Software Testing Document

CSE3112: Software Engineering Lab

App Name

Swapno - An Ecommerce App

Submitted By:

Name: Md Shamsur Rahman Sami

Roll No : 57

Name: Md Rakib Hossain

Roll No : 55

Submitted On :

May 15, 2024

Submitted To :

Dr. Saifuddin Md. Tareeq
Redwan Ahmed Rizvee

Contents

1 Unit Testing Results

User Management Subsystem:

- **Test Result:** Passed
- **Details:** All test cases related to user registration, login, authentication, and authorization functionalities were executed successfully. User creation, login, profile management, and password management functionalities were thoroughly tested and passed.

Product Management Subsystem:

- **Test Result:** Passed
- **Details:** The test cases for adding, editing, and deleting products were executed without errors. The subsystem successfully handled product management functionalities, ensuring accurate updates to the product catalog.

Cart Management Subsystem:

- **Test Result:** Passed
- **Details:** Test cases for adding products to the cart, updating cart quantities, and removing items from the cart were executed successfully. The subsystem effectively managed cart-related functionalities, providing a seamless shopping experience for users.

Order Management Subsystem:

- **Test Result:** Passed
- **Details:** The subsystem successfully handled order placement, order processing, and order tracking functionalities. Test cases related to placing orders, updating order statuses, and retrieving order details were executed without issues, ensuring efficient order management.

Payment Management Subsystem:

- **Test Result:** Passed
- **Details:** Test cases for processing payments, handling payment transactions securely, and generating payment invoices were executed successfully. The subsystem accurately managed payment-related functionalities, ensuring secure and reliable transactions.

1.1 Static Testing

1.1.1 Walkthrough

User Management Subsystem

Table 1: Static Testing Results for User Management Subsystem

Issue	**Comments**
Uninitialized Variables	No uninitialized variables found in the user management subsystem. All variables properly initialized and utilized according to the naming conventions.
Undocumented Empty Block	No undocumented empty blocks detected. All blocks are appropriately commented, indented, and documented, enhancing code readability.
No Effect Assignment	Optimized code to prevent redundant segments. Assignments meaningful and contribute to system functionality.
Code Guideline Violation	Adhered to MVC pattern guidelines, ensuring a modular structure with component reusability.
Code Anomalies	No significant anomalies detected. Secure JSON web token used for authentication and authorization.
Structural Anomalies	Maintained structural modularity throughout the system, reducing errors and promoting maintainability.

Product Management Subsystem

Table 2: Static Testing Results for Product Management Subsystem

Issue	**Comments**
Uninitialized Variables	No uninitialized variables found in the product management subsystem. All variables properly initialized and utilized according to the naming conventions.
Undocumented Empty Block	No undocumented empty blocks detected. All blocks are appropriately commented, indented, and documented, enhancing code readability.
No Effect Assignment	Optimized code to prevent redundant segments. Assignments meaningful and contribute to system functionality.
Code Guideline Violation	Adhered to MVC pattern guidelines, ensuring a modular structure with component reusability.
Code Anomalies	No significant anomalies detected. Efficient product management functionalities observed.
Structural Anomalies	Structurally modular with easy error localization. Maintained code cleanliness and separation of concerns.

Cart Management Subsystem

Table 3: Static Testing Results for Cart Management Subsystem

Issue	**Comments**
Uninitialized Variables	No uninitialized variables found in the cart management subsystem. All variables properly initialized and utilized according to the naming conventions.
Undocumented Empty Block	No undocumented empty blocks detected. All blocks are appropriately commented, indented, and documented, enhancing code readability.
No Effect Assignment	Optimized code to prevent redundant segments. Assignments meaningful and contribute to system functionality.
Code Guideline Violation	Adhered to coding guidelines and maintained the MVC pattern for a modular structure.
Code Anomalies	Free from significant anomalies. Efficient cart management functionalities observed.
Structural Anomalies	Structurally modular with easy error localization. Maintained code cleanliness and separation of concerns.

Order Management Subsystem

Table 4: Static Testing Results for Order Management Subsystem

Issue	**Comments**
Uninitialized Variables	No uninitialized variables found in the order management subsystem. All variables properly initialized and utilized according to the naming conventions.
Undocumented Empty Block	No undocumented empty blocks detected. All blocks are appropriately commented, indented, and documented, enhancing code readability.
No Effect Assignment	Optimized code to prevent redundant segments. Assignments meaningful and contribute to system functionality.
Code Guideline Violation	Adhered to coding guidelines and maintained the MVC pattern for a modular structure.
Code Anomalies	Free from significant anomalies. Efficient order management functionalities observed.
Structural Anomalies	Structurally modular with easy error localization. Maintained code cleanliness and separation of concerns.

Payment Management Subsystem

Table 5: Static Testing Results for Payment Management Subsystem

Issue	**Comments**
Uninitialized Variables	No uninitialized variables found in the payment management subsystem. All variables properly initialized and utilized according to the naming conventions.
Undocumented Empty Block	No undocumented empty blocks detected. All blocks are appropriately commented, indented, and documented, enhancing code readability.
No Effect Assignment	Optimized code to prevent redundant segments. Assignments meaningful and contribute to system functionality.
Code Guideline Violation	Adhered to coding guidelines and maintained the MVC pattern for a modular structure.
Code Anomalies	Free from significant anomalies. Payment process implemented accurately, including cash on delivery option.
Structural Anomalies	Structurally modular with easy error localization. Maintained code cleanliness and separation of concerns.

1.1.2 Code Review

1.1.2.1 Reviewer 1: Himel Roy: Swapno, your ecommerce application, demonstrates a strong commitment to software engineering principles and best practices, resulting in a well-structured and maintainable codebase. The combination of Flutter for the user app, Java for the admin app, and Firebase as the backend database showcases a modern and robust technology stack, ensuring scalability, reliability, and efficiency.

One notable aspect of your codebase is the effective implementation of the Model-View-Controller (MVC) architecture. This architectural pattern promotes code organization, separation of concerns, and maintainability, contributing to the overall quality and stability of your application.

Furthermore, the strategic use of services within your application enhances modularity, extensibility, and code reuse. Services encapsulate complex business logic, enabling easier maintenance and testing while ensuring a clear separation of concerns. This architectural decision reflects a comprehensive understanding of software engineering principles and fosters a scalable and maintainable codebase.

The choice of Firebase as the backend database solution is commendable, as it provides real-time data synchronization, robust security features, and scalability, which are essential for an ecommerce platform. The integration of Firebase enhances

1.1.2.2 Reviewer 2: Sudipto Das Sukanto: Swapno, your ecommerce application, showcases a well-implemented MVC architecture, leveraging the power of Flutter for the user app and Java for the admin app, with Firebase as the backend database. The codebase demonstrates a strong understanding of software engineering principles and follows best practices for maintainability and scalability.

One notable aspect of your codebase is the effective use of services, which encapsulate complex business logic and promote code reuse. By employing services, the codebase achieves modularity and extensibility, ensuring a clear separation of concerns and enabling easier maintenance and testing.

In summary, Swapno demonstrates a well-structured and maintainable architecture, leveraging modern technologies effectively. The emphasis on services and testing practices further enhances the codebase's quality and reliability. Great job in creating a scalable and robust ecommerce application!

1.2 Dynamic Testing

1.2.1 Black Box Testing

The input-output module has been tested with several test cases:

1.2.1.1 Range Partitioning

SignUpPage: Here is a table illustrating the test cases for the SignUpPage:

Test Case Identifier	1.2.1.1.1
Statement of Purpose	To test the signup page functionality with different input scenarios.
Description of Preconditions	The user is on the signup page.
Inputs	Various combinations of valid and invalid email addresses, passwords, and other required fields.
Expected Outputs	Successful signup with valid inputs; appropriate error messages for invalid inputs.
Description of Expected Postconditions	Upon successful signup, the user account is created.
Execution History	The signup page handles different input scenarios correctly.

LoginPage: To test the login page functionality with different input scenarios.

Test Case Identifier	1.2.1.1.2
Statement of Purpose	To test the login page functionality with different input scenarios.
Description of Preconditions	The user is on the login page.
Inputs	Various combinations of valid and invalid email addresses and passwords.
Expected Outputs	Successful login with valid credentials; appropriate error messages for invalid credentials.
Description of Expected Postconditions	Upon successful login, the user is authenticated and redirected to the homepage.
Execution History	The login page handles different input scenarios correctly.

HomePage: To test the functionality of the home page.

Test Case Identifier	1.2.1.1.3
Statement of Purpose	To test the functionality of the home page.
Description of Preconditions	The user is logged in and navigated to the home page.
Inputs	User interactions with various elements on the home page such as navigation links, product categories, and search functionality.
Expected Outputs	Proper rendering of the home page content, smooth navigation between sections, and accurate display of product listings.
Description of Expected Post-conditions	The user can browse the home page seamlessly without encountering any errors.
Execution History	The home page functions as expected during testing, providing a smooth user experience.

CartPage: To test the cart page functionality with different scenarios
(e.g., adding items, updating quantities, removing items).

Test Case Identifier	1.2.1.1.4
Statement of Purpose	To test the cart page functionality with different scenarios (e.g., adding items, updating quantities, removing items).
Description of Preconditions	The user has added items to the cart.
Inputs	Various actions such as adding items, updating quantities, and removing items from the cart.
Expected Outputs	The cart page should display the updated list of items with correct quantities and prices.
Description of Expected Post-conditions	The user can proceed to checkout or continue shopping.
Execution History	The cart page updates correctly based on user actions.

OrderPage: To test the order page functionality for placing orders.

Test Case Identifier	1.2.1.1.5
Statement of Purpose	To test the order page functionality for placing orders.
Description of Preconditions	The user has items in the cart and is on the order page.
Inputs	Various actions such as selecting delivery options, entering shipping details, and confirming the order.
Expected Outputs	Successful order placement with valid inputs; appropriate error messages for invalid inputs.
Description of Expected Post-conditions	Upon successful order placement, the user receives an order confirmation.
Execution History	The order page handles different input scenarios correctly.

PurchasePage: To test the purchase page functionality for completing transactions.

Test Case Identifier	1.2.1.1.6
Statement of Purpose	To test the purchase page functionality for completing transactions.
Description of Preconditions	The user has placed an order and is on the purchase page.
Inputs	Payment details such as credit card information, billing address, and security code.
Expected Outputs	Successful transaction completion with valid payment details; appropriate error messages for invalid details.
Description of Expected Post-conditions	Upon successful transaction, the user receives a payment confirmation.
Execution History	The purchase page handles different payment scenarios correctly.

ProfilePage: To test the profile page functionality for viewing and updating user information.

Test Case Identifier	1.2.1.1.7
Statement of Purpose	To test the profile page functionality for viewing and updating user information.
Description of Preconditions	The user is logged in and navigates to the profile page.
Inputs	User information such as name, email, contact details, and profile picture.
Expected Outputs	Successful update of user information with valid inputs; appropriate error messages for invalid inputs.
Description of Expected Post-conditions	The user profile reflects the updated information.
Execution History	The profile page allows users to view and update their information correctly.

ProductPage: To test the product page functionality for viewing product details and adding items to the cart.

Test Case Identifier	1.2.1.1.8
Statement of Purpose	To test the product page functionality for viewing product details and adding items to the cart.
Description of Preconditions	The user navigates to a specific product page.
Inputs	Interaction with product details such as size, color, quantity, and add-to-cart button.
Expected Outputs	Product details are displayed accurately; successful addition of items to the cart.
Description of Expected Post-conditions	The user can proceed to the cart with the selected items.
Execution History	The product page displays accurate product information and handles item addition correctly.

1.2.1.2 Set Partitioning

SignUpPage: Here is a table illustrating the test cases for the SignUpPage:

Test Case Identifier	1.2.1.2.1
Statement of Purpose	To test the signup page functionality with empty any field input scenarios.
Description of Preconditions	The user is on the signup page.
Inputs	name: ;email: sami@gmail.com;mobile: 01866362585; password: 123456
Expected Outputs	Unsuccessful signup with invalid inputs;
Description of Expected Post-conditions	Upon unsuccessful signup, the user account is not created.
Execution History	The signup page handles different input scenarios correctly.

Test Case Identifier	1.2.1.2.2
Statement of Purpose	To test the signup page functionality with invalid email input scenarios.
Description of Preconditions	The user is on the signup page.
Inputs	name:rahman sami ;email: sami;mobile: 01866362585; password: 123456
Expected Outputs	Unsuccessful signup with invalid inputs;
Description of Expected Post-conditions	Upon unsuccessful signup, the user account is not created.
Execution History	The signup page handles different input scenarios correctly.

Test Case Identifier	1.2.1.2.3
Statement of Purpose	To test the signup page functionality with different input scenarios.
Description of Preconditions	The user is on the signup page.
Inputs	name: md shamsur rahman sami;email: sami@gmail.com;mobile: 01866362585; password: 123456
Expected Outputs	Successful signup with valid inputs; appropriate error messages for invalid inputs.
Description of Expected Post-conditions	Upon successful signup, the user account is created.
Execution History	The signup page handles different input scenarios correctly.

LoginPage: To test the login page functionality with different input scenarios.

Test Case Identifier	1.2.1.2.4
Statement of Purpose	To test the login page functionality with invalid email input scenarios.
Description of Preconditions	The user is on the login page.
Inputs	email: sami;password: 123456
Expected Outputs	Unsuccessful login with invalid credentials;
Description of Expected Post-conditions	Upon unsuccessful login, the user is not authenticated and redirected to the homepage.
Execution History	The login page handles different input scenarios correctly.

Test Case Identifier	1.2.1.2.5
Statement of Purpose	To test the login page functionality with incorrect password input scenarios.
Description of Preconditions	The user is on the login page.
Inputs	email: sami@gmail.com;password: 123
Expected Outputs	Unsuccessful login with invalid credentials;
Description of Expected Post-conditions	Upon successful login, the user is not authenticated and redirected to the homepage.
Execution History	The login page handles different input scenarios correctly.

Test Case Identifier	1.2.1.2.6
Statement of Purpose	To test the login page functionality with correct input scenarios.
Description of Preconditions	The user is on the login page.
Inputs	email: sami@gmail.com;password: 123456
Expected Outputs	Successful login with valid credentials;
Description of Expected Post-conditions	Upon successful login, the user is authenticated and redirected to the homepage.
Execution History	The login page handles different input scenarios correctly.

