**Stockory: Inventory Management System**

**MAD Lab Project Report**

**Submitted by,**

|  |  |  |
| --- | --- | --- |
| **Name** | **Registration Number** | **Roll Number** |
| **Alok Kumar** | **220953464** | **37** |
| **Mahesh Soni** | **220953570** | **48** |
| **Saswat Dash** | **220953638** | **58** |
|  |  |  |

**Abstract**  
Managing inventory efficiently is a critical aspect for both merchants and vendors in today’s competitive and fast-moving market. However, traditional inventory processes often involve fragmented systems, manual record-keeping, and lack of real-time synchronization between suppliers and resellers. These inefficiencies can lead to stock mismanagement, delayed communication, and poor coordination. The Inventory Management System Android App addresses these challenges by offering a centralized, mobile-based solution designed specifically for merchants and vendors. This user-friendly application allows merchants to manage their stock by adding, viewing, and updating inventory items, while enabling vendors to search for merchants, import products, update item details, and manage sales activities. With distinct login systems for each user type, secure account creation, and real-time inventory adjustments, the app ensures smooth and transparent operations. Its intuitive interface, dark-themed UI, and SQLite-powered backend make it both accessible and efficient, streamlining the entire supply chain interaction. This solution enhances productivity, reduces operational friction, and fosters better communication between suppliers and buyers in the inventory ecosystem.

**Introduction**

Inventory management is a critical component in the supply chain processes of businesses, directly impacting operational efficiency, customer satisfaction, and overall profitability. Traditionally, merchants and vendors have relied on manual or fragmented digital systems to track inventory, communicate transactions, and manage stock levels. These approaches often result in data inconsistency, lack of real-time visibility, and inefficient coordination between parties, leading to stockouts, overstocking, and lost sales opportunities.

This project introduces an Android-based Inventory Management System that streamlines the management of goods between merchants and vendors through a unified mobile platform. Developed using Java and SQLite, the application provides distinct user roles for merchants and vendors, enabling tailored functionalities for each. Merchants can log in to add, update, and view inventory items, while vendors can search for merchants, import items based on availability, modify item details for resale, and track sales performance.

The system ensures data integrity by implementing strict validation during user registration, including checks for unique usernames, phone numbers, and email addresses. Real-time inventory updates are handled efficiently through SQLite, ensuring that any item transferred between merchant and vendor is accurately reflected in both user databases. The application also features a structured user interface designed with a dark theme to enhance readability and usability.

By consolidating inventory operations into a single mobile application, this system enhances coordination between merchants and vendors, reduces the risk of stock discrepancies, and simplifies the overall inventory workflow. It serves as a scalable solution for small to medium-sized enterprises looking to digitize and optimize their inventory processes without relying on costly or complex enterprise solutions.

**Project Objective**

1. Develop an Android-based inventory management system that provides dedicated features for both merchants and vendors.
2. Streamline the process of inventory tracking, product transfers, and sales operations through a unified mobile platform.
3. Ensure secure and unique user registration for both merchants and vendors with validation for usernames, emails, and phone numbers.
4. Enable merchants to efficiently manage inventory by adding, updating, and viewing product details including name, price, quantity, and tags.
5. Allow vendors to search merchants, import available inventory items, and maintain a personalized inventory database with update and sales management features.
6. Maintain real-time data consistency using SQLite to reflect accurate inventory levels across both merchant and vendor databases.
7. Provide an intuitive and responsive user interface with a consistent dark theme, ensuring usability and accessibility across all pages.
8. Facilitate better coordination and transparency between suppliers and resellers by automating key inventory functions within a single application**.**

**Tools and Technologies Used**

In the making of this application, the various tools and technologies that we employed are as follow:

1. Android Studio
2. Language: Java
3. SQLite

**System Requirements**

Minimum Android version: Android 8.0 (API Level 26 - Oreo)

