**Send Money Application**:

**Tech Stack**

* **Frontend**: Flutter (for cross-platform mobile app development)
* **Backend**: Firebase or Node.js with Express.js (for managing user authentication, wallet balance, and transactions)
* **Database**: Firebase Firestore or MySQL/PostgreSQL

**Screen Details**

**1. Login Screen**

**UI Elements**:

* Textfields: Username, Password
* Buttons: Login, Logout

**Business Logic**:

* **Login**: Validate credentials via backend (Firebase Auth or a custom endpoint).

**2. Wallet Balance Screen**

**UI Elements**:

* **TextView**: Display the wallet balance (default: "500.00 PHP").
* **Show/Hide Icon**: Toggles between showing and masking the wallet balance (e.g., "\*\*\*\*\*\*").
* **Buttons**: Send Money, View Transactions.

**Business Logic**:

* Fetch balance from the database after login.
* Show/hide functionality toggles the visibility of the balance text.
* Buttons navigate to respective screens: Send Money → **Screen 3**, View Transactions → **Screen 4**.

**3. Send Money Screen**

**UI Elements**:

* **Textfield**: Accepts only numeric input (restrict alphabets and special characters).
* **Submit Button**: Validates the amount and submits the transaction.
* **Bottom Sheet**: Displays "Success" or "Error" based on the transaction status.

**Business Logic**:

* Validate the amount (non-empty, positive value, less than or equal to the balance).
* Deduct the amount from the user’s wallet on successful transaction.
* Save the transaction to the database.
* Bottom sheet displays:
  + Success → “Amount sent successfully!”
  + Error → “Invalid input or insufficient funds.”

**4. Transaction History Screen**

**UI Elements**:

* **ListView**: Displays a list of transactions with details like amount, recipient, and timestamp.
* **Logout Button**: Logs out and redirects to the login screen.

**Business Logic**:

* Fetch transactions from the database after navigating to this screen.
* Display transactions in descending order (most recent first).

**Folder/File Structure (Flutter)**

lib/

├── main.dart (App entry point)

├── screens/

│ ├── login\_screen.dart

│ ├── wallet\_balance\_screen.dart

│ ├── send\_money\_screen.dart

│ ├── transaction\_history\_screen.dart

├── widgets/

│ ├── custom\_button.dart

│ ├── transaction\_card.dart

├── services/

│ ├── auth\_service.dart (Handles login/logout)

│ ├── wallet\_service.dart (Handles balance & transactions)

│ ├── database\_service.dart (Interacts with the database)

├── models/

│ ├── transaction.dart (Transaction model)

**Database Structure (Example with Firebase Firestore)**

{

"users": {

"userID123": {

"username": "john\_doe",

"password": "hashed\_password",

"balance": 500.00,

"transactions": [

{

"id": "txn123",

"amount": 100,

"timestamp": "2025-01-23T10:00:00Z"

},

{

"id": "txn124",

"amount": 50,

"timestamp": "2025-01-24T15:30:00Z"

}

]

}

}

}

**Basic Navigation Flow**

1. **Login Screen** → On successful login → **Wallet Balance Screen**
2. **Wallet Balance Screen**
   * Send Money → **Send Money Screen**
   * View Transactions → **Transaction History Screen**
3. **Send Money Screen** → After transaction → Bottom Sheet (Success/Error) → Redirect back to Wallet Balance Screen.
4. **Transaction History Screen** → View past transactions.

**Backend API Design**

1. **Login API**  
   **Endpoint**: POST /api/login  
   **Payload**: { "username": "user", "password": "pass" }  
   **Response**: { "success": true, "balance": 500.00 }
2. **Send Money API**  
   **Endpoint**: POST /api/send  
   **Payload**: { "userId": "123", "amount": 100 }  
   **Response**: { "success": true, "newBalance": 400.00 }
3. **Transaction History API**  
   **Endpoint**: GET /api/transactions?userId=123  
   **Response**:

[

{"amount": 100, "timestamp": "2025-01-27 13:00", "type": "received"},

  {"amount": 50, "timestamp": "2025-01-27 15:30", "type": "sent"},

]