DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Shahbad Daulatpur, Bawana Road, Delhi 110042

DEPARTMENT OF SOFTWARE ENGINEERING

Lab Code: Software Engineering Methodologies

Lab File

Submitted To:

Mr. Ankur

Submitted By:

SOUMYA SURANA(23/SE/151)

SOURAV KUMAR(23/SE/152)

SURENDRA KUMAR MEENA(23/SE/153)

TANISH AGGARWAL(23/SE/154)

EXPERIMENT - 1

AIM : To write the problem statement for the E Wallet Management System

THEORY : An E-wallet management system is a digital platform designed to facilitate electronic transactions, enabling users to store, manage, and transfer funds securely. It serves as a virtual wallet that can hold various payment methods, including credit/debit information, bank account details, and other digital assets. This system operates using encrypted technology to safeguard sensitive data, ensuring secure transactions and protecting user privacy. Through features like account balance tracking, transaction history, and instant transfers, E-wallets enhance financial convenience and accessibility, supporting cashless payments and promoting a seamless user experience in both personal and commercial contexts.

CHALLENGES OF SYSTEM :

- **Security Risks**: E-wallet systems face threats like hacking, phishing, and malware attacks, which can compromise user data and funds.
- **Regulatory Compliance**: E-wallet providers must comply with strict financial regulations and data protection laws, which vary across regions and can be complex.
- **User Trust and Adoption**: Building trust is crucial as users may be reluctant to adopt e-wallets due to privacy concerns or unfamiliarity with digital transactions.
- **Fraud Detection**: Identifying and preventing fraud in real-time is challenging, requiring advanced algorithms and constant monitoring.
- **Integration with Banking Systems**: Compatibility with multiple banks and financial institutions can be difficult, as each may have unique protocols and standards.
- **Transaction Failures and Delays**: Network or system issues can lead to transaction failures or delays, impacting user experience and satisfaction.

OBJECTIVES OF E WALLET MANAGEMENT SYSTEM :

- **Enhance Transaction Convenience**: Enable users to make fast, cashless payments anytime, anywhere, improving transaction efficiency.
- **Ensure Security of Funds and Data**: Protect user information and funds through encryption and secure authentication methods.

- Provide Seamless User Experience: Offer an easy-to-use interface with quick access to balance information, transaction history, and payment options.
- Promote Financial Inclusion: Facilitate access to digital payments, especially for those without traditional banking services.
- Support Multiple Payment Methods: Allow integration with various payment sources, including credit cards, bank accounts, and loyalty rewards.
- Real-time Transaction Tracking: Enable users to monitor their transactions instantly, promoting transparency and financial awareness.

