INFORMATION MANAGEMENT FINAL PROJECT

GCASH

USER INTERFACE OVERVIEW, ENTITY RELATIONSHIP DIAGRAM, AND PURPOSES

MEET THE MAN BEHIND THE APP

National University - Manila (‘27)

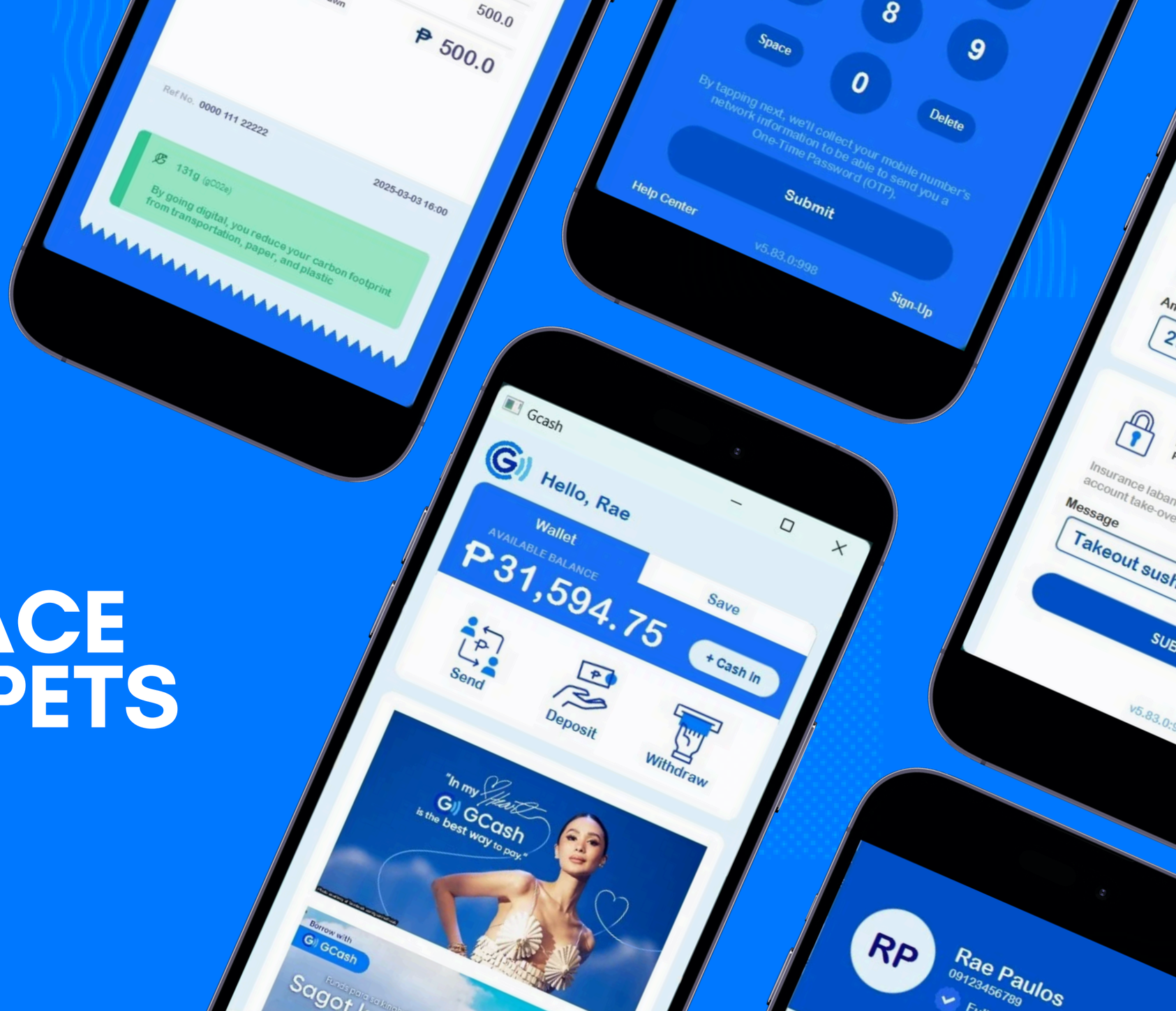
Bachelor of Science in Computer Science

his project is a Java-based e-wallet application with an SQL database, designed to provide a seamless and efficient digital transaction experience. Inspired by the functionality of GCash, it incorporates three key features: Express Send, allowing users to transfer funds instantly; Deposit, enabling users to add money to their accounts; and Withdrawal, facilitating secure cash-out transactions.

The system is developed as part of the Information Management course, taught by @robitussin on GitHub. It emphasizes secure and efficient financial transactions while demonstrating the integration of Java-based application development with database management using SQL. The project aims to enhance the understanding of digital financial services and database implementation while ensuring a user-friendly experience.

@rpaulos

GCASH
USER INTERFACE & CODE SNIPPETS



LOGIN PAGE

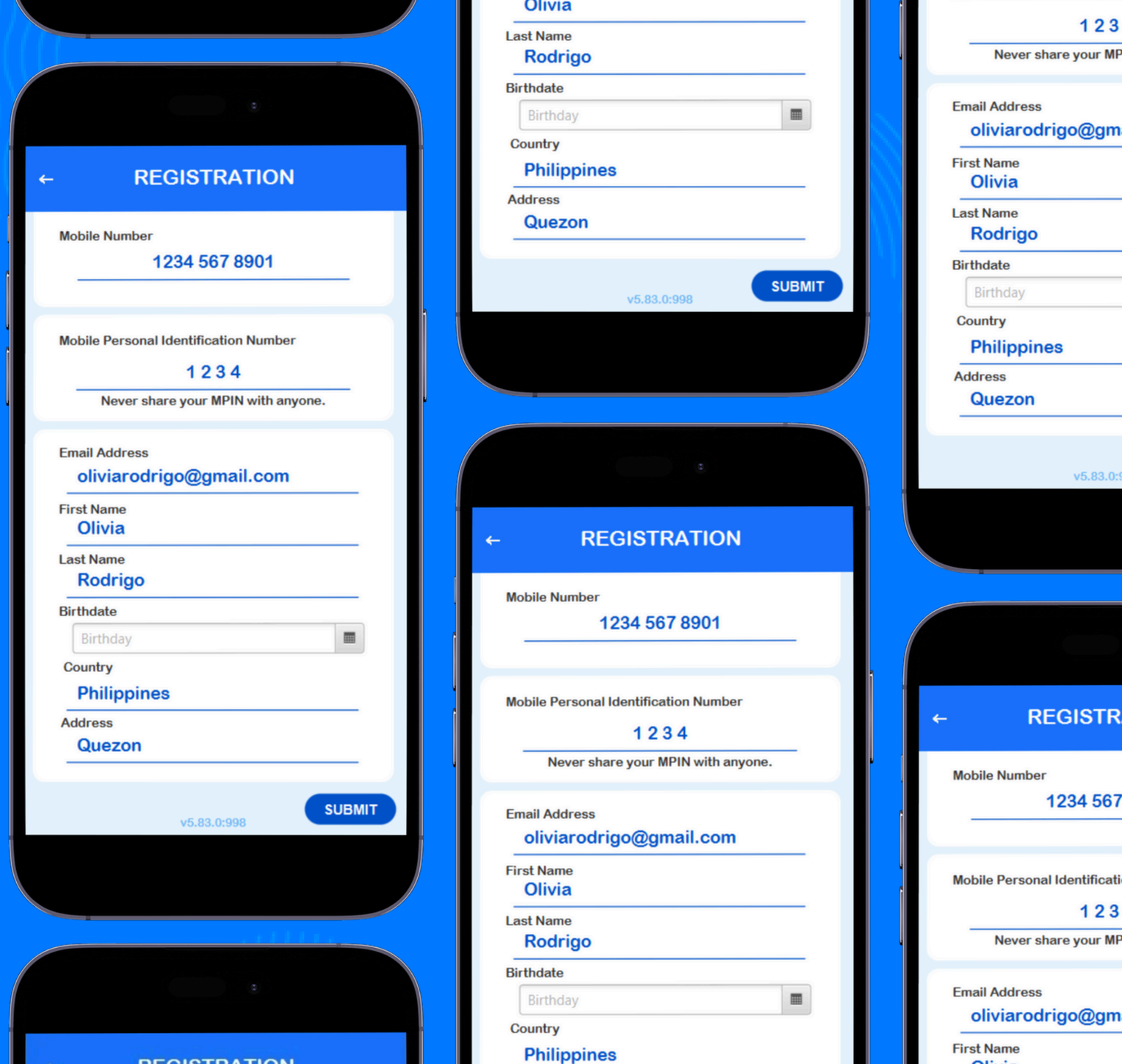
Login page is where the User logs their specific number account and PIN to log in to their Gcash. Only authorized individuals may log in and helps safeguard their financial information.

```java
public static boolean validateMobileNumber(String phone_number){

    getInstance();
    String query = "SELECT * FROM users WHERE phone_number = '" + phone_number + "'";

    System.out.println(query);

    ResultSet result = handler.execQuery(query);
    try {
        if (result.next()) {
            return true;
        }
    }
    catch (SQLException e){
        e.printStackTrace();
    }
    return false;
}
```

```java
public static boolean validateMobileNumberAndMPIN(String phone_number, String PIN){
    getInstance();
    String query = "SELECT * FROM users WHERE phone_number = '" + phone_number + "' AND PIN = '" + PIN + "'";

    System.out.println(query);

    ResultSet result = handler.execQuery(query);
    try {
        if (result.next()) {
            return true;
        }
    }
    catch (SQLException e){
        e.printStackTrace();
    }
    return false;
}
```