IDE: Android Studio **Giraffe (2022.3.1**)

Android SDK: API Level 33 (Android 13)

System requirements to develop the application:

* OS: Windows 10/11, macOS (Monterey or later), or Ubuntu 20.04+
* Processor: Intel i5 (8th Gen or higher) / AMD Ryzen 5 or better
* RAM: Minimum 8 GB (16 GB recommended for smooth emulation and multitasking)
* Storage**:** At least 10 GB free (SSD recommended)
* Graphics: Hardware acceleration support (e.g., Intel HAXM or AMD Hypervisor for emulator)

System requirements to run the application on mobile:

* OS: Android 8.0 (API 26) or above
* RAM: 3 GB or more
* Storage: At least 200 MB of free space for app installation
* Connectivity: Wi-Fi or mobile data for real-time features (e.g., chat, event sync)

**Project Modules/Features**

1. **Home Screen**

Serves as the entry point to the application, allowing users to choose their role for login. Two distinct options—**Login as Merchant** and **Login as Vendor**—are provided as clearly styled buttons. Based on the selected role, users are redirected to their respective login and functionality modules. The UI is clean, dark-themed, and user-friendly, ensuring a smooth navigation experience from the very first interaction.

.

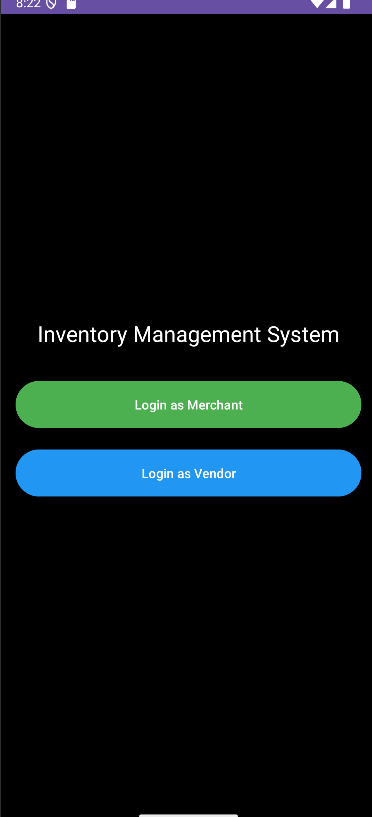


Figure 1 Home Screen

**2. Login and Account Creation (Merchant & Vendor)**

The login screens for both merchants and vendors provide a simple interface with a single input field for entering the username, followed by two clearly styled buttons: **Login** and **Create Account**. The login button verifies user credentials, while the create account option redirects to a registration form ensuring unique username, email, and phone number validation. Both screens follow the app’s dark theme with a black background, white text, and distinct green and blue buttons, offering a clean and intuitive user experience.

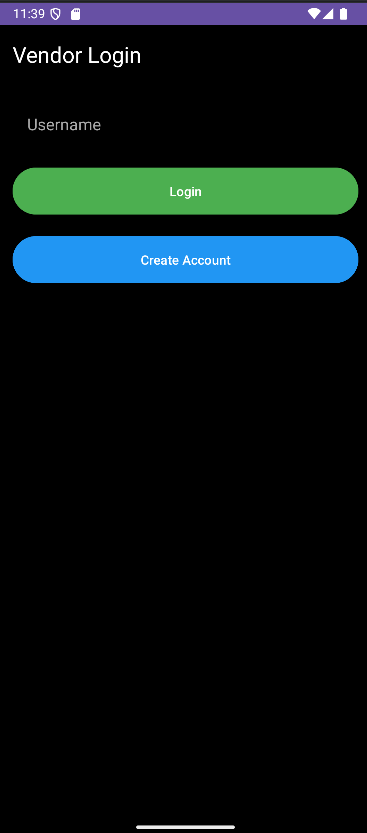
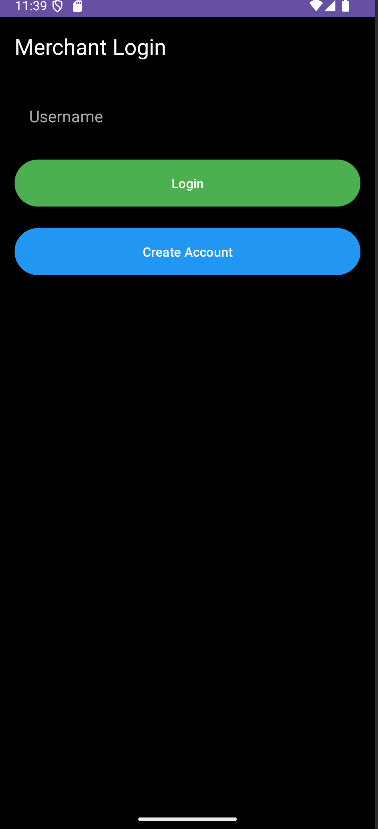
 