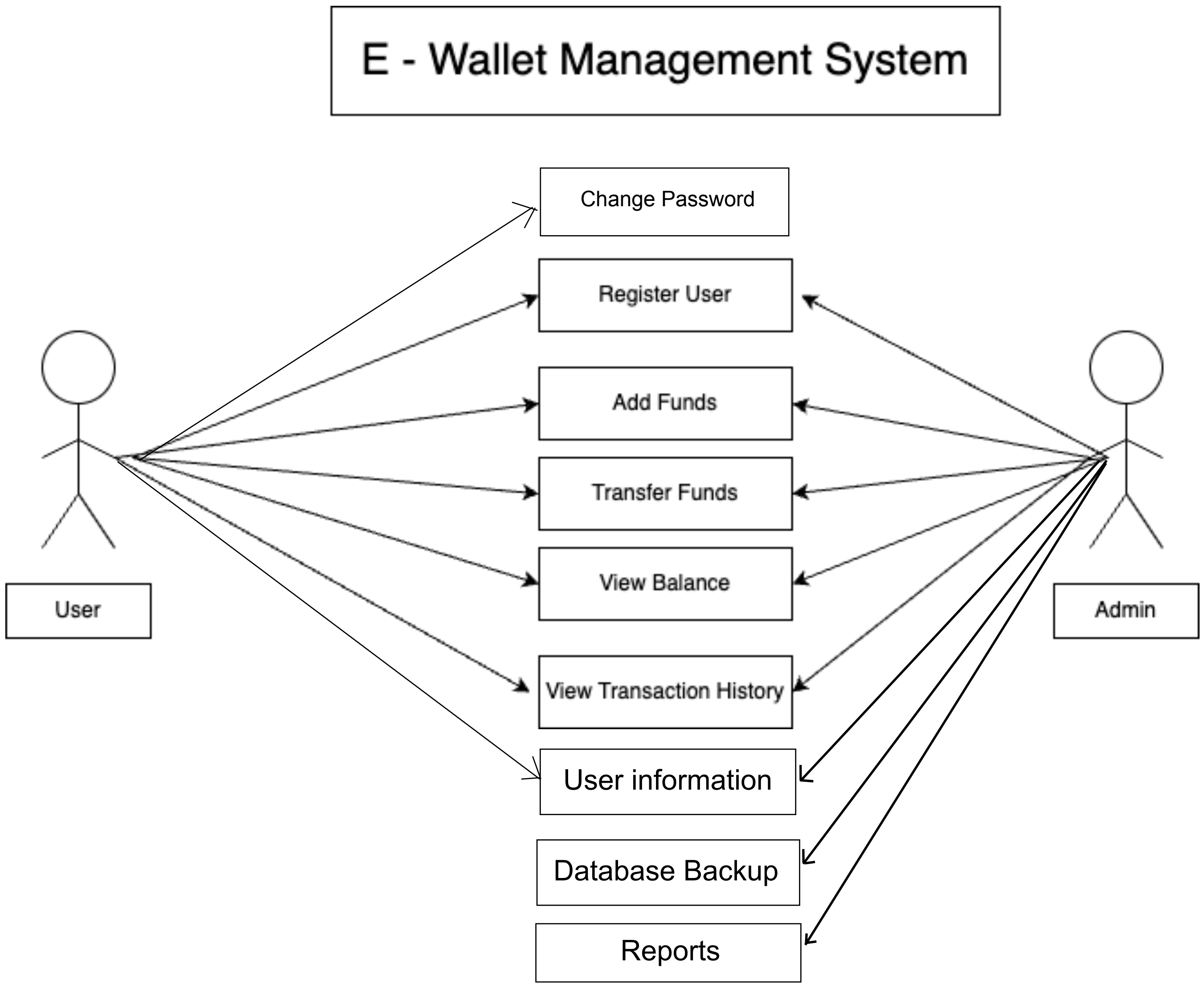
BENEFITS OF A E WALLET MANAGEMENT SYSTEM :

- Increased Convenience: Allows users to make quick, cashless transactions, eliminating the need to carry physical cash or cards.
- Enhanced Security: Reduces the risks associated with carrying cash by providing secure, encrypted transactions and multi-factor authentication.
- Improved Financial Tracking: Enables users to monitor expenses, view transaction history, and manage finances more effectively through in-app insights.
- Faster Transactions: Facilitates instant payments, reducing wait times at checkout and making online shopping and in-person transactions faster.
- Cost Savings: Lowers transaction fees compared to traditional banking, especially for small businesses and frequent users.
- Financial Inclusion: Provides access to digital payments for unbanked or underbanked individuals, promoting wider financial access and inclusion.

EXPERIMENT – 2

AIM : Use case Diagram of E-Wallet Management System

Theory: Use Case Diagram is a visual representation of the interactions between a system and its users. It helps in understanding the functional requirements of the system.
Use case diagram provides a high-level overview of the functionality of the library management system. It can be used as a basis for further development and refinement of the system's requirements.



EXPERIMENT – 3

AIM : Use case Discription of E-Wallet Management System

Theory:

Use Case Descriptions

---

1. Register User

- **Use Case ID**: UC-01
- **Use Case Name**: Register User
- **Description**: Allows a new user to register with the e-wallet system by providing a name and unique email address.
- **Actors**: User
- **Preconditions**: The user must not be registered in the system (no existing account with the same email).
- **Postconditions**: A new user account is created and stored in the system.
- **Main Flow**:
    1. The user selects the option to register.
    2. The system prompts the user to enter a name and email address.
    3. The user inputs their details.
    4. The system checks if the email address already exists.
    5. If the email is unique, the system creates a new account with the entered details.
    6. The system confirms registration success to the user.
- **Alternative Flow**:
    - **Email Already Exists**: If the email already exists, the system notifies the user and prompts them to use a different email or log in if they already have an account.