SIGN UP PAGE

Is where the User creates their account as well their designated numbers.



```java
public static boolean addUser(User user) {
    try {
        pstatement = getDBConnection().prepareStatement("INSERT INTO users (phone_number, first_name, last_name, email_address, PIN, birthdate, country, address) VALUES (?, ?, ?, ?, ?, ?, ?, ?)");
        pstatement.setString(1, user.getPhone_number());
        pstatement.setString(2, user.getFirst_name());
        pstatement.setString(3, user.getLast_name());
        pstatement.setString(4, user.getEmail_address());
        pstatement.setString(5, user.getPIN());
        pstatement.setString(6, user.getBirthdate());
        pstatement.setString(7, user.getCountry());
        pstatement.setString(8, user.getAddress());

        return pstatement.executeUpdate() > 0;
    } catch (Exception e) {
        e.printStackTrace();
    }
    return false;
}
```

HOME PAGE

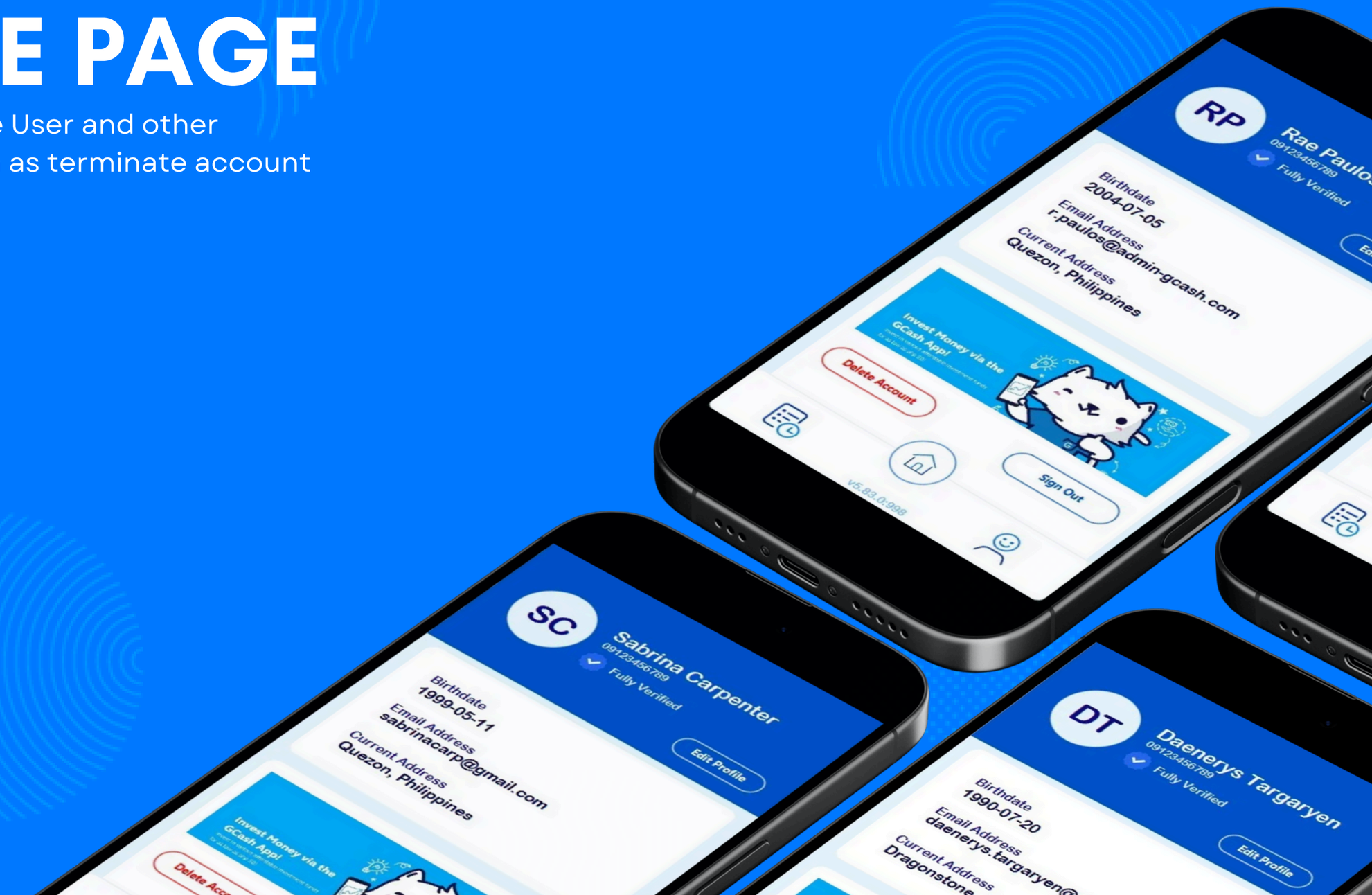
Where it serves the main important part of the UI. Providing the User information such as balance, sending money and receiving money.

```java
public static String getFirstName(String phone_number) {
    String query = "SELECT first_name FROM users WHERE phone_number = ?";
    String first_name = null;
    Connection conn = null;
    PreparedStatement stmt = null;
    ResultSet result = null;

    try {
        conn = getDBConnection();
        stmt = conn.prepareStatement(query);
        stmt.setString(1, phone_number);
        result = stmt.executeQuery();

        if(result.next()) {
            first_name = result.getString("first_name");
        }

    } catch (SQLException e) {
            e.printStackTrace();
        }
    return first_name;
}
```

```java
public static float getUserBalance(String phone_number) {
    String query = "SELECT balance FROM wallet WHERE phone_number = ?";
    float balance = 0.0f;
    Connection conn = null;
    PreparedStatement stmt = null;
    ResultSet result = null;

    try {
        conn = getDBConnection();
        stmt = conn.prepareStatement(query);
        stmt.setString(1, phone_number);
        result = stmt.executeQuery();

        if(result.next()) {
            balance = result.getFloat("balance");
        }

    } catch (SQLException e) {
            e.printStackTrace();
        }
    return balance;
}
```

PROFILE PAGE

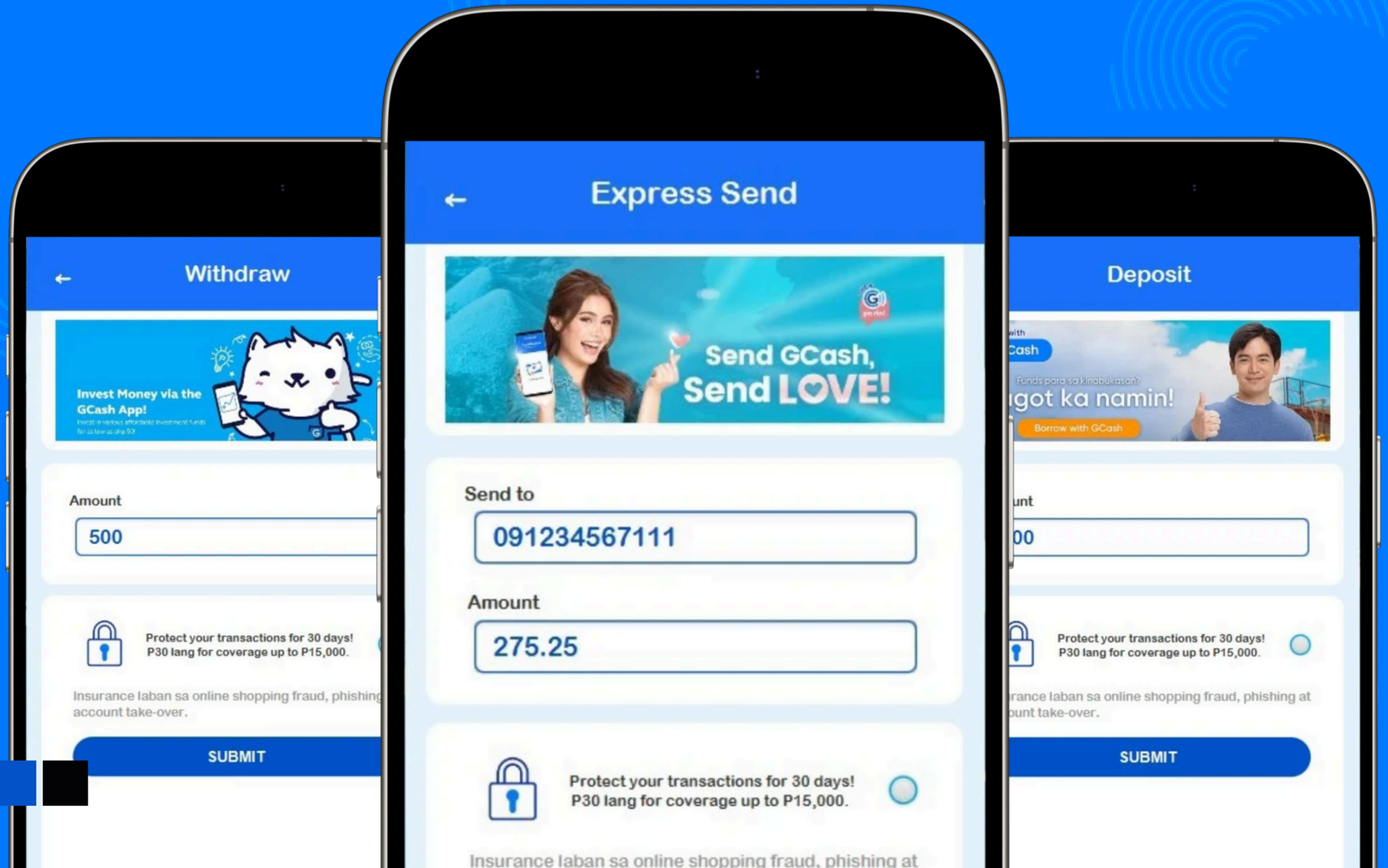
The information of the User and other account settings such as terminate account or log out.