Figure 2 and 3 Login and Account Creation (Merchant & Vendor)

**3.Vendor and Merchant Dashboards**

Upon successful login, users are directed to their respective dashboards based on their roles—Vendor or Merchant. The **Vendor Dashboard** features five main functionalities: *Contact Merchants*, *Add Item*, *View Items*, *Update Items*, and *View Sales*, each clearly labeled and color-coded for intuitive access. These options allow vendors to manage their imported inventory, interact with merchants, and track and update items seamlessly. The **Merchant Dashboard**, on the other hand, provides two primary actions: *Add Item* and *View Inventory*, enabling merchants to maintain and organize their stock efficiently. A personalized greeting adds a friendly touch to the interface, while the “Back to Home” button ensures easy navigation. The UI is minimalistic and user-centric, ensuring efficient inventory operations for both roles.

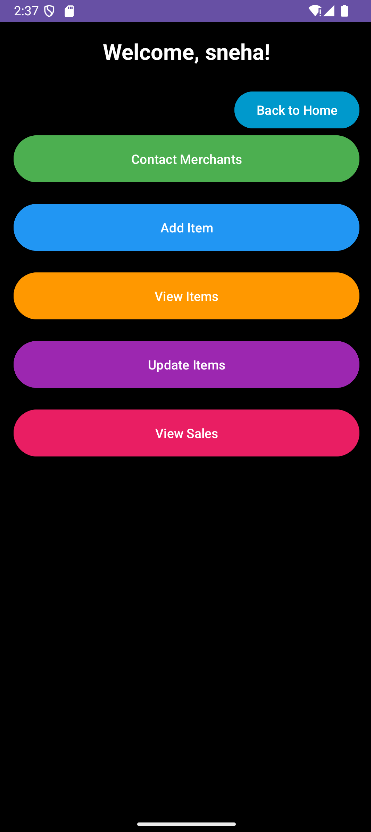
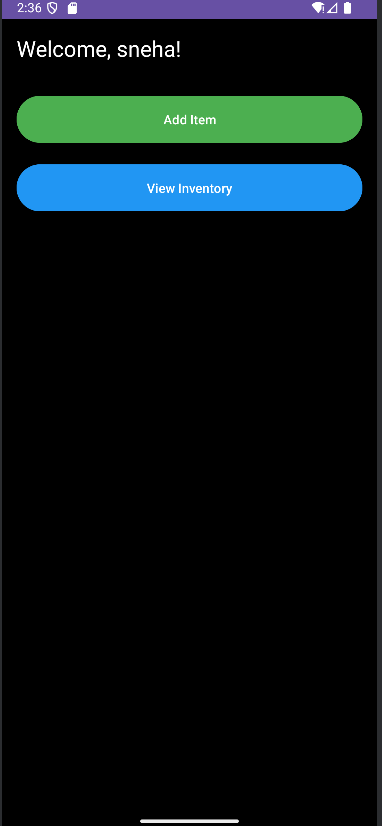
 

Figure 4 and 5 Vendor and Merchant Dashboards

1. Merchant- Add Item:

-Merchant can add items to his/her inventory by specifying item name ,price ,quantity and tag.