HomePage: To test the functionality of the home page.

Test Case Identifier	1.2.1.2.7
Statement of Purpose	To test the functionality of the home page.
Description of Preconditions	The user is logged in and navigated to the home page.
Inputs	User interactions with various elements on the home page such as navigation links, product categories, and search functionality.
Expected Outputs	Proper rendering of the home page content, smooth navigation between sections, and accurate display of product listings.
Description of Expected Post-conditions	The user can browse the home page seamlessly without encountering any errors.
Execution History	The home page functions as expected during testing, providing a smooth user experience.

CartPage: To test the cart page functionality with different scenarios (e.g., adding items, updating quantities, removing items).

Test Case Identifier	1.2.1.2.8
Statement of Purpose	To test the cart page functionality with different scenarios (e.g., adding items, updating quantities, removing items).
Description of Preconditions	The user has added items to the cart.
Inputs	Various actions such as adding items, updating quantities, and removing items from the cart.
Expected Outputs	The cart page should display the updated list of items with correct quantities and prices.
Description of Expected Post-conditions	The user can proceed to checkout or continue shopping.
Execution History	The cart page updates correctly based on user actions.

OrderPage: To test the order page functionality for placing orders.

Test Case Identifier	1.2.1.2.9
Statement of Purpose	To test the order page functionality for placing orders.
Description of Preconditions	The user has items in the cart and is on the order page.
Inputs	Various actions such as selecting delivery options, entering shipping details, and confirming the order.
Expected Outputs	Successful order placement with valid inputs; appropriate error messages for invalid inputs.
Description of Expected Post-conditions	Upon successful order placement, the user receives an order confirmation.
Execution History	The order page handles different input scenarios correctly.

PurchasePage: To test the purchase page functionality for completing transactions.

Test Case Identifier	1.2.1.2.10
Statement of Purpose	To test the purchase page functionality for completing transactions.
Description of Preconditions	The user has placed an order and is on the purchase page.
Inputs	Payment details such as credit card information, billing address, and security code.
Expected Outputs	Successful transaction completion with valid payment details; appropriate error messages for invalid details.
Description of Expected Post-conditions	Upon successful transaction, the user receives a payment confirmation.
Execution History	The purchase page handles different payment scenarios correctly.

ProfilePage: To test the profile page functionality for viewing and updating user image.

Test Case Identifier	1.2.1.2.11
Statement of Purpose	To test the profile page functionality for viewing and updating user image.
Description of Preconditions	The user is logged in and navigates to the profile page.
Inputs	User information such as name, email, contact details, and profile picture.
Expected Outputs	Successful update of user image with valid inputs; appropriate error messages for invalid inputs.
Description of Expected Postconditions	The user profile reflects the updated image.
Execution History	The profile page allows users to view and update their image correctly.

ProductPage: To test the product page functionality for viewing product details and adding items to the cart.

Test Case Identifier	1.2.1.2.12
Statement of Purpose	To test the product page functionality for viewing product details and adding items to the cart.
Description of Preconditions	The user navigates to a specific product page.
Inputs	Interaction with product details such as size, color, quantity, and add-to-cart button.
Expected Outputs	Product details are displayed accurately; successful addition of items to the cart.
Description of Expected Postconditions	The user can proceed to the cart with the selected items.
Execution History	The product page displays accurate product information and handles item addition correctly.

1.2.2 White Box Testing

1.2.2.1 Code Coverage

Add Product to Cart Here is a table illustrating the test cases for the SignUpPage:

Test Case Identifier	1.2.2.1.2
Test Case Title	Add Product to Cart
Test Case Objective	Verify that users can successfully add products to their shopping cart.
Preconditions	- The Swapno e-commerce app is opened on a compatible device. - The user is logged in to their account.
Test Steps	1. Browse the product listings within the Swapno app. 2. Select a product of interest by tapping on its listing. 3. Click on the "Add to Cart" button or icon associated with the selected product.
Expected Results	- The selected product should be added to the user's shopping cart. - The cart icon or indicator should display the updated quantity of items. - Users should receive a confirmation message or notification indicating successful addition to the cart.
Test Result	- The selected product was successfully added to the user's shopping cart. - The cart icon accurately reflected the updated quantity of items. - A confirmation message was displayed, confirming the addition of the product to the cart.
Comments	This test case ensures that the functionality to add products to the shopping cart works as intended in the Swapno app. It validates a crucial step in the shopping process and ensures a seamless user experience.

Retrieve Product Listings Here is a table illustrating the test cases for the Retrieve Product Listings:

Test Case Identifier	1.3.0.1.2
Test Case Title	Retrieve Product Listings
Test Case Objective	Verify that users can successfully retrieve product listings in the Swapno e-commerce app.
Preconditions	- The Swapno e-commerce app is opened on a compatible device. - The user is logged in to their account.
Test Steps	1. Navigate to the product listings section within the Swapno app. 2. Browse through the available product categories or use the search functionality to find specific products. 3. View the details of individual products by tapping on their listings.
Expected Results	- The product listings should be displayed correctly, showing relevant information such as title, price, and images. - Users should be able to navigate through different product categories and search results smoothly. - Product details should be accurate and up-to-date.
Test Result	- The product listings were successfully retrieved and displayed in the app. - Navigation between product categories and search results worked smoothly. - Product details were accurate and matched the displayed listings.
Comments	This test case ensures that users can effectively explore and find products within the Swapno e-commerce app. It validates the functionality related to retrieving and displaying product listings, which is essential for a user-friendly shopping experience.

Place Order Here is a table illustrating the test cases for the Place Order functionality:

Test Case Identifier	1.3.0.1.3
Test Case Title	Place Order
Test Case Objective	Verify that users can successfully place orders for selected products in the Swapno e-commerce app.
Preconditions	- The Swapno e-commerce app is opened on a compatible device. - The user is logged in to their account. - The user has added products to their shopping cart.
Test Steps	1. Navigate to the shopping cart section within the Swapno app. 2. Review the selected products and their quantities. 3. Proceed to checkout and enter the required shipping and payment information. 4. Confirm the order details and finalize the purchase.
Expected Results	- The order should be successfully placed, and the user should receive an order confirmation. - The selected products should be removed from the shopping cart. - The user's account should reflect the placed order and updated order history.
Test Result	- The order was successfully placed, and an order confirmation was received. - The selected products were removed from the shopping cart. - The user's account accurately reflected the placed order in the order history.
Comments	This test case ensures that users can seamlessly complete the checkout process and place orders for selected products. It validates the functionality related to order placement, which is a critical aspect of the e-commerce platform's functionality.

View Order History Here is a table illustrating the test cases for the View Order History functionality:

Test Case Identifier	1.3.0.1.4
Test Case Title	View Order History
Test Case Objective	Verify that users can view their order history in the Swapno e-commerce app.
Preconditions	- The Swapno e-commerce app is opened on a compatible device. - The user is logged in to their account.
Test Steps	1. Navigate to the order history section within the Swapno app. 2. View the list of past orders made by the user. 3. Select a specific order to view its details.
Expected Results	- The user should be able to see a list of their past orders, including order numbers, dates, and order statuses. - Detailed information about each order, including product details, quantities, and total prices, should be available for viewing.
Test Result	- The user could view their order history, including all past orders with relevant details. - Detailed information about each order was accessible, allowing the user to review their purchase history.
Comments	This test case ensures that users have access to their order history within the Swapno e-commerce app. It validates the functionality related to viewing past orders, which is essential for tracking purchases and managing orders effectively.

Update Profile image Here is a table illustrating the test cases for updating profile image in the Swapno e-commerce app:

Test Case Identifier	1.3.0.1.5
Test Case Title	Update Profile image
Test Case Objective	Verify that users can successfully update their profile image in the Swapno e-commerce app.
Preconditions	- The Swapno e-commerce app is opened on a compatible device. - The user is logged in to their account.
Test Steps	1. Navigate to the profile section within the Swapno app. 2. Access the profile settings or edit image option. 3. Update the desired profile image.
Expected Results	- The user's profile image should be successfully updated with the new changes. - Any modifications made to the profile image should be reflected accurately.
Test Result	- The user's profile image was successfully updated with the new changes. - All modifications made to the profile were accurately reflected, ensuring that the user's information was up-to-date.
Comments	This test case ensures that users have the ability to update their profile image within the Swapno e-commerce app. It validates the functionality related to profile management, which is crucial for maintaining accurate user image and preferences.

1.2.2.2 Branch Coverage

Add Product to Cart Here is a table illustrating the test cases for the Add Product to Cart functionality with branch coverage:

Test Case Identifier	1.2.2.1.2
Test Case Title	Add Product to Cart
Test Case Objective	Verify that users can successfully add products to their shopping cart.
Preconditions	- The Swapno e-commerce app is opened on a compatible device. - The user is logged in to their account.
Test Steps	1. Browse the product listings within the Swapno app. 2. Select a product of interest by tapping on its listing. 3. Click on the "Add to Cart" button or icon associated with the selected product.
Expected Results	- The selected product should be added to the user's shopping cart. - The cart icon or indicator should display the updated quantity of items. - Users should receive a confirmation message or notification indicating successful addition to the cart.
Test Result	- The selected product was successfully added to the user's shopping cart. - The cart icon accurately reflected the updated quantity of items. - A confirmation message was displayed, confirming the addition of the product to the cart.
Comments	This test case ensures that the functionality to add products to the shopping cart works as intended in the Swapno app. It validates a crucial step in the shopping process and ensures a seamless user experience.

1.2.2.3 Condition Coverage

Retrieve Product Listings Here is a table illustrating the test cases for the Retrieve Product Listings functionality with condition coverage:

Test Case Identifier	1.3.0.1.2
Test Case Title	Retrieve Product Listings
Test Case Objective	Verify that users can successfully retrieve product listings in the Swapno e-commerce app.
Preconditions	- The Swapno e-commerce app is opened on a compatible device. - The user is logged in to their account.
Test Steps	1. Navigate to the product listings section within the Swapno app. 2. Browse through the available product categories or use the search functionality to find specific products. 3. View the details of individual products by tapping on their listings.
Expected Results	- The product listings should be displayed correctly, showing relevant information such as title, price, and images. - Users should be able to navigate through different product categories and search results smoothly. - Product details should be accurate and up-to-date.
Test Result	- The product listings were successfully retrieved and displayed in the app. - Navigation between product categories and search results worked smoothly. - Product details were accurate and matched the displayed listings.
Comments	This test case ensures that users can effectively explore and find products within the Swapno e-commerce app. It validates the functionality related to retrieving and displaying product listings, which is essential for a user-friendly shopping experience.

1.2.2.4 Path Coverage for Swapno E-commerce App (Single Vendor) Achieving full path coverage for every feature in Swapno an e-commerce application can be very complex. This document outlines potential test cases to illustrate path coverage for some core functionalities in Swapno (single vendor).

User Login & Registration

1. Valid Login (existing user with correct credentials)
2. Invalid Login (incorrect username or password)
3. New User Registration (successful registration with valid data)

4. Incomplete Registration (missing required fields)

Browse Products

1. Successful browsing of product categories (user can navigate and view different categories)
2. Search Functionality (user searches for a specific product using keywords and gets relevant results)
3. Product Details (user taps on a product listing and sees detailed information)
4. Out-of-Stock Product (user tries to view details of an out-of-stock product and sees appropriate message)

Add to Cart

1. Adding In-Stock Product (user adds a product to their cart successfully)
2. Adding Multiple Products (user adds several different products to their cart)
3. Adding Out-of-Stock Product (user tries to add an out-of-stock product and receives an error message)
4. Update Cart Quantity (user increases or decreases the quantity of a product in the cart)
5. Remove from Cart (user removes a product from their cart)

Checkout Process

1. Initiate Checkout (user proceeds to checkout from the shopping cart)
2. Logged-in User Checkout (user completes checkout with their saved shipping information)
3. Payment Processing (user successfully completes payment using a valid payment method)
4. Payment Failure (user's payment is declined and receives an error message)

Order History

1. View Order History (logged-in user views a list of their past orders)
2. Order Statuses (user can see the current status of their orders - placed, Pending, In Progress, Delivered)

Profile Management (Optional)

1. Update Profile Information (user edits their profile picture)
2. Change Profile Picture (user uploads a new profile picture)

Note: This is not an exhaustive list, and additional test cases can be created based on specific functionalities within the Swapno app.

Key Considerations for Path Coverage

- **Error Handling:** Ensure your test cases explore scenarios that might trigger errors or exceptions in the code (e.g., network errors during checkout).
- **Negative Testing:** Include test cases that cover invalid inputs and unexpected user actions.
- **Branching Points:** Identify conditional statements, loops, and function calls in the code and design tests that cover all possible outcomes based on those branches.

2 Integration Testing

At this stage, we conducted meticulous testing of subsystem interfaces and module interactions to identify and localize any errors or issues. Using a top-down approach, we evaluated modules in descending order of hierarchy, following the data flow from higher to lower priority subsystems. This comprehensive methodology ensured a rigorous and professional evaluation process.

2.1 Bottom Up Testing

Not applicable.

2.2 Top Down Testing

Test Case Number	**Statement of Purpose**	**Interface(s) Being Tested**
2.2.1	Validate the integration between the Product Search Interface and Product Database Interface.	Product Search Interface, Product Database Interface
2.2.2	Validate the integration between the Cart Interface and Checkout Interface.	Cart Interface, Checkout Interface
2.2.3	Validate the integration between the User Authentication Interface and Profile Management Interface.	User Authentication Interface, Profile Management Interface
2.2.4	Validate the integration between the Order History Interface and Payment Interface.	Order History Interface, Payment Interface

2.2.1 Test case 1

Test Case Number	2.2.1
Statement of Purpose	Validate the integration between the Product Search Interface and Product Database Interface.
Interface(s) Being Tested	Product Search Interface, Product Database Interface.
Description of Precondition	The Product Search Interface and Product Database Interface are integrated and functional.
Test Case Inputs	`{ "keywords": "banana", "category": "Fruits", }`
Method Stub	N/A
Expected Output	A list of available banana in the Fruits category
Description of Expected Post-conditions	The Product Search Interface successfully retrieves the list of available banana by querying the Product Database Interface.
Execution History	The `searchProducts()` method in the Product Search Interface is invoked with the given inputs. The method internally communicates with the Product Database Interface to retrieve the relevant product data by the `queryProducts` method. The Product Database Interface processes the query and returns the list of available bananas, which is then presented by the Product Search Interface to the user.

