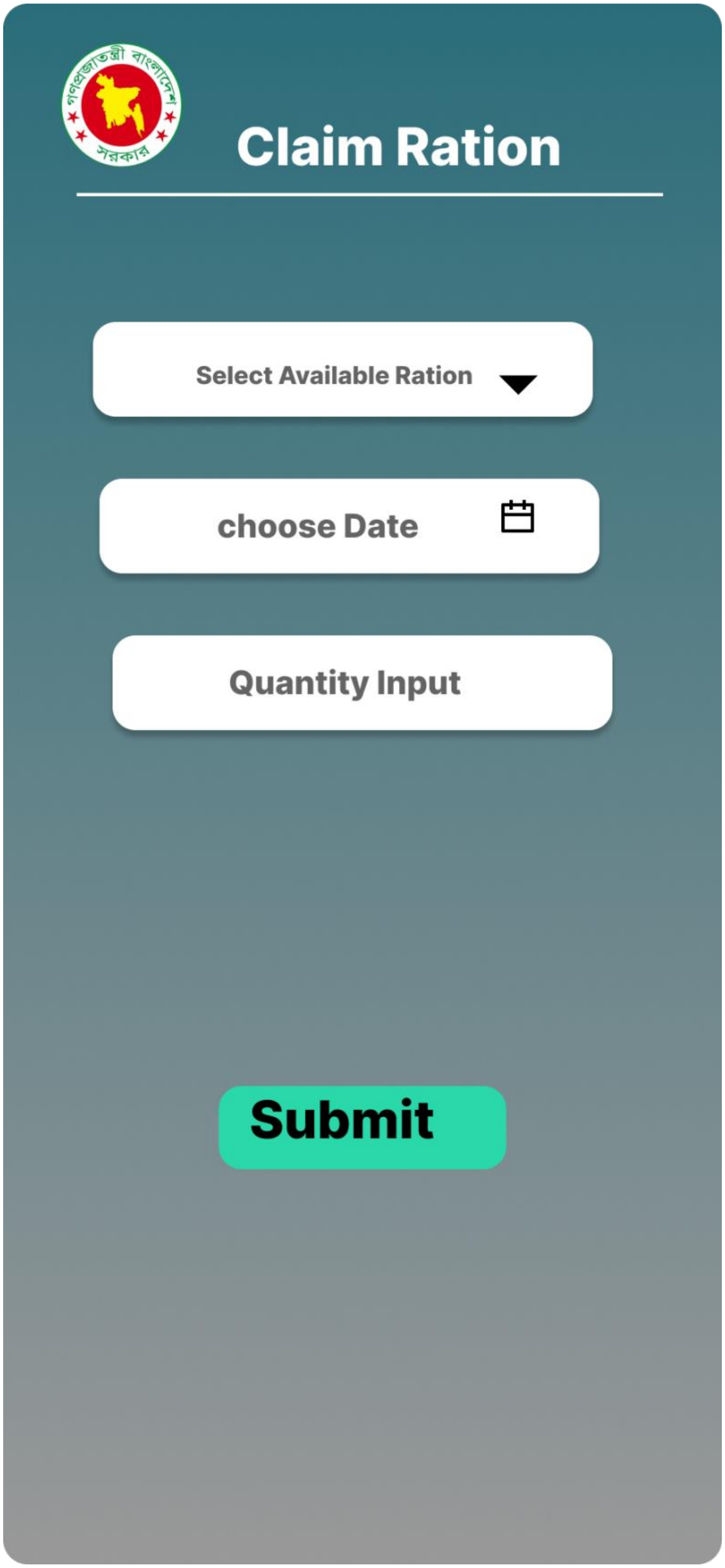


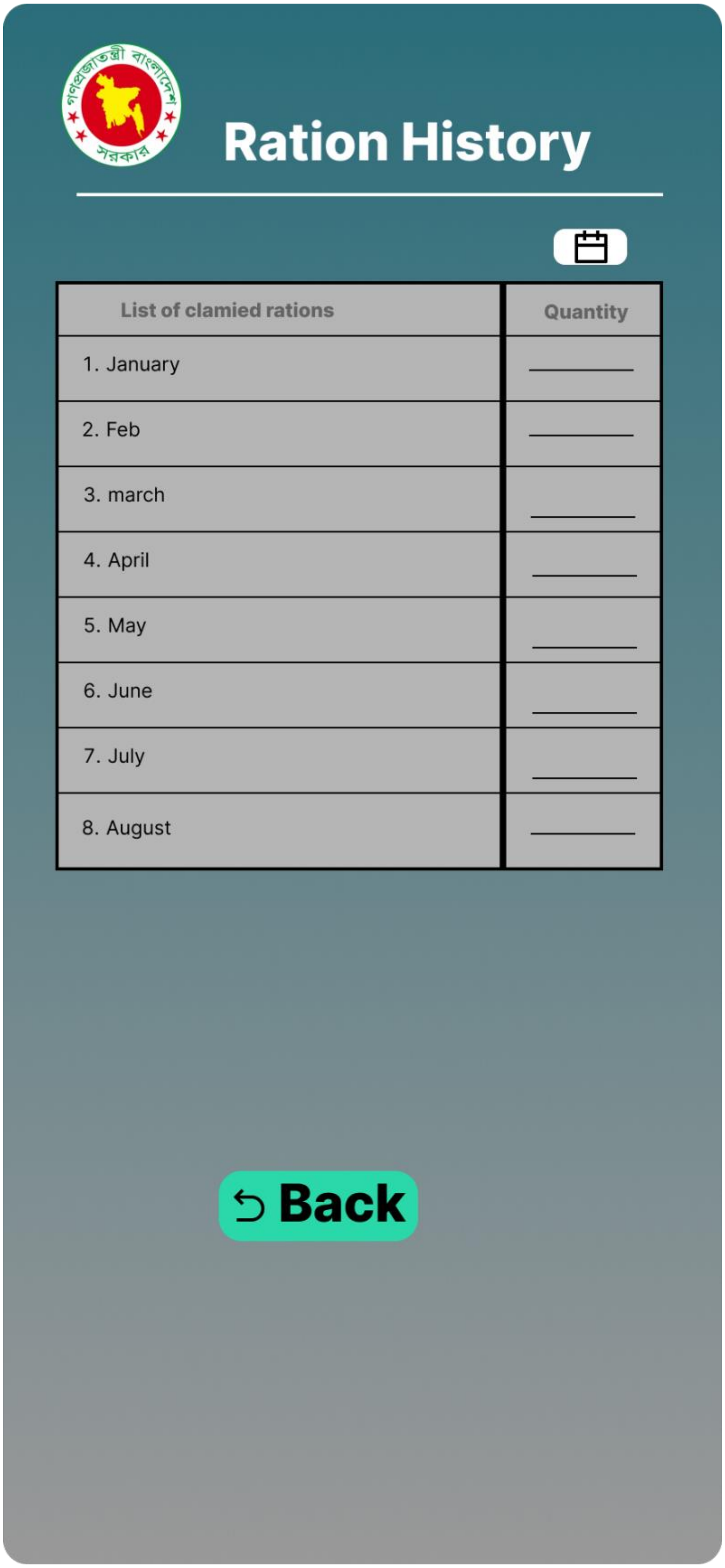
5. SOFTWARE DESIGN


গণপ্রজাতন্ত্রী বাংলাদেশ সরকার
DIGITAL RATION DISTRIBUTION SYSTEM
Login
Email address
Password
Log In →
Forget Password?
OR
Sign Up →


Registration
Enter Full Name
Enter Your Mobile Number
Enter Your NID Number
Enter Your Email
Create a Password
Re-Type Password
Register