---

2. Add Funds

- **Use Case ID**: UC-02
- **Use Case Name**: Add Funds
- **Description**: Allows a user to add a specified amount of money to their e-wallet balance.
- **Actors**: User
- **Preconditions**: The user must be registered in the system and logged in.
- **Postconditions**: The specified amount is added to the user's balance, and a transaction record is created.
- **Main Flow**:
    1. The user selects the option to add funds.
    2. The system prompts the user to enter the amount they wish to add.

3. The user enters a valid amount.
4. The system adds the specified amount to the user’s balance.
5. The system creates a transaction record indicating the amount added.
6. The system confirms the addition to the user.

- **Alternative Flow**:
  - **Invalid Amount**: If the amount is less than or equal to zero, the system rejects the input and prompts the user to enter a valid positive amount.

---

3. Transfer Funds

- **Use Case ID**: UC-03
- **Use Case Name**: Transfer Funds
- **Description**: Allows a user to transfer a specified amount of money to another registered user's account.
- **Actors**: User
- **Preconditions**:
  - The user must be registered in the system and logged in.
  - The recipient must be registered in the system.
  - The sender’s balance must be greater than or equal to the transfer amount.
- **Postconditions**:
  - The specified amount is deducted from the sender’s balance and added to the recipient’s balance.
  - A transaction record is created for both the sender and the recipient.
- **Main Flow**:
  1. The user selects the option to transfer funds.
  2. The system prompts the user to enter the recipient's email and the transfer amount.
  3. The user enters the recipient's email and a valid positive amount.
  4. The system verifies that the recipient exists and that the sender has enough balance.
  5. If both conditions are met, the system deducts the specified amount from the sender’s balance and adds it to the recipient’s balance.
  6. The system creates transaction records for both users.
  7. The system confirms the successful transfer to the user.
- **Alternative Flows**:
  - **Recipient Not Found**: If the recipient email does not exist, the system notifies the user and prompts them to enter a valid recipient email.
  - **Insufficient Balance**: If the sender’s balance is insufficient, the system notifies the user and cancels the transfer.

---

4. View Balance

- **Use Case ID**: UC-04
- **Use Case Name**: View Balance
- **Description**: Allows a user to view their current e-wallet balance.
- **Actors**: User
- **Preconditions**: The user must be registered in the system and logged in.
- **Postconditions**: The user's current balance is displayed.
- **Main Flow**:
  1. The user selects the option to view their balance.
  2. The system retrieves the user's current balance.
  3. The system displays the balance to the user.
- **Alternative Flow**:
  - **User Not Found**: If the user is not registered or logged in, the system displays an error message.

---

5. View Transaction History

- **Use Case ID**: UC-05
- **Use Case Name**: View Transaction History
- **Description**: Allows a user to view a record of their past transactions.
- **Actors**: User
- **Preconditions**: The user must be registered in the system and logged in.
- **Postconditions**: The user's transaction history is displayed.
- **Main Flow**:
  1. The user selects the option to view their transaction history.
  2. The system retrieves the user's transaction records.
  3. The system displays the transaction history to the user.
- **Alternative Flow**:
  - **No Transactions**: If there are no transactions in the user's history, the system informs the user that no transactions were found.

---

6. Exit System

- **Use Case ID**: UC-06
- **Use Case Name**: Exit System
- **Description**: Ends the user's session in the application.
- **Actors**: User
- **Preconditions**: The user is logged into the system.
- **Postconditions**: The user's session is terminated, and they are logged out of the application.
- **Main Flow**:

1. The user selects the option to exit.
2. The system logs the user out and terminates the session.
3. The system displays a message confirming the exit.

Use Case	Description	Actors
Register User	The user provides a name and email to register in the system.	User
Add Funds	Allows the user to add a specified amount to their wallet balance.	User
Transfer Funds	Enables the user to transfer a specified amount to another registered user's wallet.	User
View Balance	Displays the current balance in the user's wallet.	User
View Transaction History	Shows a list of past transactions made by the user.	User
Exit System	Ends the user's session in the application.	User

