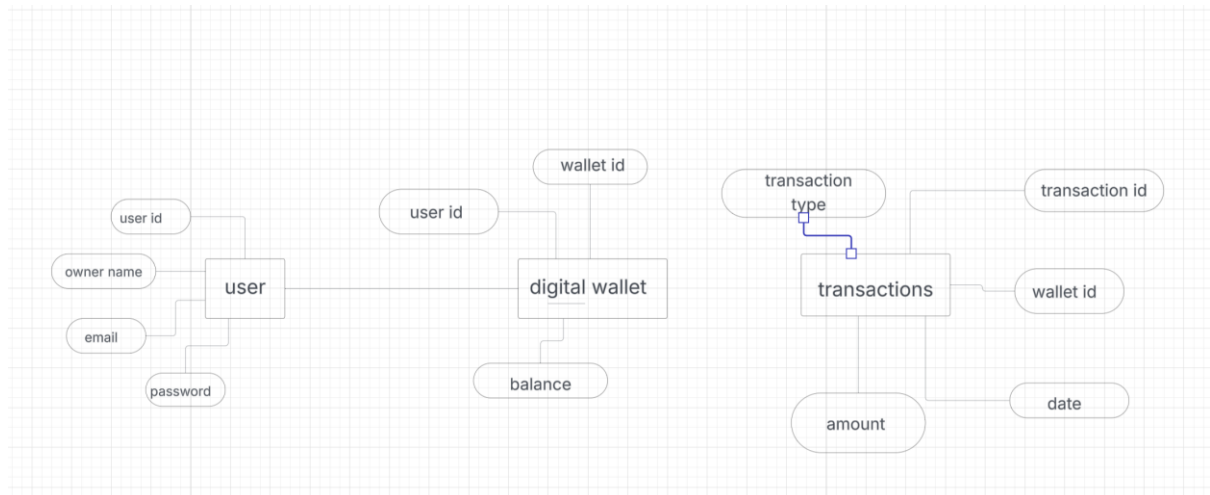


## ER DIAGRAM:



## 1. Introduction

The Digital Wallet project is a C++ application designed to facilitate the management of personal finances through a digital wallet interface. As digital transactions become increasingly prevalent, the need for secure and efficient financial management tools has grown. This project aims to provide users with a simple yet effective way to manage their money, track their spending, and save their financial data.

### 1.1 Purpose

The primary purpose of the Digital Wallet application is to allow users to:

- Add funds to their wallet.
- Spend money while ensuring they do not exceed their balance.
- Save their wallet data to a file for future reference.
- Display wallet details in a user-friendly manner.

## **1.2 Importance**

Digital wallets are essential in today's economy, providing convenience and security for transactions. This project contributes to the growing field of financial technology (FinTech) by offering a basic yet functional digital wallet solution.

## **2. Project Objectives**

The objectives of the Digital Wallet project include:

- Developing a robust C++ application that implements core wallet functionalities.
- Ensuring user-friendly interactions through console-based input and output.
- Implementing error handling to manage invalid operations gracefully.
- Providing data persistence through file operations.

## **3. Key Features**

### **3.1 User Management**

- Each wallet is associated with an owner's name, allowing for personalized management of funds.

### **3.2 Balance Management**

- Users can set an initial balance when creating their wallet.
- The application ensures that the balance cannot go negative, promoting responsible financial management.

### **3.3 Polymorphism**

- The application utilizes polymorphism to allow the addition of funds using both double and integer types, enhancing flexibility in user input.

### **3.4 File Operations**

- Users can save their wallet data to a text file, ensuring that their financial information is preserved between sessions.

## 4. Technical Implementation

### 4.1 Programming Language

- The application is developed in C++, leveraging its object-oriented features for better code organization and reusability.

### 4.2 Key Classes

#### 4.2.1 *DigitalWallet Class*

- **Attributes:**
  - **std::string ownerName:** Stores the name of the wallet owner.
  - **double balance:** Represents the current balance in the wallet.
- **Methods:**
  - **Constructor:** Initializes the wallet with the owner's name and an initial balance.
  - **addMoney(double amount):** Adds a specified amount to the wallet if it is positive.
  - **addMoney(int amount):** Overloaded method to add an integer amount to the wallet.
  - **spendMoney(double amount):** Deducts a specified amount from the wallet if sufficient balance exists.
  - **saveToFile(const std::string& filename):** Saves the wallet's owner name and balance to a specified file.

#### 4.2.2 *Friend Function*

- **displayWalletDetails:** A friend function that allows access to private members of the **DigitalWallet** class to display wallet information.

### 4.3 User Interaction

The main function serves as the entry point for the application, guiding users through the following steps:

1. **Creating a Wallet:** Users can create a new wallet instance by providing their name and an initial balance.
2. **Displaying Wallet Details:** The application displays the current wallet owner and balance.