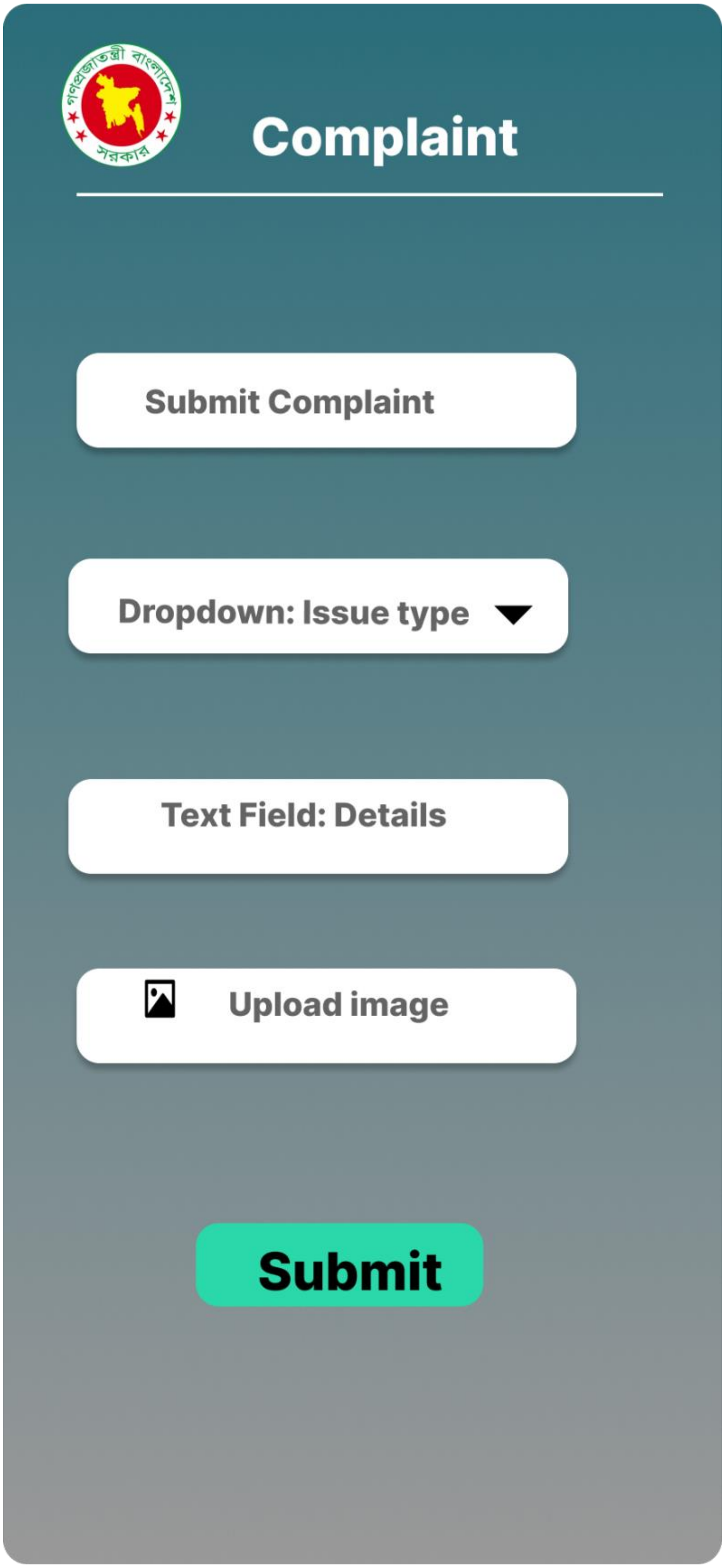

Homepage
view Ration Card
Claim Ration
Ration History
Submit Complaint
Notification
Profile