EDIT PROFILE

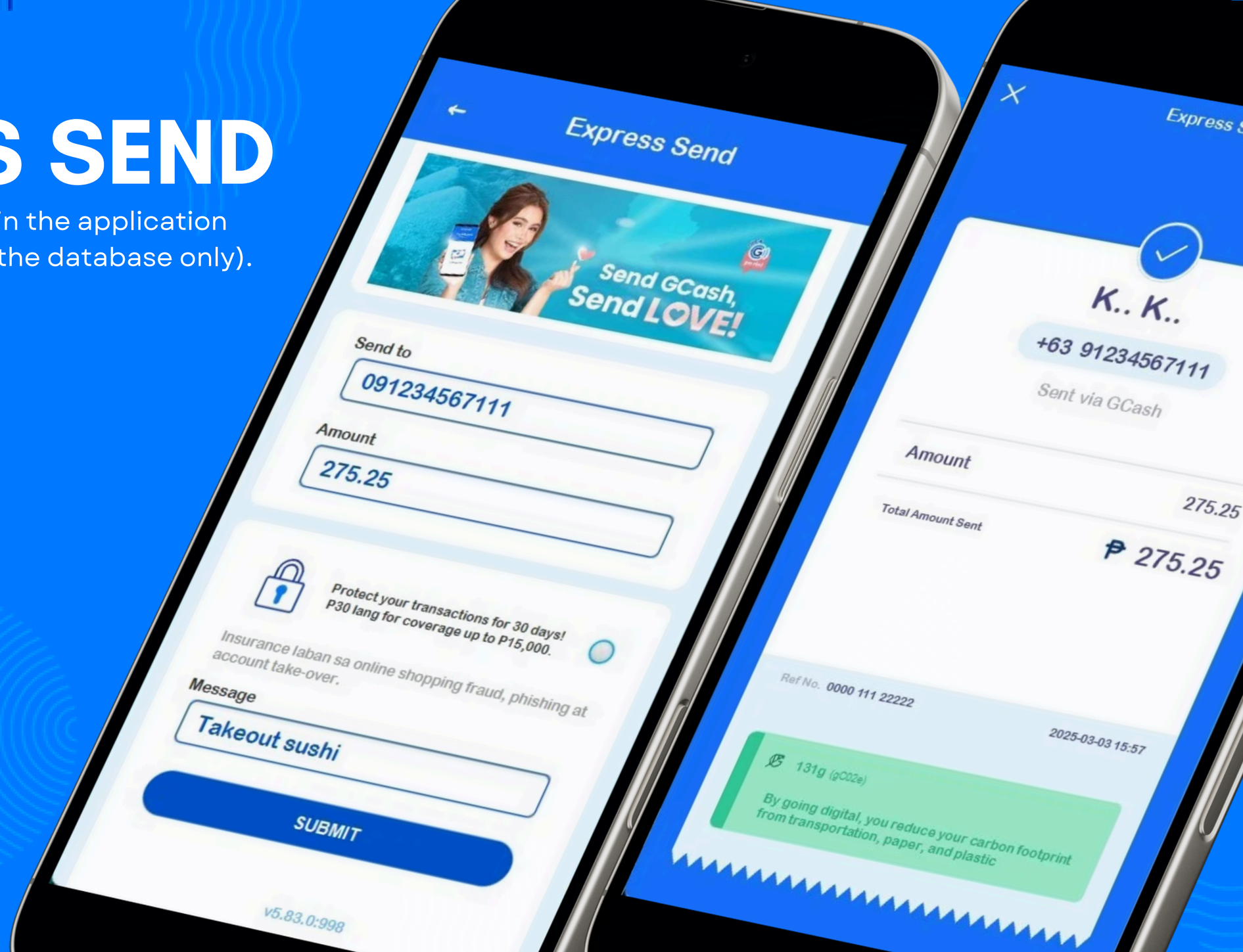
The user may edit his/her account information.

MAIN FEATURES



EXPRESS SEND

User can send money within the application
to other accounts (Within the database only).



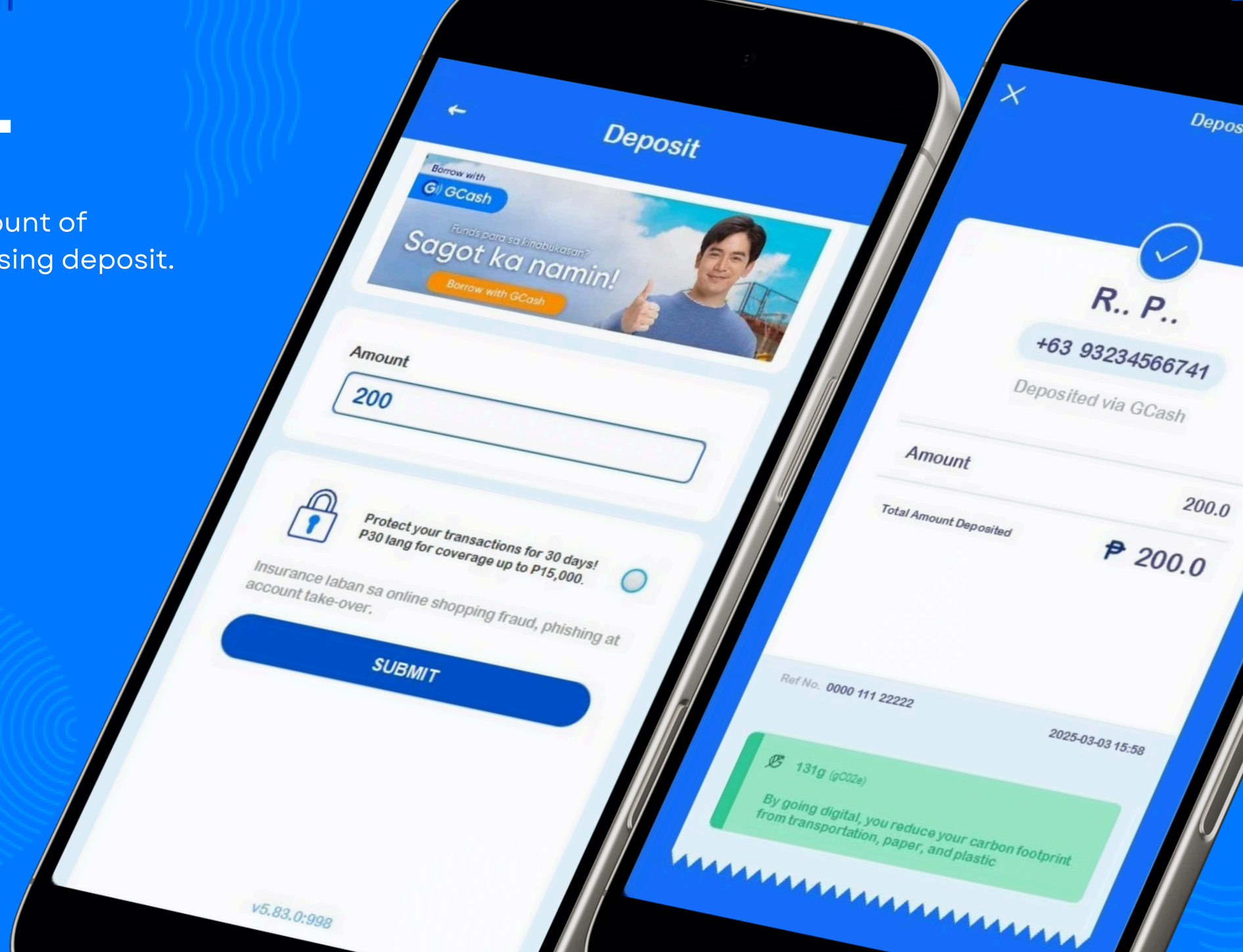
```java
public static float expressSend(String numberToSendTo, float amountToSend) {
    String query = "UPDATE wallet SET balance = balance + ? WHERE phone_number = ?";
    float balance = 0.0f;
    Connection conn = null;
    PreparedStatement stmt = null;

    try {
        conn = getDBConnection();
        stmt = conn.prepareStatement(query);
        stmt.setFloat(1, amountToSend);
        stmt.setString(2, numberToSendTo);

        int affectedRows = stmt.executeUpdate();

    } catch (SQLException e) {
        e.printStackTrace();
    }
    return balance;
}
```

```java
public static void negateBalance(Float negateFromBalance, String myNumber) {
    String query = "UPDATE wallet SET balance = balance - ? WHERE phone_number = ?";
    Connection conn = null;
    PreparedStatement stmt = null;

    try {
        conn = getDBConnection();
        stmt = conn.prepareStatement(query);
        stmt.setFloat(1, negateFromBalance);
        stmt.setString(2, myNumber);

        int affectedRows = stmt.executeUpdate();

    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

```java
public static void recordExpressSend(String sender_number, String receiver_number, Float amount) {
    String query = "INSERT INTO send_transactions (sender_number, receiver_number, amount) VALUES (?, ?, ?);";
    Connection conn = null;
    PreparedStatement stmt = null;

    try {
        conn = getDBConnection();
        stmt = conn.prepareStatement(query);
        stmt.setString(1, sender_number);
        stmt.setString(2, receiver_number);
        stmt.setFloat(3, amount);
        int affectedRows = stmt.executeUpdate();

    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

DEPOSIT

The User may add an amount of money to their account using deposit.

