ONLINE WALLET MANAGEMENT SYSTEM

Team Members

Utkarsh Singh

Venkatesh M.

**INTRODUCTION**

**Problem Statement:**

To create a wallet management system in which user can transfer, add and the admin can approve the account.

**Description:**

This project is aimed at developing an online Wallet Management System. This is a web-based application that can be accessed over the web. The user can Show payment wallet account balance, Transfer funds from one account to another, adding amount to the payment wallet account, Create payment wallet account. This is an integrated system that contains both the user component, and the Admin component. There are features like report generators etc. in this system.

**Scope:**

User can only transfer money to the account which has registered on the e wallet application. User can add money by giving his/her card payment info. User can get a list of transactions from one date to another.

**Out of Scope:**

User cannot work with real money because the server hosted on the localhost. Admin cannot do anything with the user’s account,he can just approve the account.

## Functional components of the project

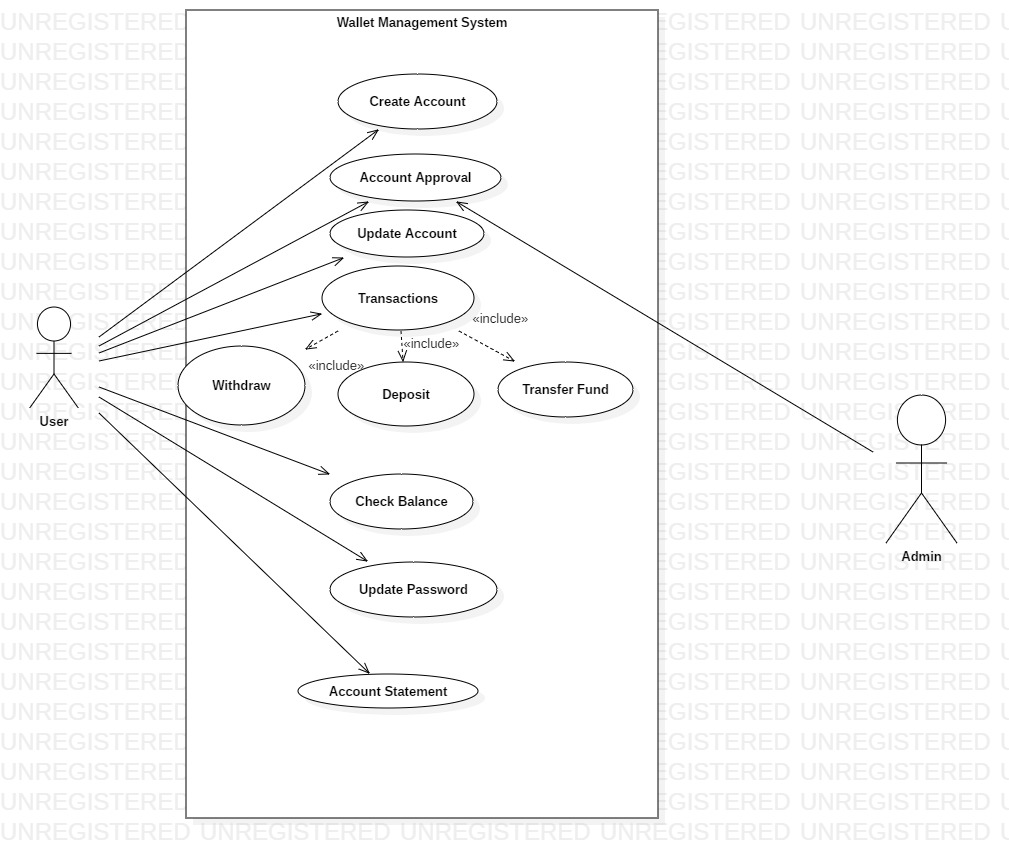
**Admin Operation**

Admin can only approve account.