2.2.2 Test case 2

Test Case Number	2.2.2
Statement of Purpose	Validate the integration between the Cart Interface and Checkout Interface.
Interface(s) Being Tested	Cart Interface, Checkout Interface.
Description of Precondition	The Cart Interface and Checkout Interface are integrated and functional.
Test Case Inputs	N/A
Method Stub	N/A
Expected Output	Successful transition from the Cart Interface to the Checkout Interface with the selected products.
Description of Expected Post-conditions	The Checkout Interface displays the selected products from the Cart Interface and proceeds with the checkout process.
Execution History	The user selects products in the Cart Interface and initiates the checkout process. The Cart Interface communicates with the Checkout Interface to transfer the selected products. The Checkout Interface displays the selected products and allows the user to proceed with payment.

2.2.3 Test case 3

Test Case Number	2.2.3
Statement of Purpose	Validate the integration between the User Authentication Interface and Profile Management Interface.
Interface(s) Being Tested	User Authentication Interface, Profile Management Interface.
Description of Precondition	The User Authentication Interface and Profile Management Interface are integrated and functional.
Test Case Inputs	N/A
Method Stub	N/A
Expected Output	Successful user authentication and access to profile management features.
Description of Expected Post-conditions	The user successfully logs in through the User Authentication Interface and gains access to profile management functionalities.
Execution History	The user enters login credentials in the User Authentication Interface. The User Authentication Interface verifies the credentials and grants access to the Profile Management Interface upon successful authentication.

2.2.4 Test case 4

Test Case Number	2.2.4
Statement of Purpose	Validate the integration between the Order History Interface and Payment Interface.
Interface(s) Being Tested	Order History Interface, Payment Interface.
Description of Precondition	The Order History Interface and Payment Interface are integrated and functional.
Test Case Inputs	N/A
Method Stub	N/A
Expected Output	Successful payment processing for an order displayed in the Order History Interface.
Description of Expected Post-conditions	The payment for the selected order is successfully processed through the Payment Interface.
Execution History	The user selects an order in the Order History Interface for which payment is pending. The Order History Interface communicates with the Payment Interface to process the payment. Upon successful payment processing, the status of the order is updated in the Order History Interface.

2.3 Sandwich Testing

Not applicable.

3 System Testing

System testing is done to monitor how each component of the system is behaving.

3.1 Functional Testing

3.1.1 Requirement 1 : Admin Dashboard

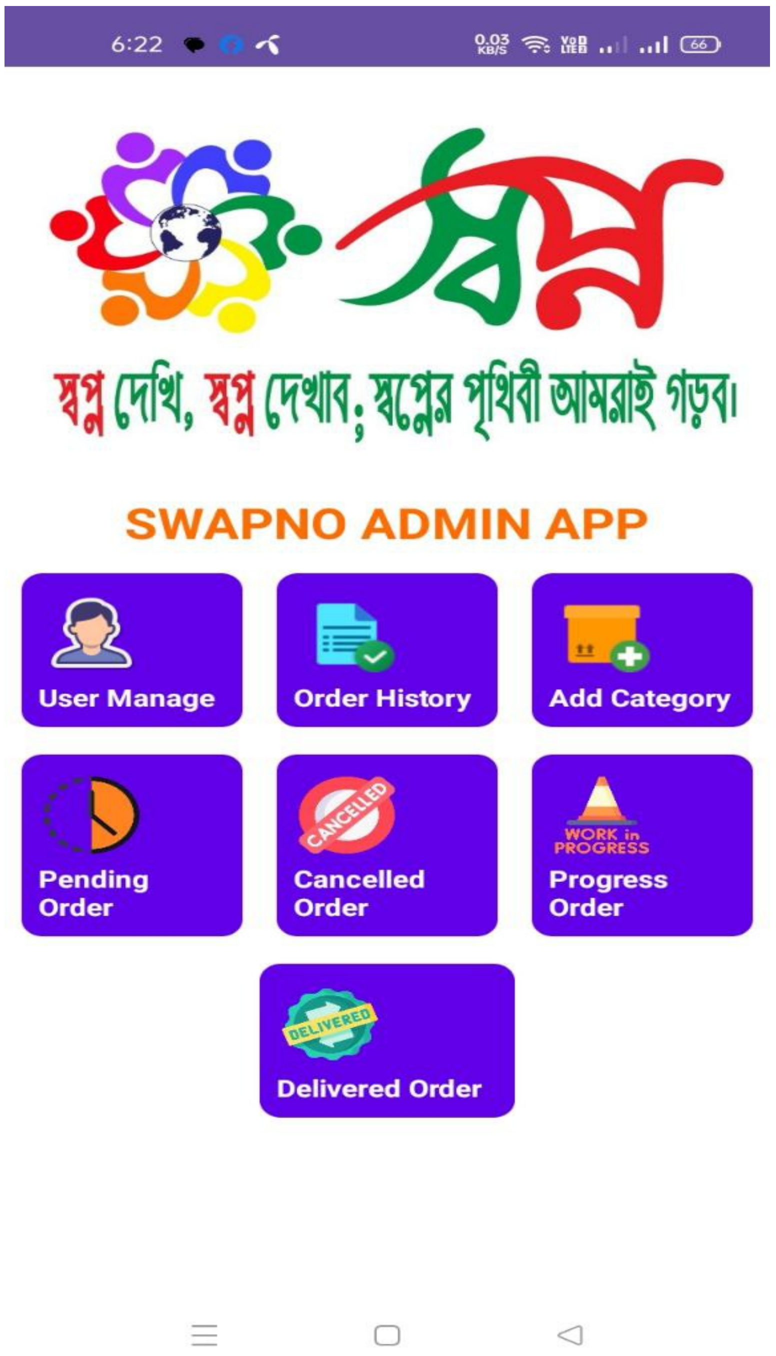
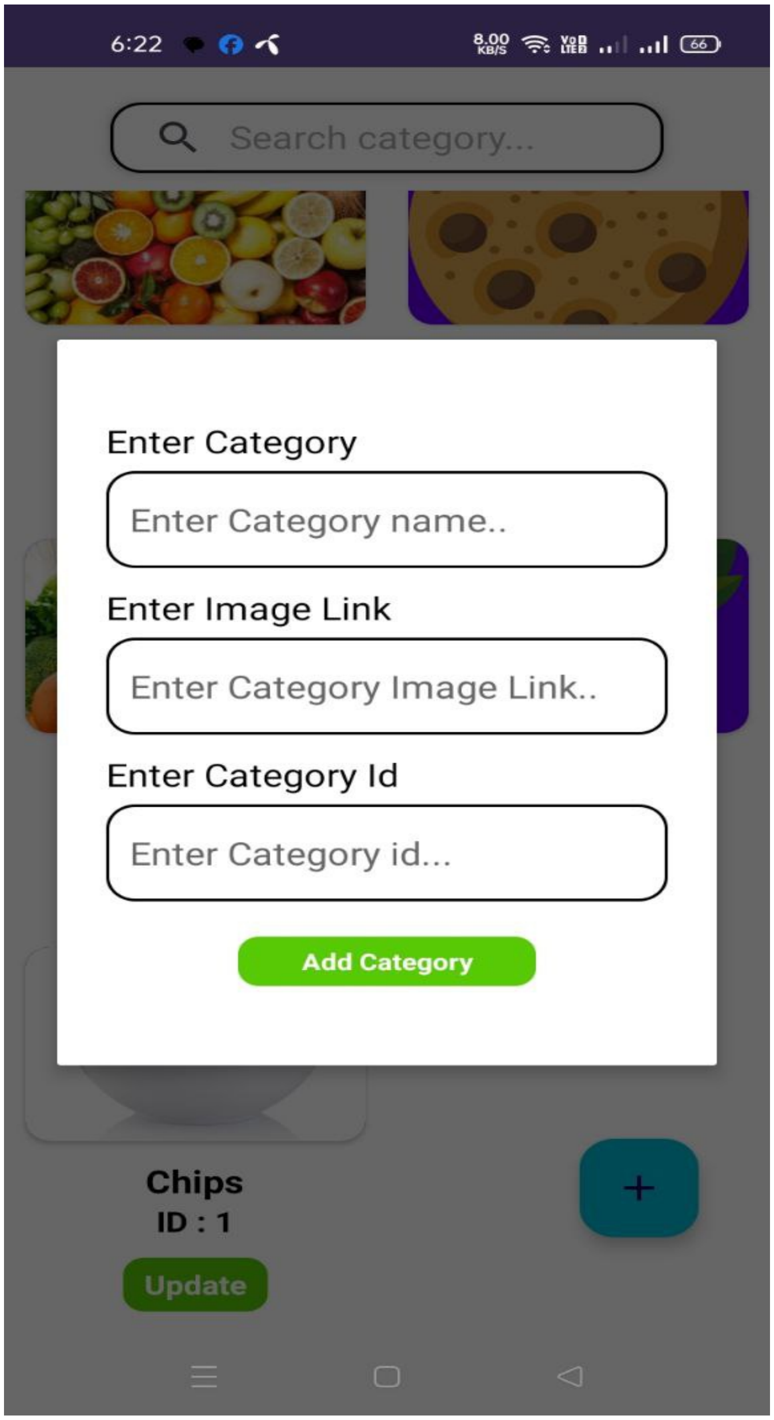


Figure 1: Admin dashboard

3.1.2 Requirement 2 : Add Category



Enter Category

Enter Category name..

Enter Image Link

Enter Category Image Link..

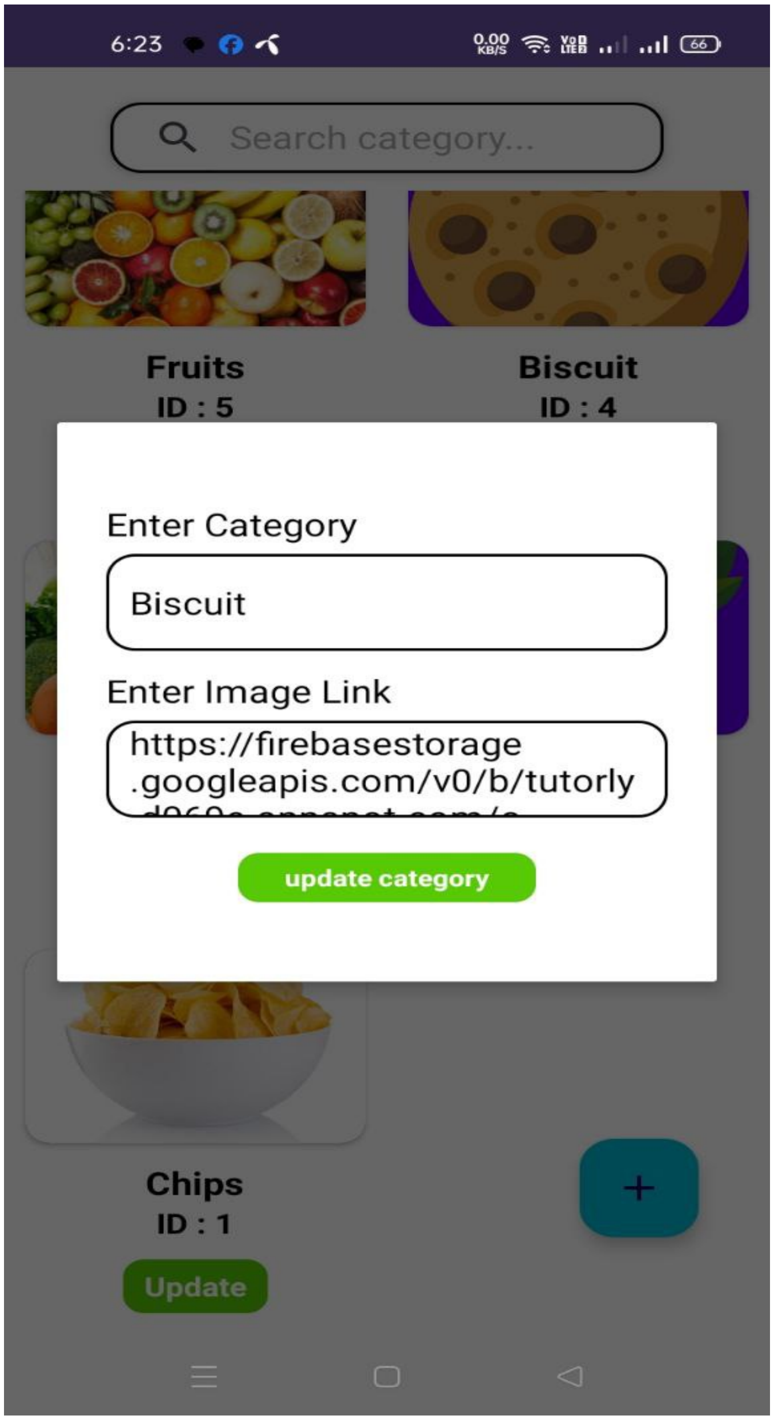
Enter Category Id

Enter Category id...