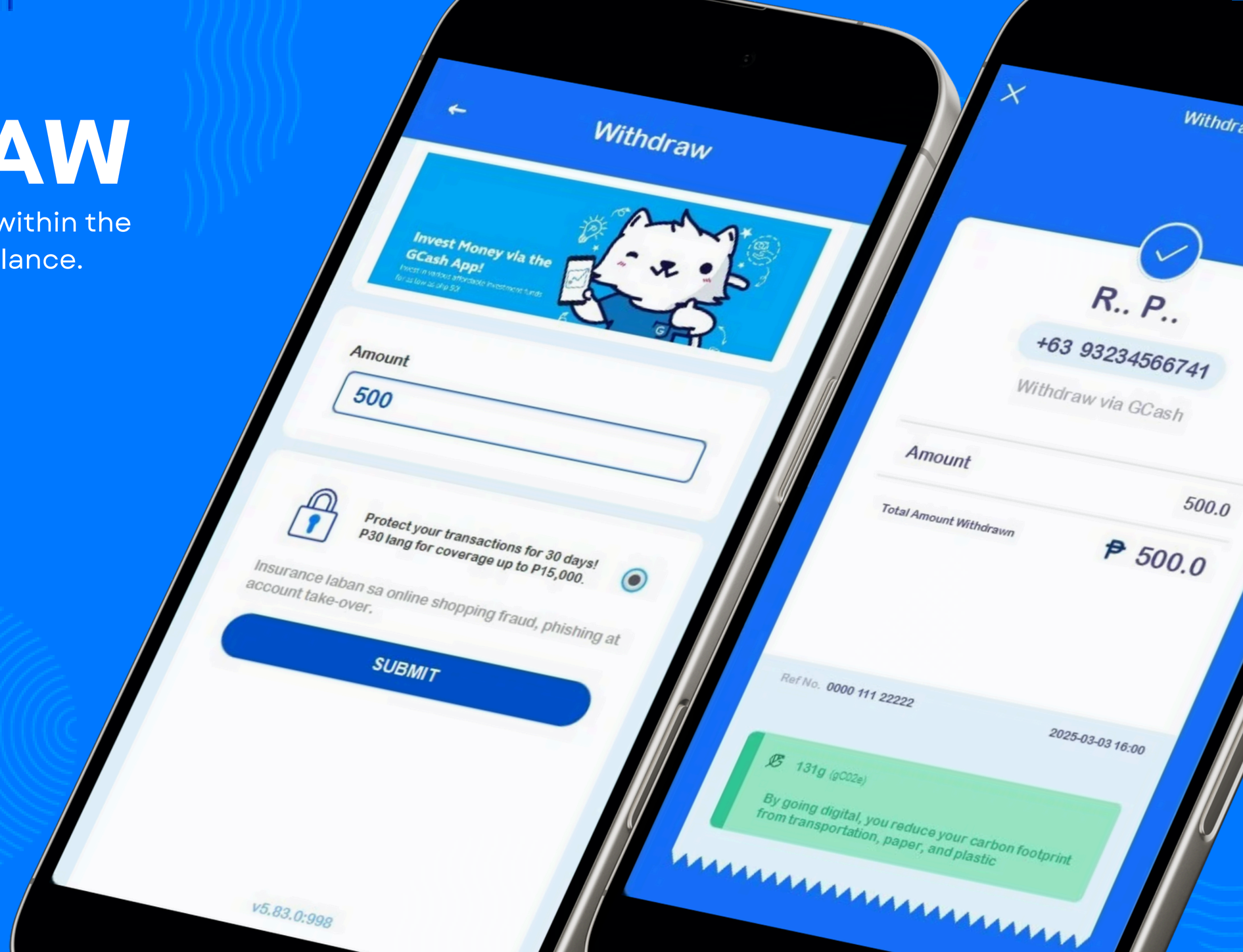


```java
public static void deposit(Float addToBalance, String phone_number) {
    String query = "UPDATE wallet SET balance = balance + ? WHERE phone_number = ?";
    Connection conn = null;
    PreparedStatement stmt = null;

    try {
        conn = getDBConnection();
        stmt = conn.prepareStatement(query);
        stmt.setFloat(1, addToBalance);
        stmt.setString(2, phone_number);
        int affectedRows = stmt.executeUpdate();

    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

```java
public static void recordDeposit(String phone_number, Float amount) {
    String query = "INSERT INTO deposit_transactions (depositor_number, amount) VALUES (?, ?);";
    Connection conn = null;
    PreparedStatement stmt = null;

    try {
        conn = getDBConnection();
        stmt = conn.prepareStatement(query);
        stmt.setString(1, phone_number);
        stmt.setFloat(2, amount);
        int affectedRows = stmt.executeUpdate();

    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

WITHDRAW

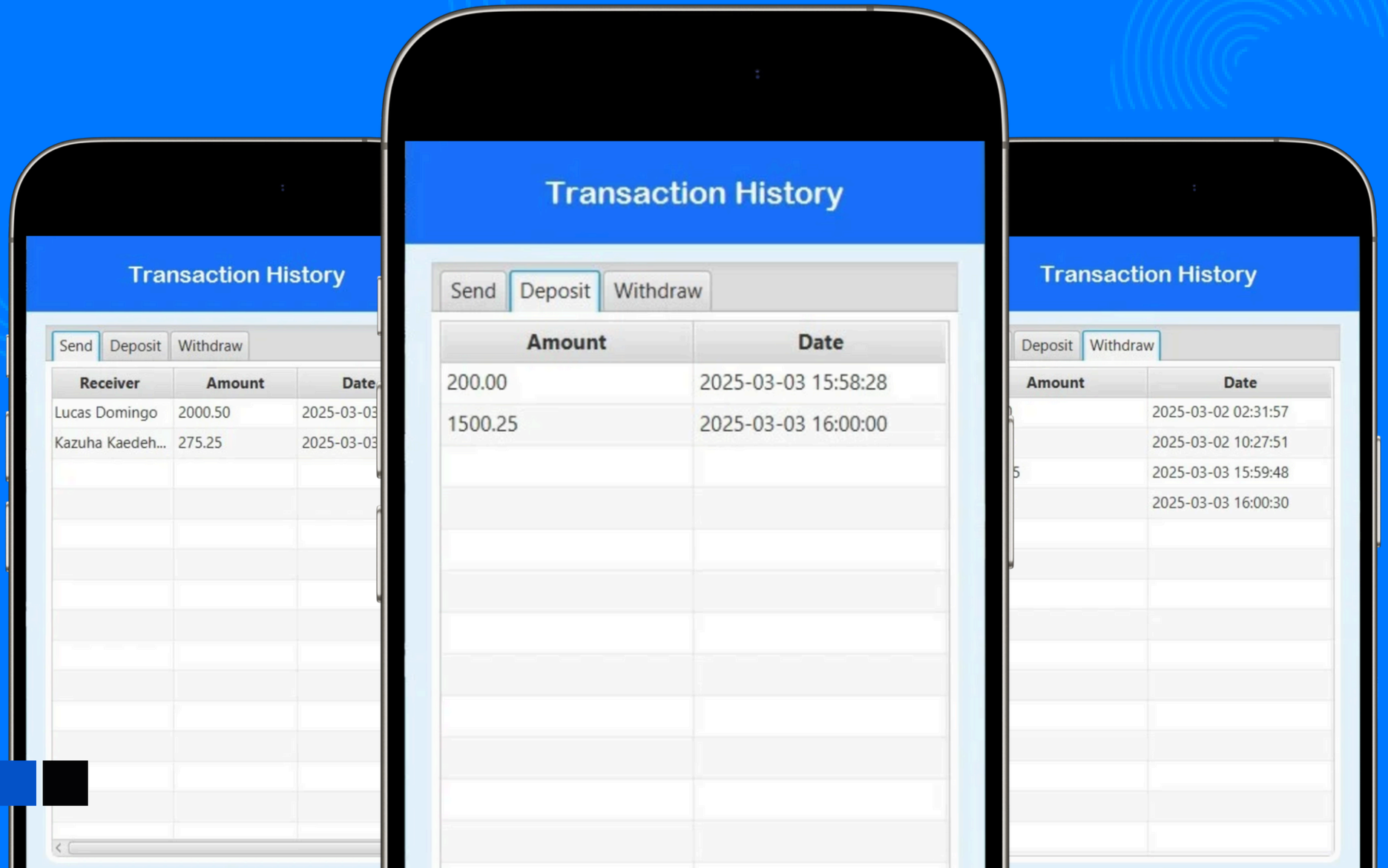
User may withdraw cash within the range of their account balance.