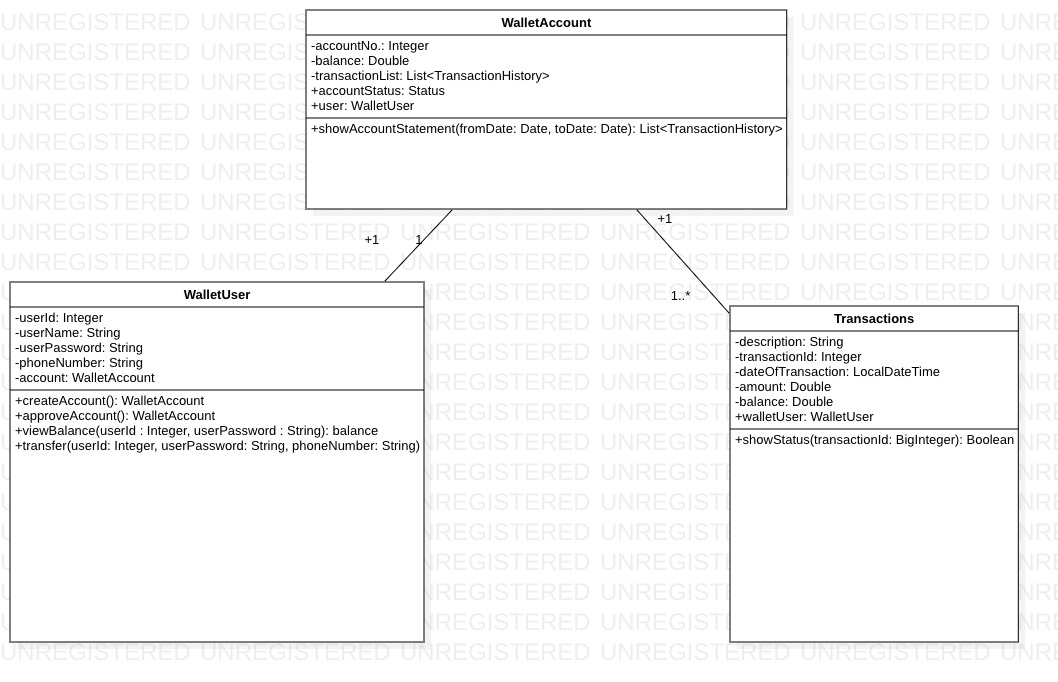
**User Operation**

User can Show payment wallet account balance, Transfer funds from one account to another, adding amount to the payment wallet account, Create payment wallet account. Status should be approved by Admin.

USE CASE VIEW



**CLASS DIAGRAM**



CLASSES AND METHODS

The mobile wallet consists of the following classes:-

1. WalletUser – This class stores the details of user who can be customer or administrator.

Attributes –

userId : BigInteger

userName : String

userPassword : String

phoneNumber : String

account : WalletAccount

Methods:

viewBalance(userId : Integer, userPassword : String) : balance

Users can view their wallet balance by using their credentials.

createAccount(): WalletAccount

This method is used for the creation of new user account.

transfer(in userId:Integer, in userPassword:String, in phoneNumber:String)

This method helps to transfer money from current user to another user of same wallet or different account

approveAccount(): WalletAccount

This method is used by the admin to approve a newly created account.

1. Transactions – This class contains the details of transactions.

Attributes:

description : String

transactionId : BigInteger

dateOfTx : Date

amount : BigDouble

balance : BigDouble

Methods:

showStatus(in transactionId:BigInteger): Boolean

It shows whether the transaction is a success or not.

1. WalletAccount- This class is used to use new bank accounts by user.

Attributes :

accountNo.: Integer

balance : BigDouble

transactionList: List<TransactionHistory>

accountStatus: Status

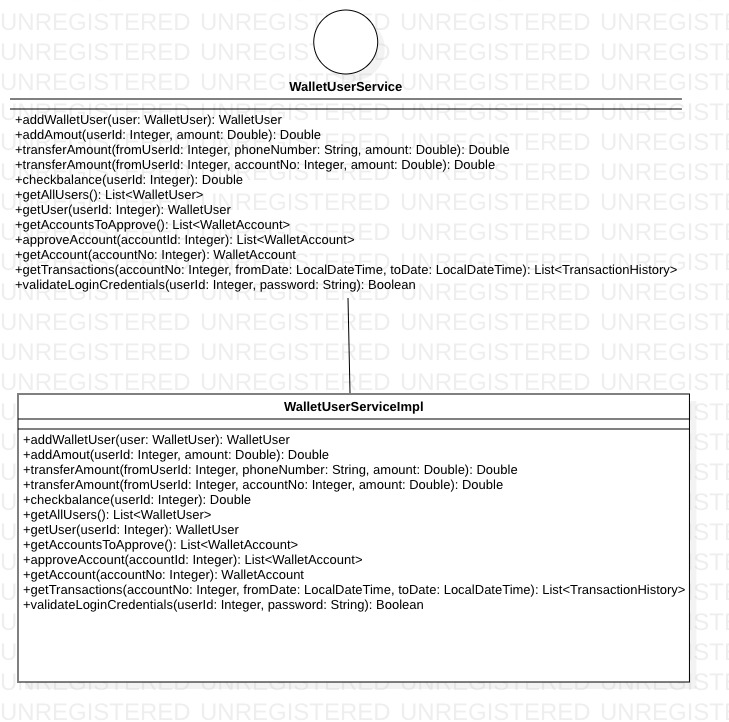
user: WalletUser

Methods :

