Cairo University Faculty of Computers and Artificial Intelligence



**Software design specification document**

**2022**

**Project Team**

|  |  |  |
| --- | --- | --- |
| **ID** | **Name** | **Email** |
| 20210612 | Omar Hussien Ibrahim Ibrahim | [omarhus98@gmail.com](mailto:omarhus98@gmail.com) |
| 20201152 | Mohamed Ezzat Awad Badawy | [moezza39@gmail.com](mailto:moezza39@gmail.com) |
| 20200060 | Ahmed Waleed Shawky | [theassassin103@gmail.com](mailto:theassassin103@gmail.com) |
| 20200343 | Omar ayman hussien salem | [11410120200343@fci-cu.edu.eg](mailto:11410120200343@fci-cu.edu.eg) |

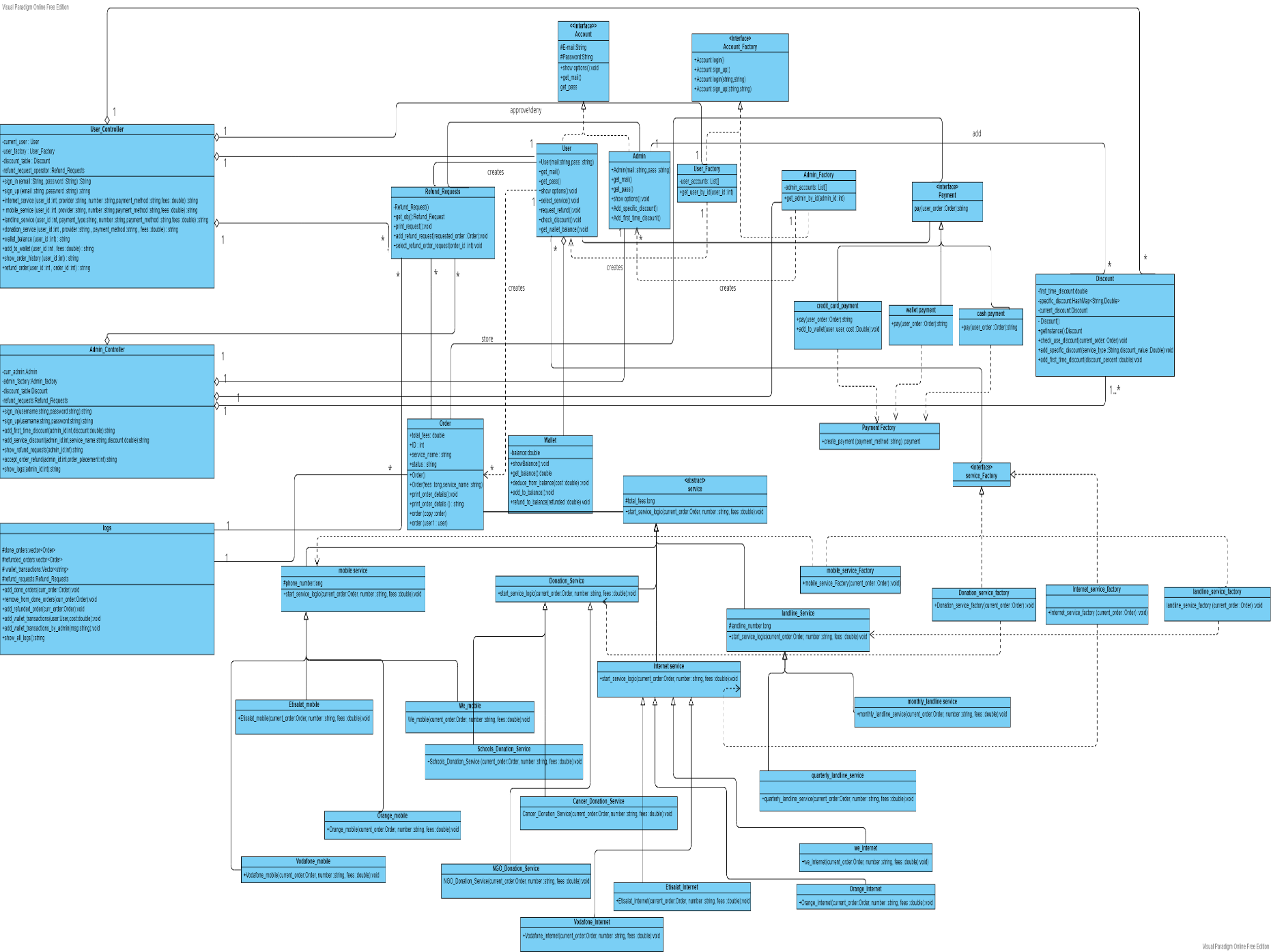
Contents

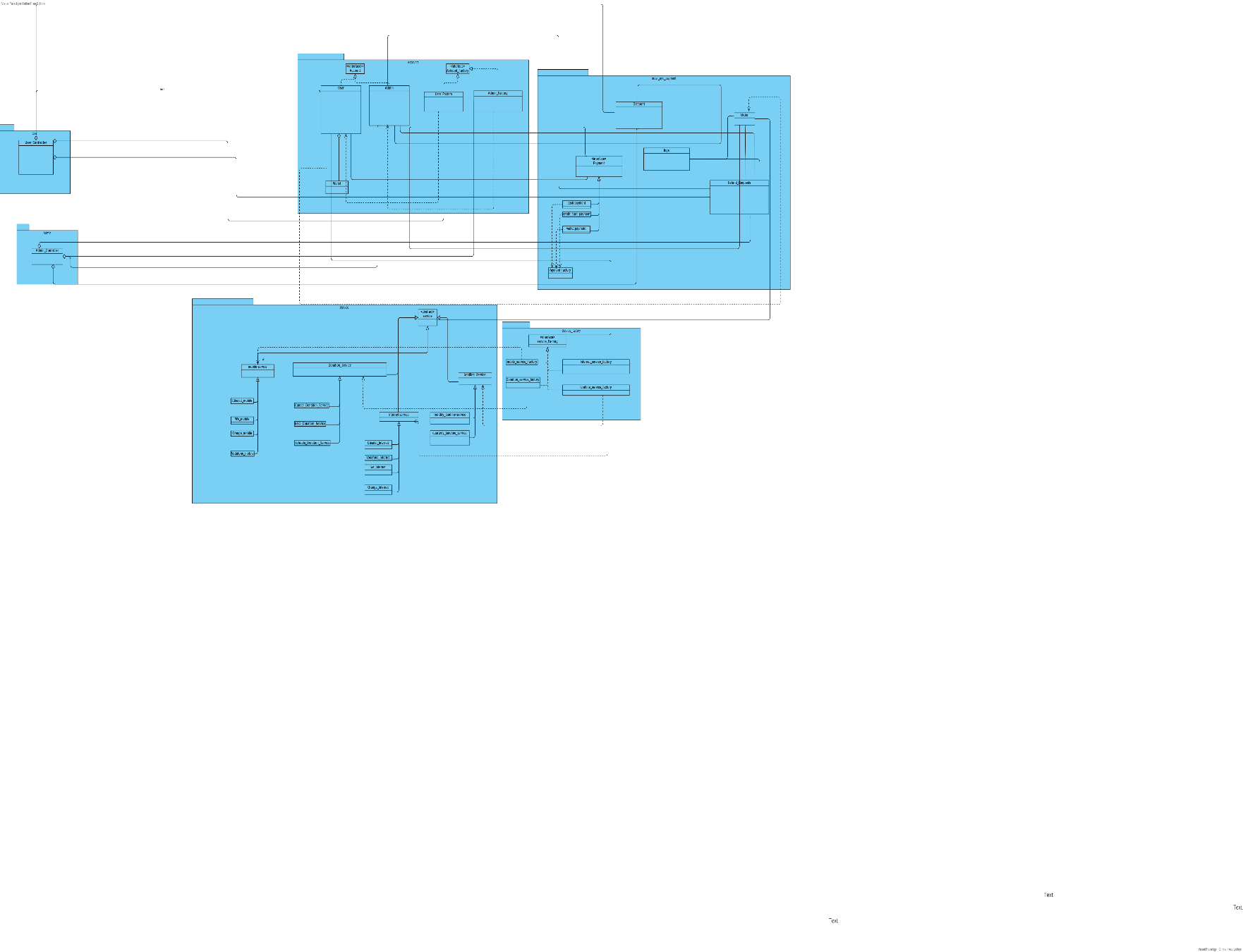
[Instructions[To be removed] 3](#_Toc120811426)