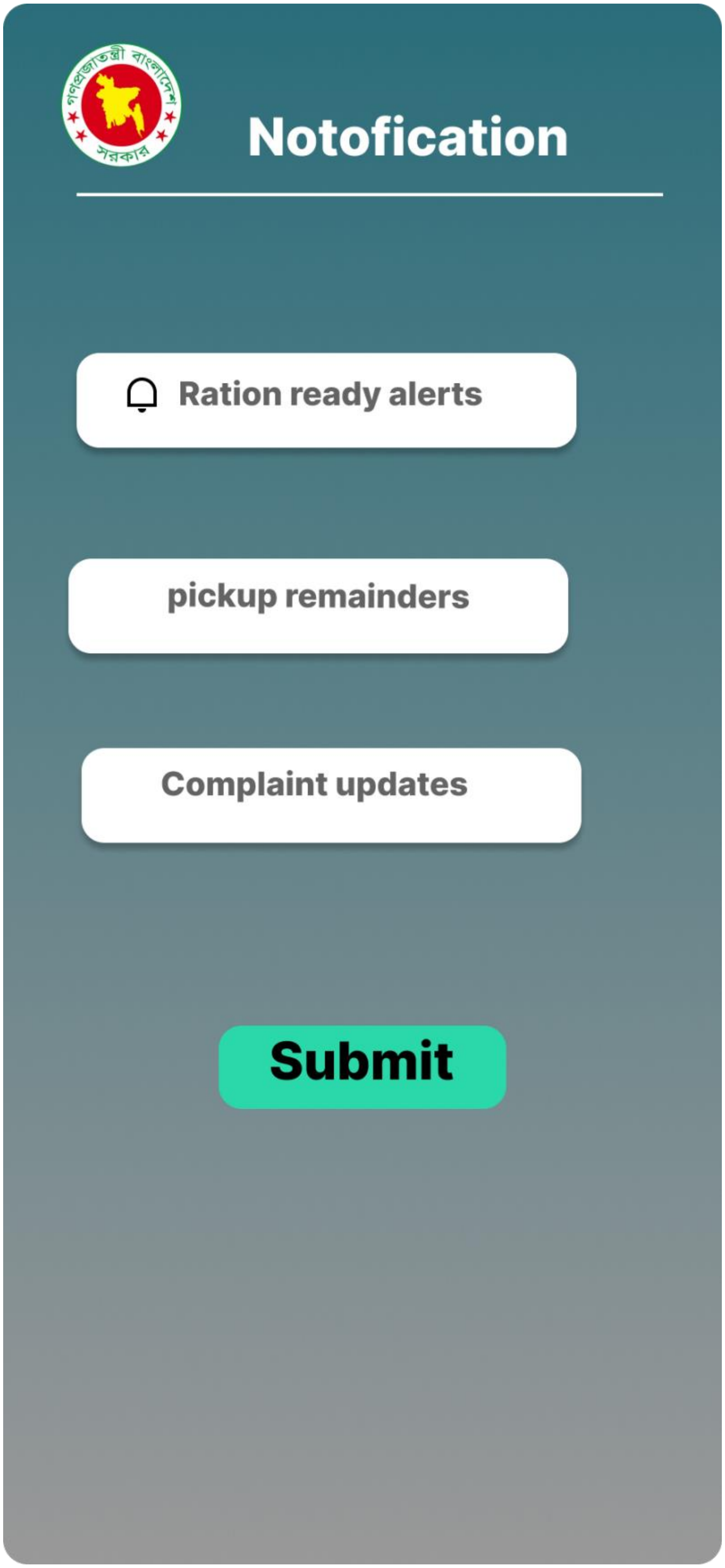

Claim Ration
Select Available Ration
choose Date
Quantity Input
Submit



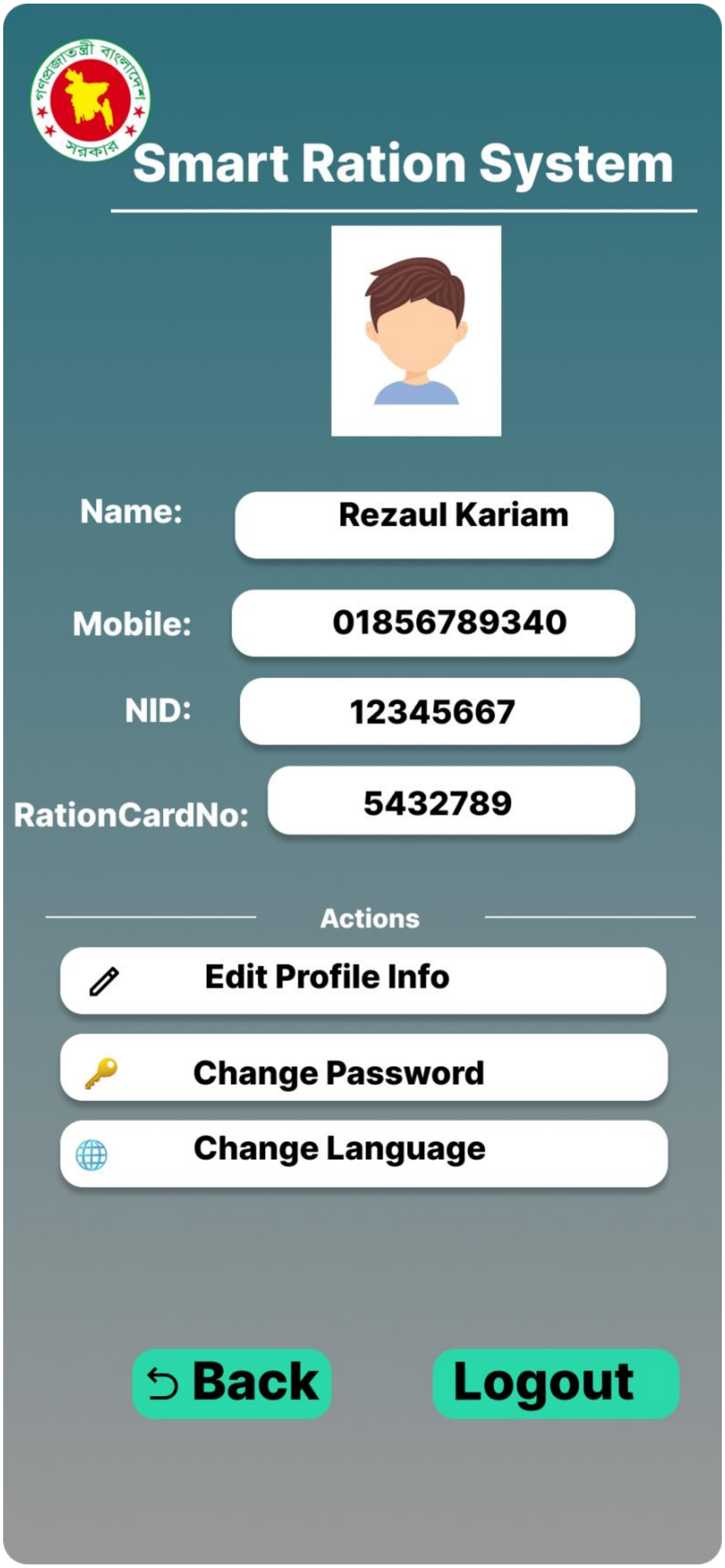
Ration History

List of clamied rations	Quantity
1. January	
2. Feb	
3. march	
4. April	
5. May	
6. June	
7. July	
8. August	


গণপ্রজাতন্ত্রী বাংলাদেশ সরকার
Complaint
Submit Complaint
Dropdown: Issue type
Text Field: Details
Upload image
Submit



Notofication



Smart Ration System

Name: Rezaul Kariam

Mobile: 01856789340

NID: 12345667

RationCardNo: 5432789

6. Test Plan and Test Case:

Unit Test: The process of unit testing involves assessing individual units or components of software to ensure they operate correctly in isolation, meeting specified requirements. This practice aids in early detection of defects, thereby reducing the cost of fixing issues and enhancing the overall quality and reliability of the software.

1. **Unit Test Planning:** Test Managers (3-week).
2. **Unit Test Case Implementation:** Carried out by Software Test Engineers or Quality Assurance Engineers within an 16-week timeframe.
3. **Unit Test Execution:** Conducted by Developers within a 15-week timeframe.
4. **Unit Test Documentation:** Developers are responsible for recording unit test results over a 15-week period.

System Test: System testing evaluates the cohesive functioning of all software components, a kin to ensuring all gears mesh effectively in a machine. This process verifies that the software behaves correctly in real-world scenarios and meets user requirements prior to release.