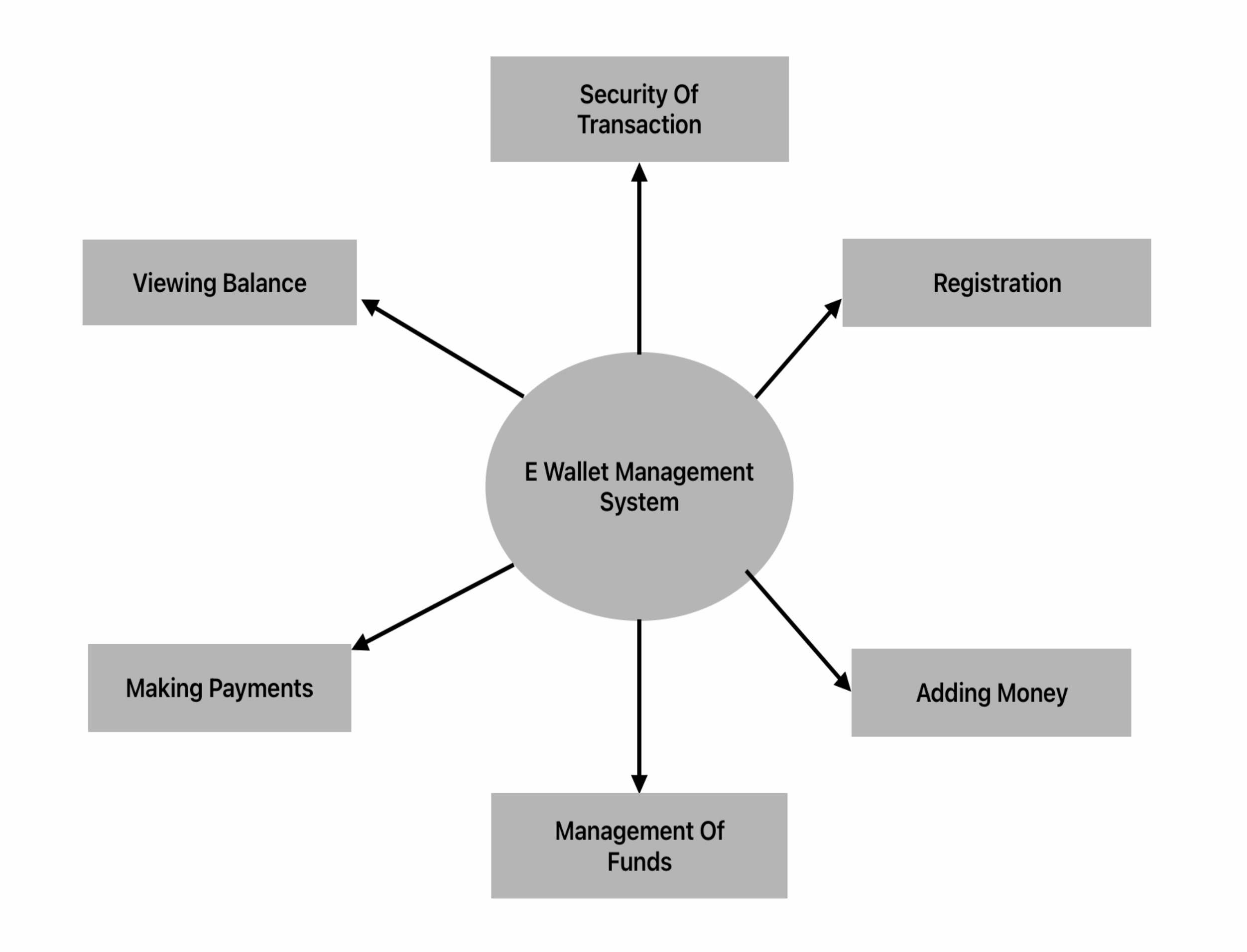
EXPERIMENT - 4

AIM : Data Flow Diagrams For E Wallet Management System

THEORY :