3. **Adding Money:** Users can add money to their wallet using both double and integer values.
4. **Spending Money:** Users can spend money, with checks in place to prevent overspending.
5. **Saving Data:** The wallet's state can be saved to a text file for future reference.

## 4.4 User Feedback

The application provides console messages to inform users of successful operations or errors, enhancing the user experience.

## 5. Error Handling

The application incorporates robust error handling to ensure:

- Only positive amounts can be added or spent.
- Spending does not exceed the current balance.
- File operations handle potential errors, such as inability to open a file.

## 6. Data Persistence

The wallet data is saved to a text file (**wallet\_data.txt**) in a simple format:

RunCopy code

```
1Owner: John Doe
2Balance: 190
```

This allows users to retrieve their wallet information in subsequent sessions.

## 7. Development Environment

### 7.1 Tools Used

- **C++ Compiler:** Clang is used for compiling the application.
- **IDE:** Visual Studio Code provides a user-friendly environment for development.

- **Debugger:** LLDB is utilized for debugging the application.

## 7.2 Configuration Files

- **settings.json:** Configures compiler paths, warning settings, and other preferences for the C++ runner.
- **c\_cpp\_properties.json:** Defines include paths and IntelliSense settings for code completion.
- **launch.json:** Configures the debugger settings for running the application.

## 8. Testing and Validation

### 8.1 Test Cases

The application is tested through various scenarios, including:

- Adding money (both double and integer).
- Spending money with sufficient and insufficient balance.
- Saving and retrieving wallet data.

### 8.2 Validation

User inputs are validated to ensure that only valid operations are performed, with appropriate error messages displayed for invalid actions.

## 9. Future Enhancements

The project can be expanded with additional features, such as:

- **User Authentication:** Implementing a secure login system to protect user data.
- **Integration with Online Payment Systems:** Allowing users to make online purchases directly from their wallet.
- **Mobile Application Version:** Developing a mobile-friendly version for greater accessibility.
- **Enhanced User Interface:** Creating a graphical user interface (GUI) for improved user experience.

## 10. Conclusion

The Digital Wallet project successfully demonstrates the core functionalities of a digital wallet application using C++. It highlights the importance of financial management tools in the digital age and sets the foundation for future enhancements and developments in the FinTech space.

## 11. References

- C++ programming resources, including textbooks and online tutorials.
- Documentation for libraries used in file handling and console I/O operations.

Code snippet:

The screenshot shows the Visual Studio Code editor with a C++ project named "c++ project". The Explorer sidebar on the left lists the project files: `.vscode`, `digitalwallet`, `DigitalWallet.cpp`, `DigitalWallet.exe`, `DigitalWallet.h` (selected), `main.cpp`, and `wallet_data.txt`. The main editor displays the content of `DigitalWallet.h`. The code defines a `DigitalWallet` class with private attributes `ownerName` and `balance`. It includes a constructor, an inline `getBalance()` function, and several public methods for adding and spending money, saving data to a file, and displaying details. A friend function `displayWalletDetails` is also declared. The terminal at the bottom shows the command prompt `nishantdhall@Nishants-MacBook-Air c++ project %`.

```
1 #ifndef DIGITALWALLET_H
2 #define DIGITALWALLET_H //structure definition of digitalwallet.cpp
3
4 #include <iostream>
5 #include <fstream>
6 #include <string>
7
8 class DigitalWallet {
9 private:
10     std::string ownerName;
11     double balance;
12
13 public:
14     // Constructor
15     DigitalWallet(std::string name, double initialBalance);
16
17     // Inline function to get the balance
18     inline double getBalance() const { return balance; }
19
20     //polymorphism
21     // Function to add money if value is double
22     void addMoney(double amount);
23
24     // Function to add money if value is int
25     void addMoney(int amount);
26
27     // Function to spend money
28     void spendMoney(double amount);
29
30     // Function to save wallet data to a file
31     void saveToFile(const std::string& filename);
32
33     // Friend function to display wallet details
34     friend void displayWalletDetails(const DigitalWallet& wallet);
35
36 #endif
```

This screenshot is similar to the first one, showing the same VS Code editor and project. However, the code in `DigitalWallet.h` is more complete. It includes the closing brace for the `DigitalWallet` class, the `displayWalletDetails` friend function definition, and the `#endif` preprocessor directive. The terminal at the bottom remains the same, showing the command prompt.

```
1 #ifndef DIGITALWALLET_H
2 #define DIGITALWALLET_H //structure definition of digitalwallet.cpp
3
4 #include <iostream>
5 #include <fstream>
6 #include <string>
7
8 class DigitalWallet {
9 private:
10     std::string ownerName;
11     double balance;
12
13 public:
14     // Constructor
15     DigitalWallet(std::string name, double initialBalance);
16
17     // Inline function to get the balance
18     inline double getBalance() const { return balance; }
19
20     //polymorphism
21     // Function to add money if value is double
22     void addMoney(double amount);
23
24     // Function to add money if value is int
25     void addMoney(int amount);
26
27     // Function to spend money
28     void spendMoney(double amount);
29
30     // Function to save wallet data to a file
31     void saveToFile(const std::string& filename);
32
33     // Friend function to display wallet details
34     friend void displayWalletDetails(const DigitalWallet& wallet);
35 };
36
37 #endif // DIGITALWALLET_H
```