1.**Module and Subsystem Testing:** Performed by Software Test Engineers or Quality Assurance Engineers in a 14-week timeframe.
2.**Integration Testing:** Carried out by Software Test Engineers or Quality Assurance Engineersover a15 -week period.
3.**Acceptance Testing:** End-users or Stakeholders conduct acceptance testing over a 12 - week period.
4.**Defect Analysis and Reporting:** Led by Test Leads or Test Managers within a 14-week time frame.

Test case 1:

Project Name: Digital ration distribution system	**Test designed by:** Ritu rani Das
Test case Id: **Test priority:** High	**Test designed date:** 25.06.2025 **Test executed by:** Abir Ahmed Antor
Module name: Log in	**Test execution date:** 26.06.2025
Test title: Verify successful login with credentials.	
Description: This test case ensures that a registered user can successfully log in to the system using their correct phone number and password.	
Precondition (if any): User must have a valid, active account in the system.	

Test step:	**Test data:**	**Expected Result:**	**Actual Result:**	**Status (pass/fail):**
1. Navigate to the login page. 2. Enter valid phone number. 3. Enter correct password. 4. Click "Login".	Phone: 01785674340 Password: Test@123	The system validates the credentials and redirects the user to the homepage.	Login successful, redirected to homepage.	Pass

Test case 2:

<table>
<tr><td colspan="3"><b>Project Name:</b> Digital ration distribution system</td><td colspan="2"><b>Test designed by:</b> Ritu Rani Das</td></tr>
<tr><td colspan="3"><b>Test case Id:</b>

<b>Test priority:</b> High</td><td colspan="2"><b>Test designed date:</b> 25.06.2025

<b>Test executed by:</b> Abir Ahmed Antor</td></tr>
<tr><td colspan="3"><b>Module name:</b> User Registration</td><td colspan="2"><b>Test execution date:</b> 26.06.2025</td></tr>
<tr><td colspan="5"><b>Test title:</b> Verify new user registration with valid and unique information.</td></tr>
<tr><td colspan="5"><b>Description:</b> This test ensures that a user can successfully create an account with valid inputs.</td></tr>
<tr><td colspan="5"><b>Precondition (if any):</b> User is not registered in the system.</td></tr>
<tr><td><b>Test step:</b>

1. Go to Registration Page.
2. Fill all fields.
3. Click "Register".</td><td><b>Test data:</b>

Mobile: 01785674340, NID: 123456987, Ration Card: 5452789, Password: Pass@123</td><td><b>Expected Result:</b>

User successfully registered. Redirected to login page.</td><td><b>Actual Result:</b>

Registration successful<b>.</b></td><td><b>Status (pass/fail):</b>

Pass</td></tr>
</table>

Test case 3:

Project Name: Digital ration distribution system			**Test designed by:** Anirban Sarkar	
Test case Id: **Test priority:** High			**Test designed date:** 25.06.2025 **Test executed by:** Abir Ahmed Antor	
Module name: Claim Ration			**Test execution date:** 26.06.2025	
Test title: Verify that a user can claim ration using valid input.				
Description: This test checks that a user can select ration items and submit claims**.**				
Precondition (if any): User must be logged in.				
Test step:	**Test data:**	**Expected Result:**	**Actual Result:**	**Status (pass/fail):**
1. Go to Claim Ration page. 2. Select item. 3. Select date. 4. Enter quantity. 5. Click Submit.	Item: Rice, Qty: 10kg, Date: 27-06-2025	Confirmation appears: "Claim submitted." Data saved.	Claim submitted successfully.	Pass

Test case 4:

Project Name: Digital ration distribution system			**Test designed by:** Anirban Sarkar	
Test case Id: **Test priority:** Medium			**Test designed date:** 25.06.2025 **Test executed by:**A M Fozlul karim	
Module name: Ration History			**Test execution date:** 26.06.2025	
Test title: Verify that a user can view previously claimed rations**.**				
Description: Checks that a user can see their claim history.				
Precondition (if any): User has previously claimed ration.				
Test step:	**Test data:**	**Expected Result:**	**Actual Result:**	**Status (pass/fail):**
1. Go to Ration History.	N/A	Table displays past ration details.	History table shown correctly.	Pass

Test case 5:

Project Name: Digital ration distribution system				**Test designed by:** Abir Ahmed Antor
Test case Id: **Test priority:** High				**Test designed date:** 25.06.2025 **Test executed by:** A M Fozlul karim
Module name: Complaint Submission				**Test execution date:** 26.06.2025
Test title: Verify complaint submission with valid data.				
Description: This test case ensures that a user can submit a complaint with issue type, description, and optional image.				
Precondition (if any): User must be logged in.				
Test step:	**Test data:**	**Expected Result:**	**Actual Result:**	**Status (pass/fail):**
1.Navigate to Complaint Page. 2. Select an issue type. 3. Enter a description. 4. Upload optional image. 5. Click Submit.		Complaint submitted successfully. A success message is shown.	Complaint submitted and confirmation displayed.	Pass

