Cairo University Faculty of Computers and Artificial Intelligence



**Software design specification document**

**2022**

**Project Team**

|  |  |  |
| --- | --- | --- |
| **ID** | **Name** | **Email** |
| 20216108 | Nada osama ahmed ali | princeegypt28@gmail.com |
| 20206012 | Toqa ahmed moatez saad | 70qahmed@gmail.com |

Contents

[decomposotion diagram 2](#_Toc120811426)

[Class diagram design 2](#_Toc120811427)

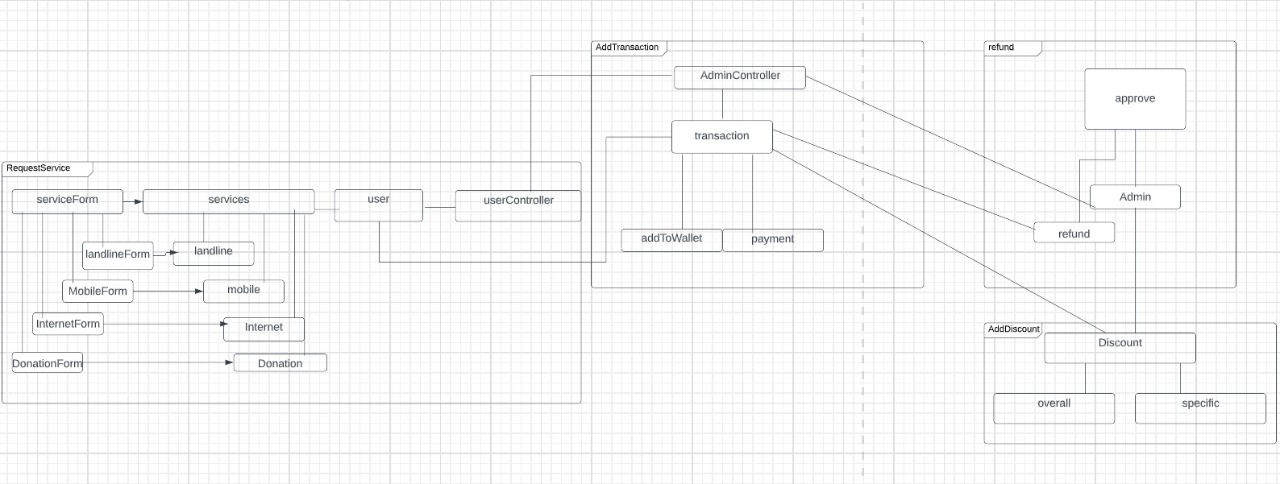
[Class diagram Explanation 2-3](#_Toc120811428)

[Sequence diagram design 3](#_Toc120811429)-4-5

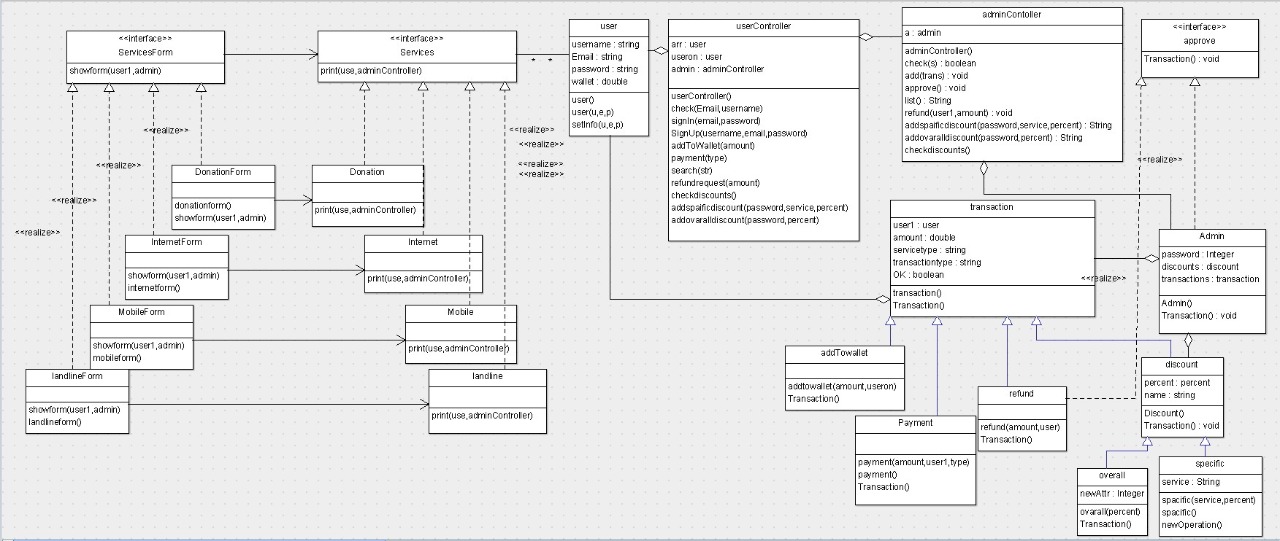
Requirements Exposure as Web Service API............................................................................................5-6-7-8