Add Category

Figure 2: Admin add category of product for user app

3.1.3 Requirement 3 : Update Category



Enter Category

Biscuit

Enter Image Link

https://firebasestorage.googleapis.com/v0/b/tutorly

update category

Figure 3: Admin can update product category

3.1.4 Requirement 4 : Add Product

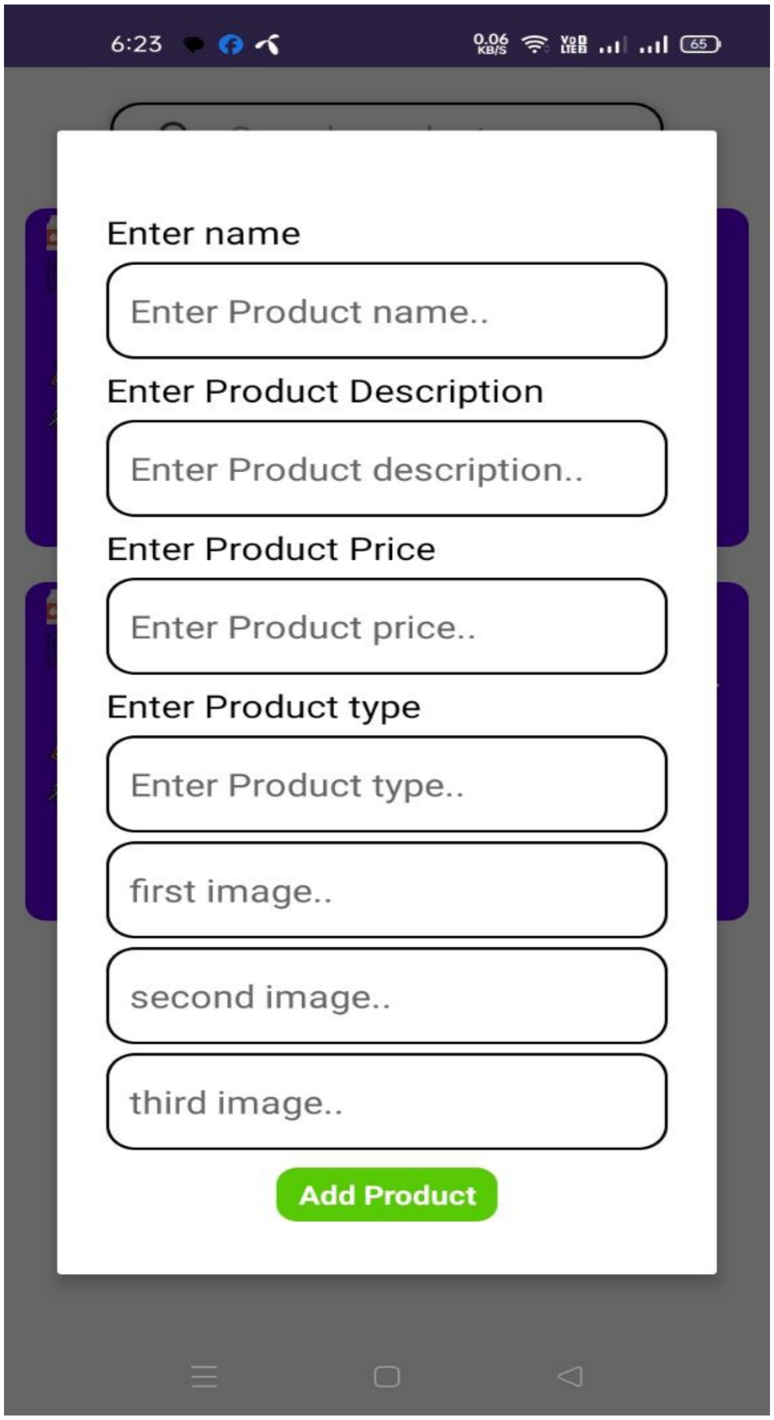
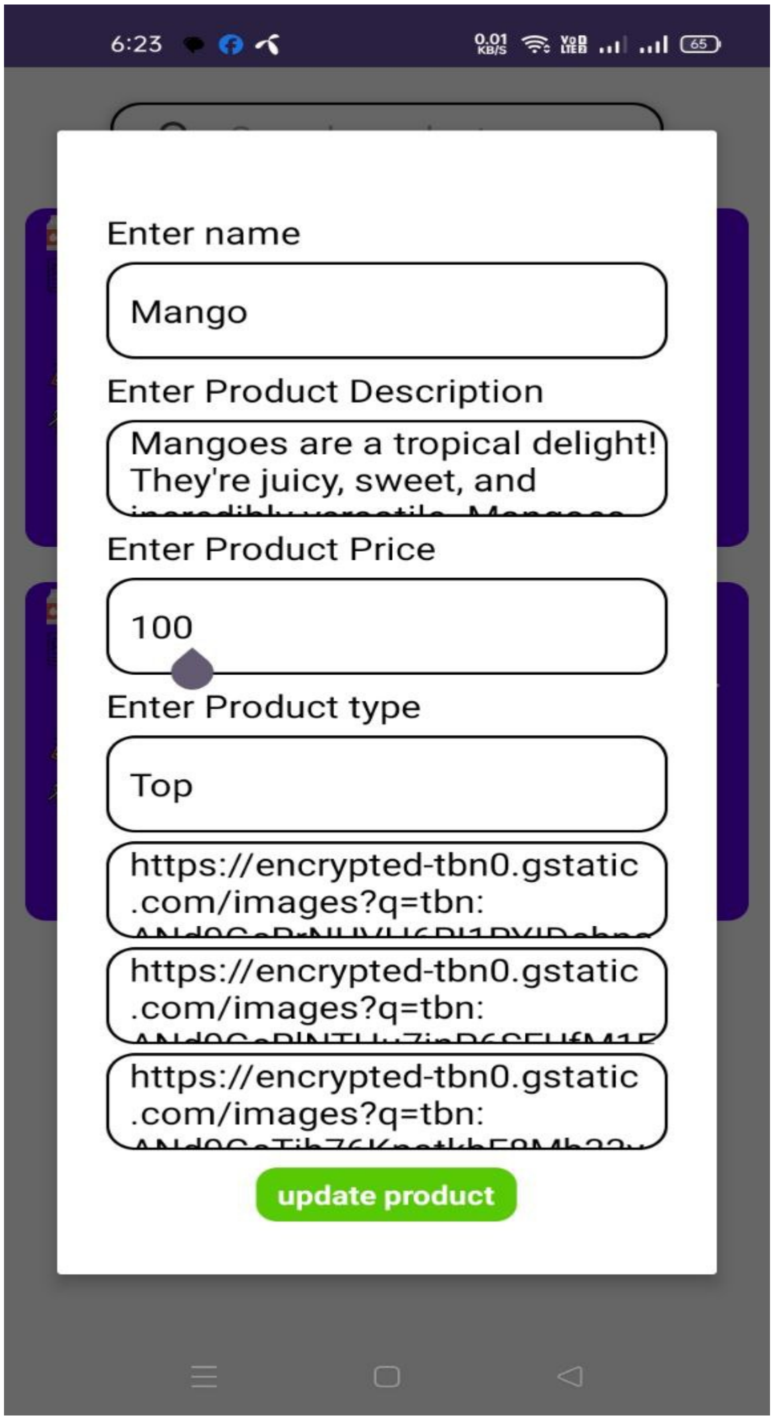


Figure 4: Admin add product of specific category

3.1.5 Requirement 5 : Update Category



Enter name

Mango

Enter Product Description

Mangoes are a tropical delight! They're juicy, sweet, and

Enter Product Price

100

Enter Product type

Top

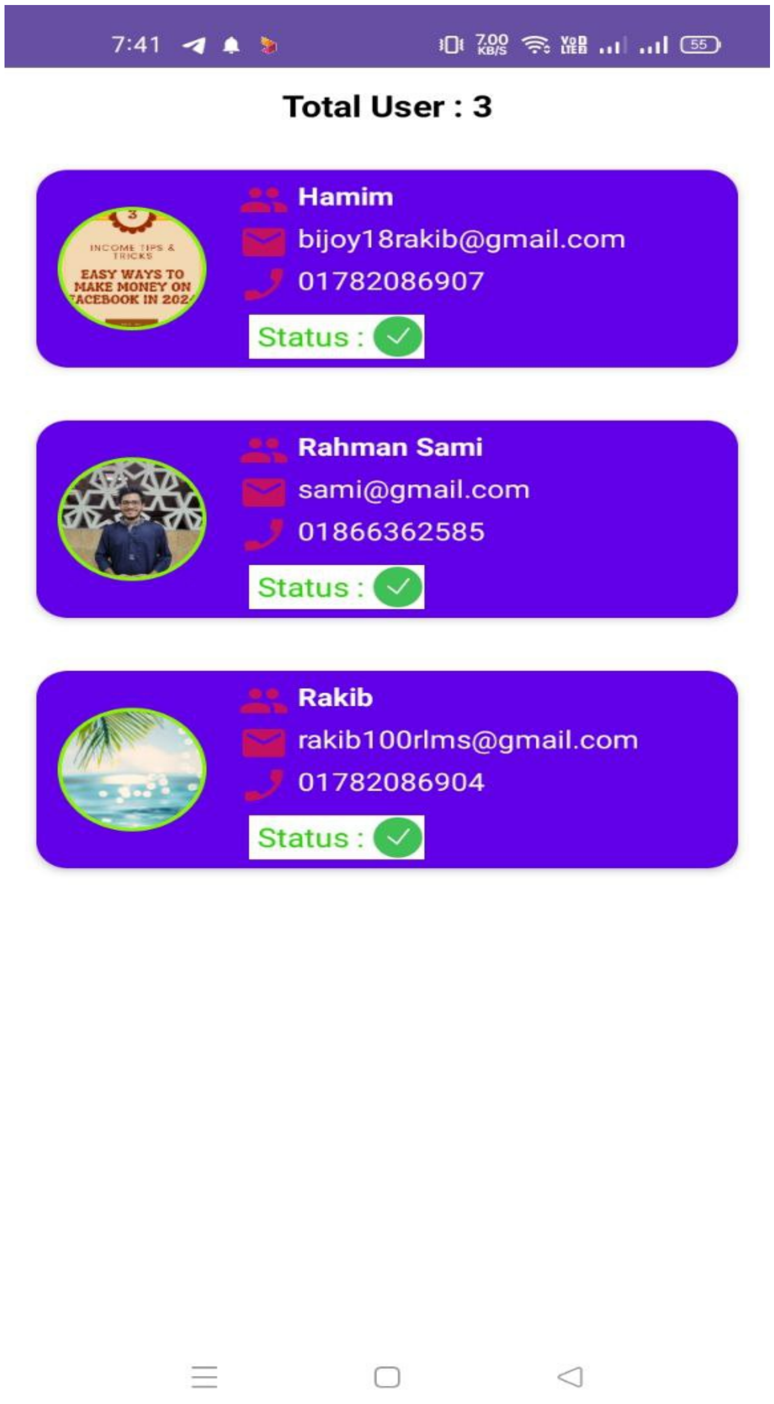
https://encrypted-tbn0.gstatic.com/images?q=tbn:

https://encrypted-tbn0.gstatic.com/images?q=tbn:

https://encrypted-tbn0.gstatic.com/images?q=tbn:

Figure 5: Admin can add product for specific category

3.1.6 Requirement 6 : User List



Total User : 3

Hamim
bijoy18rakib@gmail.com
01782086907
Status : ✓

Rahman Sami
sami@gmail.com
01866362585
Status : ✓

Rakib
rakib100rlms@gmail.com
01782086904
Status : ✓

Figure 6: User List

3.1.7 Requirement 7 : Category List

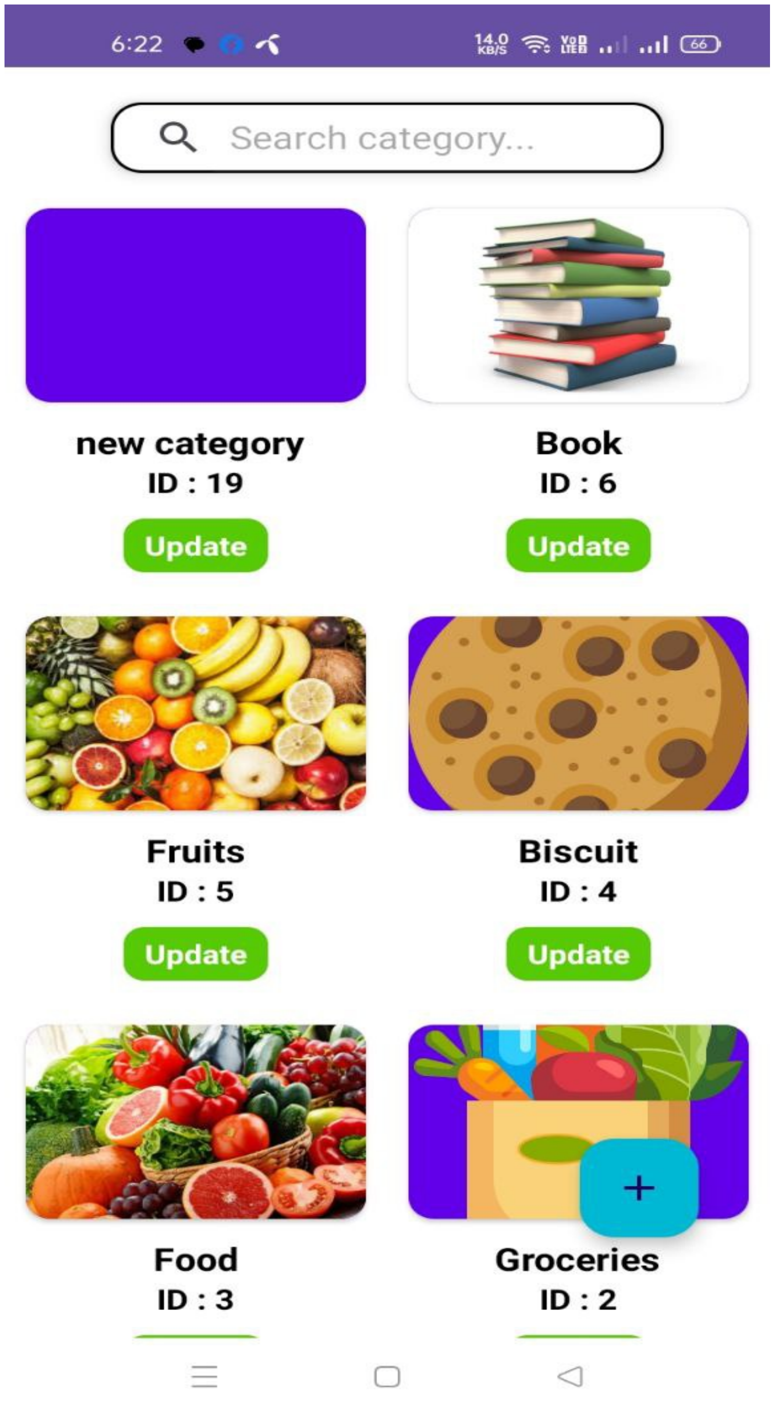


Figure 7: Admin can search category

3.1.8 Requirement 8 : Product List

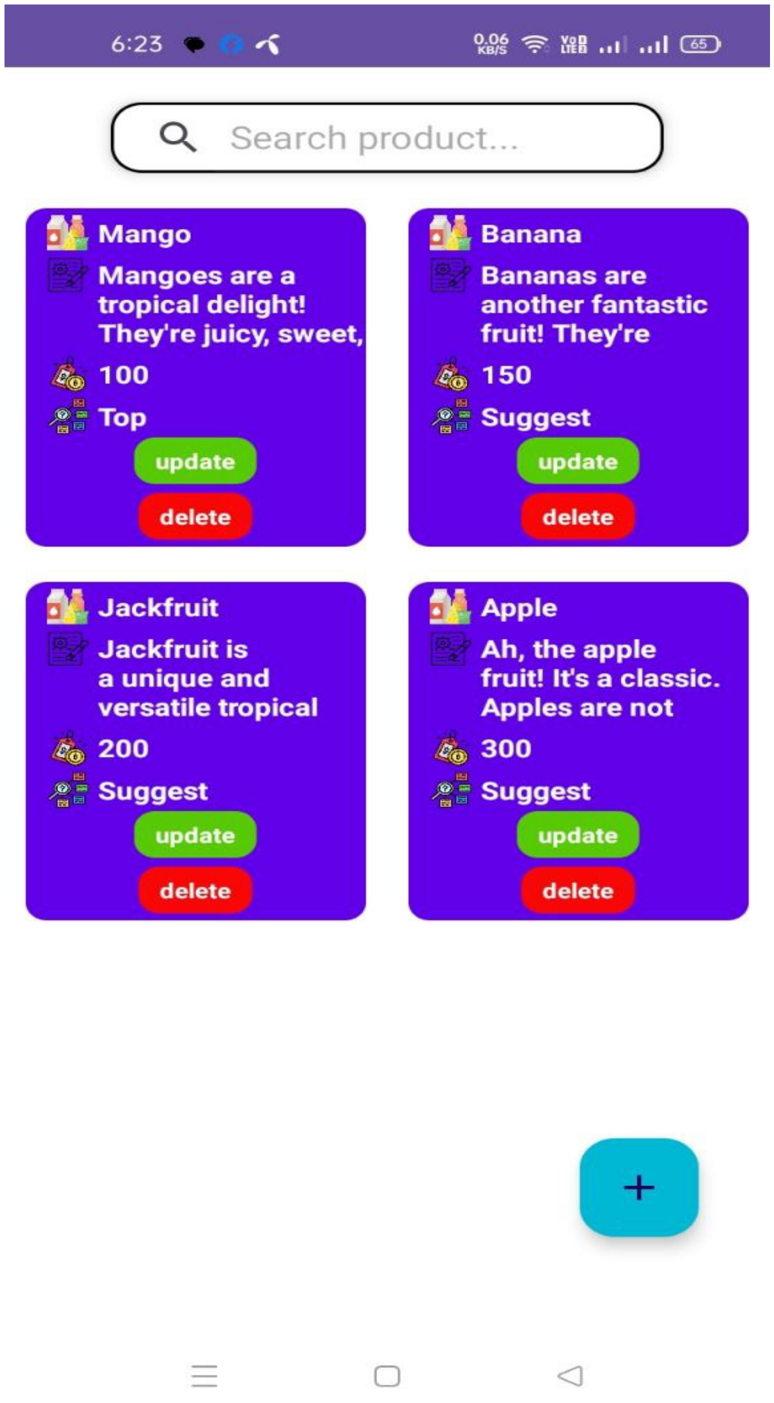
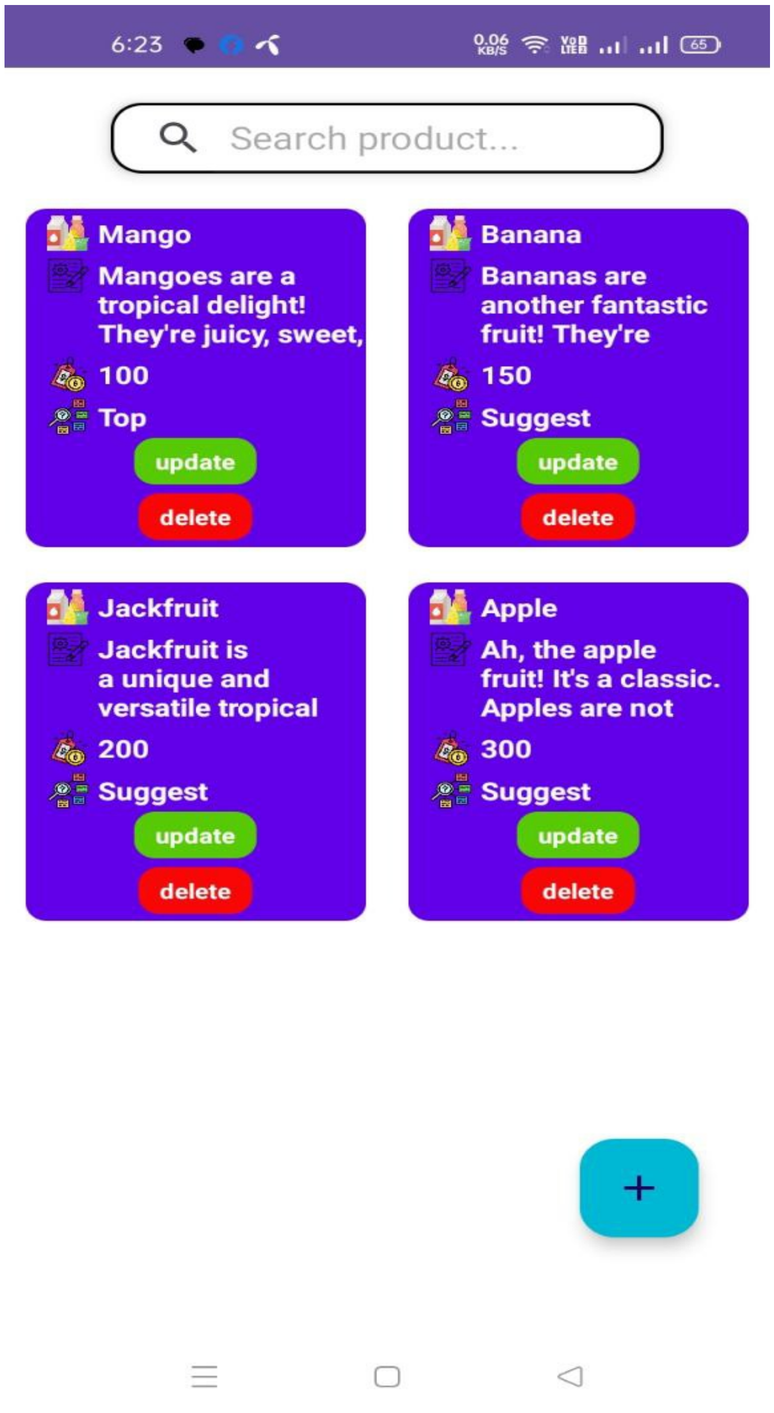

6:23
Search product...
Mango
Mangoes are a tropical delight! They're juicy, sweet,
100
Top
update
delete
Banana
Bananas are another fantastic fruit! They're
150
Suggest
update
delete
Jackfruit
Jackfruit is a unique and versatile tropical
200
Suggest
update
delete
Apple
Ah, the apple fruit! It's a classic. Apples are not
300
Suggest
update
delete
+

Figure 8: Admin can search specific category Product

3.1.9 Requirement 8 : Product List



Mango
Mangoes are a tropical delight! They're juicy, sweet,
100
Top

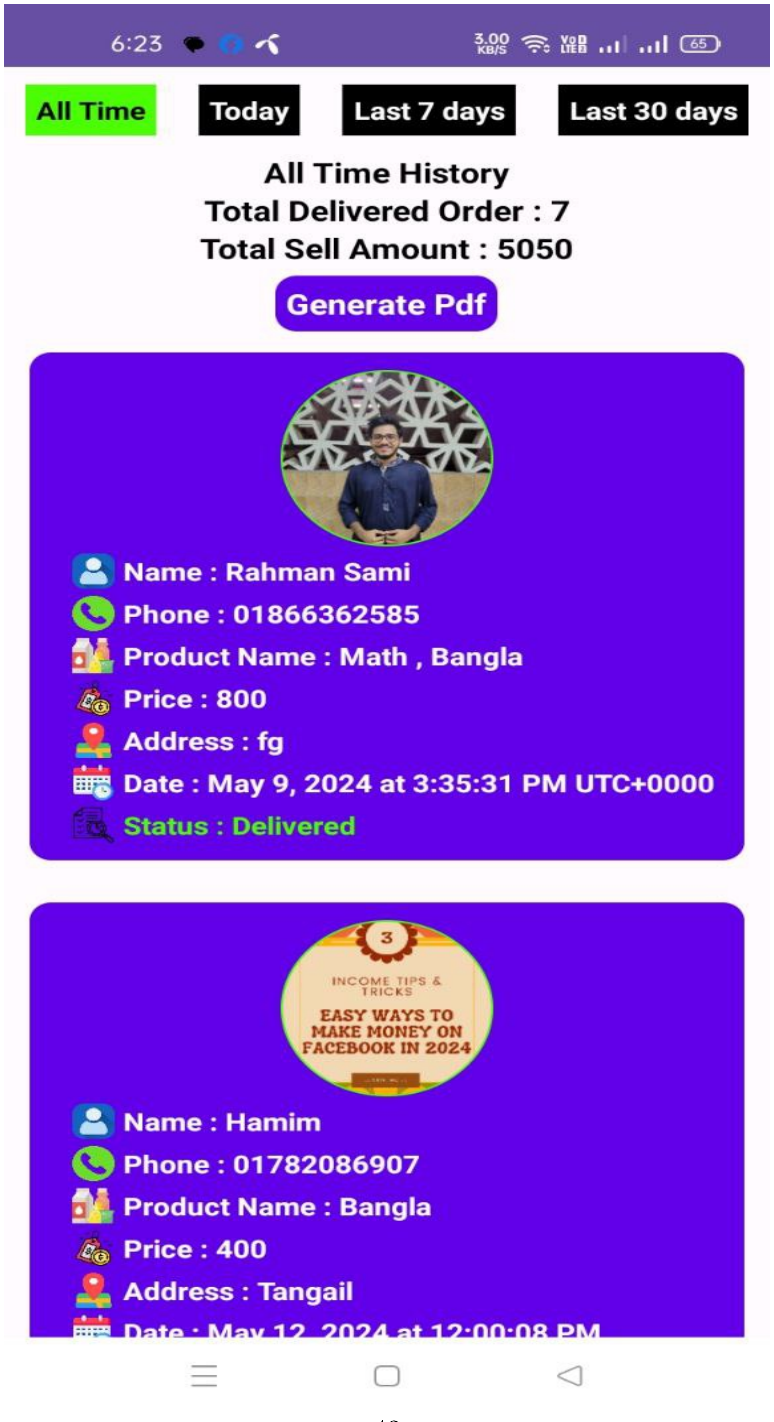
Banana
Bananas are another fantastic fruit! They're
150
Suggest

Jackfruit
Jackfruit is a unique and versatile tropical
200
Suggest

Apple
Ah, the apple fruit! It's a classic. Apples are not
300
Suggest

Figure 9: Admin can search specific category Product

3.1.10 Requirement 9 : Order List


6:23

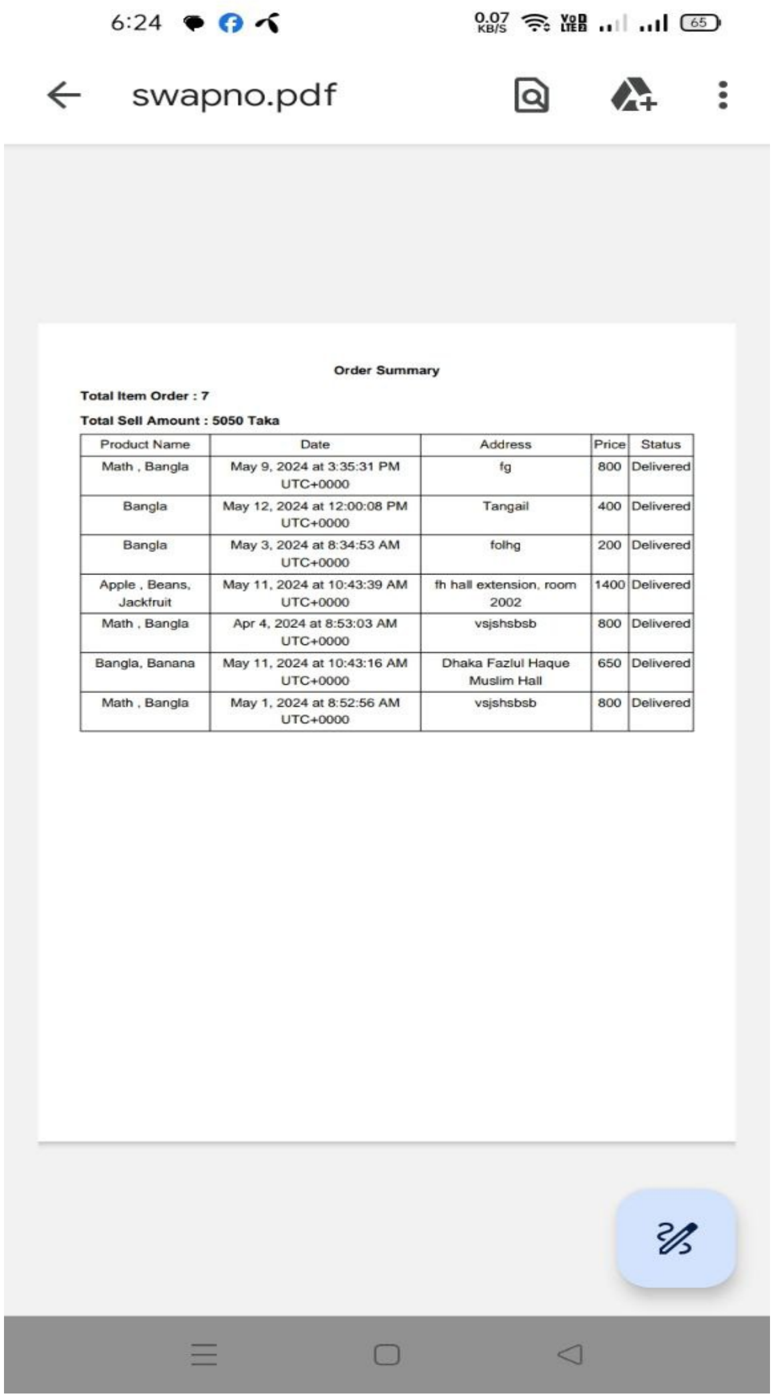
All Time History
Total Delivered Order : 7
Total Sell Amount : 5050

