

# MoneyPal: Online Wallet

*Team Members: Ashish G, Prathamesh M*

## ABSTRACT

This mini-project involves developing a basic online wallet system using MySQL, Python Flask, HTML, JavaScript and CSS. The system allows users to create accounts, manage their wallet balances, view transaction history, and delete their accounts. The project focuses on understanding database design and management, emphasizing the use of a relational database schema to store user information and transaction records. It provides a foundational understanding of how online wallets function and how databases are used to manage financial data.

## PROBLEM STATEMENT

With the rise of digital transactions, there's a need for simple systems to manage financial data. This mini-project aims to address the following database-related challenges:

- Creating a basic database schema for user and transaction data.
- Implementing fundamental database operations for financial transactions.
- Exploring challenges of maintaining data consistency in a simple application.
- Understanding database interaction with web applications.

To address these, the project develops a basic online wallet using MySQL, Python Flask, HTML, CSS and JavaScript focusing on core functionalities and database interaction.

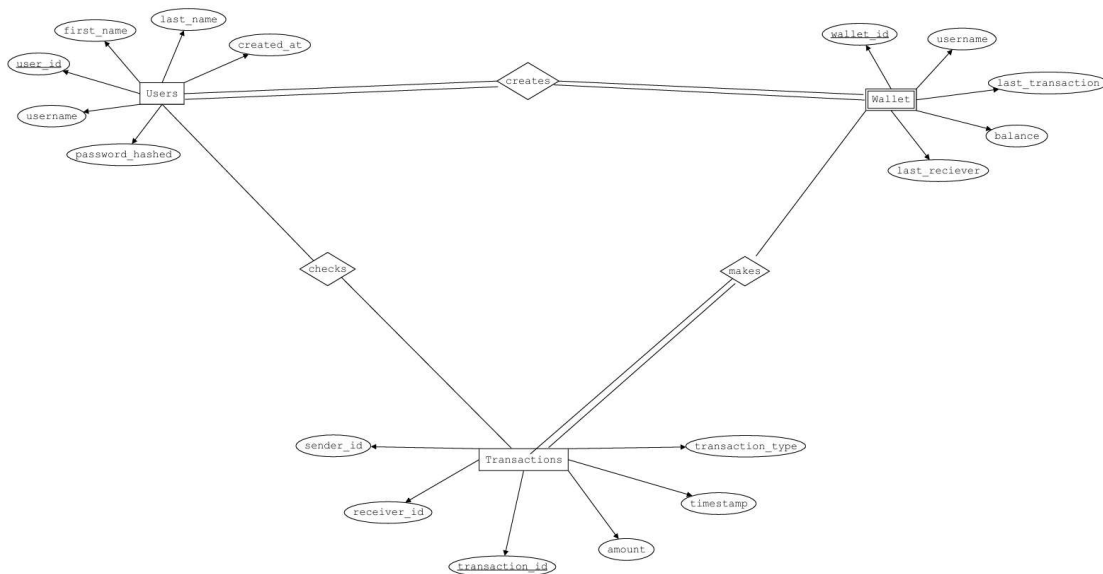
- **Functional Requirements:**
  - User Registration and Login functionality.
  - Wallet Balance Management, allowing users to Deposit funds, Withdraw funds (checking against balance).
  - Facilitation of User-to-User Fund Transfers, validating sender balance and recipient existence.
  - Viewing a chronological Transaction History for the logged-in user.
  - Allowing users to Delete their accounts and associated data.
- **Data Requirements:**
  - User Data: Must store a unique User ID, First Name, Last Name, a securely Hashed Password, and the Current Balance. Timestamps for account creation are also useful.
  - Transaction Data: Must record a unique Transaction ID, the associated User ID (linking to the user), the Transaction Type (Deposit, Withdraw, Transfer), the Amount involved, a Timestamp for the transaction, and the ID of the other user involved in a transfer.
- **Database Challenges Addressed:**
  - Designing an efficient relational database schema ([Users](#), [Wallet](#), [Transactions](#)) for storing the required data.
  - Implementing fundamental SQL operations (INSERT, SELECT, UPDATE, DELETE) for the core functionalities.
  - Exploring basic data consistency issues, such as ensuring balance updates correspond

correctly to transactions (especially transfers).