[Github repository link 4](#_Toc120811430)

# Decompsotion diagram



# Class diagram design



# Class diagram Explanation

We used the following patterns in our class diagram:

1. factory design pattern:

- why : to let the service form decide which service to instantiate without the user intervintion.

- classes : serviceform (creator) - donationform ,internetform ,mobileform ,landlineform (concrete creator) - service (product) - donation ,internet ,mobile ,landline (concrete prodeuct).

1. Proxy design pattern:

- why : so we can hide the logic of the admin approving the refund from the acutal refund work and make them act as two seprate parts.

- classes : approv (subject) - refund (real subject) - admin (proxy)

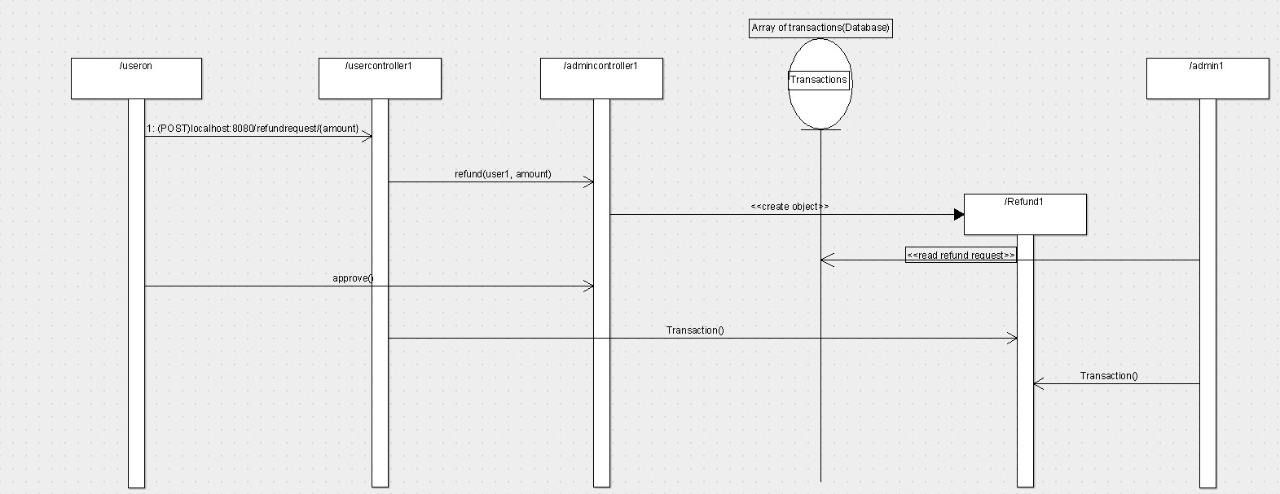
1. Decorator design pattern:

- why : to add the responability of applaying the discount to the payment dynamically.

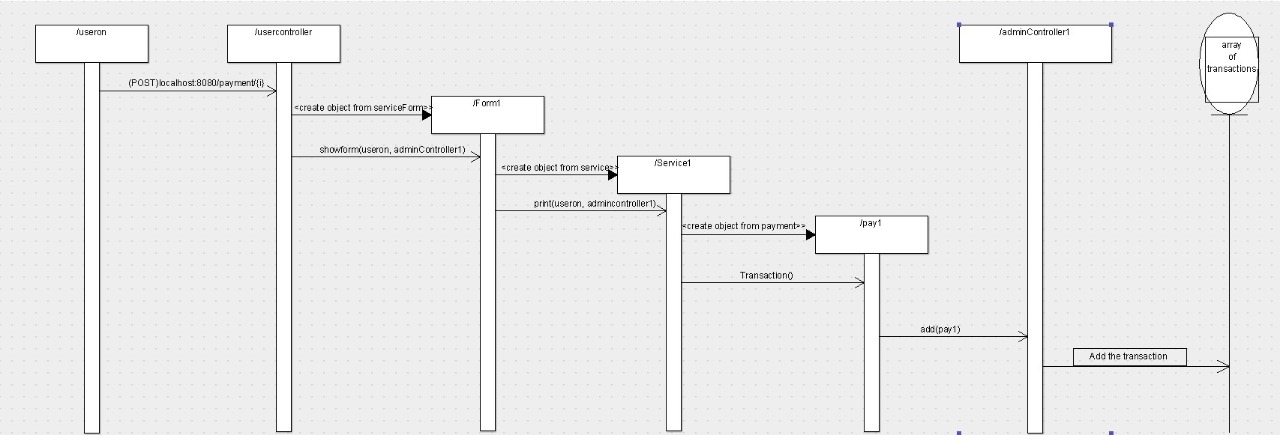
- classes : transcation (component)- payment (concrete component) - discount (decorater) - spacificdiscount ,ovaralldiscount (concrete decorator)

