

UCS1711

Mobile Application Development

Lab

Mini Project

Group Members

P T Jayannthan – 205001049

Kishaanth S – 205001054

Koushik Viswanath S - 205001055

Problem Statement:

The goal of this Android application is to provide users with a comprehensive banking management experience. Users can perform various transactions, view their account details, and manage their financial activities seamlessly through the mobile app. The application aims to simplify and enhance the user's banking experience, offering features such as account management, transactions, and financial insights.

Software Requirements Specification:

1. Functional Requirements:

a. User Authentication:

- Users should be able to register a new account.
- Existing users should be able to log in securely.

b. Account Management:

- Users can view account details.
- Add, modify, or delete accounts.

c. Transaction Handling:

- Perform various transactions such as deposits, transfers, and payments.
- View transaction history.

d. Dashboard:

- Display a personalized dashboard with relevant financial information.

2. Non-Functional Requirements:

a. Security:

- User data must be securely stored and transmitted.
- Implement secure authentication measures.

b. Performance:

- The application should be responsive and provide real-time updates.
- Efficient data retrieval and processing.

Identified Functionalities:

1. Login & Registration Module:

- Allows users to register a new account or log in securely.

2. Account Module:

- Enables users to view, add, modify, or delete their accounts.

3. Transaction Module:

- Facilitates various transactions such as deposits, transfers, and payments.
- Provides a transaction history feature.

4. Dashboard Module:

- Displays a personalized dashboard with relevant financial information.

Appropriate Logic:

1. Secure Authentication Logic:

- Implement session management to handle user authentication tokens securely.

2. Logic for Account Management (CRUD operations):

- Design logic for creating a new user account with mandatory fields (first name, last name, country, username, password).
- Implement logic for retrieving and displaying existing user profiles.
- Enable users to update their account information, such as changing passwords or personal details.
- Provide functionality to delete a user account if needed, ensuring proper data integrity and security checks.

3. Transaction Logic with Validations:

- Develop logic for various financial transactions, such as deposits, transfers, and payments.
- Implement validations to ensure that transactions are within acceptable limits.
- Check for sufficient funds before processing transactions like transfers or withdrawals.
- Log and handle transaction failures gracefully, providing clear feedback to users.

4. Responsive and Informative Dashboard:

- Design a dynamic dashboard that adapts to different screen sizes and orientations.
- Utilize relevant UI components to display key information, such as user greetings, time-based messages, and account summaries.
- Fetch and display real-time data, such as transaction history or account balances, in an organized and visually appealing manner.
- Implement intuitive navigation within the dashboard, allowing users to access different sections of the application seamlessly.

Functionalities of each module:

1. LaunchActivity.java:

- Responsibilities:
 1. App Initialization:
 - Handles the initialization of the application, setting up essential components and configurations.
 2. User Navigation:
 - Manages the initial user interface, often serving as the entry point and facilitating navigation to other parts of the application.

2. TransactionFragment.java:

- Responsibilities:
 1. Transaction Processing:
 - Manages the user interface and logic for processing various financial transactions, such as deposits, withdrawals, or transfers.
 2. Transaction History:
 - Displays a history of user transactions, providing insights into past financial activities.

3. DrawerActivity.java:

- Responsibilities:
 1. Navigation Drawer:

- Implements a navigation drawer, allowing users to access different sections or modules of the application.
2. Global App Functions:
 - Manages functionality common across multiple fragments, such as handling user authentication or settings.

4. DashboardFragment.java:

- Responsibilities:
 1. User Dashboard:
 - Creates and manages the main dashboard interface, providing users with personalized information, greetings, and visual elements.
 2. Account Information:
 - Retrieves and displays information about the user's accounts and transactions.

5. CreateProfileFragment.java:

- Responsibilities:
 1. User Profile Creation:
 - Manages the UI and logic for creating a new user profile, collecting necessary information for account setup.
 2. Profile Validation:
 - Validates user inputs, ensuring completeness and uniqueness, and handles the creation of a new user profile.

6. AccountOverviewFragment.java:

- Responsibilities:
 1. Account Overview:
 - Displays an overview of the user's accounts, including balances and recent activities.
 2. Account Management:
 - Allows users to perform account-related actions, such as updating account details or deleting an account.

