Cairo University Faculty of Computers and Artificial Intelligence



**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](mailto:marwamubarak6@gmail.com‏) |
| 20200268 | Doha Abd-ElBasset Ahmed | [dohae7222@gmail.com](mailto:dohae7222@gmail.com) |

# Class diagram design

If the photo isn’t clear, please open it as a photo from github repository or you found it also at the source code uploaded

**Diagram, schematic

Description automatically generated**

# 

Subsystem decomposition diagram

Graphical user interface

Description automatically generated with medium confidence

# 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.

* MVC

We use mvc to separate all application's concerns into model, view, and controller

• 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,

• 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 (1-admin add discount)

# Diagram Description automatically generated

2- user add money to wallet

**Diagram

Description automatically generated**

3-admin set cash availability to service

**Diagram

Description automatically generated**

4-user ask to refund service

**Diagram

Description automatically generated**

5- user sign up

**Diagram

Description automatically generated**

6- admin show refund transactions

**Diagram

Description automatically generated**

# 

# Requirements Exposure as Web Service API

**Part 1: Exposed Postman Collection**

**You will find the exposed postman collection at the uploaded source code**

**(named: MoneySystem.postman\_collection.json )**

**Part 2:**

|  |  |  |
| --- | --- | --- |
| Requirement |  | Exposed API |
| (user Login) The system should check if the user registered before or not |  | GET /login  Ex: http://localhost:8080/login  the body:  json  {  "userEmail": "mrmr@gmail.com",  "password": "123"  }  (if the user doesn't exist return failed else return the user) |
| The user should be able to sign-up to the system. Given the user’s email and a password and the username,(the system should check if the user already exist or not) |  | POST / signUp (if the user already exists return failed else return that successfully signed up)  ex:  <http://localhost:8080/signUp>  Body (json)  json  {  "userEmail": "mrmr@gmail.com",  "userName": "m",  "password": "123"  } |
| The user can show the services he did (the system will loop all the complete services of the login user ) |  | GET/showCompleteService (Any successful service has done by the user will be displayed, if there is no service done return empty)  You should login or signup then login to do this operation  Ex:  <http://localhost:8080/showCompleteService> |
| The user can ask for refund request of any complete service he did (the system will put it in the pending requests and send it to the admin)but before that you should login |  | POST/refund?id={id} (any successful refund process return request Pending when this id contains in complete services else return invalid id)  Ex:  <http://localhost:8080/refund?id=1>  to make Refund-- write the above URL with value of parameter id ->(int) which you want to be refunded  You should login or signup then login to do this operation  Query Params  id=1 |
| The user can show the pending requests he did(the system will loop all the the pending requests of the login user ) |  | GET/showPendingRequest (Any pending request has done by the admin will be displayed, if there is no pending request return empty)  Ex:  [http://localhost:8080/showPendingRequest](E:\\StudioProjects\\software-project\\ http:\\localhost:8080\\showPendingRequest)  You should login or signup then login to do this operation |
| The user can show the rejected requests he did(the system will loop all the the rejected requests of the login user ) |  | GET/showRejectedRequest (Any rejected request has done by the admin will be displayed, if there is no rejected request return empty)  Ex:  [http://localhost:8080/showRejectedRequest](http://localhost:8080/showRejectedRequest  )    You should login or signup then login to do this operation |
| The user can show the accepted requests he did(the system will loop all the accepted requests of the login user ) |  | GET/showAcceptedService (Any accepted request has done by the admin will be displayed, if there is no accepted request return empty)  Ex:  <http://localhost:8080/showAcceptedService>  You should login or signup then login to do this operation |
| The user can add money to his wallet from his credit card(the system go to check if the credit card has the amount he asked) |  | POST/addToWallet?amount={amount} (If his credit card has enough money to add this amount in wallet return done successfully else return there isn't enough money)  Ex:  <http://localhost:8080/addToWallet?amount=430>  to add money to user's wallet, write the above URL with parameter amount->(int)  Query Params  amount=430  You should login or signup then login to do this operation |
| The user can show all the discounts that added by the admin(system will return the available discounts) |  | GET/showDiscounts (if there is discounts added by the admin return this discounts, else return there is no discount )  Ex:  <http://localhost:8080/showDiscounts>  You should login or signup then login to do this operation |