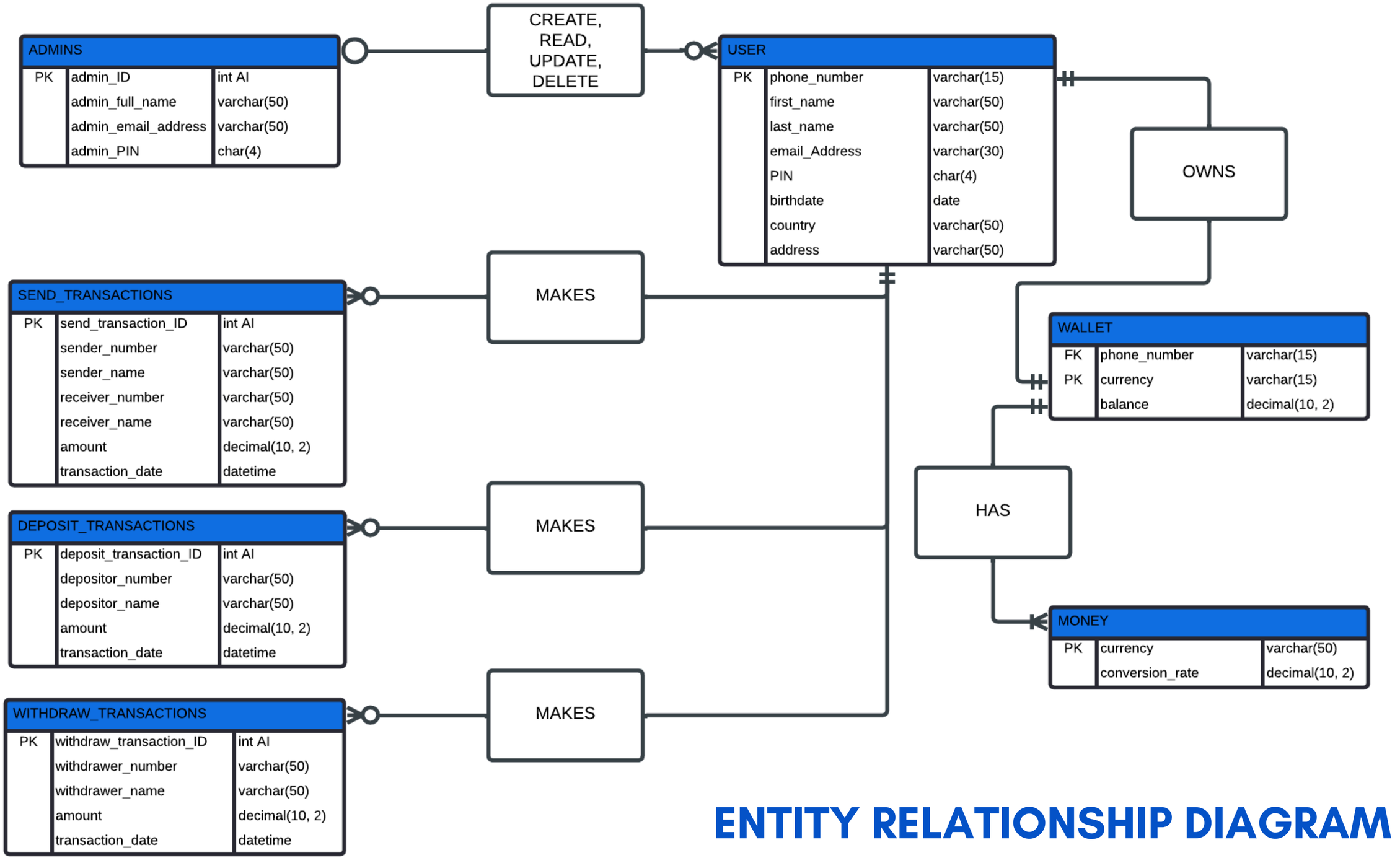
```java
public static void withdraw(Float negateFromBalance, String phone_number) {
    String query = "UPDATE wallet SET balance = balance - ? WHERE phone_number = ?";
    Connection conn = null;
    PreparedStatement stmt = null;

    try {
        conn = getDBConnection();
        stmt = conn.prepareStatement(query);
        stmt.setFloat(1, negateFromBalance);
        stmt.setString(2, phone_number);
        int affectedRows = stmt.executeUpdate();

    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

```java
public static void recordWithdraw(String phone_number, Float amount) {
    String query = "INSERT INTO withdraw_transactions (withdrawer_number, amount) VALUES (?, ?)";
    Connection conn = null;
    PreparedStatement stmt = null;

    try {
        conn = getDBConnection();
        stmt = conn.prepareStatement(query);
        stmt.setString(1, phone_number);
        stmt.setFloat(2, amount);
        int affectedRows = stmt.executeUpdate();