7. TransferFragment.java:

- Responsibilities:
 1. Fund Transfers:
 - Manages the UI and logic for transferring funds between different user accounts.
 2. Validation:
 - Validates transfer requests, ensuring the availability of funds and handling any errors or exceptions.

8. PaymentFragment.java:

- Responsibilities:
 1. Payment Processing:
 - Handles UI and logic for making payments or financial transactions, possibly involving external entities.
 2. Validation and Authorization:
 - Validates payment details, authorizing transactions securely.

9. ApplicationDB.java:

- Responsibilities:
 1. Database Interaction:

- Manages interactions with the local database, handling tasks like storing and retrieving user profiles, transactions, and other relevant data.
2. Data Persistence:
- Ensures the persistence of user-related and application data between sessions.

System Design:

1. Components:

- Mobile App:
 - User Interface (UI) for various functionalities.
 - Fragments for different modules (e.g., Dashboard, Transaction, Profile).
- Database:
 - Local Database (SQLite) for storing user profiles, account details, and transaction history.
- Server (Optional):
 - If the app communicates with a server for more advanced features or external transactions.

2. User Authentication:

- Description:
 - Secure user login and registration.
- Components:
 - Login Module (LaunchActivity):
 - Validates user credentials against the local database.
 - Registration Module (CreateProfileFragment):
 - Collects user details and creates a new profile in the database.

3. Account Management:

- Description:
 - Create, Read, Update, and Delete (CRUD) operations for user accounts.
- Components:
 - Account Overview Module (AccountOverviewFragment):
 - Displays account details and recent activities.
 - Profile Creation Module (CreateProfileFragment):
 - Manages the creation of new user profiles.

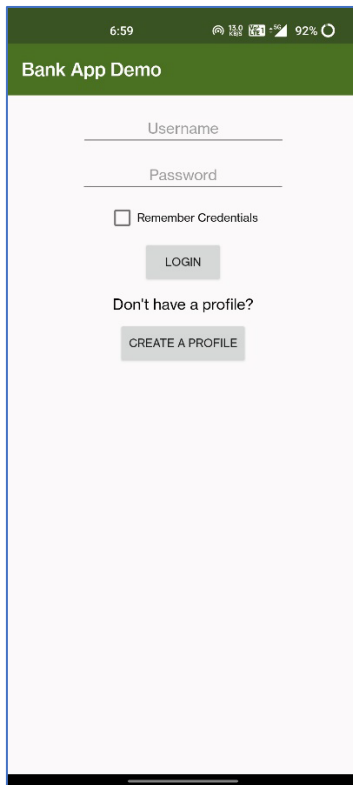
4. Transaction Processing:

- Description:
 - Facilitates financial transactions and maintains transaction history.
- Components:
 - Transaction Module (TransactionFragment):
 - Handles deposit, withdrawal, and transfer functionalities.
 - Database (ApplicationDB):
 - Stores and retrieves transaction data.

Deployment:

- Build the Android application for deployment.
- Install the application on a mobile device for testing and usage.
- Ensure the application functions seamlessly on the target device.

Output:



6:59 92%

Bank App Demo

Username

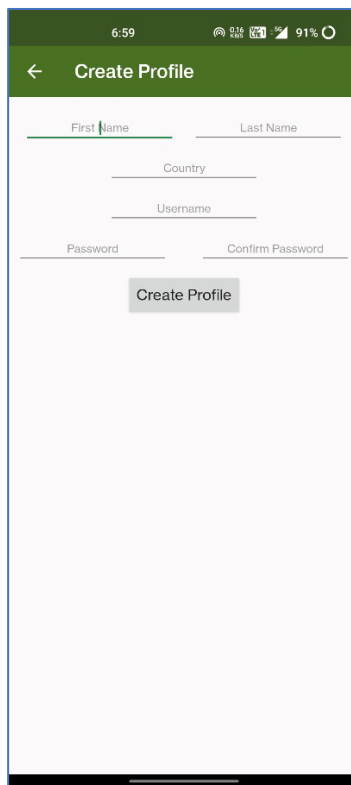
Password

☐ Remember Credentials

LOGIN

Don't have a profile?