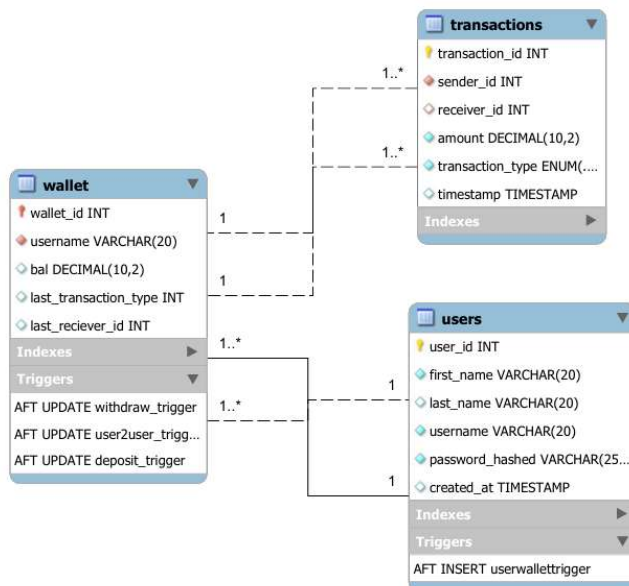
- Demonstrating the interaction between a web application backend (Python Flask) and the MySQL database to manage and present financial data.

## ER DIAGRAM & RELATIONAL SCHEMA ALONG WITH SAMPLE DATA

### ER Diagram



### Relational Tables



## Sample Data

### 1. Users Table

|   | user_id | first_name | last_name | username | password_hashed                               | created_at          |
|---|---------|------------|-----------|----------|---|---------------------|
| ▶ | 1       | First1     | Last1     | 1        | pbkdf2:sha256:1000000\$Z47ahuls\$076edf804... | 2025-04-10 00:49:34 |
|   | 2       | First2     | Last2     | 2        | pbkdf2:sha256:1000000\$wm87Pyxy\$67892cf4...  | 2025-04-10 00:49:34 |
|   | 3       | First3     | Last3     | 3        | pbkdf2:sha256:1000000\$XDnLBHhu\$46314053...  | 2025-04-10 00:49:35 |
|   | 4       | First4     | Last4     | 4        | pbkdf2:sha256:1000000\$l8k7pvMG\$b4594942e... | 2025-04-10 00:49:36 |
|   | 5       | First5     | Last5     | 5        | pbkdf2:sha256:1000000\$xjKCIvtn\$499cad276... | 2025-04-10 00:49:37 |

### 2. Wallet Table

|   | wallet_id | username | bal      | last_transaction_type | last_reciever_id |
|---|-----------|----------|----------|-----------------------|------------------|
| ▶ | 1         | 1        | 15000.00 | 3                     | NULL             |
|   | 2         | 2        | 5000.00  | 3                     | 1                |
|   | 3         | 3        | 10000.00 | 1                     | NULL             |
|   | 4         | 4        | 10000.00 | 1                     | NULL             |
|   | 5         | 5        | 10000.00 | 1                     | NULL             |
|   | 6         | 6        | 10000.00 | 1                     | NULL             |

### 3. Transactions Table

|   | transaction_id | sender_id | receiver_id | amount   | transaction_type | timestamp           |
|---|----------------|-----------|-------------|----------|------------------|---------------------|
| ▶ | 1              | 1         | NULL        | 10000.00 | Deposit          | 2025-04-10 00:49:34 |
|   | 2              | 2         | NULL        | 10000.00 | Deposit          | 2025-04-10 00:49:34 |
|   | 3              | 3         | NULL        | 10000.00 | Deposit          | 2025-04-10 00:49:35 |
|   | 4              | 4         | NULL        | 10000.00 | Deposit          | 2025-04-10 00:49:36 |
|   | 5              | 5         | NULL        | 10000.00 | Deposit          | 2025-04-10 00:49:37 |
|   | 6              | 6         | NULL        | 10000.00 | Deposit          | 2025-04-10 00:49:38 |

## DDL COMMANDS

### 1. Users Table

```
CREATE TABLE IF NOT EXISTS users (  
  
    user_id INT PRIMARY KEY AUTO_INCREMENT,  
  
    first_name VARCHAR(20) NOT NULL,  
  
    last_name VARCHAR(20),  
  
    username VARCHAR(20) NOT NULL UNIQUE,  
  
    password_hashed VARCHAR(255) NOT NULL,  
  
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP  
  
);
```

### 2. Wallet Table

```
CREATE TABLE IF NOT EXISTS wallet (  
  
    wallet_id INT PRIMARY KEY AUTO_INCREMENT,
```

```

username VARCHAR(20) NOT NULL UNIQUE,

bal DECIMAL(10, 2) DEFAULT 0.00,

last_transaction_type INT,

last_reciever_id INT,

FOREIGN KEY (wallet_id) REFERENCES users(user_id) ON DELETE CASCADE,

FOREIGN KEY (username) REFERENCES users(username)

);