Name : Rahman Sami
Phone : 01866362585
Product Name : Math , Bangla
Price : 800
Address : fg
Date : May 9, 2024 at 3:35:31 PM UTC+0000
Status : Delivered

Name : Hamim
Phone : 01782086907
Product Name : Bangla
Price : 400
Address : Tangail
Date : May 12, 2024 at 12:00:08 PM

Figure 10: Admin can see differnt types of order list pending , inorder ,delivered(Today,all,last 7 days,last 30 days)

3.1.11 Requirement 10 : Order List PDF



Order Summary

Total Item Order : 7

Total Sell Amount : 5050 Taka

Product Name	Date	Address	Price	Status
Math , Bangla	May 9, 2024 at 3:35:31 PM UTC+0000	fg	800	Delivered
Bangla	May 12, 2024 at 12:00:08 PM UTC+0000	Tangail	400	Delivered
Bangla	May 3, 2024 at 8:34:53 AM UTC+0000	folhg	200	Delivered
Apple , Beans, Jackfruit	May 11, 2024 at 10:43:39 AM UTC+0000	fh hall extension, room 2002	1400	Delivered
Math , Bangla	Apr 4, 2024 at 8:53:03 AM UTC+0000	vsjshsbsb	800	Delivered
Bangla, Banana	May 11, 2024 at 10:43:16 AM UTC+0000	Dhaka Fazlul Haque Muslim Hall	650	Delivered
Math , Bangla	May 1, 2024 at 8:52:56 AM UTC+0000	vsjshsbsb	800	Delivered

Figure 11: Admin can see differnt types of order list pending , inorder ,delivered(Today,all,last 7 days,last 30 days)

3.1.12 Requirement 11 : User Sign up

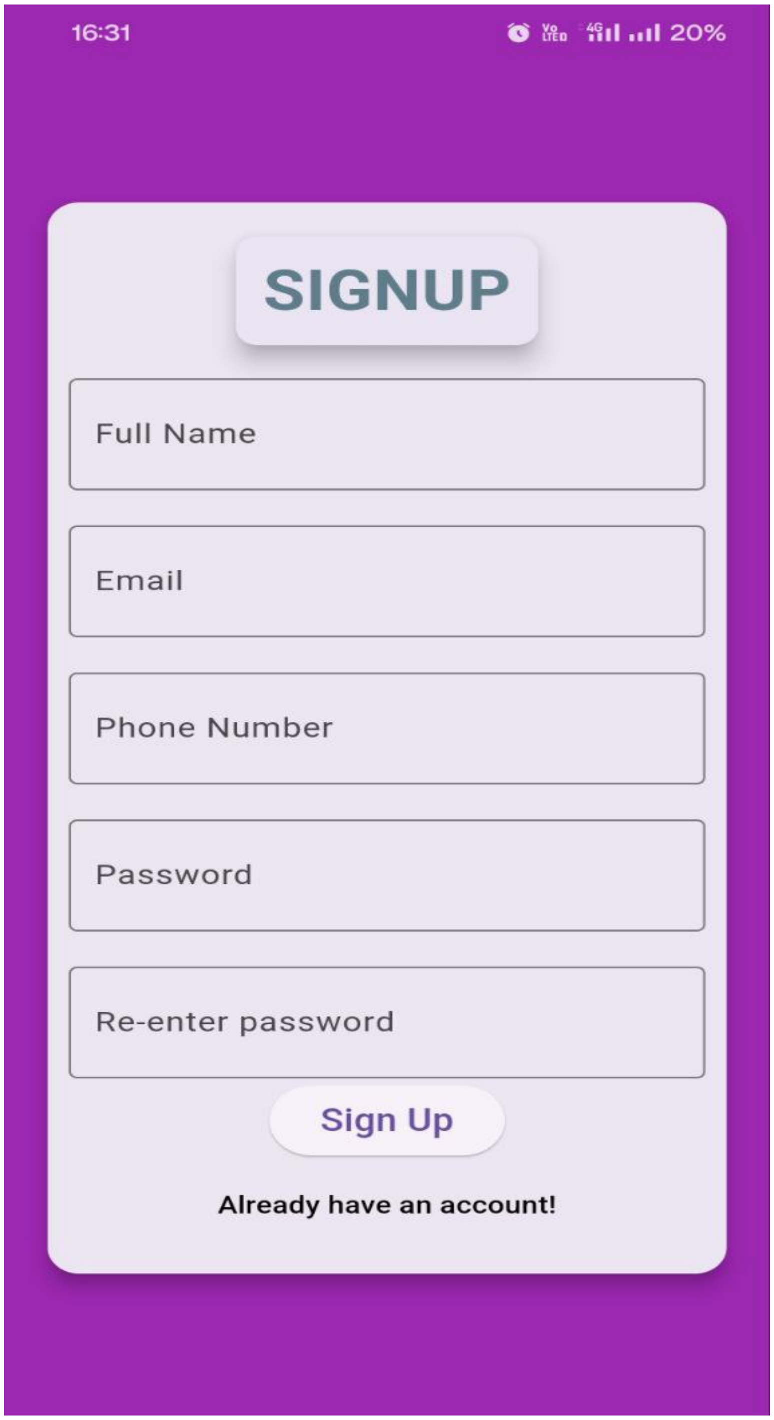


Figure 12: User Sign up

3.1.13 Requirement 12 : User Login

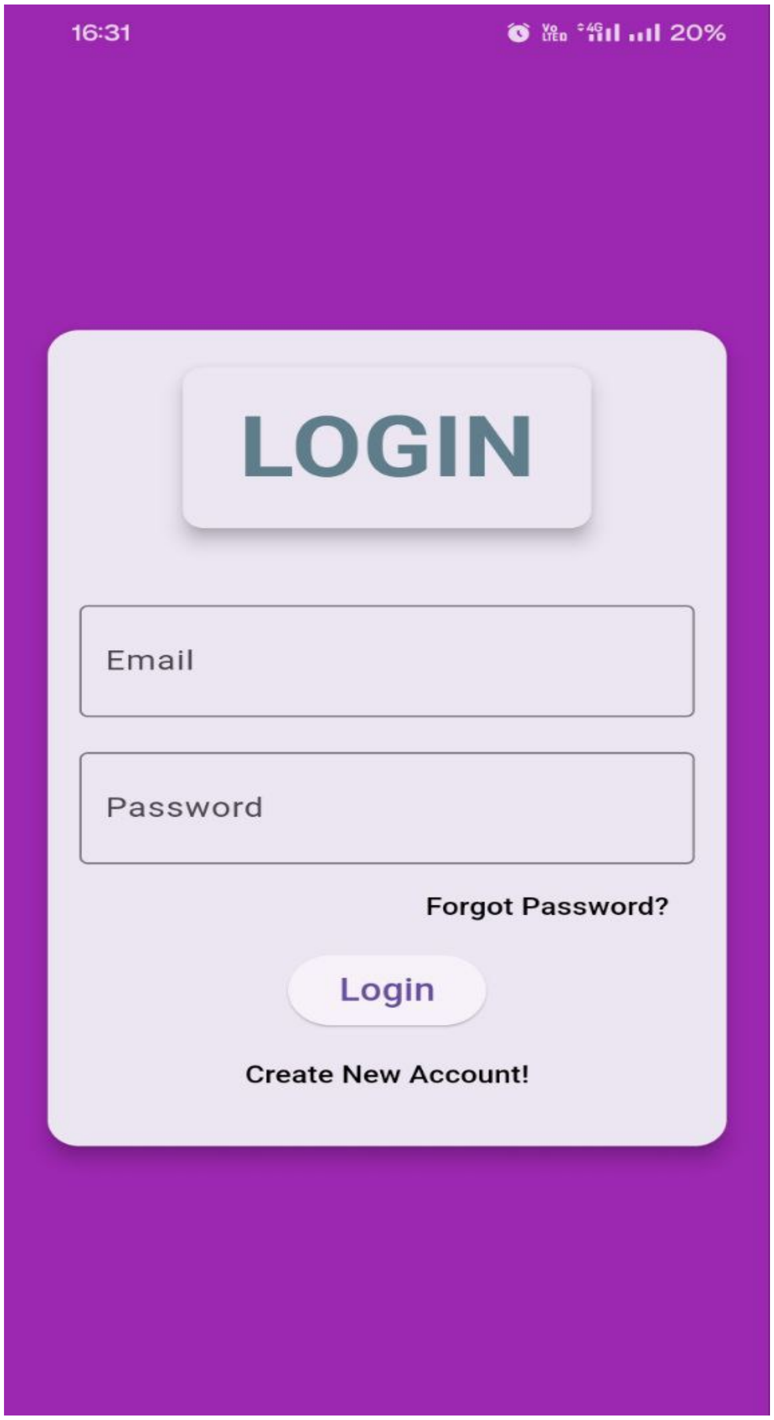


Figure 13: User login

3.1.14 Requirement 13 : User Dashboard

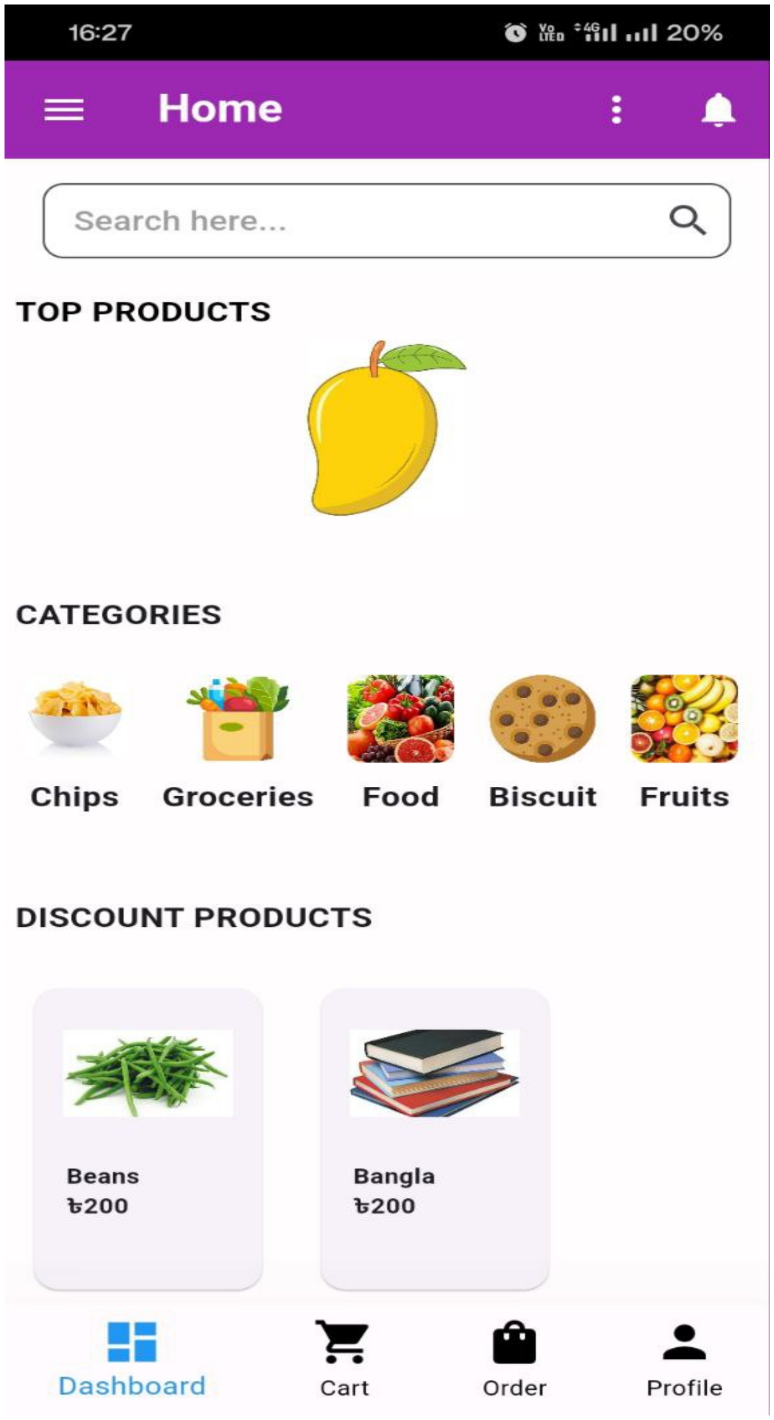
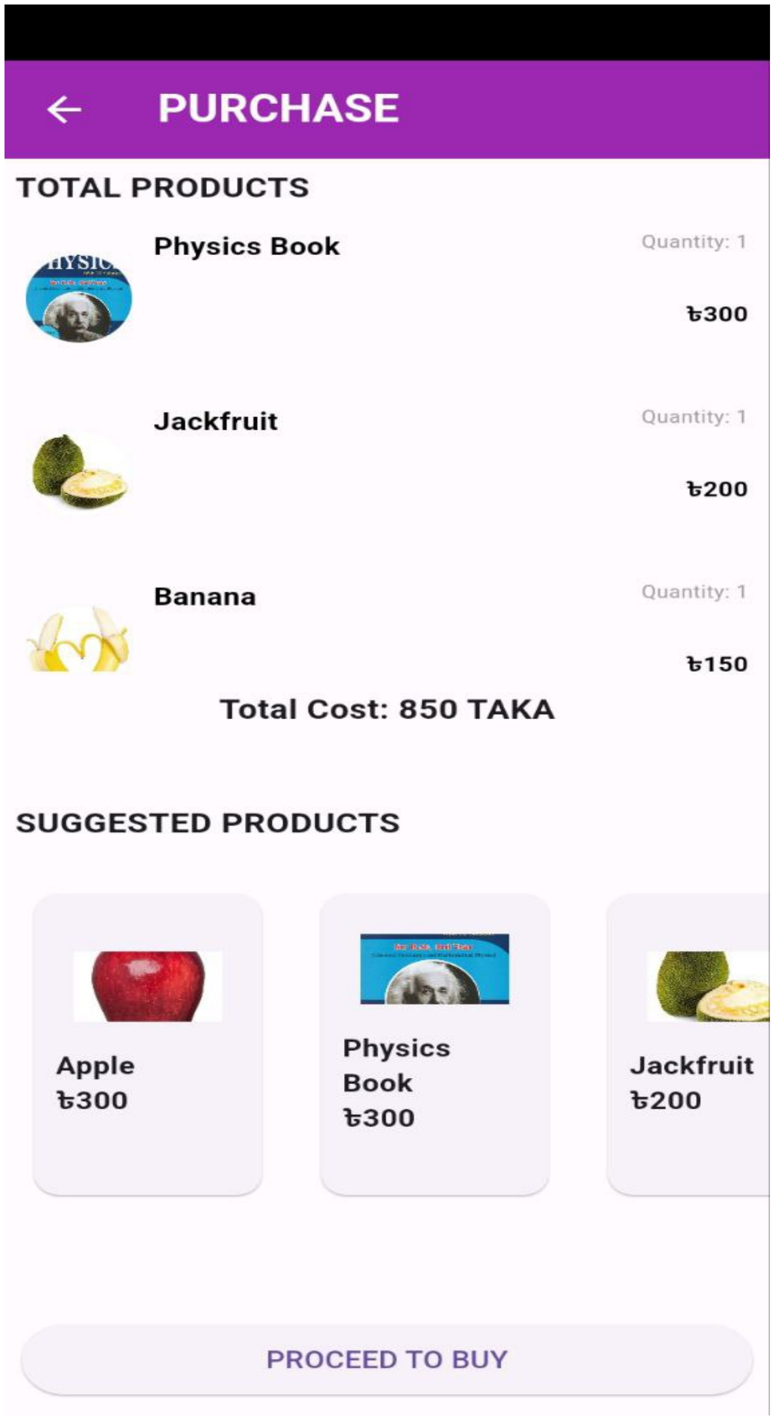