[Class diagram design **Error! Bookmark not defined.**](#_Toc120811427)

[Class diagram Explanation 4](#_Toc120811428)

[Sequence diagram design 5](#_Toc120811429)

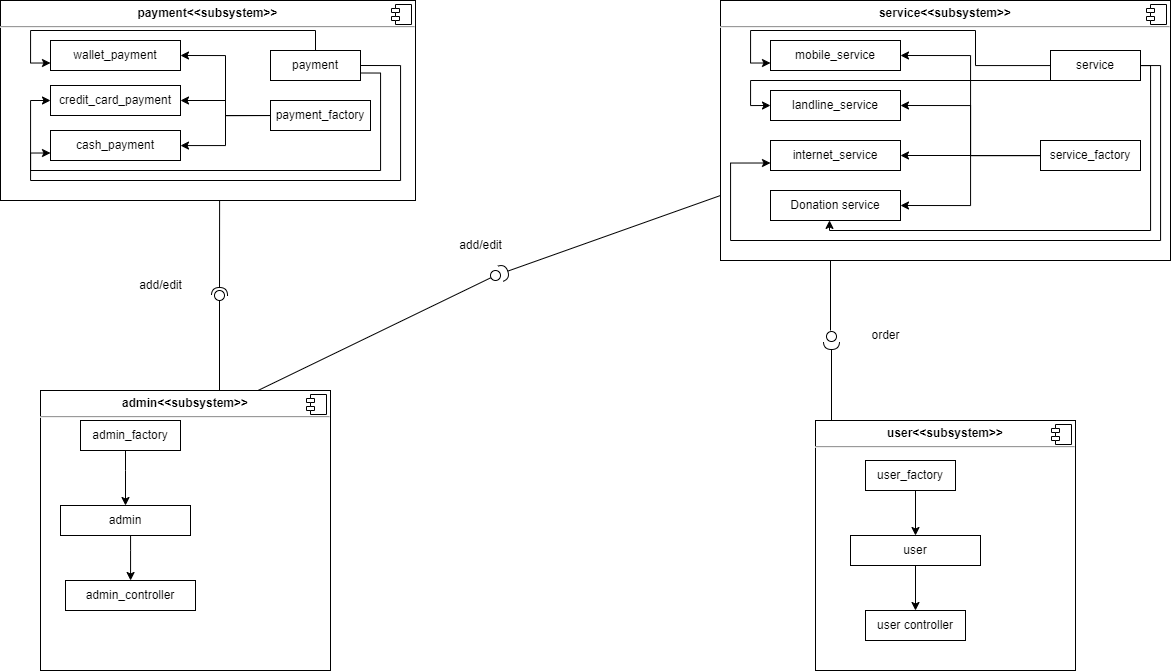
****Github repository link : https://github.com/Ahmed-Waleed-01/Advanced\_Software\_Phase\_2/tree/master