```

### 3. Transactions Table

```

CREATE TABLE IF NOT EXISTS transactions (

transaction_id INT PRIMARY KEY AUTO_INCREMENT,

sender_id INT NOT NULL,

receiver_id INT DEFAULT NULL,

amount DECIMAL(10, 2) NOT NULL,

transaction_type ENUM('Deposit', 'Withdraw', 'User to User') NOT NULL,

timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

FOREIGN KEY (sender_id) REFERENCES wallet(wallet_id),

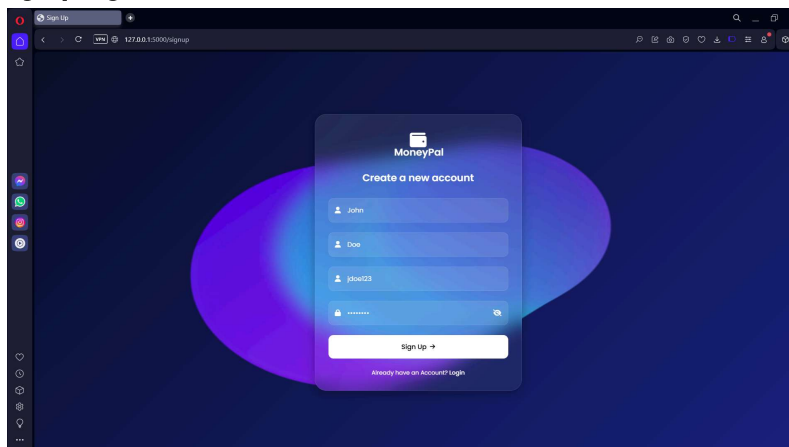
FOREIGN KEY (receiver_id) REFERENCES wallet(wallet_id)

);

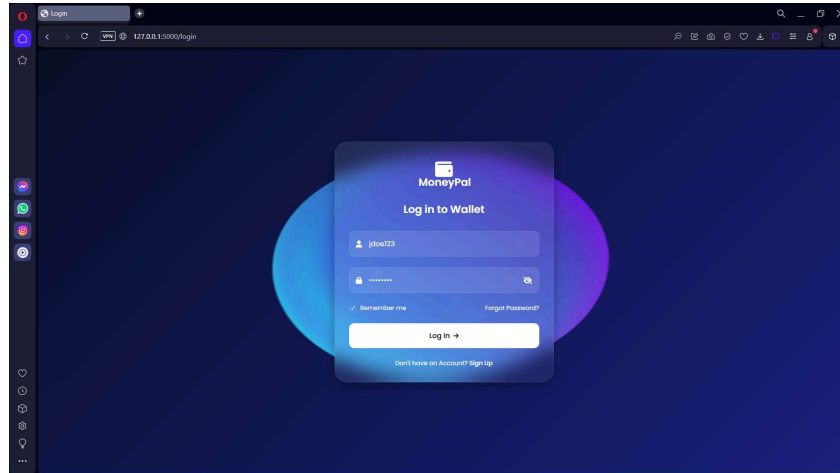
```