| The user can pay for any service in the system. The system should prompt the user to the payment form when the signed in user asks to pay for any service. |  | POST/serve Input body ex below  (return done successfully and the service receipt if we can do the service ,else return message with the reason of why it is not completed)  Ex:  <http://localhost:8080/serve>  to serve any service ---write the above URL with the following body  Bodyraw (json)  json  {   "paymentWayID": 1,   "serviceProviderID": 1,   "amount": 33,   "info": "01112247338",   "serviceName": "Landline" }  You should login or signup then login to do this operation  Note: if you change the base url change it also from the saved data |
| The user should be able to search for any service in the system. The user can type the service name and the system will return all services that match the user query. |  | GET/search?id={id} (it will return the results that contains the Id ,If there is not return Not Found) Ex: <http://localhost:8080/search?id=r>  to search about any service ---write the above URL with id parameter  Query Params  id=r  You should login or signup then login to do this operation |
| The admin should be able to list all refund requests |  | GET/showRefund (Return the refund requests if there is else return empty)  Ex:  [http://localhost:8080/showRefund](http://localhost:8080/showRefund )  to show the current refund requests that require response --->write the above URL |
| The admin should be able to accept or reject any refund request and if any refund request got accepted a refund transaction should be processed. |  | POST/refundResponse?id={id}&op={op} (it will return the refund accepted or rejected or invalid id)  Ex: <http://localhost:8080/refundResponse?id=1&op=2> to accept or reject any refund request --> write the above URL with 2paramters: id(the id of the service) and the op(write 1 to accept the request and write 2 to reject it )  Query Params  id=1  op=2 |
| The admin should be able to list all user Payment transaction |  | GET/showPaymentTransaction (if it is empty return empty ,else return list of the payment transactions )  Ex:  <http://localhost:8080/showPaymentTransaction>  write the above URL to show payment transactions |
| If the service that should to receive the payment accepts cache on delivery, then this option should be visible too. |  | **PUT**  /cashAvailability?serviceName={serviceName}&visible={visible}  (return done if the service found ,else return not found)  Ex:  <http://localhost:8080/cashAvailability?serviceName=Landline&visible=false>  to set the availability of cash of any service write the above URL with the following parameters  serviceName and visible(true or false)  Query Params  serviceName=Landline  Visible=false |
| The admin should be able to add discounts to the system. There are two types of discounts. a. Overall discounts. For example the user should have 10% discount for the first transaction (regardless the service) b. Specific discount.. For example the admin can apply 20% discount for all mobile recharge services. For any given service. All overall discounts and specific discounts for this service should apply |  | POST  /addDiscount?choice={choice}&amount={amount}&serviceName={serviceName}  choice(1 to specific discount or 2 to add to all services) and amount (the discount like 0.1 for 10%)and the serviceName(write it if you choose option 1)  Ex:  <http://localhost:8080/addDiscount?choice=2&amount=10>  to add discount write the above url with the following paramters:  Query Params  choice=2  amount=10 |
| The admin should be able to list all user refund transactions |  | **GET**  **/showRefundTransaction**  (if the list is empty return empty else return the list)  **Ex:**  [http://localhost:8080/showRefundTransaction](http://localhost:8080/showRefundTransaction )    to show refund transactions -->write the above URL |
| The admin should be able to list all user refund transactions |  | **GET**  **/showAddToWalletTransaction**  (if the list is empty return empty else return the list)  Ex:  <http://localhost:8080/showAddToWalletTransaction>  to show add to wallet transactions -->write the above URL |
| The system should prompt the user to the payment form when the user asks to pay for any service |  | GET/pay?amount={amount}&discount={discount}&paymentWay={paymentWay} if you have the amount return done successfully else return there is not enough money  ex:  <http://localhost:8080/pay?amount=15&discount=0.1&paymentWay=2>  to pay any service -->write the above URL with the following parameters  amount(the amount you want to spend) ,discount(the discount on the service) ,paymentWay (to check if it is valid to pay by it or no)  Query Params  amount =15  discount =0.1  paymentWay =2  You should login or (signup then login) then use any service to call this endpoint |
| The provider will get the amount and info for the service the user asks for |  | GET/setInfo?amount={amount}&info={info}&providerId={providerId} Ex:  [http://localhost:8080/setInfo?amount=10&info=012234455&providerId=1](E:\\StudioProjects\\software-project\\Open Request   http:\\localhost:8080\\setInfo?amount=10&info=012234455&providerId=1)  to send the info and the amount required to the service provider write the above url with the parameters amount(the amount of money),info(the phone number),and providerId(to specify provider)  Query Params  amount =10  info=012234455  providerId=1  You should login or (signup then login) then use any ervice to do this operation |

# Github repository link

* <https://github.com/MarwaMubarak/software-project>