Figure 14: User Dashboard

3.1.15 Requirement 14 : User Purchase



PURCHASE

TOTAL PRODUCTS

Physics Book Quantity: 1 ৳300

Jackfruit Quantity: 1 ৳200

Banana Quantity: 1 ৳150

Total Cost: 850 TAKA

SUGGESTED PRODUCTS

Apple ৳300

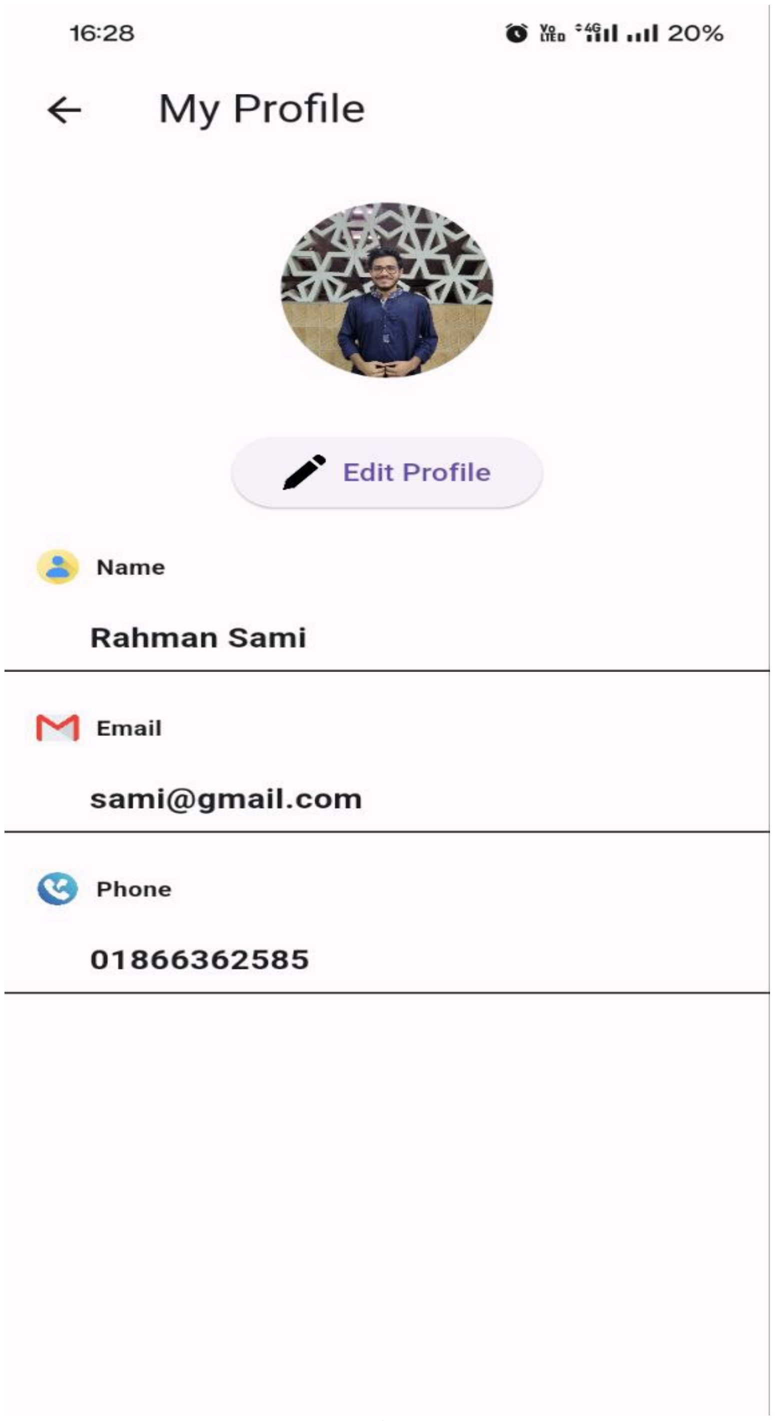
Physics Book ৳300

Jackfruit ৳200

PROCEED TO BUY

Figure 15: User Dashboard

3.1.16 Requirement 15 : User Profile


16:28

20%

My Profile

Name

Rahman Sami

Email

sami@gmail.com

Phone

01866362585

Figure 16: User Profile

3.1.17 Requirement 16 : Product Search

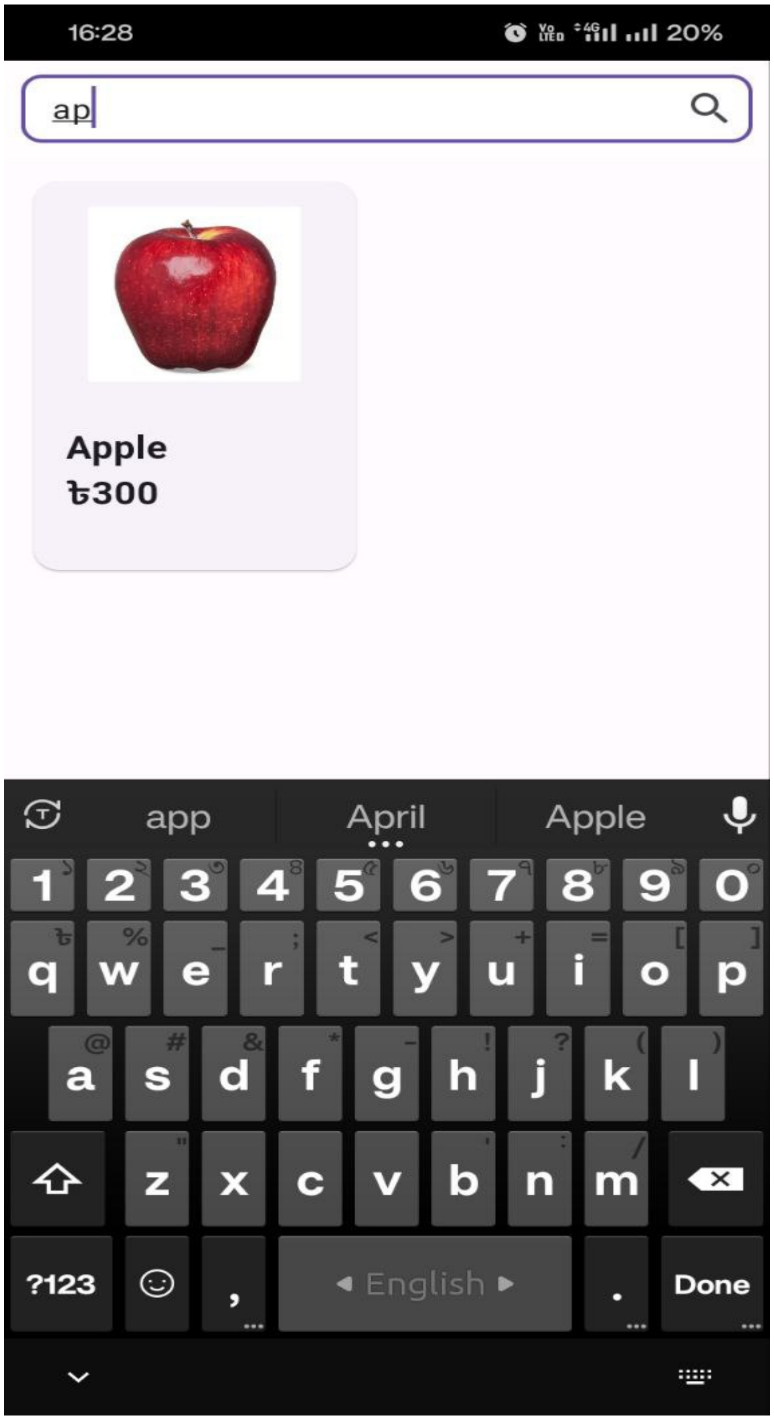
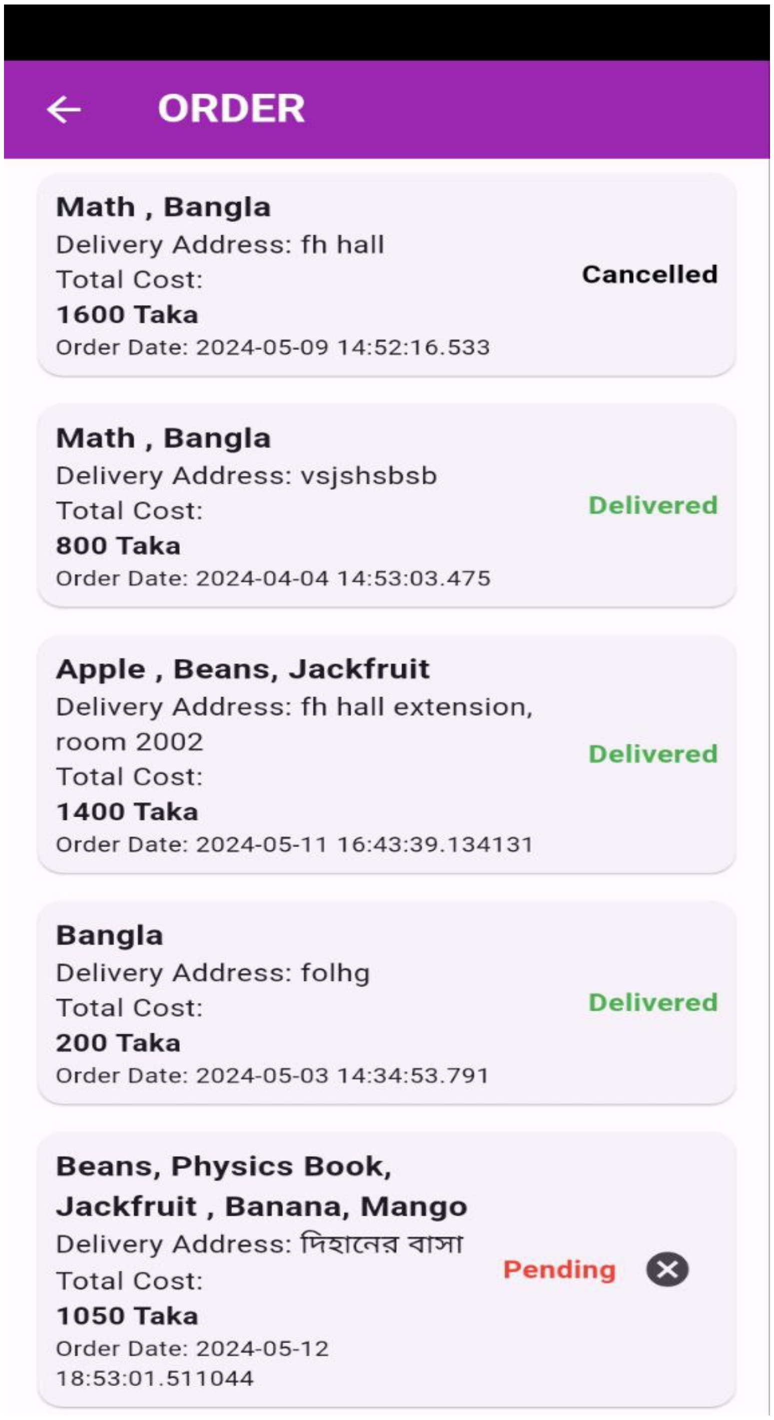


Figure 17: User can search Product

3.1.18 Requirement 17 : Order



Math , Bangla
Delivery Address: fh hall
Total Cost:
1600 Taka
Order Date: 2024-05-09 14:52:16.533
Cancelled

Math , Bangla
Delivery Address: vsjshsbsb
Total Cost:
800 Taka
Order Date: 2024-04-04 14:53:03.475
Delivered