CREATE A PROFILE



6:59 91%

Create Profile

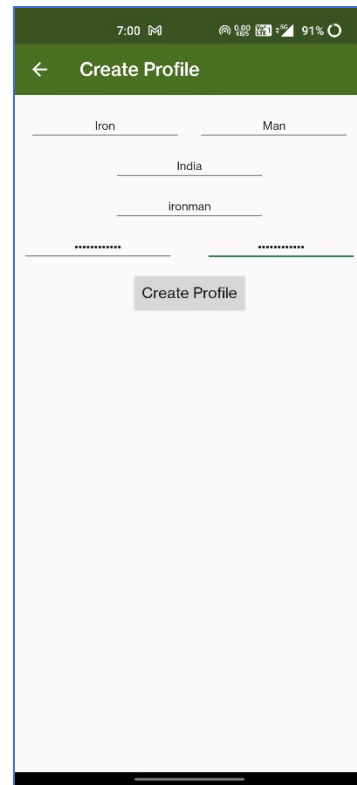
First Name Last Name

Country

Username

Password Confirm Password

Create Profile



7:00 91%

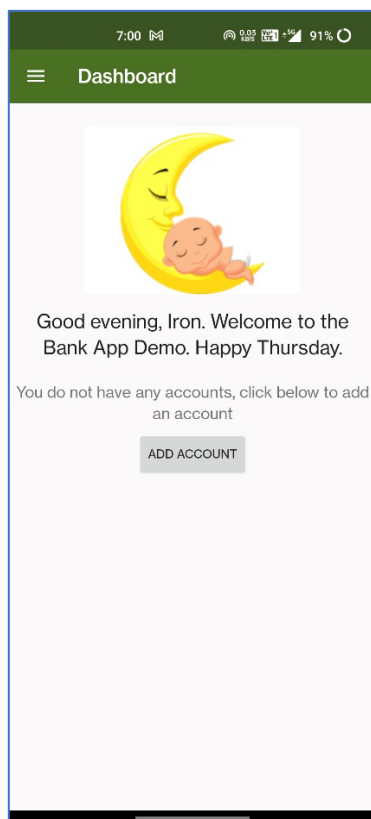
Create Profile

Iron Man

India


ironman

Create Profile



7:00 91%

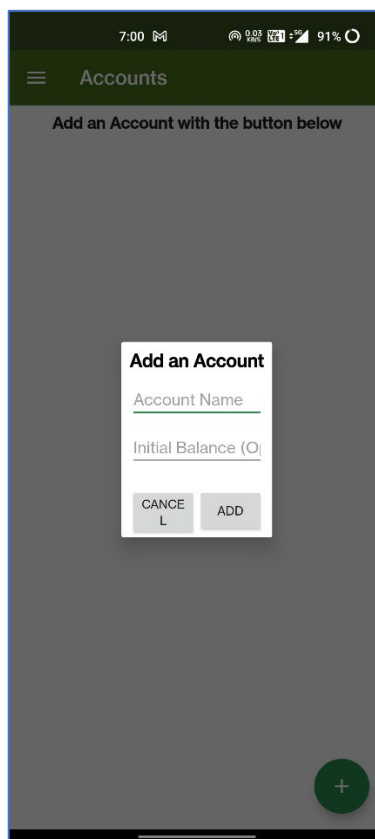
Dashboard



Good evening, Iron. Welcome to the Bank App Demo. Happy Thursday.

You do not have any accounts, click below to add an account

ADD ACCOUNT



7:00 91%

Accounts

Add an Account with the button below

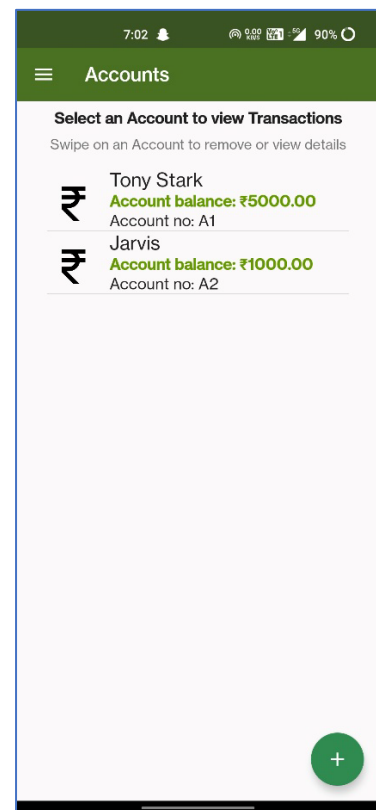
Add an Account

Account Name

Initial Balance (0)