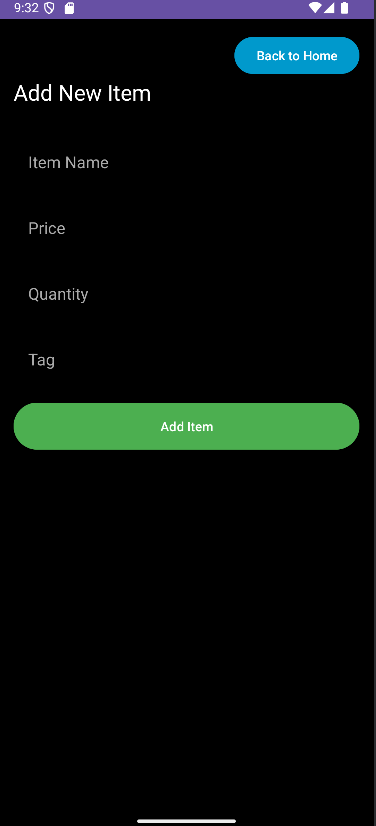


Figure 6 Merchant- Add Item

1. Merchant- View Inventory:

-Merchant can view the items stored in his/her inventory.

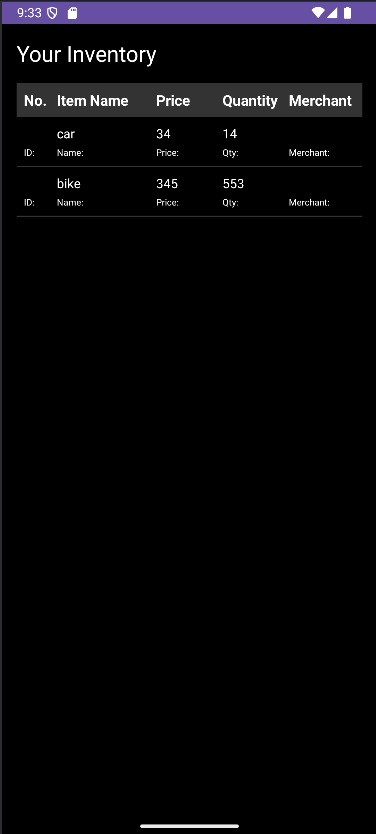


Figure 7 Merchant- View Inventory

1. Vendor- Contact Merchants:

-Displays Email and contact no of merchants.  
-Vendor can place a call directly to the merchant regarding any queries.

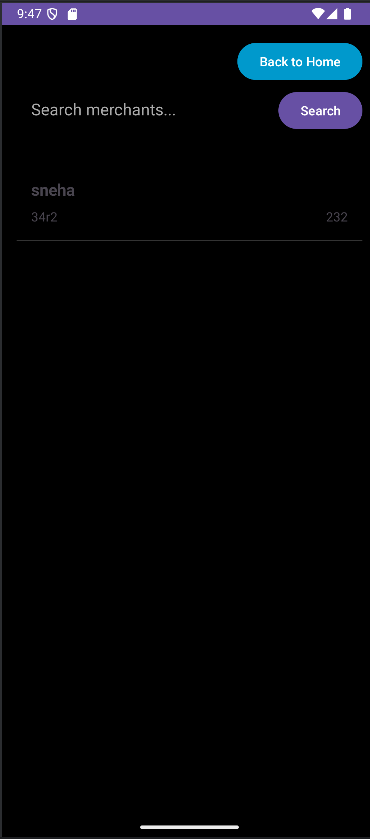


Figure 8 Vendor- Contact Merchants

1. Vendor- Add Item:

-Vendor can navigate through different merchants and add items that he/she wants to his/her inventory.