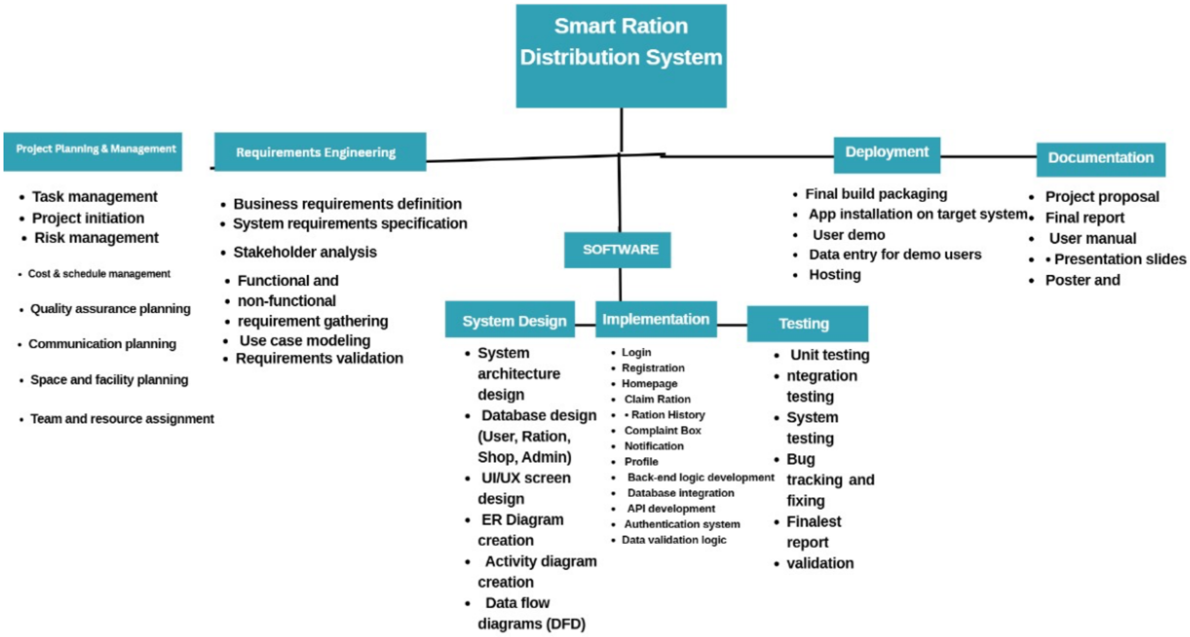
Test case 6:

<table>
<tr><td colspan="3"><b>Project Name:</b> Digital ration distribution system</td><td colspan="2"><b>Test designed by:</b> Abir Ahmed Antor</td></tr>
<tr><td colspan="3"><b>Test case Id:</b>
<b>Test priority:</b> Medium</td><td colspan="2"><b>Test designed date:</b> 25.06.2025
<b>Test executed by:</b> A M Fozlul karim</td></tr>
<tr><td colspan="3"><b>Module name:</b> Notification Module</td><td colspan="2"><b>Test execution date:</b> 26.06.2025</td></tr>
<tr><td colspan="5"><b>Test title:</b> Verify that notification messages appear correctly.</td></tr>
<tr><td colspan="5"><b>Description:</b> This test case checks that alerts like "Ration Ready", "Pickup Reminder" appear under Notifications.</td></tr>
<tr><td colspan="5"><b>Precondition (if any):</b> Notification data exists in backend for the user.</td></tr>
<tr><td><b>Test step:</b></td><td><b>Test data:</b></td><td><b>Expected Result:</b></td><td><b>Actual Result:</b></td><td><b>Status (pass/fail):</b></td></tr>
<tr><td>1. Login.
2. Navigate to Notification Page.</td><td>N/A</td><td>All relevant alerts are shown in a list.</td><td>All test notifications shown correctly.</td><td>Pass</td></tr>
</table>

Test case 7:

Project Name: Digital ration distribution system			**Test designed by:** Md Fahim Muntasir	
Test case Id: **Test priority:** High			**Test designed date:** 25.06.2025 **Test executed by:** A M Fozlul karim	
Module name: Profile Module			**Test execution date:** 26.06.2025	
Test title: Verify profile data visibility and edit options.				
Description: Ensures that users can view their personal information and access edit/change options.				
Precondition (if any): User must be logged in.				
Test step:	**Test data:**	**Expected Result:**	**Actual Result:**	**Status (pass/fail):**
1. Navigate to Profile Page. 2. View user info. 3. Click Edit Profile Info or Change Password/Language.	Name: Rezaul Karim, Mobile: 01785674340	Profile fields displayed. Edit options open respective dialogs.	All info shown. Buttons work as expected.	Pass

7. WBS


Smart Ration Distribution System
Project Planning & Management
• Task management
• Project initiation
• Risk management
• Cost & schedule management
• Quality assurance planning
• Communication planning
• Space and facility planning
• Team and resource assignment
Requirements Engineering
• Business requirements definition
• System requirements specification
• Stakeholder analysis
• Functional and
• non-functional
• requirement gathering
• Use case modeling
• Requirements validation
SOFTWARE
System Design
• System architecture design
• Database design (User, Ration, Shop, Admin)
• UI/UX screen design
• ER Diagram creation
• Activity diagram creation
• Data flow diagrams (DFD)
Implementation
• Login
• Registration
• Homepage
• Claim Ration
• • Ration History
• Complaint Box
• Notification
• Profile
• Back-end logic development
• Database integration
• API development
• Authentication system
• Data validation logic
Testing
• Unit testing
• ntegration testing
• System testing
• Bug tracking and fixing
• Finalest report
• validation
Deployment
• Final build packaging
• App installation on target system
• User demo
• Data entry for demo users
• Hosting
Documentation
• Project proposal
• Final report
• User manual
• • Presentation slides
• Poster and

8. Effort Estimation and Timeline:

Effort Estimation:

Our project is small, simple software project in which a small team with good application experience work in software development. As a result, the project is considered organic.