    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

TRANSACTION HISTORY

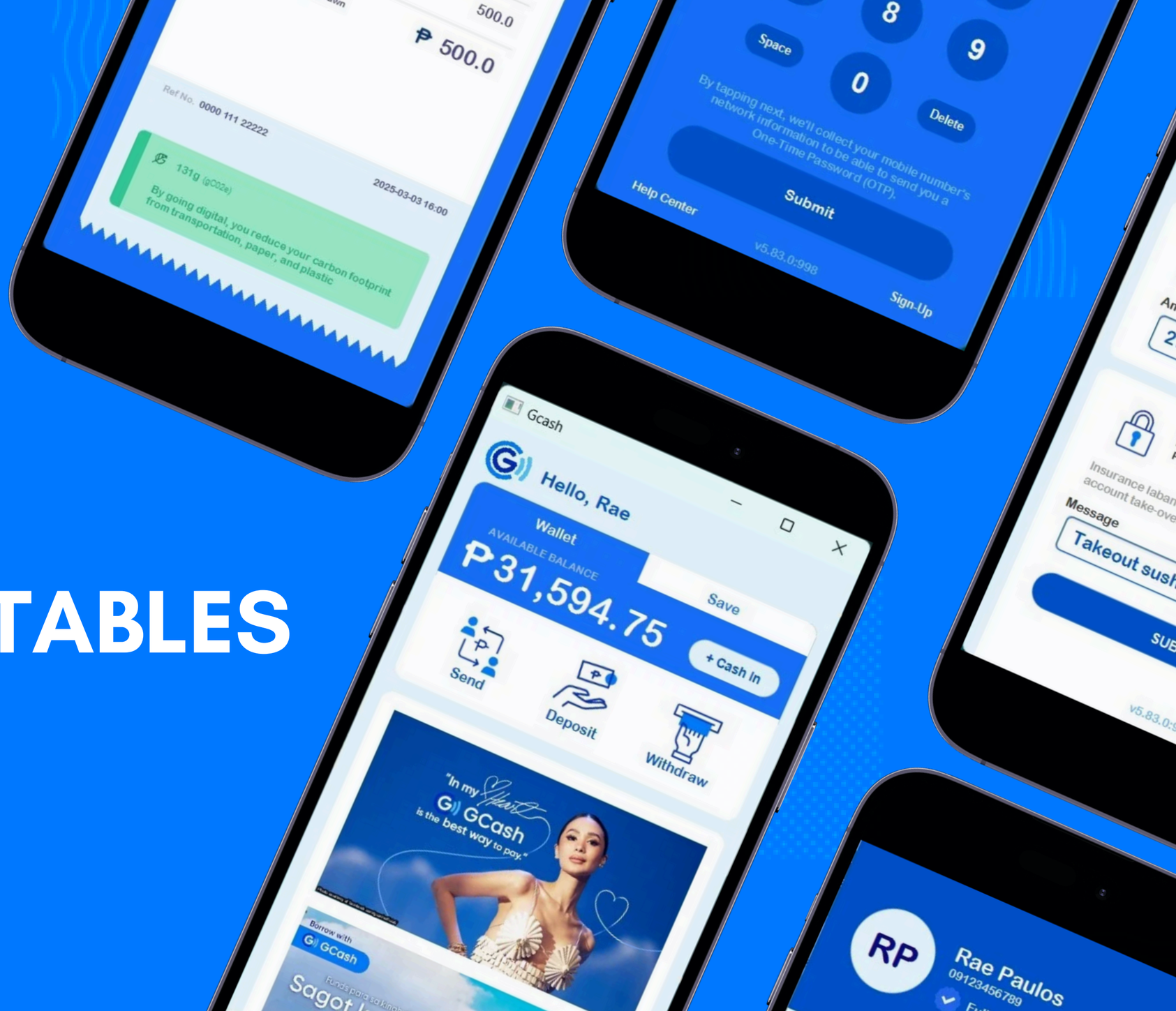


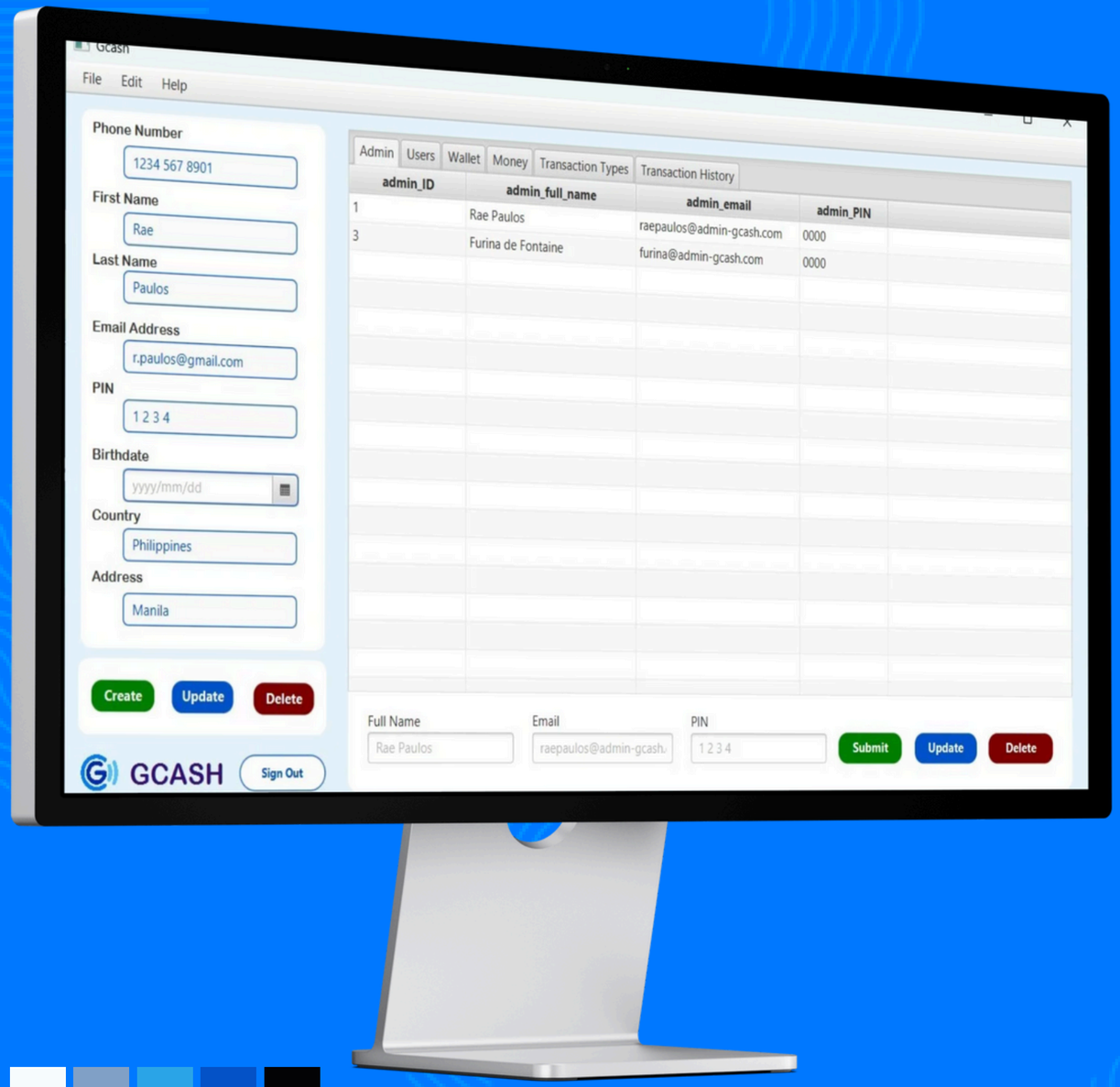
GCASH
ENTITY RELATIONSHIP DIAGRAM



ENTITY RELATIONSHIP DIAGRAM

GCASH
PURPOSE OF TABLES

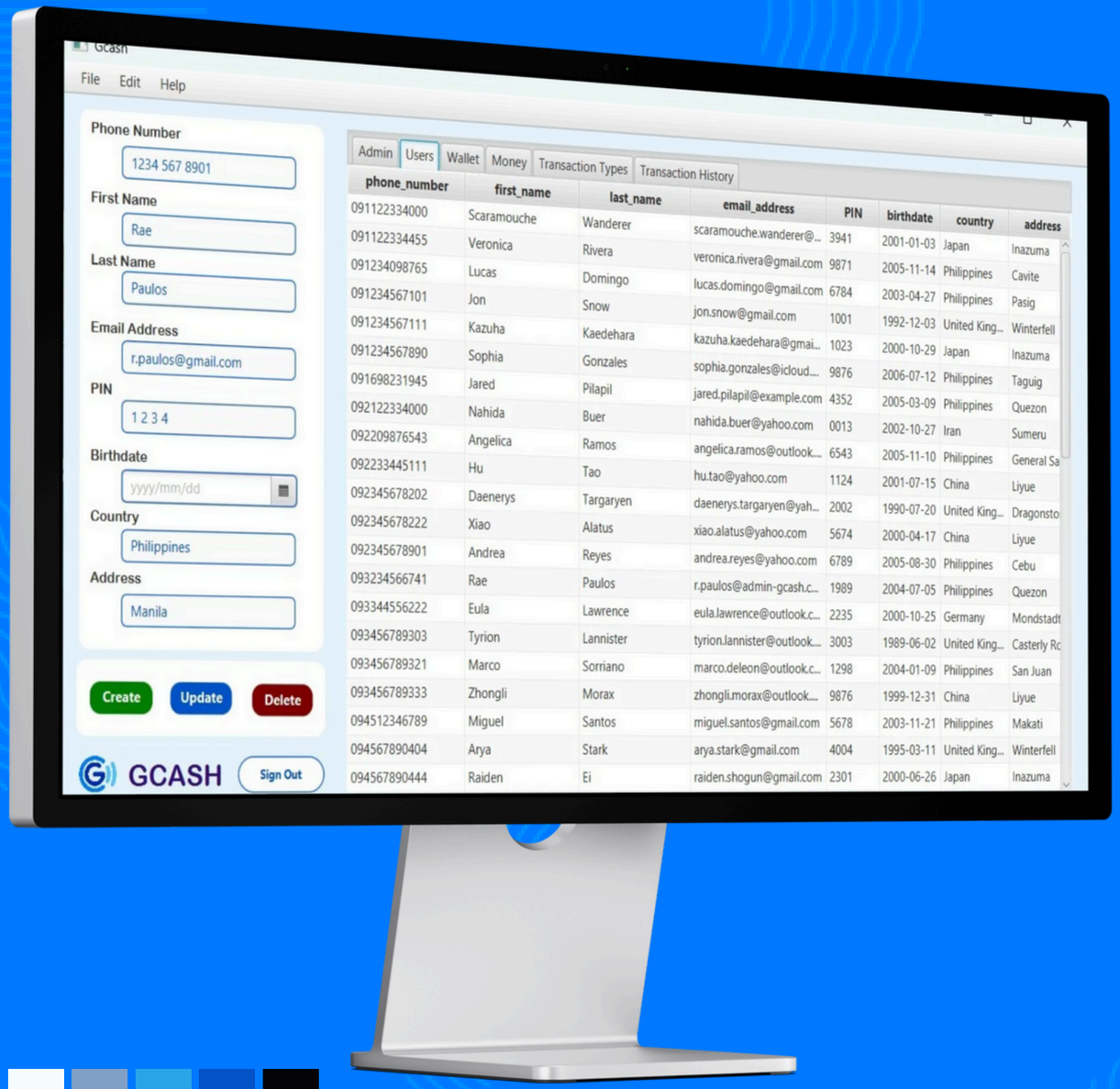




ADMIN TABLE

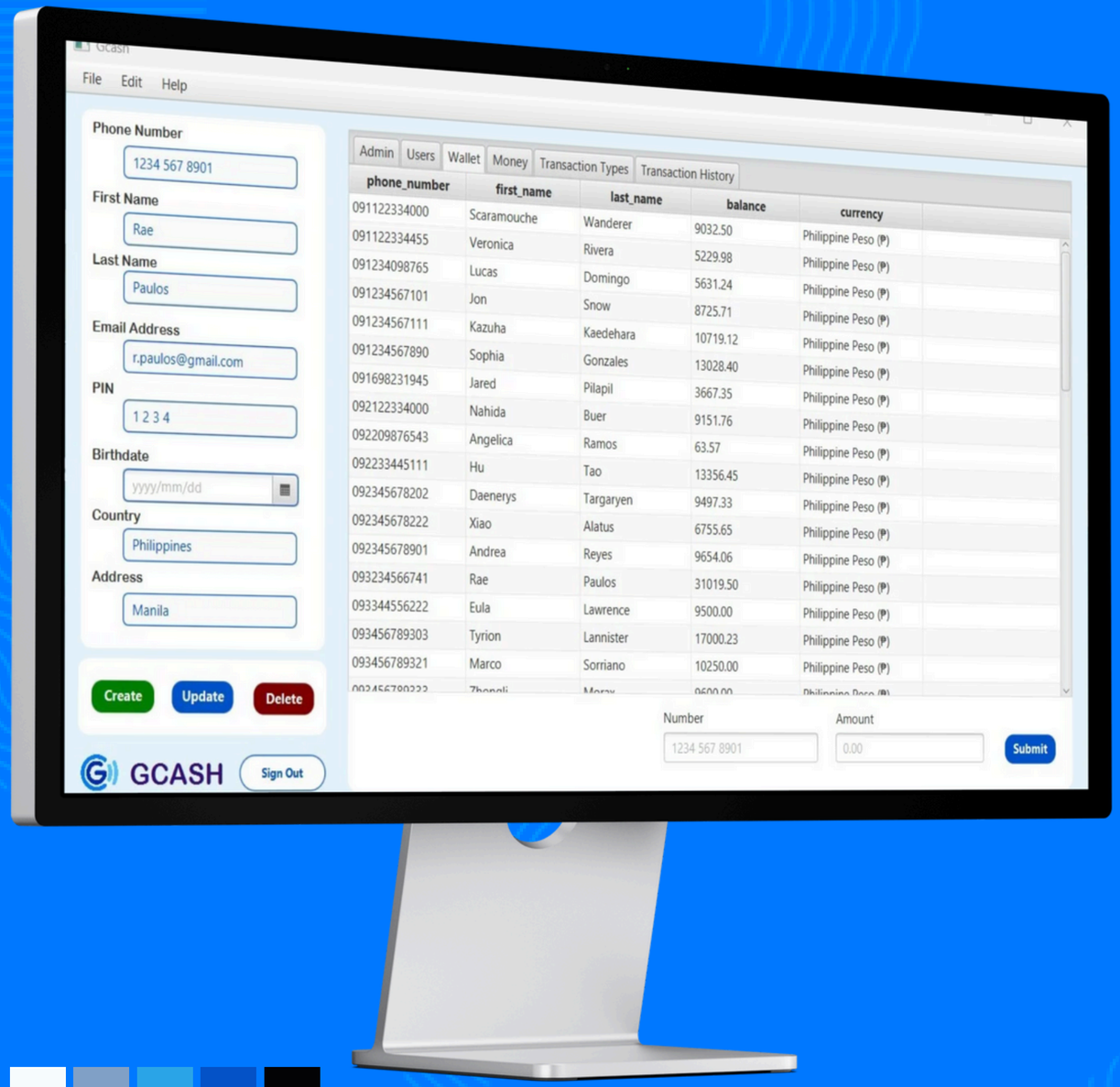
Admins are those capable of creating, reading, updating, and deleting accounts.