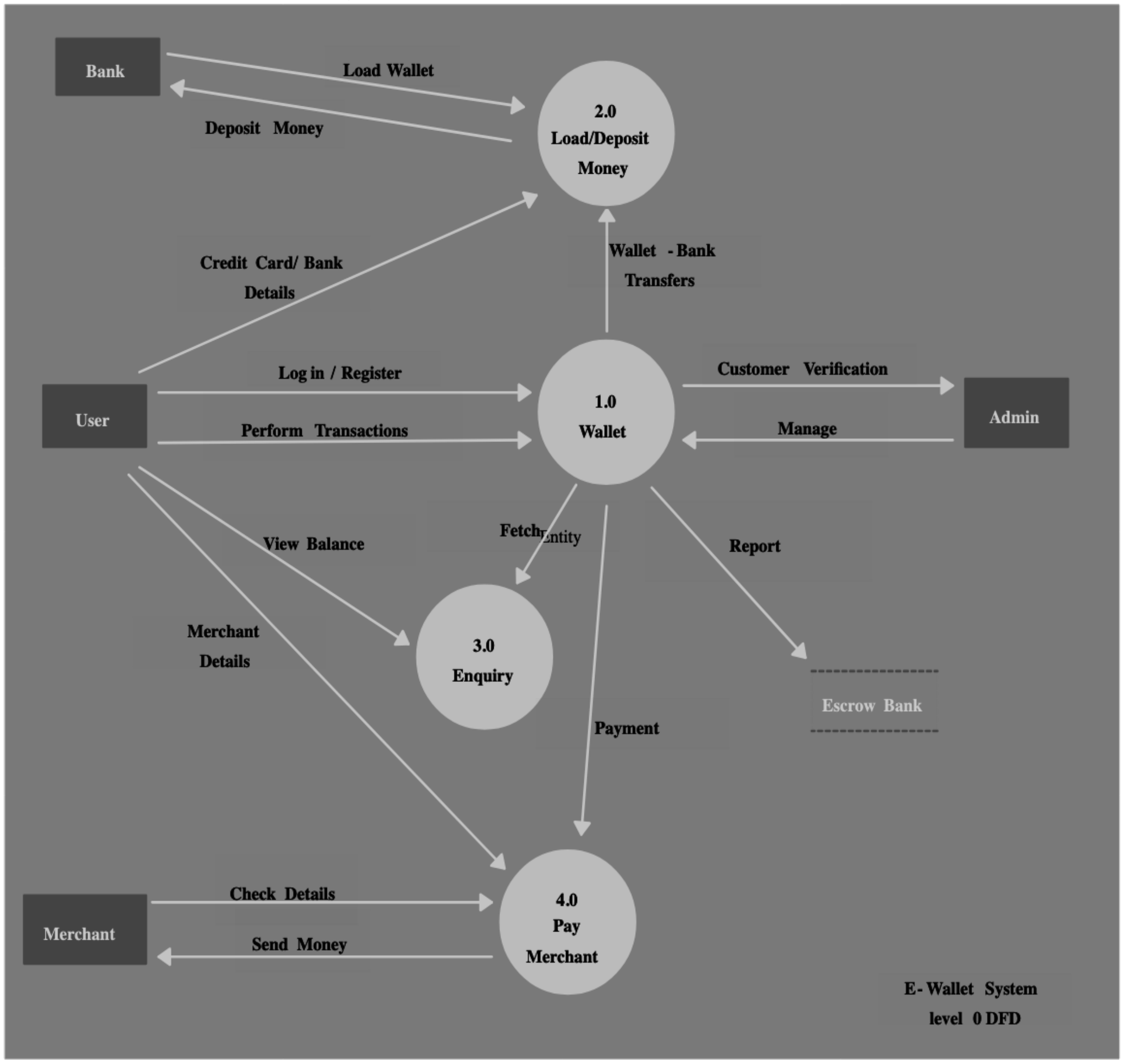
Level 0 DFD :

A Level 0 Data Flow Diagram (DFD), also known as a context diagram, provides a high-level overview of a system, illustrating the system's boundaries and its interactions with external entities. It serves as a foundational representation that highlights the main inputs and outputs without delving into the internal processes. In a Level 0 DFD, the entire system is represented as a single process, typically labeled with the system name, while external entities, such as users or other systems, are depicted as circles or rectangles. Arrows indicate the flow of data between the system and these external entities, showcasing how information enters and exits the system. This diagram is crucial for understanding the system's overall context and scope, making it an essential tool in the early stages of system design.



Level 1 DFD :

A Level 1 Data Flow Diagram (DFD) provides a more detailed view of a system than a Level 0 DFD, breaking down the main process from the context diagram into its sub-processes. In this diagram, the overall system is represented as a single process, but it is decomposed into multiple components or functions that illustrate how data flows between them. Each sub-process is labeled with a descriptive name, and data stores may be included to show where information is held within the system. External entities remain present, indicating where data originates and where it is sent. Arrows depict the flow of data between the processes, data stores, and external entities, allowing stakeholders to understand the specific interactions and operations that take place within the system. This level of detail helps in analyzing and designing system functionalities more effectively.