Constructive Cost Model (COCOMO):

Project Name: Smart Ration Distribution System

Group No: 3
Supervised by: Saeeda Sharmeen

Team Members:

Name	ID
Ritu Rani Das	22-49459-3
Abir Ahmed Antor	22-49460-3
Anirban Sarkar	22-49680-3
A.M. Fozlul Karim Sabbir	22-49677-3
M.D. Fahim Montasir	22-49400-3

Let’s consider:
SLOC (Source Lines of Code) = 18,000
Mode: Organic (simple & small team, in-house web system)

COCOMO Organic Mode Coefficients:

- Coefficient a=2.4a=2.4
- Exponent p=1.05p=1.05
- Time multiplier T=0.38T=0.38

Step 1: Effort (Person-Months)

Effort (PM)=2.4×(18000/1000)^1.05
=2.4×18^1.05
=2.4×20.79
=49.91 person-months

Step 2: Development Time (in months/weeks)

Dev Time = 2.5×(49.91)^0.38

= 2.5×4.418

= 11.04 weeks ≈ 3months

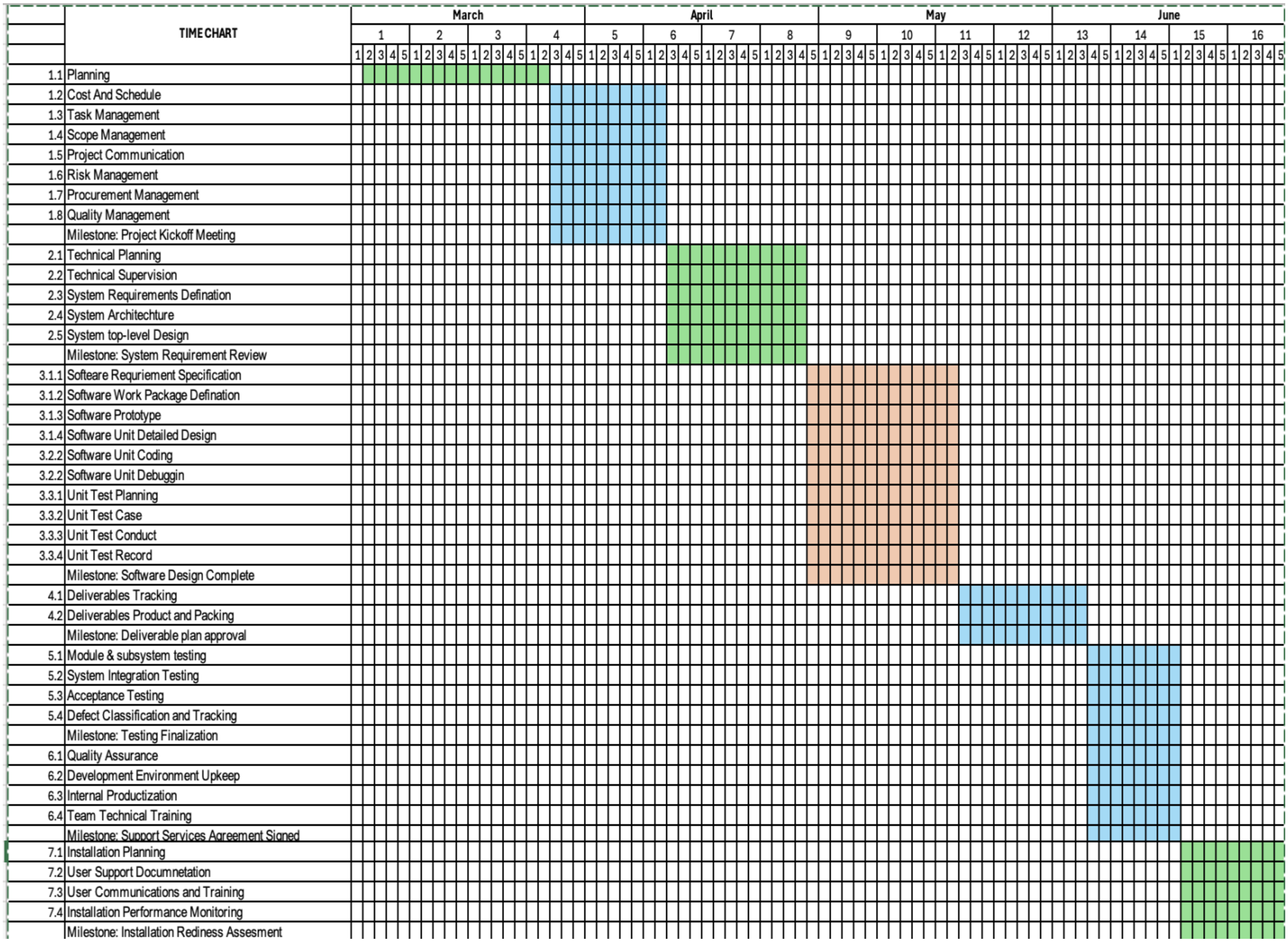
Step 3: Required People

Required People = Effort/Development Time

= 49.91/11.04

= 4.52≈5

9. Timeline chart



	TIME CHART	March																				April																				May																				June																			
		1					2					3					4					5					6					7					8					9					10					11					12					13					14					15					16				
		1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
1.1	Planning		🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩																																																															
1.2	Cost And Schedule																	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦																																																					
1.3	Task Management																	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦																																																					
1.4	Scope Management																	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦																																																					
1.5	Project Communication																	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦																																																					
1.6	Risk Management																	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦																																																					
1.7	Procurement Management																	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦																																																					
1.8	Quality Management																	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦																																																					
	Milestone: Project Kickoff Meeting																	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦																																																					
2.1	Technical Planning																											🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩																																									
2.2	Technical Supervision																											🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩																																									
2.3	System Requirements Defination																											🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩																																									
2.4	System Architechture																											🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩																																									
2.5	System top-level Design																											🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩																																									
	Milestone: System Requirement Review																											🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩																																									
3.1.1	Softeare Requriement Specification																																							🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧																											
3.1.2	Software Work Package Defination																																							🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧																											
3.1.3	Software Prototype																																							🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧																											
3.1.4	Software Unit Detailed Design																																							🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧																											
3.2.2	Software Unit Coding																																							🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧																											
3.2.2	Software Unit Debuggin																																							🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧																											
3.3.1	Unit Test Planning																																							🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧																											
3.3.2	Unit Test Case																																							🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧																											
3.3.3	Unit Test Conduct																																							🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧																											