They are the only ones who has access to the admin side of Gcash.



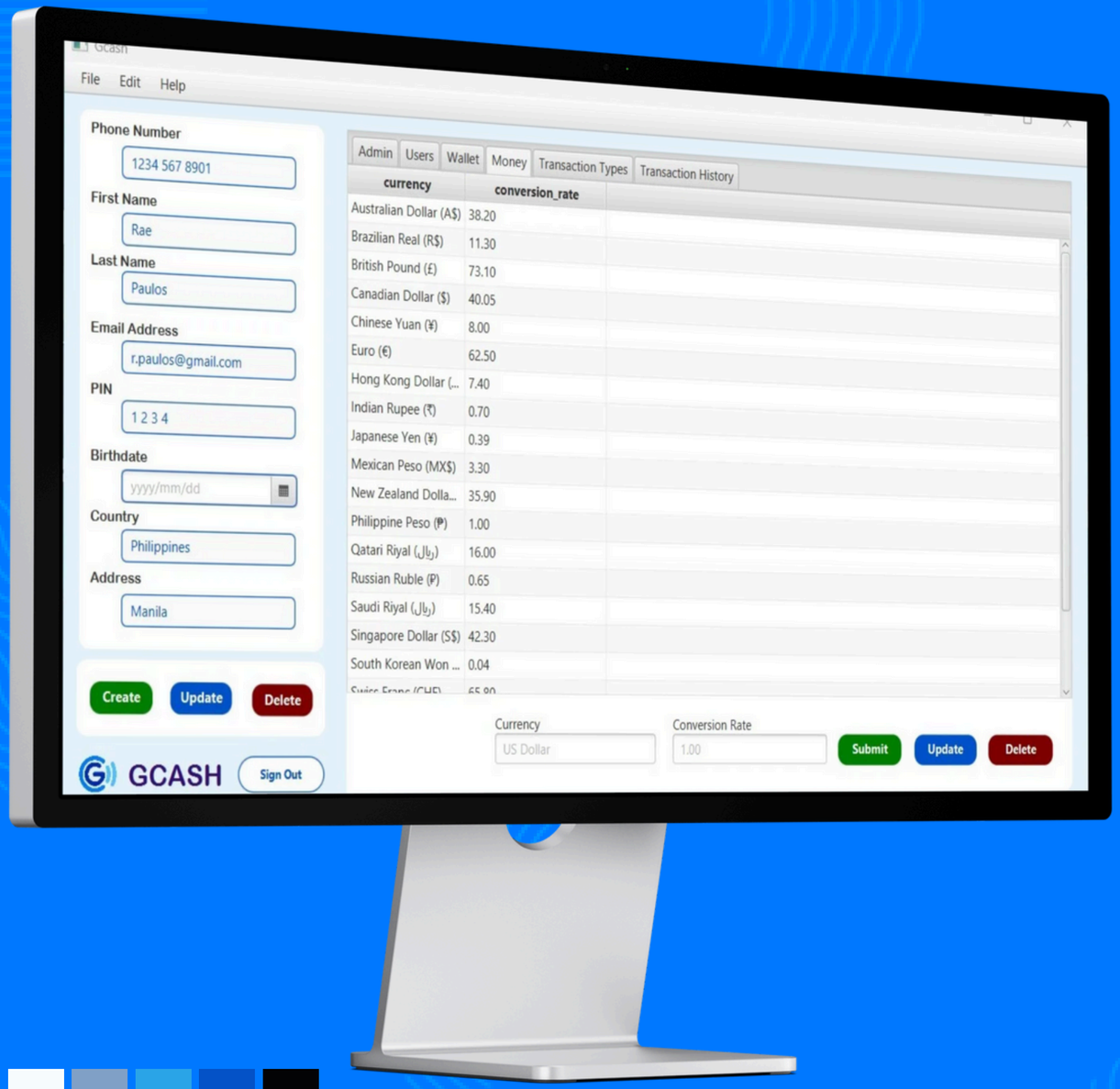
USER TABLE

The user table are the ones who uses Gcash. They are the ones who uses the main three features of the app.



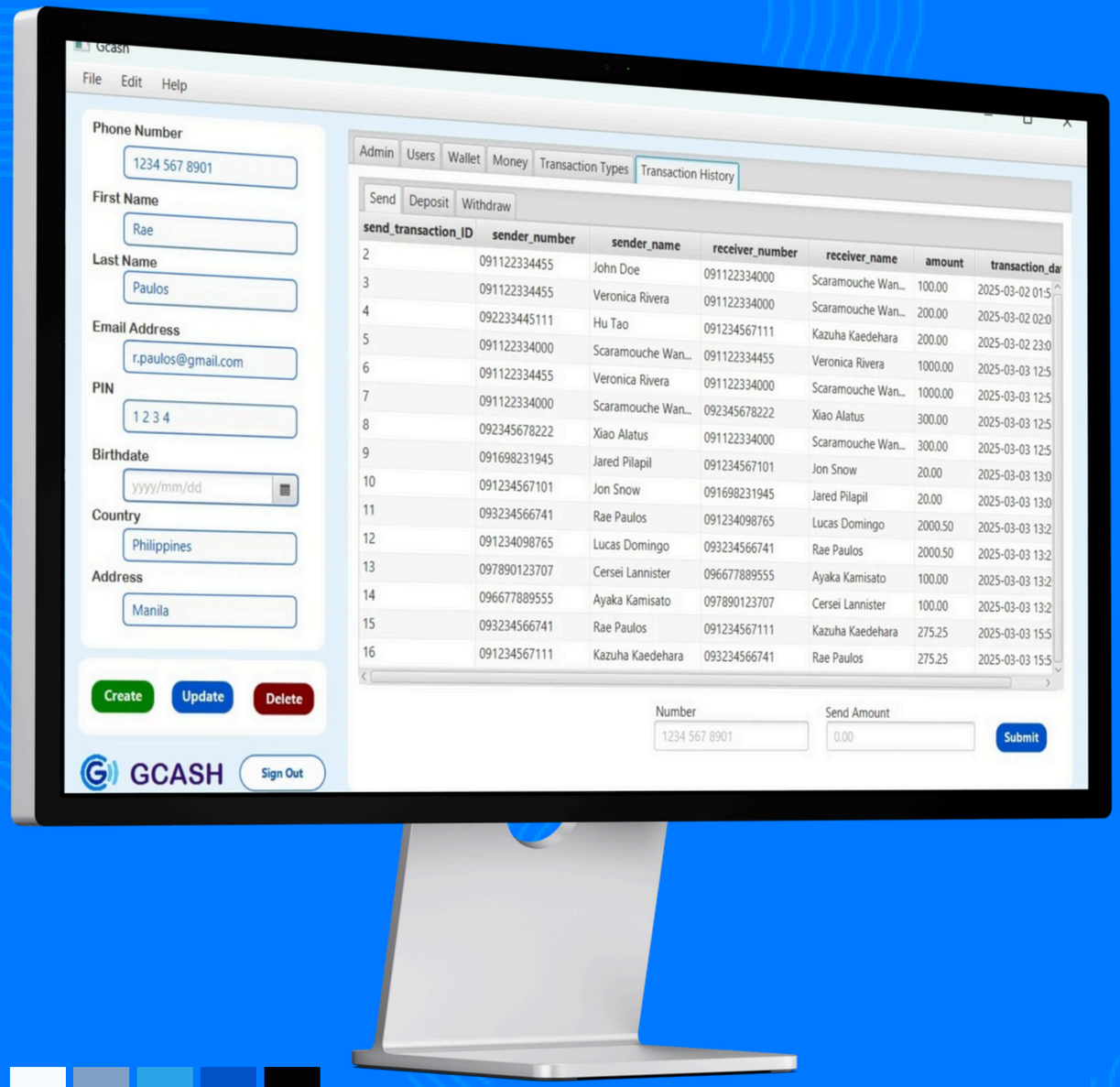
WALLET TABLE

The wallet table is where the data for storing user balance is located.