showAccountStatement(fromDate: Date, toDate: Date): void

This method give the information about the transactions done from one date to another.

**Service Layer**

**Class Diagram**

Interfaces:

1. WalletUserService

Methods

1. addUser(in user:WalletUser): WalletUser

To add a object user having all the details of user with the class WalletUser.

2. addAmount(in userId:Integer, in amount:Double): Double

To add the required amount in the valid user account

3. transferAmount(in fromUserId:Integer, in phoneNumber:String, in amount:Double): Double

To transfer amount from one user to another user of known phone number

4.transferAmount(in fromUserId:Integer, in accountNo:Integer, in amount:Double): Double

To transfer money to account of non e-wallet users

5.checkbalance(in userId:Integer): Double

Check the amount present in the wallet account

6. getAllUsers(): List<WalletUser>

Method used by admin to get all users present in database

7.getUser(in userId:Integer): WalletUser

Method used by admin to get a user’s details

through userId

8.getAccountsToApprove(): List<WalletAccount>

Method called when accounts need to be approved

9.approveAccount(in accountId:Integer): WalletAccount

Method used by admin to approve account

10.getAccount(in accountNo:Integer): WalletAccount

To get an account object through accountId

11.getTransactions(in accountNo:Integer, in fromDate:LocalDateTime, in toDate:LocalDateTime): List<TransactionHistory>

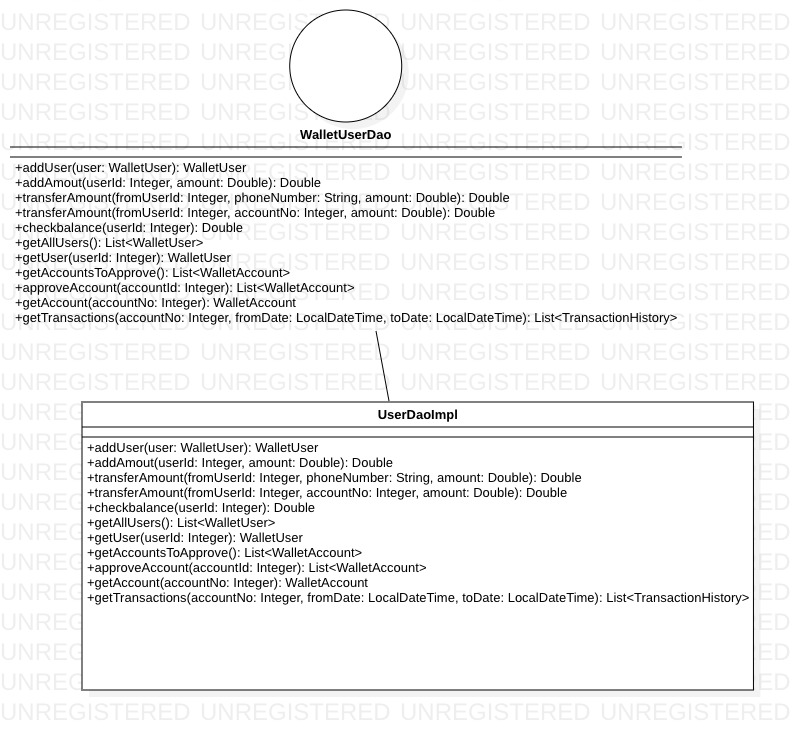
To get the whole list of transaction details from one date to another

12.validateLoginCredentials(in userId:Integer, in password:String): Boolean

To check whether the userId and password is valid or not.