EXPERIMENT - 5

AIM : Entity Relationship Diagram for E Wallet Management System

THEORY :

Entities and Attributes

1. **User**
   - **Attributes**:
     - `user_id` (Primary Key): Unique identifier for each user.
     - `name`: The name of the user.
     - `email`: The user’s unique email address.
     - `balance`: The current balance of the user's e-wallet.
   - **Description**: The `User` entity represents each individual who registers and interacts with the e-wallet system. Each user has a unique email address and a balance that indicates the amount of money available in their wallet.
2. **Transaction**
   - **Attributes**:
     - `transaction_id` (Primary Key): Unique identifier for each transaction.
     - `transaction_type`: Type of transaction, e.g., "add funds," "deduct funds," "transfer."
     - `amount`: The amount of money involved in the transaction.
     - `timestamp`: Date and time when the transaction took place.
     - `user_id` (Foreign Key): Refers to the `user_id` of the `User` who performed the transaction.
     - `related_user_id` (Optional Foreign Key): Refers to the `user_id` of the recipient in case of a transfer transaction.
   - **Description**: The `Transaction` entity logs each financial transaction that a user performs, including adding funds, deducting funds, or transferring money. It stores details about the transaction type, amount, and timestamp.
3. **Account**
   - **Attributes**:
     - `account_id` (Primary Key): Unique identifier for the account.
     - `user_id` (Foreign Key): Refers to the `user_id` in the `User` entity.
     - `created_at`: The date and time the account was created.
     - `account_status`: Status of the account (e.g., active, inactive).
   - **Description**: The `Account` entity represents the account associated with each user in the e-wallet system. It links to the `User` entity and includes account status and creation details.

Relationships

1. **User has Account**
   - **Type**: One-to-One

    - **Description**: Each `User` has exactly one `Account` associated with them in the e-wallet system. The account contains additional information like creation date and status.
2. **User performs Transaction**
    - **Type**: One-to-Many
    - **Description**: A `User` can perform multiple `Transactions` in the e-wallet system, but each transaction is associated with only one user. This is a one-to-many relationship where a single user can add funds, transfer money, or view multiple transactions, while each transaction record is unique to a specific user.
3. **Transaction involves User**
    - **Type**: Optional Many-to-Many (for Transfers)
    - **Description**: In case of a transfer, there's a relationship between the `Transaction` and a second `User`, who is the recipient. For other types of transactions, this relationship might be null.

Example ER Diagram Explanation

In this setup:

- **Entities**: Users, Accounts, and Transactions form the core of the e-wallet system.
- **Relationships**:
    - **One-to-One** (`User` and `Account`): Each user has one account.
    - **One-to-Many** (`User` and `Transaction`): Users can perform multiple transactions, such as adding or transferring money.
    - **Many-to-Many** (conditional, for transfers): For transfers, there may be a link from the sender to the recipient within the `Transaction` entity, where one user's transaction record involves another user as the recipient.

Detailed Example Scenario

Consider the following scenario to see how this ER model works in practice:

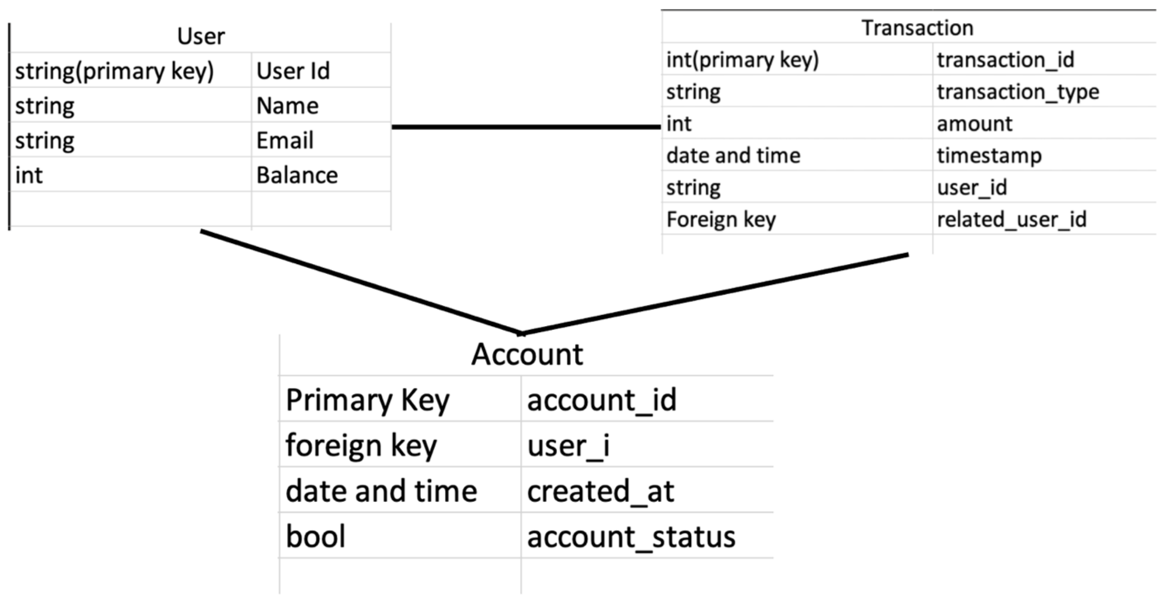
1. **User Registration**:
    - A new user registers, and a record is created in the `User` entity with a unique `user_id`. An associated `Account` record is also created.
2. **Adding Funds**:
    - When the user adds funds to their wallet, a new `Transaction` entry is created. This entry has a `transaction_type` of "add funds," the amount added, and a timestamp.
3. **Transferring Money**:
    - When the user transfers money to another user, a `Transaction` entry is created with `transaction_type` as "transfer." The `related_user_id` points to the recipient's `user_id`, creating a link between the sender and recipient through the `Transaction` table.
4. **Viewing Transactions**:

- The user can view their transaction history by querying the `Transaction` entity for all entries where `user_id` matches their own `user_id`. For each transaction, they can see the type, amount, timestamp, and, in the case of transfers, the related user (recipient).

Key Points of the ER Diagram

- **Primary Keys**: Each entity has a primary key (`user_id`, `transaction_id`, `account_id`) to uniquely identify each record.
- **Foreign Keys**: Relationships are maintained via foreign keys (e.g., `user_id` in `Transaction` links back to `User`).
- **Attributes**: Each entity's attributes capture essential data for operations within the e-wallet, like account status, transaction type, and timestamps.
- **Normalization**: This ER diagram is normalized to avoid redundancy by ensuring separate tables for `User`, `Transaction`, and `Account`.

This ER diagram provides a scalable and flexible structure, allowing for future expansions, such as adding multiple payment methods, supporting refunds, or integrating loyalty points.

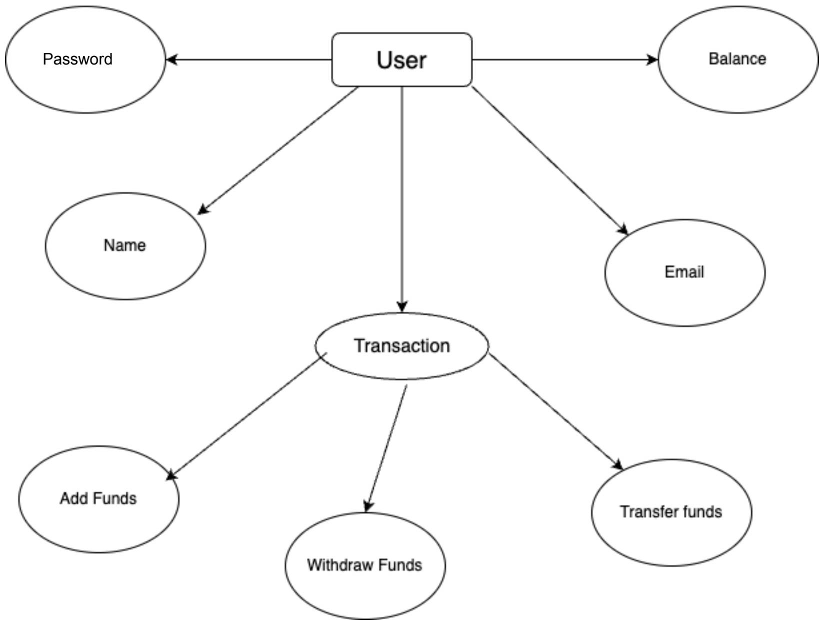


User	
string(primary key)	User Id
string	Name
string	Email
int	Balance

Transaction	
int(primary key)	transaction_id
string	transaction_type
int	amount
date and time	timestamp
string	user_id
Foreign key	related_user_id

Account	
Primary Key	account_id
foreign key	user_i
date and time	created_at
bool	account_status

E - Wallet Management System



EXPERIMENT - 6

AIM : Software Requirement Specification For E Wallet Management System

THEORY :

1. Introduction

1.1 Purpose
The purpose of this document is to outline the functional and non-functional requirements of an E-Wallet Management System. This system will provide users with a secure, efficient, and convenient way to conduct electronic transactions, manage funds, and monitor transaction history.

