Cairo University Faculty of Computers and Artificial Intelligence



**Advanced SE**

**Software design specification document**

**2022**

**Project Team**

|  |  |  |
| --- | --- | --- |
| **ID** | **Name** | **Email** |
| 20200359 | Omar Mohamed Mostafa | [omar.m.elesawy2002@gmail.com](mailto:omar.m.elesawy2002@gmail.com) |
| 20200514 | Marwa Ahmed Mohamed | marwamubarak6@gmail.com‏ |
| 20200268 | Doha Abd-ElBasset Ahmed | [dohae7222@gmail.com](mailto:dohae7222@gmail.com) |

# Class diagram design

**Diagram

Description automatically generated**

# Class diagram Explanation

* **We used:**

• Strategy design pattern,

We found that we need to create something specific in a lot of different ways and extract all of these algorithms into separate classes.

Participating:1. Payment<interface> implements WalletPayment, CashPayment, ddddddddddd CreaditPayment.

2. Discount <interface> implements OverAllDiscount, SpecificDiscount.

• Factory method

We found that we need to create objects in a superclass, but allows subclasses to alter the type of objects that will be created.

Participating:1. PaymentFactory , Payment, WalletPayment, CreditPayment, CashPayment.

2. ServiceFactory, Service, MobileRechargeSerice, InternetPaymentService, LandLineService, DonationService.

3. DiscountFacotry, Discount, OverAllDisount, SpecificDiscount.

• Abstract Factory method

We found that we need to produce families of related objects without specifying their concrete classes.

Participating: FactoryOfServiceProviderFactory, ServiceProviderFactory, DonationProviderFactory, LandlineProivderFactory, MobileProviderFactory, InternetProviderFactory, ServiceProvider, We, Vodafone, Etisalat, Orange, MothlyRicpitLandLine, QuarterRicpitLandLine,Schools, CancerHospital, MGOs.

• Decorator design pattern

We found that we need to attach new behaviors to objects by placing these objects inside special wrapper objects that contain the behaviors.

Participation: Service , ServiceDecorator, DiscountDecorator,

• Command design pattern

We suggest that UI objects shouldn’t send these requests directly. Instead, we should extract all the request details, such as the object being called, the name of the method and the list of arguments into a separate command class with a single method that trigger

Participation: Invoker, Command <interface>implements Register, Login

• Singelton design pattern

We found that we need to ensure that a class has only one instance, while providing a global access point to this instance.

Participating: SavedData

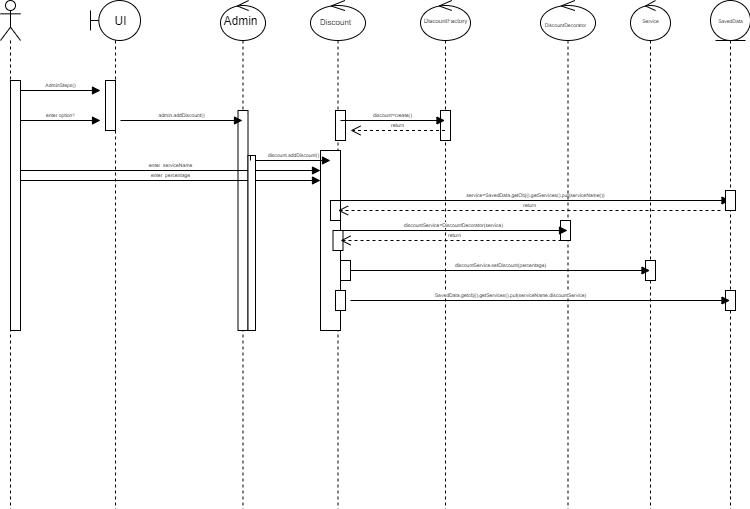
• Template method

We found that defines the skeleton of an algorithm in the superclass but lets subclasses override specific steps of the algorithm without changing its structure.

Participating: Service, MobileRechargeSerice, InternetPaymentService, LandLineService, DonationService

# Sequence diagram design

• Admin Add discount



**Diagram

Description automatically generated**

Login and add fund

**Diagram

Description automatically generated**

Make service

**Diagram

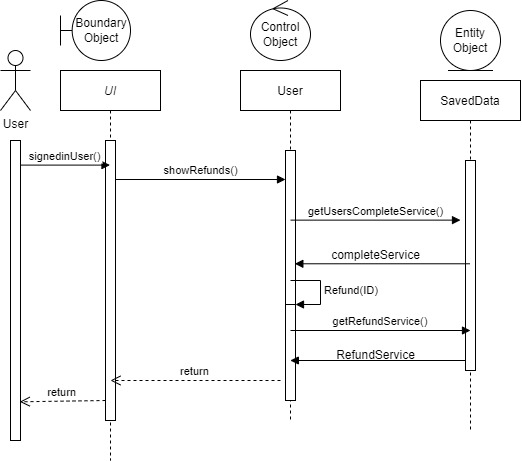
Description automatically generated**

Show Refunds admin

Diagram, schematic

Description automatically generated

SignUP



User make refund in complete service

# Github repository link

* [omar-el-esawy/SW-Project-: project in software course (github.com)](https://github.com/omar-el-esawy/SW-Project-)