CANCEL ADD

+



7:02 90%

Accounts

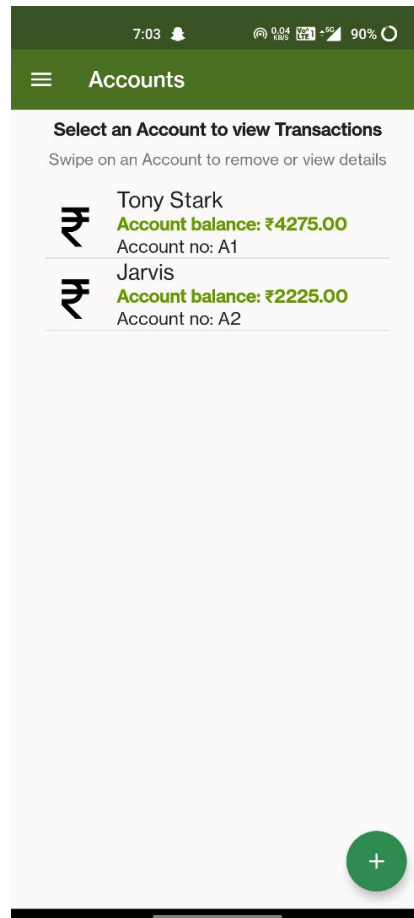
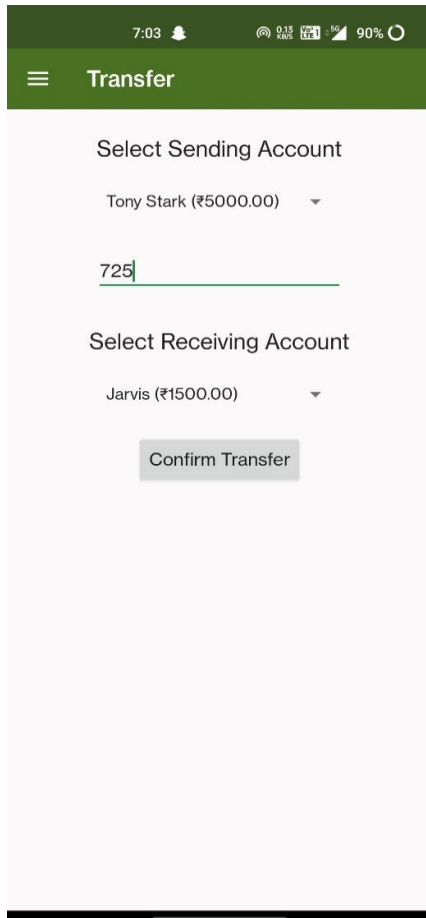
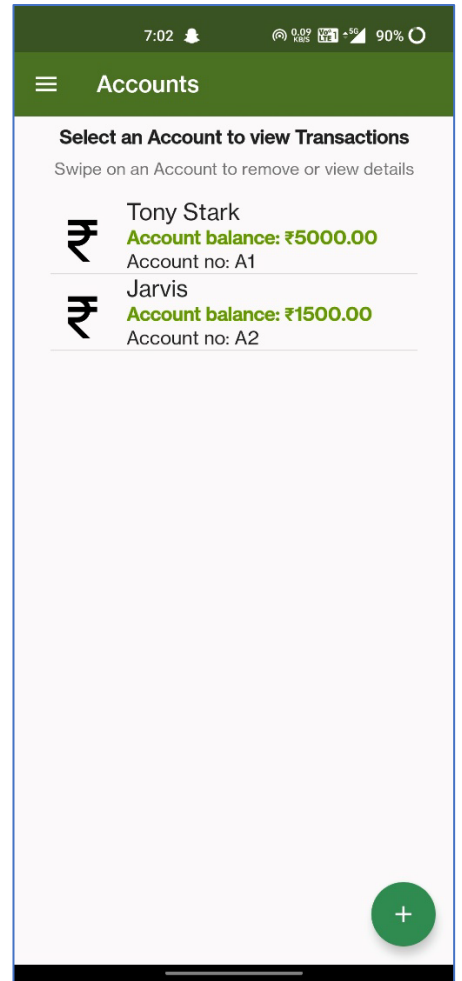