**DAO Layer**

**Class Diagram**



Interfaces:

WalletUserDao

Methods

1. addUser(in user:WalletUser): WalletUser

2. addAmout(in userId:Integer, in amount:Double): Double

3. transferAmount(in fromUserId:Integer, in phoneNumber:String, in amount:Double): Double

4.transferAmount(in fromUserId:Integer, in accountNo:Integer, in amount:Double): Double

5.checkbalance(in userId:Integer): Double

6. getAllUsers(): List<WalletUser>

7.getUser(in userId:Integer): WalletUser

8.getAccountsToApprove(): List<WalletAccount>

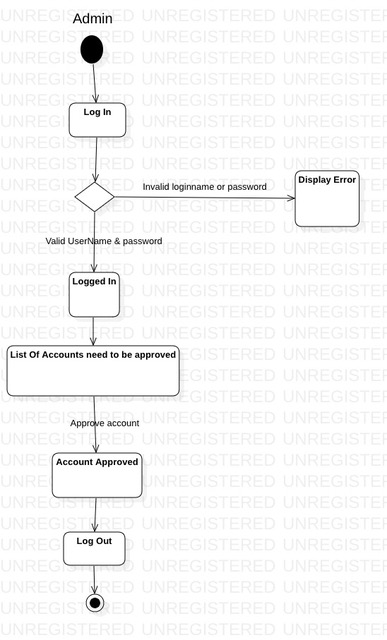
9.approveAccount(in accountId:Integer): List<WalletAccount>

10.getAccount(in accountNo:Integer): WalletAccount

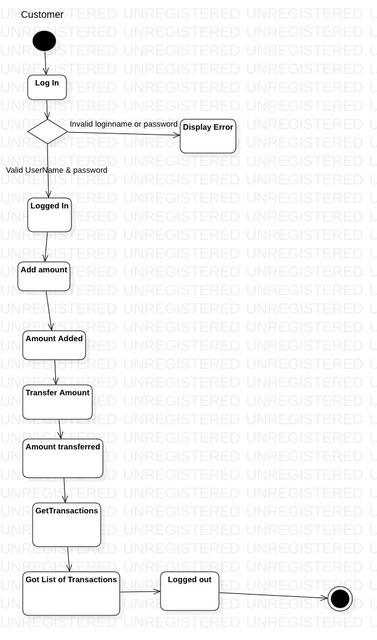
11.getTransactions(in accountNo:Integer, in fromDate:LocalDateTime, in toDate:LocalDateTime): List<TransactionHistory>

**Activity Diagram**

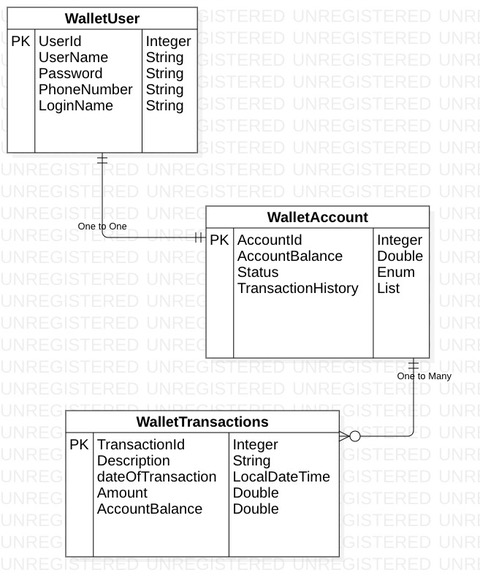
**Admin**



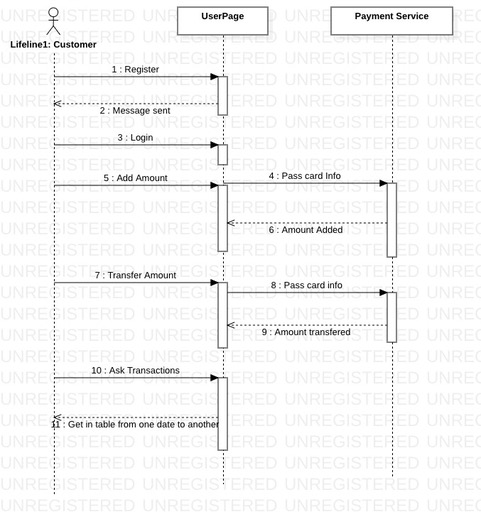
**Customer**



**E-R Diagram**



**Sequence Diagram**



**Validation Rules**

1. UserName should be of atleast 3 characters.
2. Phone no should be of 10 digits.
3. First letter of name should be capital letter.
4. Password should be of atleast 8 characters.
5. FromDate should be less than the ToDate.
6. FromDate shouldn’t be present date.
7. ToDate couldn’t be greater than PresentDate.