****

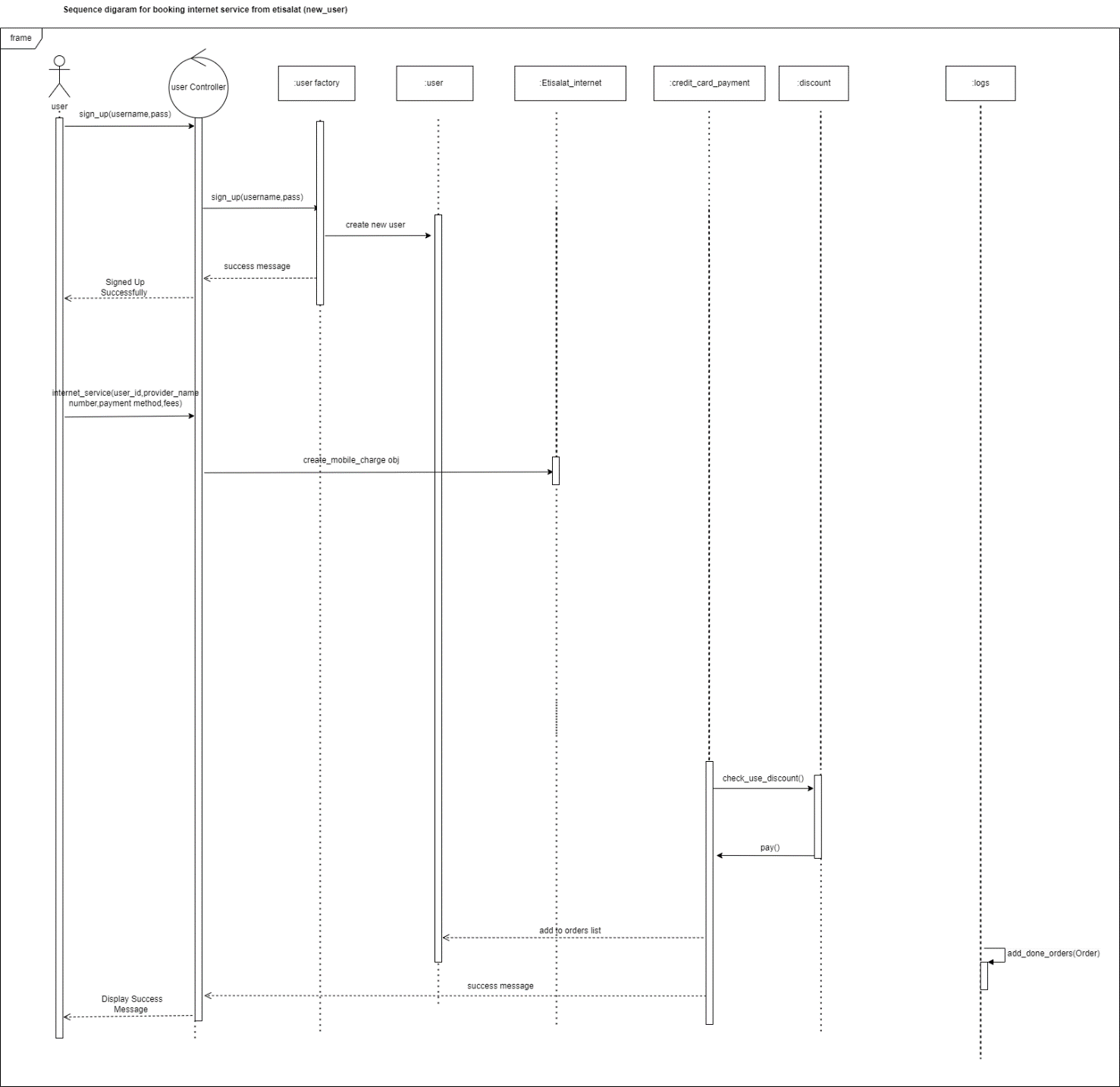
# Note: class diagrams here are not clear so you should open it as an image from the project file, I provide two versions of class diagram one before packaging and one after packaging