3.3.4	Unit Test Record																																							🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧																											
	Milestone: Software Design Complete																																							🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧																											
4.1	Deliverables Tracking																																																				🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦																
4.2	Deliverables Product and Packing																																																				🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦																
	Milestone: Deliverable plan approval																																																				🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦																
5.1	Module & subsystem testing																																																															🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦								
5.2	System Integration Testing																																																															🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦								
5.3	Acceptance Testing																																																															🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦								
5.4	Defect Classification and Tracking																																																															🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦								
	Milestone: Testing Finalization																																																															🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦								
6.1	Quality Assurance																																																															🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦								
6.2	Development Environment Upkeep																																																															🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦								
6.3	Internal Productization																																																															🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦								
6.4	Team Technical Training																																																															🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦								
	Milestone: Support Services Agreement Signed																																																															🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦								
7.1	Installation Planning																																																																							🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
7.2	User Support Documnetation																																																																							🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
7.3	User Communications and Training																																																																							🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
7.4	Installation Performance Monitoring																																																																							🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
	Milestone: Installation Rediness Assesment																																																																							🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩

https://docs.google.com/spreadsheets/d/1-N2AOC40ULP-rpE7PpfhWZEbn127v6Xu/edit?usp=share_link&ouid=101948731966891919033&rtpof=true&sd=true

10. Risk Management :

ID	Risk Description	Impact	Response Strategy	Level	Owner	Notes
1	Data breach or security failure	High	Implement secure login (OTP/NID), encryption, and perform regular security audits.	High	Developer Team	Use HTTPS and secure database queries
2	Government policy changes	Medium	Design system with flexible configuration for policy update.	Medium	Product Owner	Keep track of Gov notifications
3	Server crash due to traffic spike	High	Load balancing, use cloud services, auto-scaling.	High	DevOps	Stress testing required
4	Untrained users (FPS/shop dealers)	Medium	Provide simple UI, training videos, and guides.	Medium	UI/UX + Trainer	Demo to shopkeepers
5	Incomplete or invalid user data	Medium	Add field validation, connect to NID database, and verify before approval.	Medium	Backend Developers	Use regex, API check for NID
6	Delay in task execution by team	Medium	Weekly SCRUM meetings, clear deadline tracking using Trello or Jira.	Medium	Scrum Master	Maintain daily stand-up
7	Lack of coordination among 5 members	Medium	Define clear roles and communicate over WhatsApp/Discord regularly.	Medium	All Team Members	Weekly progress reviews
8	Unexpected technical bugs	High	Code review, unit testing, fix during QA phase.	High	Developer & QA Tester	Use GitHub issues
9	Staff turnover or absence	Medium	Cross-training of modules among members.	Medium	Scrum Master	Document everything properly
10	Maintenance complexity post-delivery	High	Plan for long-term support, handover documentation and admin guide.	High	DevOps + Backend Team	Create proper release notes