Cairo University Faculty of Computers and Artificial Intelligence



**Software design specification document**

**2023**

**Instapay**

**Project Team**

|  |  |  |
| --- | --- | --- |
| **ID** | **Name** | **Email** |
| 20201098 | Shreen Samhi Abdo | shreensamhi01@gmail.com |
| 20200588 | Nada Arfat Saber | n.1762002@gmail.com |
| 20201065 | Rahma Mohamed Sayed | rahmamuhammed285@gmail.com |
| 20200330 | Alaa Elden Ebrahim | alaaebrahim387@gmail.com |

Contents

[Instructions[To be removed] 2](#_Toc120811426)

[Class diagram design 2](#_Toc120811427)

[Class diagram Explanation 3](#_Toc120811428)

[Sequence diagram design 3](#_Toc120811429)

[Github repository link 4](#_Toc120811430)

# Class diagram design

**A diagram of text and words

Description automatically generated with medium confidence**

# Class diagram Explanation

Design Patterns Used:

Strategy Pattern:

Classes: ElectricityBillPaymentStrategy, GasBillPaymentStrategy, WaterBillPaymentStrategy

Interface: IBillPaymentStrategy

Advantages:

Flexibility: The Strategy Pattern allows you to define a family of algorithms, encapsulate each one, and make them interchangeable. In this case, the IBillPaymentStrategy interface defines the algorithm for paying bills, and each concrete class (ElectricityBillPaymentStrategy, GasBillPaymentStrategy, WaterBillPaymentStrategy) encapsulates a specific implementation.

Code Reusability: Each bill payment strategy is encapsulated within its own class, promoting code reusability. If a new bill payment strategy needs to be added, a new class implementing IBillPaymentStrategy can be created without modifying existing code.

Easier Maintenance: Modifications to a specific bill payment strategy can be done independently of other strategies, making the codebase more maintainable.

Strategy Pattern

Classes: TransferToWalletStrategy, TransferToInstapayStrategy, TransferToBankStrategy

Interface: ITransferStratgey

Advantages:

Dependency Inversion: By using dependency injection in the constructor, these classes adhere to the Dependency Inversion Principle. They depend on abstractions (IExternalAccountService and IUserRepository), not on concrete implementations. This makes the code more flexible and easily adaptable to changes.

Separation of Concerns: Each strategy (Transfe0rToWalletStrategy, TransferToInstapayStrategy, TransferToBankStrategy) focuses on its own transfer logic. Changes in one strategy do not affect the others, promoting modularity and maintainability.

Template Method Pattern:

Class: UserAuthenticationService

Advantages:

Common Structure: The UserAuthenticationService class provides a template method (RegisterUser) with a common structure, allowing subclasses (WalletAuthenticationService, BankAuthenticationService) to override specific steps without changing the overall algorithm.

Enforces a Process: The template method defines the steps of user registration, ensuring that certain steps are followed in a specific order. This helps in maintaining a consistent process across different authentication services.

Singleton Pattern:

Class: InMemoryUserRepository

Advantages:

Single Instance: The InMemoryUserRepository class follows the Singleton Pattern, ensuring that only one instance of the repository exists. This is useful for scenarios where a single point of access to the user repository is required throughout the application.

Global Access: The static method Repository provides a global point of access to the singleton instance, allowing other parts of the code to retrieve the repository without creating multiple instances.