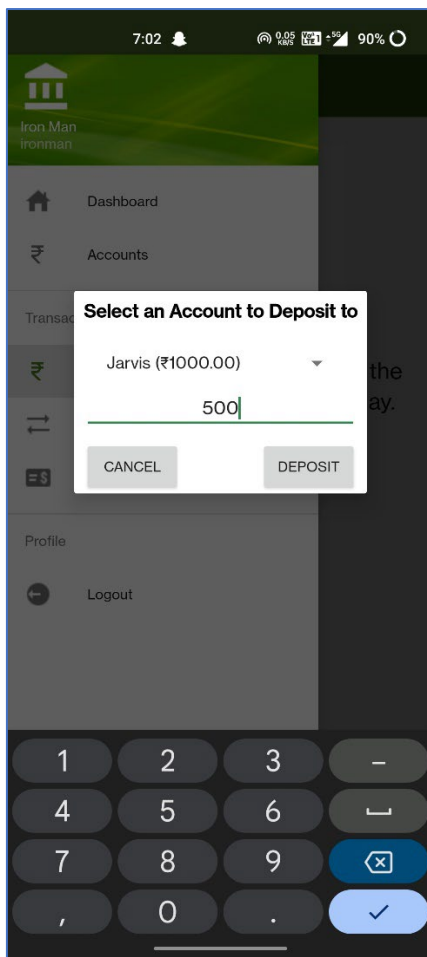
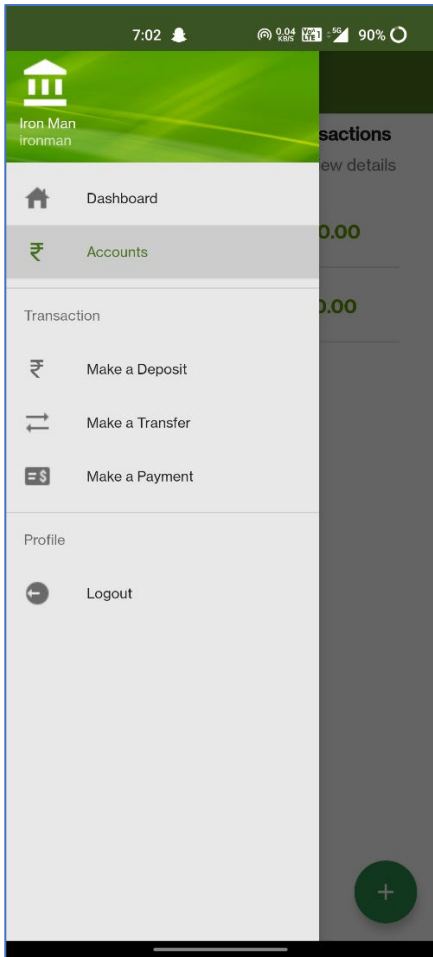
Select an Account to view Transactions