This screenshot shows the Visual Studio Code editor with the file `DigitalWallet.cpp` open. The Explorer sidebar on the left shows the project structure: `.vscode`, `digitalwallet`, `DigitalWallet.cpp`, `DigitalWallet.exe`, `DigitalWallet.h`, `main.cpp`, and `wallet_data.txt`. The main editor displays the following C++ code:

```
1 #include "DigitalWallet.h"
2
3 // Constructor implementation
4 DigitalWallet::DigitalWallet(std::string name, double initialBalance) {
5     ownerName = name;
6     balance = initialBalance >= 0 ? initialBalance : 0; // Ensure balance is non-negative
7 }
8
9 // Function to add money if value is double—polymorphism
10 void DigitalWallet::addMoney(double amount) {
11     if (amount > 0) {
12         balance += amount;
13         std::cout << "Added " << amount << " to wallet. New balance: " << balance << std::endl;
14     } else {
15         std::cout << "Amount must be positive!" << std::endl;
16     }
17 }
18
19 // Function to add money if value is int—polymorphism
20 void DigitalWallet::addMoney(int amount) {
21     if (amount > 0) {
22         balance += amount;
23         std::cout << "Added " << amount << " to wallet. New balance: " << balance << std::endl;
24     } else {
25         std::cout << "Amount must be positive!" << std::endl;
26     }
27 }
28
29 // Function to spend money
30 void DigitalWallet::spendMoney(double amount) {
31     if (amount > 0 && amount <= balance) {
32         balance -= amount;
33         std::cout << "Spent " << amount << " from wallet. New balance: " << balance << std::endl;
34     } else {
35         std::cout << "Insufficient balance or invalid amount!" << std::endl;
36     }
37 }
38
39 // Function to save wallet data to a file
40 void DigitalWallet::saveToFile(const std::string& filename) {
41     std::ofstream outFile(filename);
42     if (outFile.is_open()) {
43         outFile << "Owner: " << ownerName << std::endl;
44         outFile << "Balance: " << balance << std::endl;
45         outFile.close();
46         std::cout << "Wallet data saved to " << filename << std::endl;
47     } else {
48         std::cout << "Unable to open file!" << std::endl;
49     }
50 }
51
52 // Friend function to display wallet details
53 void displayWalletDetails(const DigitalWallet& wallet) {
54     std::cout << "Wallet Owner: " << wallet.ownerName << std::endl;
55     std::cout << "Current Balance: " << wallet.getBalance() << std::endl;
56 }
```

The bottom panel shows the TERMINAL with the prompt `nishantdhall@Nishants-MacBook-Air c++ project %`. The status bar at the bottom indicates the cursor is at line 8, column 1, with 4 spaces, UTF-8 encoding, LF line endings, C++ language, and the macOS-clang-arm64 compiler.

This screenshot shows the Visual Studio Code editor with the file `DigitalWallet.cpp` open, displaying the continuation of the code from the previous image:

```
28
29 // Function to spend money
30 void DigitalWallet::spendMoney(double amount) {
31     if (amount > 0 && amount <= balance) {
32         balance -= amount;
33         std::cout << "Spent " << amount << " from wallet. New balance: " << balance << std::endl;
34     } else {
35         std::cout << "Insufficient balance or invalid amount!" << std::endl;
36     }
37 }
38
39 // Function to save wallet data to a file
40 void DigitalWallet::saveToFile(const std::string& filename) {
41     std::ofstream outFile(filename);
42     if (outFile.is_open()) {
43         outFile << "Owner: " << ownerName << std::endl;
44         outFile << "Balance: " << balance << std::endl;
45         outFile.close();
46         std::cout << "Wallet data saved to " << filename << std::endl;
47     } else {
48         std::cout << "Unable to open file!" << std::endl;
49     }
50 }
51
52 // Friend function to display wallet details
53 void displayWalletDetails(const DigitalWallet& wallet) {
54     std::cout << "Wallet Owner: " << wallet.ownerName << std::endl;
55     std::cout << "Current Balance: " << wallet.getBalance() << std::endl;
56 }
```

The bottom panel shows the TERMINAL with the prompt `nishantdhall@Nishants-MacBook-Air c++ project %`. The status bar at the bottom indicates the cursor is at line 8, column 1, with 4 spaces, UTF-8 encoding, LF line endings, C++ language, and the macOS-clang-arm64 compiler.