# Sequence diagram design

Sequence diagram 1 : AccountType

A screenshot of a computer

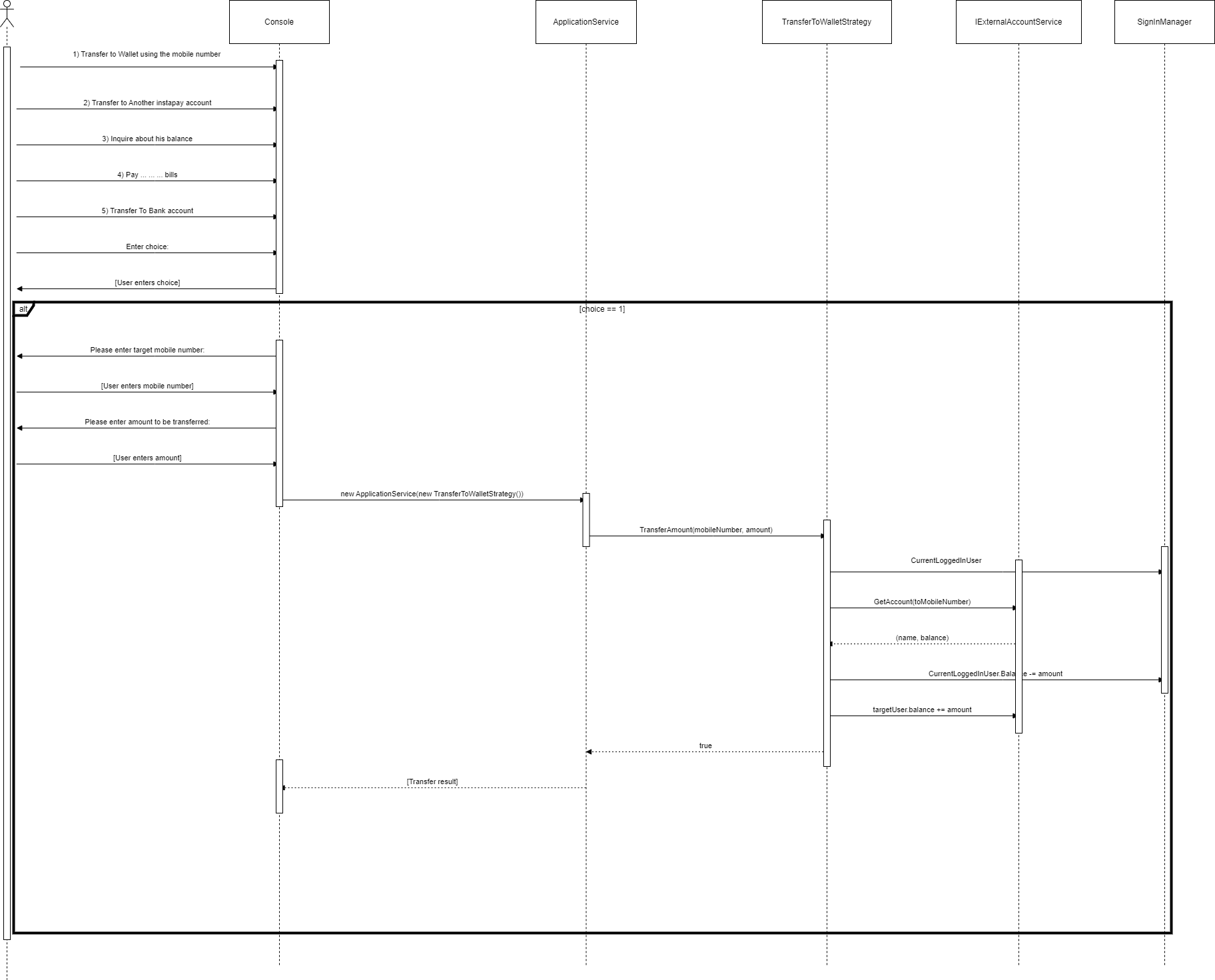
Description automatically generated

**Sequence diagram 2 : SignUp/SignIn**

**A black and white page

Description automatically generated**

**Sequence diagram 3 : Transfer To Wallet Using The Mobile Number**

****

sequence diagram 4 : Pay GazBill

A diagram of a diagram

Description automatically generated

# Github repository link

* <https://github.com/shreensamhi/Instapay_system.git>
* If the link didn’t work, please contact with anyone of us via email or in our section.