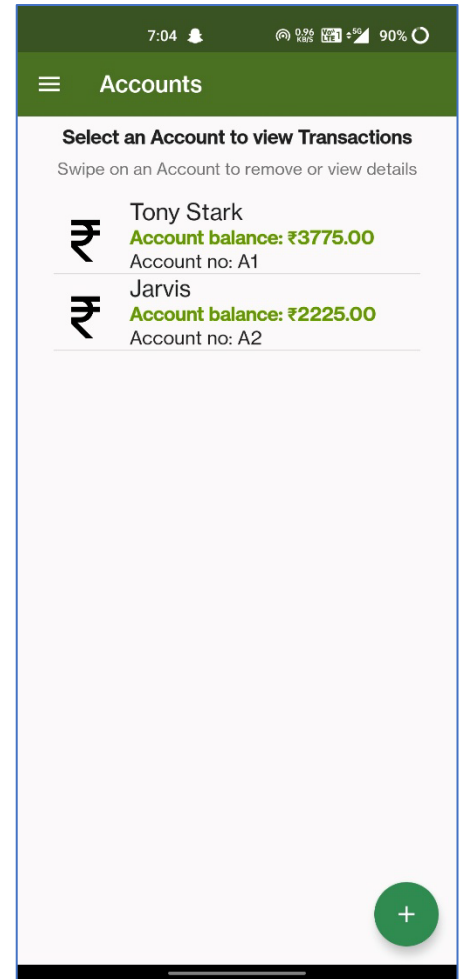
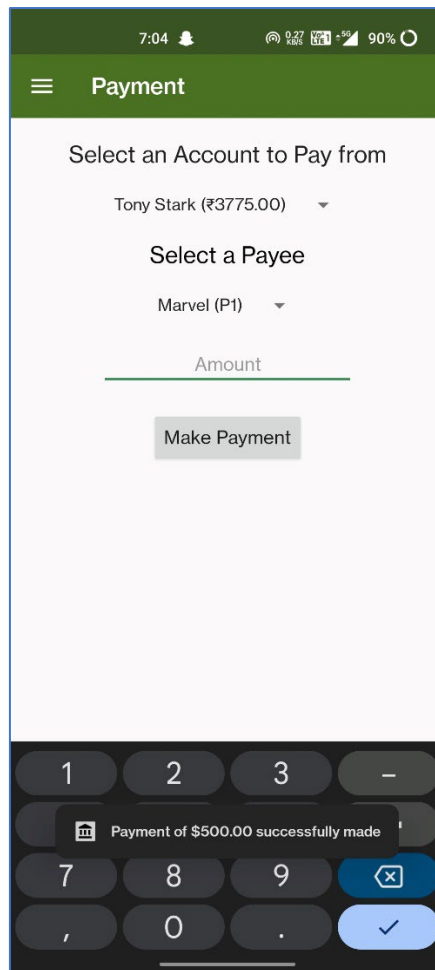
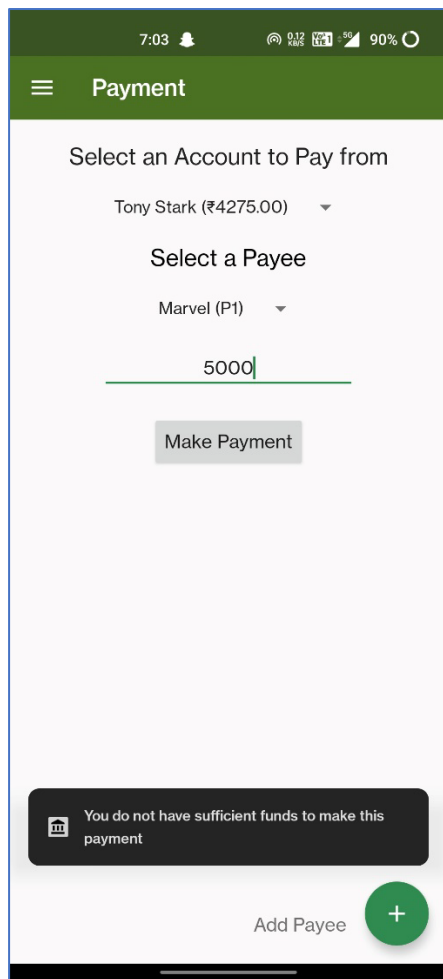
Swipe on an Account to remove or view details

₹ Tony Stark
Account balance: ₹5000.00
Account no: A1

₹ Jarvis
Account balance: ₹1000.00
Account no: A2

+





Best Practices:

1. Use meaningful variable names and comments to improve code readability.
2. Utilize XML layouts for defining UI components to maintain consistency.
3. Organize code logically and use comments for better code maintenance.
4. Implement error handling to handle unexpected scenarios gracefully.
5. Ensure smooth user interactions and animations for a better user experience.
6. Follow Android coding conventions and design guidelines.