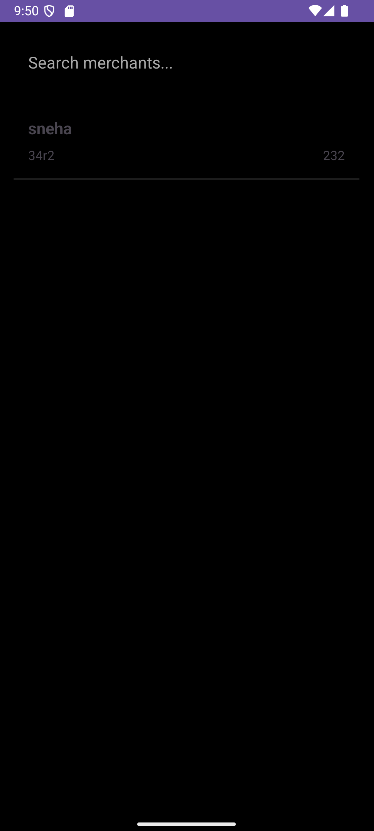


Figure 9 Vendor- Add Item

1. Vendor- View Items:

-Vendor can view items added to his/her inventory.

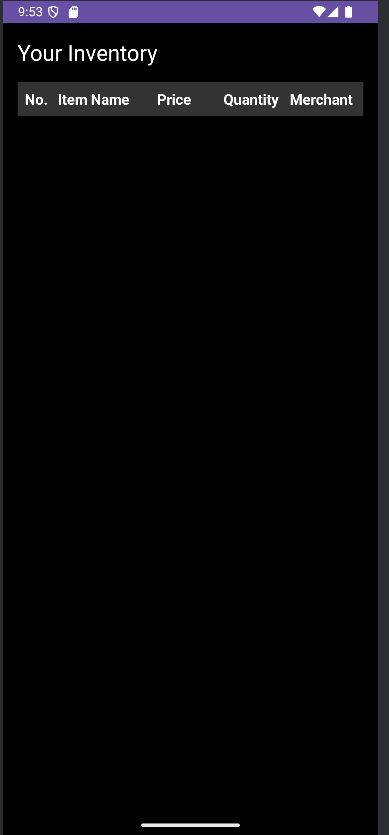


Figure 10 Vendor- View Items

1. Vendor- Update Items:

-Vendor can change the name and price of the item.

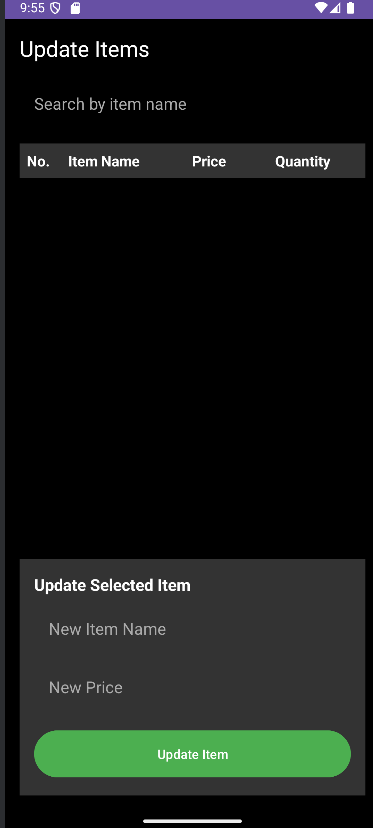


Figure 11 Vendor- Update Items

1. Vendors- View Sales:

-Vendor can click on an item and sell it.  
-on selling total revenue will be recorded.  
-View sales lists details of all the items sold.