## UI DESIGN

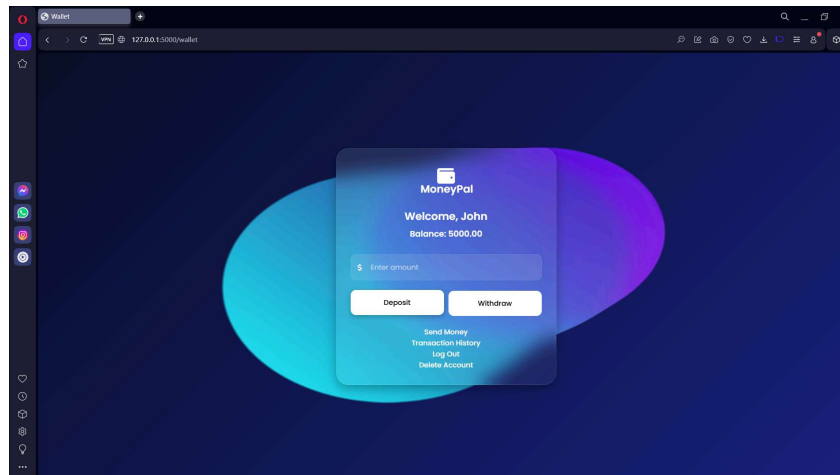
### 1. Signup Page



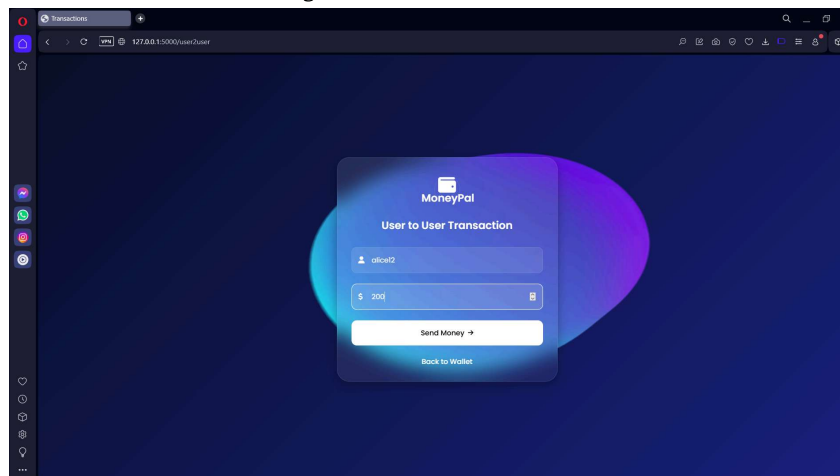
## 2. Login Page



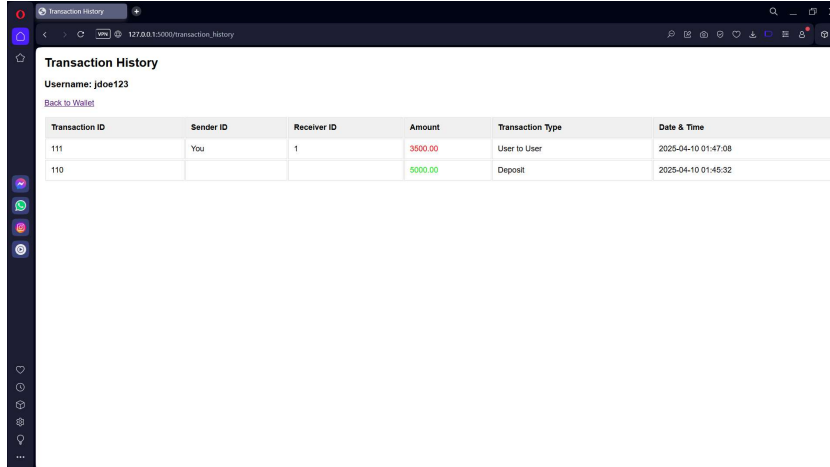
## 3. Wallet Page



## 4. User-to-User Transaction Page



## 5. Transaction History Page



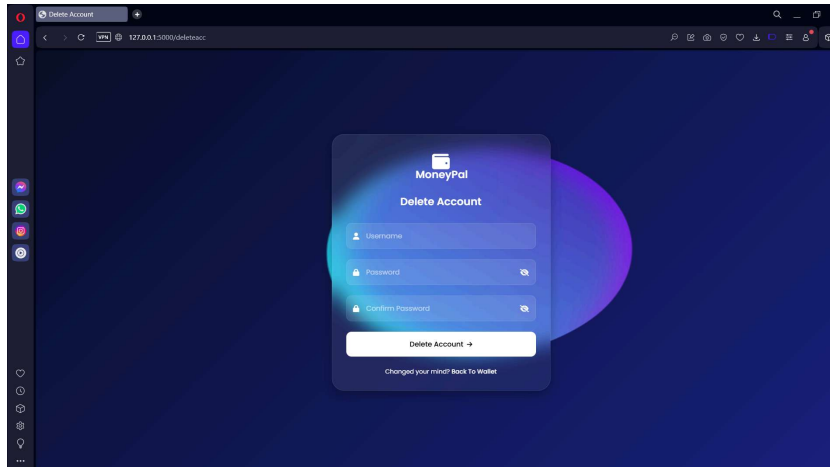
Transaction History

Username: jdoe123

[Back to Wallet](#)

| Transaction ID | Sender ID | Receiver ID | Amount  | Transaction Type | Date & Time         |
|----------------|-----------|-------------|---------|------------------|---------------------|
| 111            | You       | 1           | 3500.00 | User to User     | 2025-04-10 01:47:08 |
| 110            |           |             | 5000.00 | Deposit          | 2025-04-10 01:45:32 |

## 6. Delete Account Page



MoneyPal

Delete Account

Username

Password

Confirm Password

Delete Account →

[Changed your mind? Back To Wallet](#)