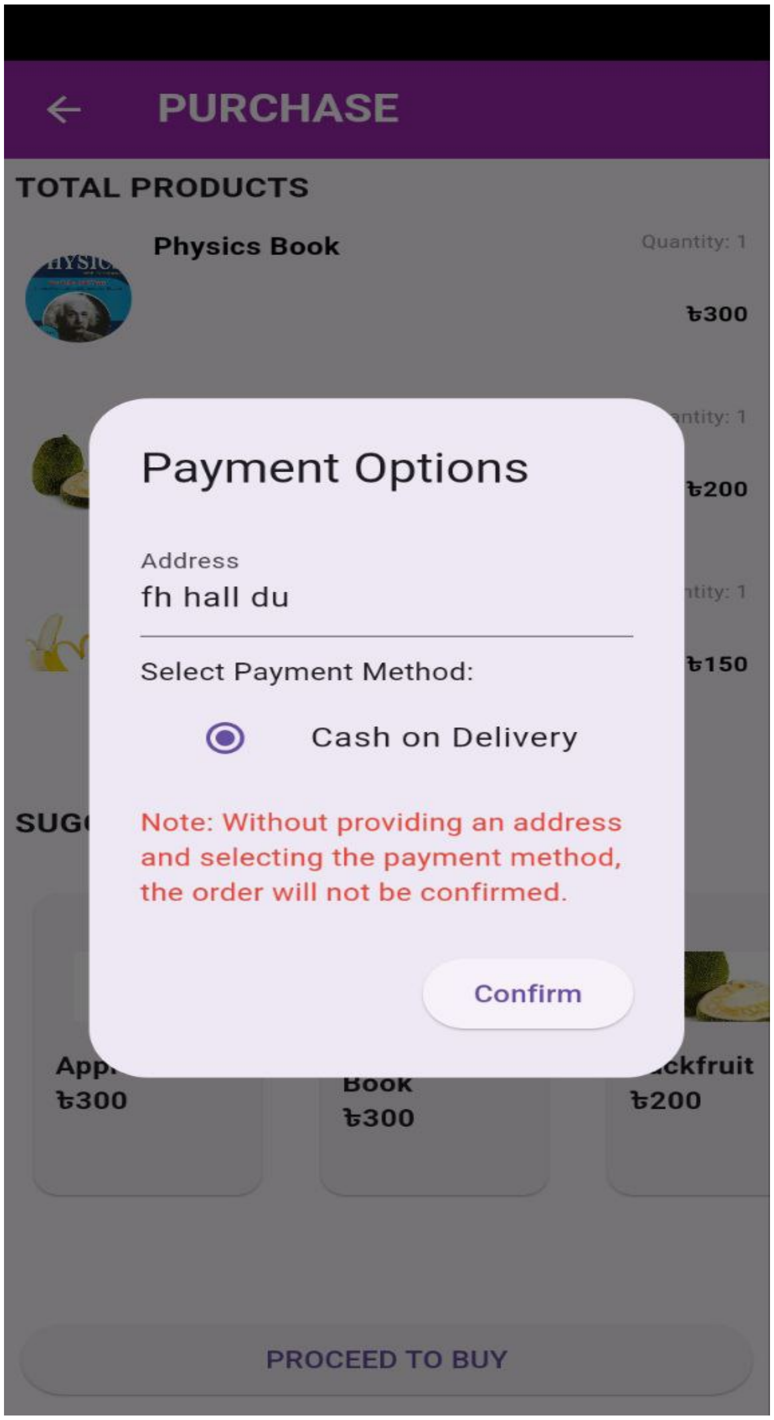
Apple , Beans, Jackfruit
Delivery Address: fh hall extension, room 2002
Total Cost:
1400 Taka
Order Date: 2024-05-11 16:43:39.134131
Delivered

Bangla
Delivery Address: folhg
Total Cost:
200 Taka
Order Date: 2024-05-03 14:34:53.791
Delivered

Beans, Physics Book, Jackfruit , Banana, Mango
Delivery Address: দিহানের বাসা
Total Cost:
1050 Taka
Order Date: 2024-05-12 18:53:01.511044
Pending

Figure 18: User order Product

3.1.19 Requirement 18 : Order Confirm


PURCHASE

TOTAL PRODUCTS

Physics Book Quantity: 1 ৳300

৳200

৳150

Payment Options

Address

fh hall du

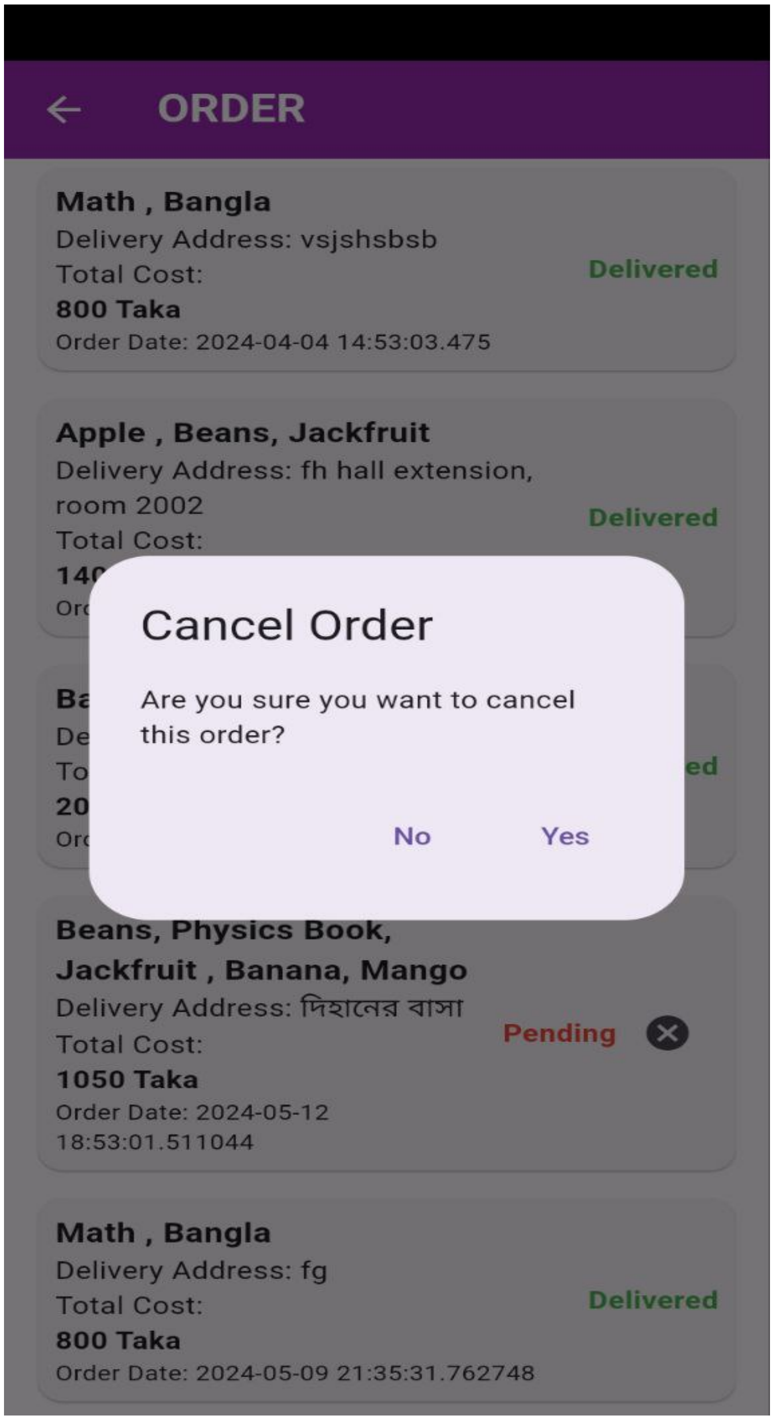
Select Payment Method:

[✓ Cash on Delivery]

Note: Without providing an address and selecting the payment method, the order will not be confirmed.

Figure 19: Order confirmation

3.1.20 Requirement 18 : Order Cancel



Math , Bangla
Delivery Address: vsjshsbsb
Total Cost:
800 Taka
Order Date: 2024-04-04 14:53:03.475

Apple , Beans, Jackfruit
Delivery Address: fh hall extension, room 2002
Total Cost:
140
Or

Ba
De
To
20
Or

Cancel Order

Are you sure you want to cancel this order?

Beans, Physics Book, Jackfruit , Banana, Mango
Delivery Address: দিহানের বাসা
Total Cost:
1050 Taka
Order Date: 2024-05-12 18:53:01.511044

Math , Bangla
Delivery Address: fg
Total Cost:
800 Taka
Order Date: 2024-05-09 21:35:31.762748

Figure 20: Cancel Order confirmation

3.2 Performance Testing

3.2.1 Security Testing

Security : We have tested the data security. After that, we tested based on each of the users and came up with this outcome

Test Case Identifier	**Test case Description**	**Outcomes**
ST-SEC-01	Unregistered User Access	Can not access any feature.
ST-SEC-02.	Registered User Authentication	We verified that a registered user can successfully log in with valid credentials and is granted access to search product and user profile and order confirm and cancel
ST-SEC-03	Registered User Authentication	We verified that a registered user cannot access or modify data that belongs to other users. We verified this by testing scenarios such as profile ,specific user order cofirm and cancel.
ST-SEC-04	Verify Admin Authorization	We verify that an admin has access to privileged functionalities such as viewing users data, total revenue, and all types order listand order list pdf generate , product add and upadate..
ST-SEC-05	Verify User Input Validation	We verified that the application performs proper validation and handling of user inputs.
ST-SEC-06	Verify Admin Input Validation	We verified that the application performs proper validation and handling of Admin inputslike category add ,update ,product add update
ST-SEC-07	Verify Password Security	We verified that the application securely stores passwords using appropriate hashing and encryption techniques. We verified this by ensuring that passwords are not stored in plain text and cannot be easily accessed or decrypted.

ST-SEC-08	Verify Data Confidentiality	We verified that sensitive user information such as passwords, order details, and user details are securely stored and transmitted to maintain data confidentiality.

3.2.2 Timing Testing

Time : We tested the timing by using multiple devices simultaneously and sending the requests on our deployed app. We used Java for backend which is very known for its swift responses. The data is given below:

Operation	**Number of request**	**Avg execution time**	**Description**
User Registration	100	0.25	We verified the average execution time for registering 100 users, measuring the time taken for each request.
Product Search	60	0.3	We verified the average execution time for searching Product 60 times, measuring the time taken for each product search request.
Order Confirmation	30	0.2	We verified the average execution time for searching Product 30 times, measuring the time taken for order request.
Order Cancel	20	0.2	We verified the average execution time for searching Product 20 times, measuring the time taken for order cancel.
Sells Pdf generate	25	0.6	We verified the average execution time for sells pdf generate.
Add Product	35	0.2	We verified the average execution time 0.2 seconds for add product 35 times
Add Category	15	0.25	We verified the average execution time 0.25 seconds for add product 15 times

3.2.3 Volume Testing

est 1: Creating 1000 product :

```java
import org.junit.jupiter.api.Test;
import org.mockito.Mockito;

import java.util.HashMap;
import java.util.Map;

import static org.junit.jupiter.api.Assertions.assertEquals;
import static org.mockito.Mockito.times;
import static org.mockito.Mockito.verify;
import static org.mockito.Mockito.when;

public class ProductServiceTest {

    @Test
    public void testCreateProductForVolume() throws Exception {
        ProductService productService = new ProductService(); // Instantiate your boo

        // Mock request data
        Map<String, Object> requestBody = new HashMap<>();
        requestBody.put("propertyId", "property123");
        requestBody.put("userId", "user123");
        // Add other necessary request data for each booking...

        // Mock request and response objects
        MockHttpServletRequest req = new MockHttpServletRequest();
        req.setBody(requestBody);

        MockHttpServletResponse res = new MockHttpServletResponse();

        // Mock behavior of response methods
        res.setStatus(Mockito.mock(MockHttpServletResponse.class).getStatus());
        res.setJson(Mockito.mock(MockHttpServletResponse.class).getJson());

        // Simulate 600 booking requests
        for (int i = 0; i < 1000; i++) {
            productService.createProduct(req, res);
        }

        // Verify the expected outcome
        assertEquals(201, res.getStatus()); // Check status code
        verify(res, times(1000)).setJson(Mockito.any()); // Verify json response call
        // Add additional assertions as needed
    }
}
```

Test 2: Creating 1000 Pdf :

```java

import java.net.URI;
import java.net.URISyntaxException;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.time.LocalDate;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutionException;
import java.util.concurrent.ThreadLocalRandom;

public class PdfVolumeTest {

    private static final String BASE_URL = "http://localhost:3000/api/invoices/bydate
    private static final int REQUESTS = 1000;

    public static void main(String[] args) throws URISyntaxException, InterruptedExce
        HttpClient httpClient = HttpClient.newHttpClient();
        CompletableFuture<Void>[] futures = new CompletableFuture[REQUESTS];

        for (int i = 0; i < REQUESTS; i++) {
            String startDate = getRandomDate();
            String endDate = getRandomDate();
            String jsonBody = String.format("{\"startDate\": \"%s\", \"endDate\": \"%

            HttpRequest request = HttpRequest.newBuilder()
                    .uri(new URI(BASE_URL))
                    .header("Content-Type", "application/json")
                    .POST(HttpRequest.BodyPublishers.ofString(jsonBody))
                    .build();

            futures[i] = httpClient.sendAsync(request, HttpResponse.BodyHandlers.ofSt
                    .thenApply(HttpResponse::statusCode)
                    .thenAccept(statusCode -> System.out.println("Request " + (i + 1)
        }

        // Wait for all requests to complete
        CompletableFuture.allOf(futures).join();
    }

    private static String getRandomDate() {
        // Generate a random date between 2020-01-01 and 2023-12-31
        LocalDate startDate = LocalDate.of(2020, 1, 1);
        LocalDate endDate = LocalDate.of(2023, 12, 31);
        long startEpochDay = startDate.toEpochDay();
        long endEpochDay = endDate.toEpochDay();
        long randomEpochDay = ThreadLocalRandom.current().nextLong(startEpochDay, end
        return LocalDate.ofEpochDay(randomEpochDay).toString();
    }
}
```

3.3 Acceptance Testing

3.3.1 Alphatest

After checking the features with some of our friends and relatives this features we rechanged and was made better in development environment

Problem	**Solve**
Users encounter errors or inconsistencies during the checkout process.	Error Handling: Implement robust error handling and validation to guide users and prevent data loss.
Searrch Item doesnt work Properly	Error Handling: Fix search item and decrease respose time
Problem on image loading from network	Error Handling: Fix image loading issue and time

3.3.2 Beta Test

Beta testing is a type of user acceptance testing (UAT) conducted by external users outside of the development team and quality assurance (QA) team. The purpose of beta testing is to evaluate the software product's usability, functionality, performance, and compatibility with different environments and use cases. Beta testers provide feedback based on their experiences, helping to identify bugs, usability issues, and areas for improvement.

Problem	**Solve**
Deployment Issues	In Payment system we only use COD : Cash on Delivery