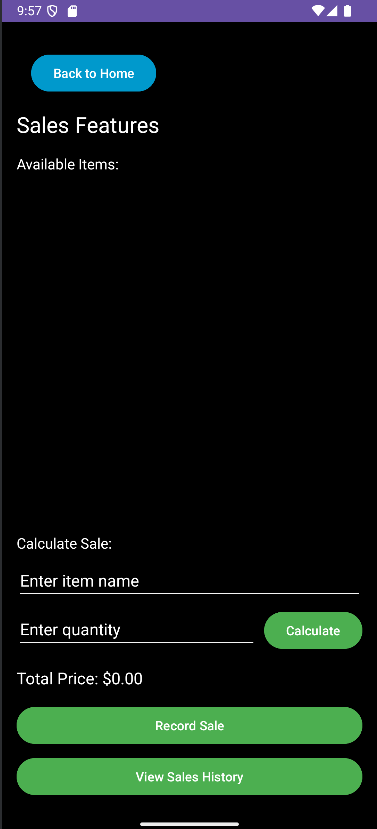


Figure 13 Vendors- View Sales

**Database Design**

* Main Activity

1. Function: Entry point of the application
2. Database IDs: None (Login/Registration selection screen)

* Merchant Activities:

1. MerchantLoginActivity
2. Function: Handles merchant login
3. Database IDs: Uses TABLE\_MERCHANTS (id, username, email, phone)
4. MerchantCreateAccountActivity
5. Function: Creates new merchant accounts
6. Database IDs: Uses TABLE\_MERCHANTS (id, username, email, phone)
7. MerchantDashboardActivity
8. Function: Main dashboard for merchants
9. Database IDs: Uses TABLE\_MERCHANTS and TABLE\_MERCHANT\_ITEMS
10. AddItemActivity
11. Function: Allows merchants to add new items
12. Database IDs: Uses TABLE\_MERCHANT\_ITEMS (id, item\_name, price, quantity, tag, owner\_id)
13. ViewInventoryActivity
14. Function: Displays merchant's inventory
15. Database IDs: Uses TABLE\_MERCHANT\_ITEMS
16. UpdateItemActivity
17. Function: Allows updating item details
18. Database IDs: Uses TABLE\_MERCHANT\_ITEMS

* Vendor Activities:

1. VendorLoginActivity
2. Function: Handles vendor login
3. Database IDs: Uses TABLE\_VENDORS (id, username, email, phone)
4. VendorCreateAccountActivity
5. Function: Creates new vendor accounts
6. Database IDs: Uses TABLE\_VENDORS (id, username, email, phone)
7. VendorDashboardActivity
8. Function: Main dashboard for vendors
9. Database IDs: Uses TABLE\_VENDORS and TABLE\_VENDOR\_ITEMS
10. MerchantItemsActivity
11. Function: Displays items from merchants
12. Database IDs: Uses TABLE\_MERCHANT\_ITEMS
13. AddItemFromMerchantActivity
14. Function: Allows vendors to add items from merchants
15. Database IDs: Uses TABLE\_VENDOR\_ITEMS and TABLE\_MERCHANT\_ITEMS

* Sales Related Activities:

1. SalesFeaturesActivity
2. Function: Handles sales operations
3. Database IDs: Uses TABLE\_VENDOR\_ITEMS, TABLE\_SALES\_HISTORY
4. SalesHistoryActivity
5. Function: Displays sales history
6. Database IDs: Uses TABLE\_SALES\_HISTORY (id, item\_id, quantity\_sold, total\_price, sale\_date, vendor\_id)

* Contact/Details Activities:

1. ContactActivity
2. Function: Displays contact information
3. Database IDs: Uses TABLE\_MERCHANTS and TABLE\_VENDORS
4. MerchantDetailsActivity
5. Function: Shows merchant details
6. Database IDs: Uses TABLE\_MERCHANTS
7. View Activities:
8. ViewImportedItemsActivity
9. Function: Shows imported items
10. Database IDs: Uses TABLE\_VENDOR\_ITEMS

Database Tables and their IDs:

* TABLE\_MERCHANTS

1. id (PRIMARY KEY)
2. username
3. email
4. phone

* TABLE\_VENDORS

1. id (PRIMARY KEY)
2. username
3. email
4. phone

* TABLE\_MERCHANT\_ITEMS

1. id (PRIMARY KEY)
2. item\_name
3. price
4. quantity
5. tag
6. owner\_id (FOREIGN KEY to TABLE\_MERCHANTS)

* TABLE\_VENDOR\_ITEMS

1. id (PRIMARY KEY)
2. item\_name
3. price
4. quantity
5. tag
6. owner\_id (FOREIGN KEY to TABLE\_VENDORS)

* TABLE\_SALES\_HISTORY

1. id (PRIMARY KEY)
2. item\_id (FOREIGN KEY to TABLE\_VENDOR\_ITEMS)
3. quantity\_sold
4. total\_price
5. sale\_date
6. vendor\_id (FOREIGN KEY to TABLE\_VENDORS)

* TABLE\_ITEM\_MODIFICATIONS

1. id (PRIMARY KEY)
2. item\_id (FOREIGN KEY to TABLE\_VENDOR\_ITEMS)
3. modified\_name
4. modified\_price

**Testing and Output**

Unit Testing: It was done on individual modules like login authentication, adding items to database , and updating item details.

Integration Testing: This ensured smooth movement from data from one activity to another, database update is handled properly and proper authentication is implemented.

Test Cases included:

* Successful and failed login attempts.
* Checking uniqueness of each login
* Items successfully getting added into inventories
* Successfully updating items and recording their sales records.

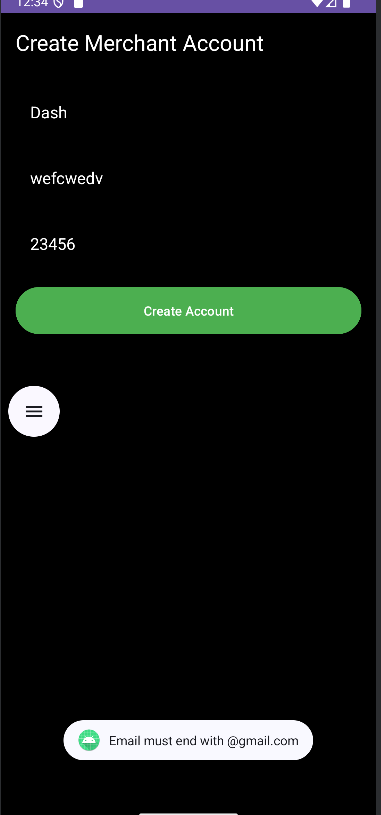


Figure failed login

**Conclusion**

**What you learned:**

Building the Inventory Management System app provided valuable insights into role-based application development, data synchronization, and secure transaction handling. Implementing distinct workflows for merchants and vendors helped us understand the importance of modular architecture and consistent user experience. It also strengthened our grasp on local database management using SQLite and handling data validation at both frontend and backend levels.ty.

**Challenges faced:**

1. Ensuring accurate and real-time updates to item quantities across merchant and vendor inventories.
2. Managing unique user data (username, email, phone) with conflict detection and appropriate feedback mechanisms.
3. Designing clean and navigable interfaces for multiple user flows without clutter or confusion.

**Future scope:**

1. Cloud Integration: Shift from local SQLite storage to cloud-based databases for cross-device accessibility and backup.
2. Analytics Dashboard: Add visual inventory insights for both vendors and merchants, including sales trends and restocking alerts.
3. Notifications: Implement real-time push notifications for low stock, successful sales, and item updates.
4. Barcode Scanning: Introduce barcode/QR code features to streamline inventory tracking and item retrieval.

**References**

Android Developer Documentation

<https://developer.android.com/develop>

StackOverflow

<https://stackoverflow.com/>