1.2 Scope
The E-Wallet Management System is a web and mobile-based application that allows users to perform various financial transactions, including sending/receiving funds, viewing transaction history, and managing their wallet balance. The system targets individual users, merchants, and financial institutions looking for a reliable digital payment solution.

1.3 Definitions, Acronyms, and Abbreviations

- **E-Wallet**: Digital application to store funds and perform transactions.
- **NFC**: Near Field Communication, for contactless payments.
- **KYC**: Know Your Customer, a verification process for user identity.

1.4 References
List any references to other documents, such as regulatory guidelines, financial standards, or existing payment system documentation.

---

2. Overall Description

2.1 Product Perspective
The E-Wallet Management System functions as a standalone application but is integrated with bank APIs, payment gateways, and security frameworks. It’s intended for individual and business use.

2.2 Product Functions
The main features of the system are:

- User Registration and Authentication
- Fund Storage and Transfer
- Transaction History and Reports
- Contactless Payment (NFC)
- Integration with bank accounts and cards
- Customer Support and Feedback System

2.3 User Classes and Characteristics

- **End Users**: Individuals using the e-wallet for personal transactions.
- **Merchants**: Business entities that accept payments through the e-wallet.
- **Administrators**: System administrators who manage the backend and user support.

2.4 Operating Environment

The system will operate on Android, iOS, and web browsers. It requires a secure backend server hosted on a cloud environment to manage data and processing.

2.5 Design and Implementation Constraints

- Must comply with data protection laws (e.g., GDPR).
- Follow PCI-DSS standards for handling payment data.

---

3. Functional Requirements

3.1 User Registration and Login

- Users should be able to register via email or mobile.
- Two-factor authentication is required for security.

3.2 Account Management

- View account balance and transaction history.
- Link and unlink bank accounts or credit cards.
- Manage personal details and security settings.

3.3 Fund Transfer

- Users can transfer funds to other e-wallet users.
- Fund transfer should be processed in real-time.
- Transaction confirmation via SMS/email.

3.4 Contactless Payment

- NFC-based payment support for compatible devices.
- QR code-based payments for all devices.

3.5 Transaction History

- Display history of all transactions with details like date, amount, and type.
- Search and filter options for transaction history.

3.6 Notifications and Alerts

- Notify users of successful transactions, low balance, and login attempts.

- Support push notifications for mobile devices.

---

4. Non-Functional Requirements

4.1 Security

- End-to-end encryption for all transactions.
- Compliance with PCI-DSS standards.
- Biometric authentication (fingerprint, face ID) on supported devices.

4.2 Performance Requirements

- Transactions should be processed within 2 seconds.
- System should support high concurrency, with a load of up to 10,000 transactions per minute.

4.3 Usability

- The interface should be intuitive and accessible.
- Support for multiple languages.

4.4 Reliability

- 99.9% uptime to ensure system availability.
- Automatic failover and backup systems in place.

4.5 Portability

- Compatible with major web browsers and mobile platforms (iOS and Android).

4.6 Compliance

- The system must adhere to GDPR for data protection and PCI-DSS for payment security standards.

---

5. System Models and Design

5.1 Use Case Diagram
Illustrates major use cases such as User Registration, Fund Transfer, and Payment.

5.2 Sequence Diagrams
Detailed sequence diagrams for main transactions like Fund Transfer and Login.

5.3 Class Diagrams
Class diagrams representing objects like User, Transaction, and Payment Gateway.

6. System Architecture

6.1 System Architecture
High-level architecture of the E-Wallet Management System, including front-end, backend, database, and API integrations.

6.2 Database Design
Outline the primary database tables (User, Transaction, Wallet, Merchant).

6.3 API Integrations
Description of APIs for banking, payment gateways, and KYC verification services.

7. Appendices

7.1 Glossary
Define any specific terms related to the system.

7.2 References
List all documents, websites, or standards referred to during development.