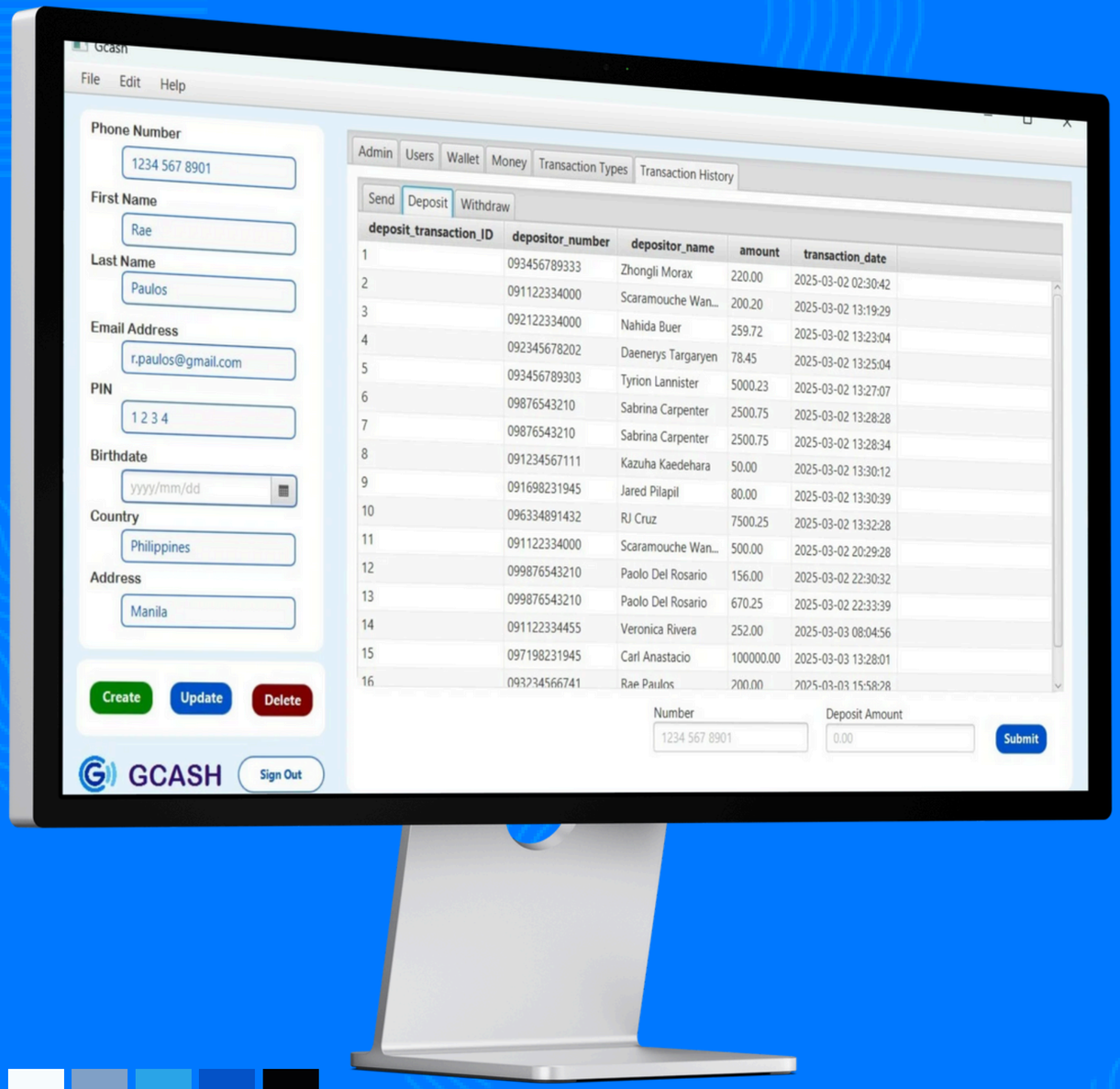
MONEY TABLE

Contains different types of supported (and soon to be supported) currencies and its conversion rate.



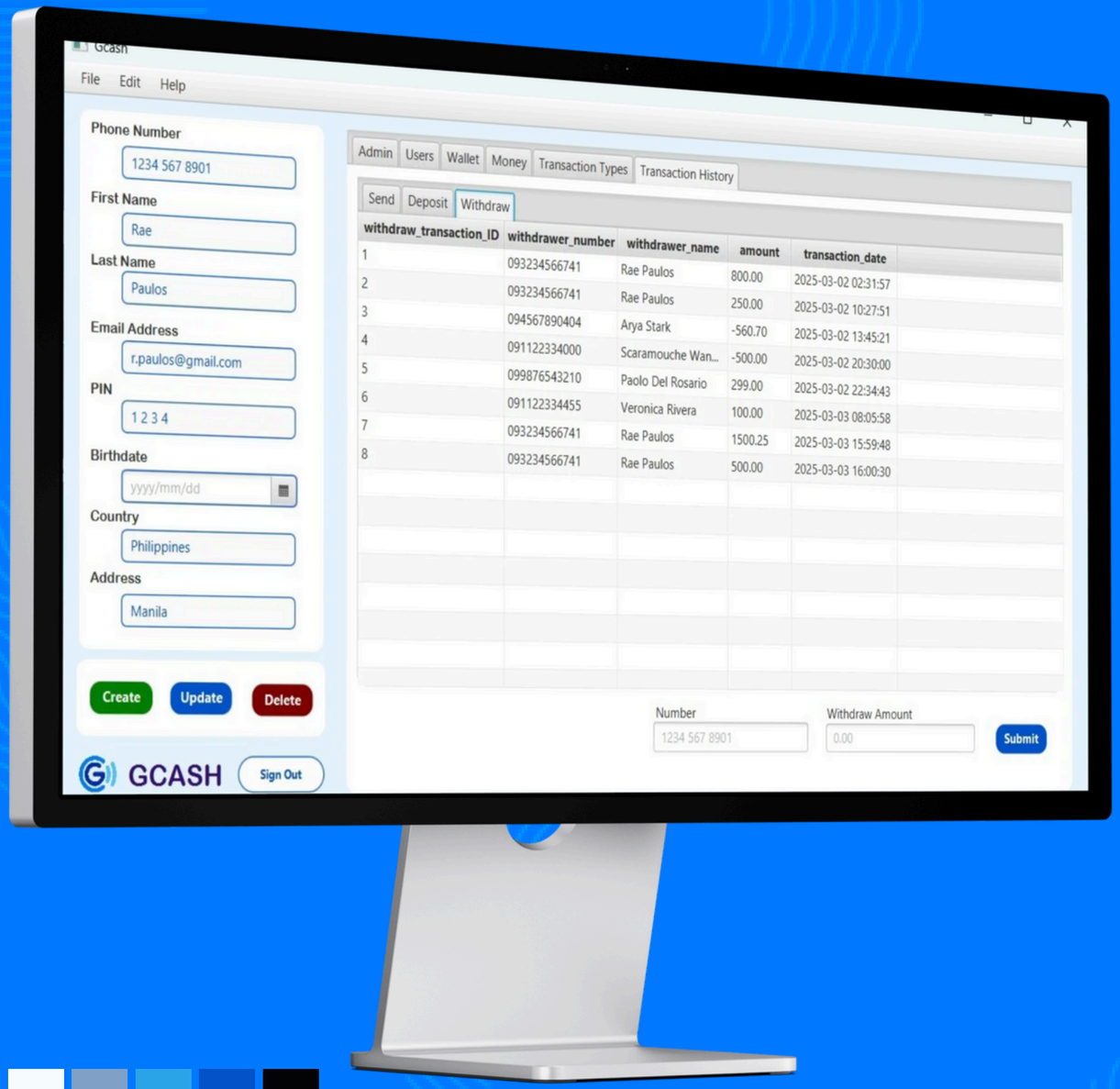
SEND TABLE

This table records the transactions made by the user when using the Express Send feature of the app.



DEPOSIT TABLE

This table records the transactions made by the user when using the deposit feature of the app.



WITHDRAW TABLE

This table records the transactions made by the user when using the withdraw feature of the app.

INFORMATION MANAGEMENT FINAL PROJECT

GCASH

USER INTERFACE OVERVIEW, ENTITY RELATIONSHIP DIAGRAM, AND PURPOSES