# Sequence diagram design

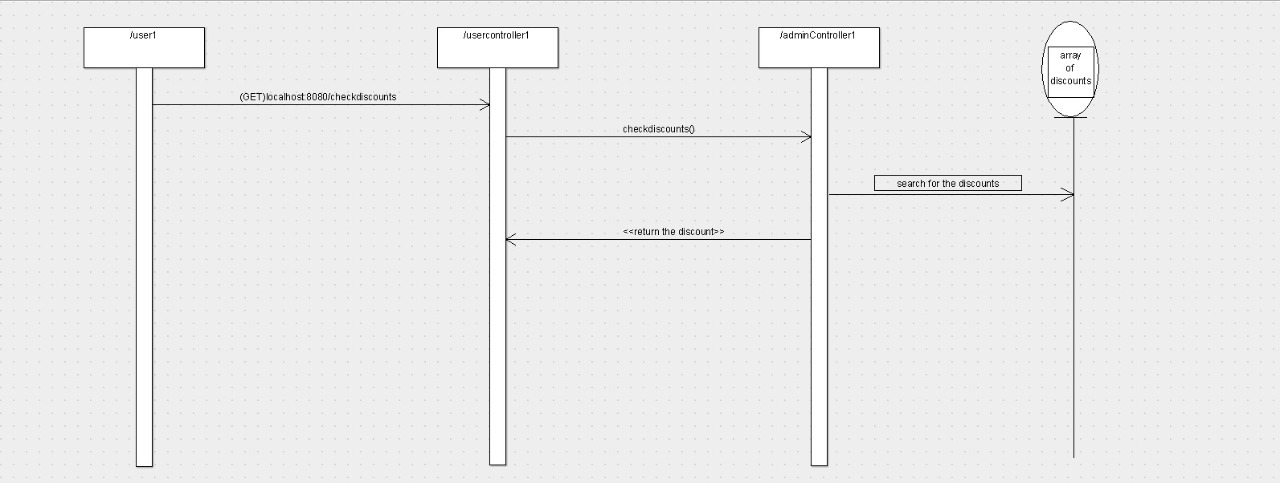
Refund sequence diagram:



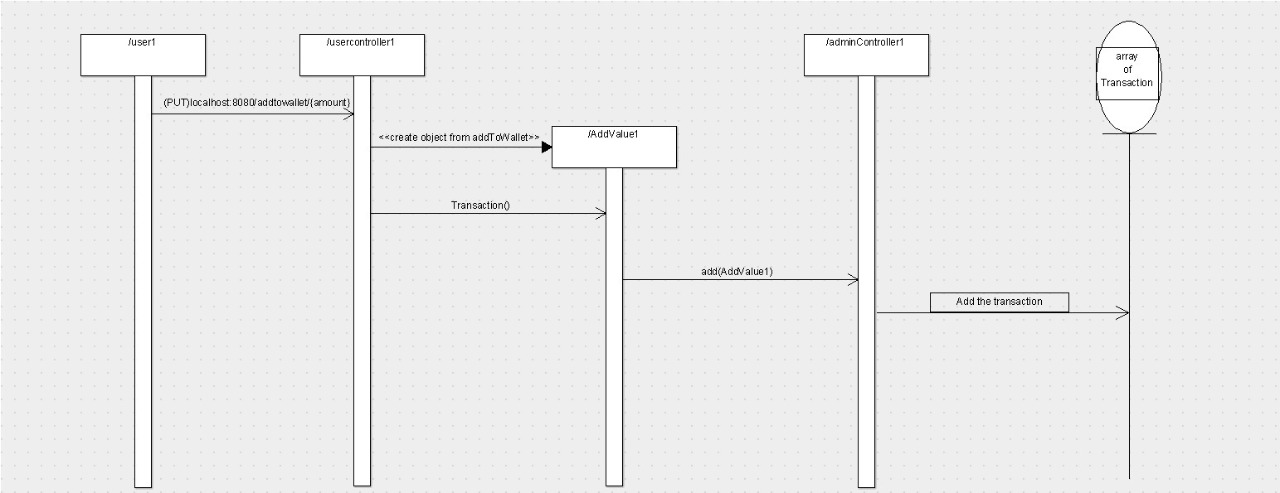
Payment sequence diagram:

****

**check discounts sequence diagram:**

****

**Add to wallet sequence diagram:**

****

# Requirements Exposure as Web Service API

**Part 1: Exposed Postman Collection link:**

https://elements.getpostman.com/redirect?entityId=24963271-9a5551e7-6210-47b8-9011-de6f8b7c2e44&entityType=collection

**Part 2:**

Note: admin password is 5555

|  |  |
| --- | --- |
| Requirement | Exposed API |
| The user should be able to sign-in to the system. Given the user’s email and a  password, the user can login to the system and use any of the system functionalities. | 1. GET /signin/{Email}/{Password}   A service to sign in to the system. Return “signed in successfully” if the user signed up to the system before, else return ”user does not exist”.  Input: email and password.  Example: localhost:8080/signin/neda/neda |
| The user should be able to sign up to the system. The user should provide his username, email and password. The system should check if the username or the email is registered before, if they are not registered before then the signup process should complete successfully, if not, the system will show an error to the user. | POST /signup/{username}/{Email}/{Password} A service to sign up to system. This service return “user already registered” if the user signed up before else it returns “user registered successfully” . after the user sign up he gets automatically signed in.  Input: username,email and password.  Example: localhost:8080/signup/neda/neda/neda |
| 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. | 1. GET /search/{str}   A service to search for a service.return service name if the service exists else return “service does not exist”  Input: service name (mobile,internet,landline,donations).  Example: localhost:8080/search/mobile |
| The user can pay for any service in the system. The system should prompt the user to the payment form when the user asks to pay for any service. The default way is to pay via credit card. The system should allow the user to consume from the wallet for this payment. If the service that should to receive the payment accepts cache on delivery, then this option should be visible too. | 1. POST /payment/{i}   A service to pay.  Input: service name (mobile,internet,landline,donations).  Example: localhost:8080/payment/mobile |
| The user can ask for a refund for any complete transaction to any given service. The refund request will be issued by the user and sent to the admin. | 1. POST /refundrequest/{amount}   A service to submit a refund request. Return “refund requested”.  Input: amount of the payment transcation that need to be refunded.  Example: localhost:8080/refundrequest/50 |
| The user should be able to add any funds to the wallet. Adding funds to the wallet should be done via credit card. | 1. PUT /addtowallet/{amount}   A service to add money to the user wallet via credit card. Return "amount successfully added to wallet amount of money on the wallet = " and the new amount of the money in the wallet  Input: amount of money you want to add to.  Example: localhost:8080/addtowallet/50 |
| The user should be able to check any discount for any service in the system. | 1. GET /checkdiscounts   A service to check the discounts added by the admin. Return discounts type and percent.  Input: no input  Example: localhost:8080/checkdiscounts |
| The admin should be able to add discounts to the system (ovarall discount) . | 1. POST /addovaralldiscount/{password}/{percent}   A service to add an ovarall discount. Return “ovarall discount added”.  Input: admin password and discount percent.  Example: localhost:8080/addovaralldiscount/5555/5 |
| The admin should be able to add discounts to the system (ovarall discount) . | 9-POST /addspaificdiscount/{password}/{service}/{percent}  A service to add an spacific discount. Return “spacific discount added”.  Input: admin password ,service type and discount percent.  Example: localhost:8080/addspaificdiscount/5555/mobile/5 |
| The admin should be able to list all user transactions. | 10-GET /list/{password}  A service to list all user transcations. Return all transcation type and amount.  Input: admin password.  Example: localhost:8080/list/5555 |
| The admin should be able to list all refund requests. The admin should be able to accept  or reject any refund request. | 1. PUT /approve/{password}   A service to approve refund requests.  Input: admin password.  Example: localhost:8080/approve/5555 |

# Github repository link

https://github.com/nadao-1707/sda-project.git