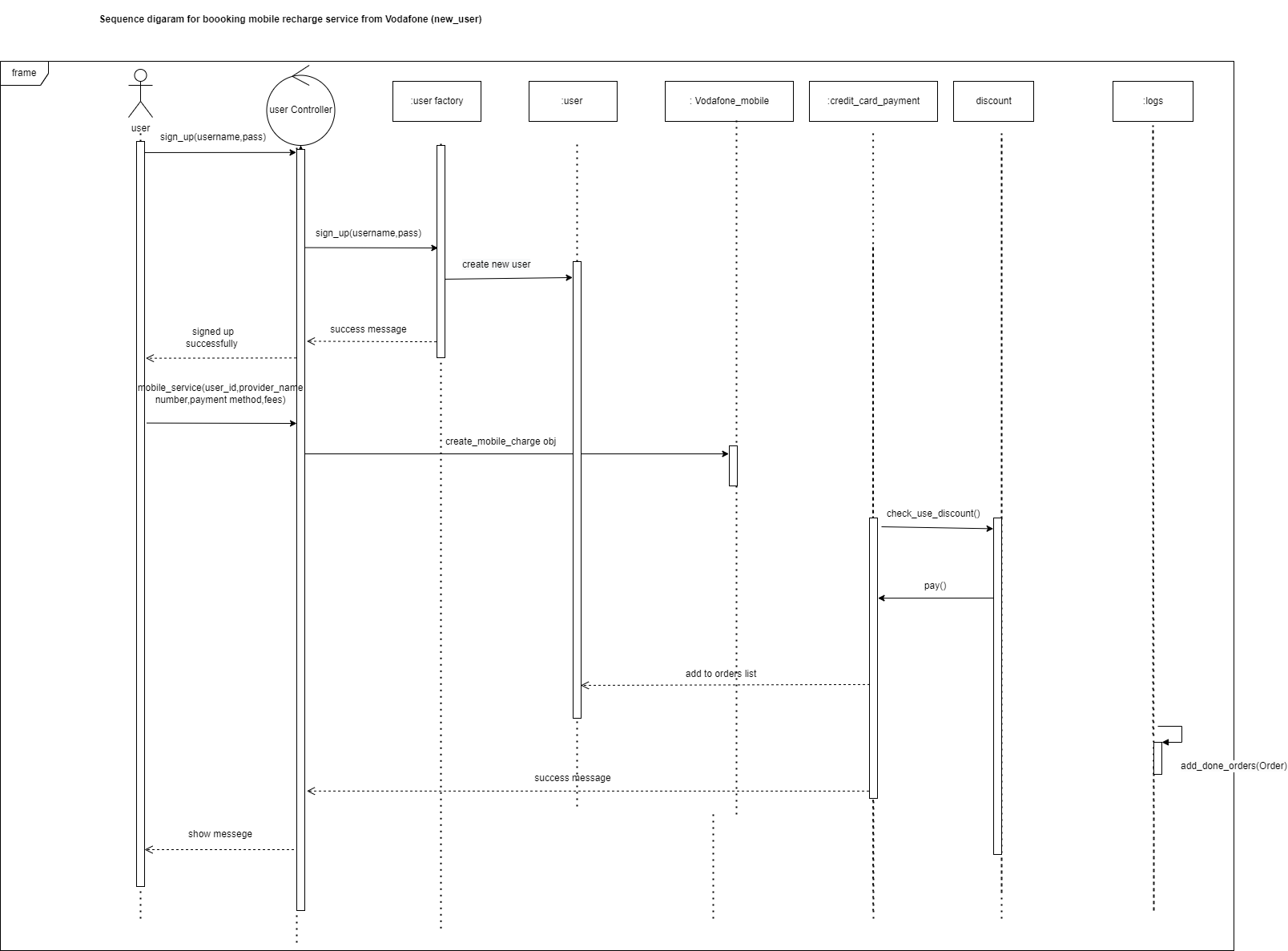
# Class diagram Explanation

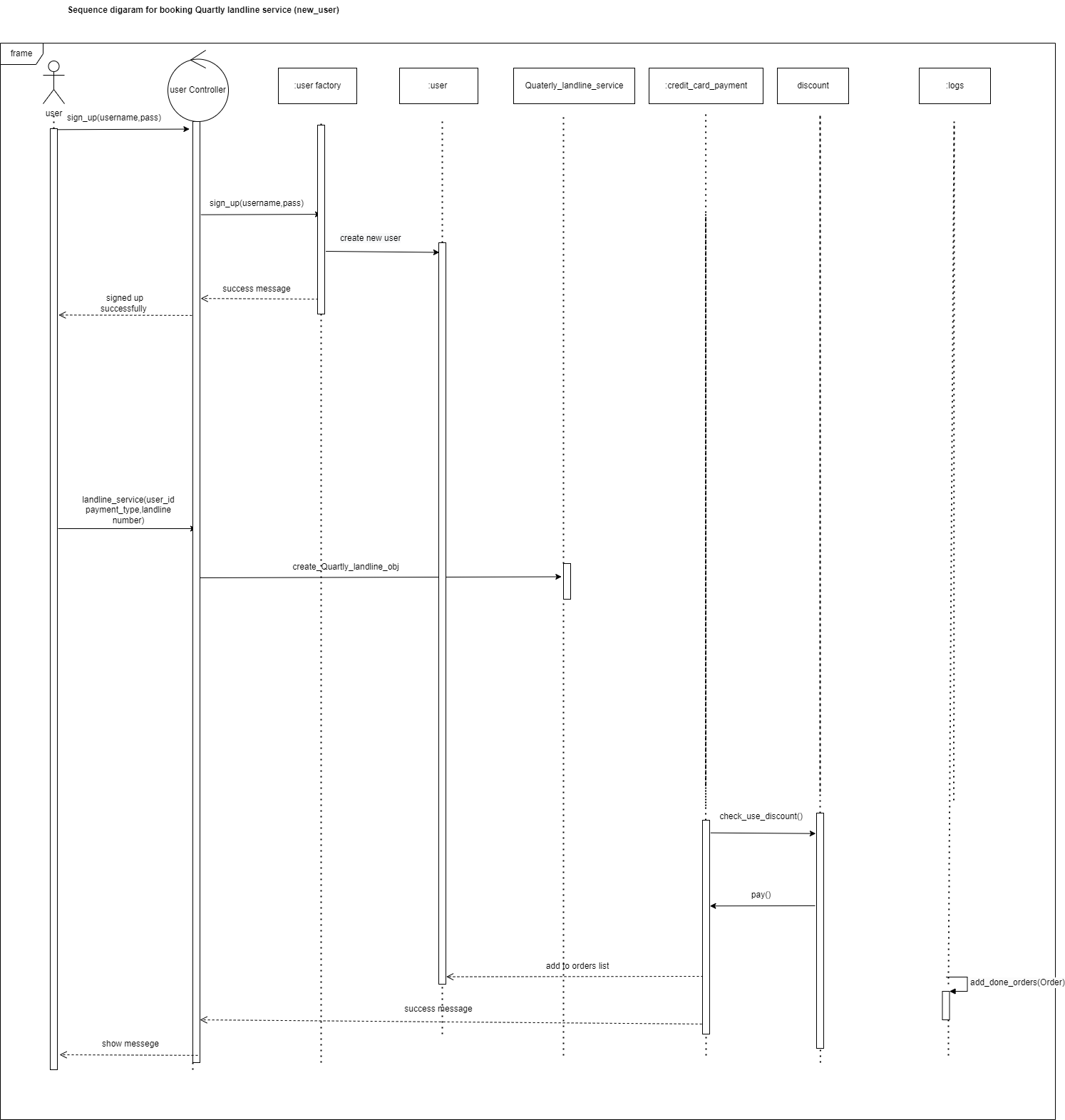
1. First things first, we used "strategy" design pattern as shown in "payment" class as it reflects different ways of paying for the service. This will allow to provide new payment methods without updating existing class, in another words, saving OCP
2. moving forward, we used "factory" design pattern in order to be able to create a new object every time we desire to do (e.g whenever a user comes to request a service, a new service is created with its handler to manage the request). It also provides the flexibility for the modification of the system without interrupting the previously working classes, which brings us back to preserving the OCP again
3. Finally, we used "single-tone" The Singleton's purpose is to control object creation, limiting the number to one but allowing the flexibility to create more objects if the situation changes. Since there is only one Singleton instance, any instance fields of a Singleton will occur only once per class, just like static fields. As shown in the provided class diagram, we can see that it's used in the following classes: logs, refund requests

**Decomposition diagram:**

# Sequence diagram design







# 

# 

# 

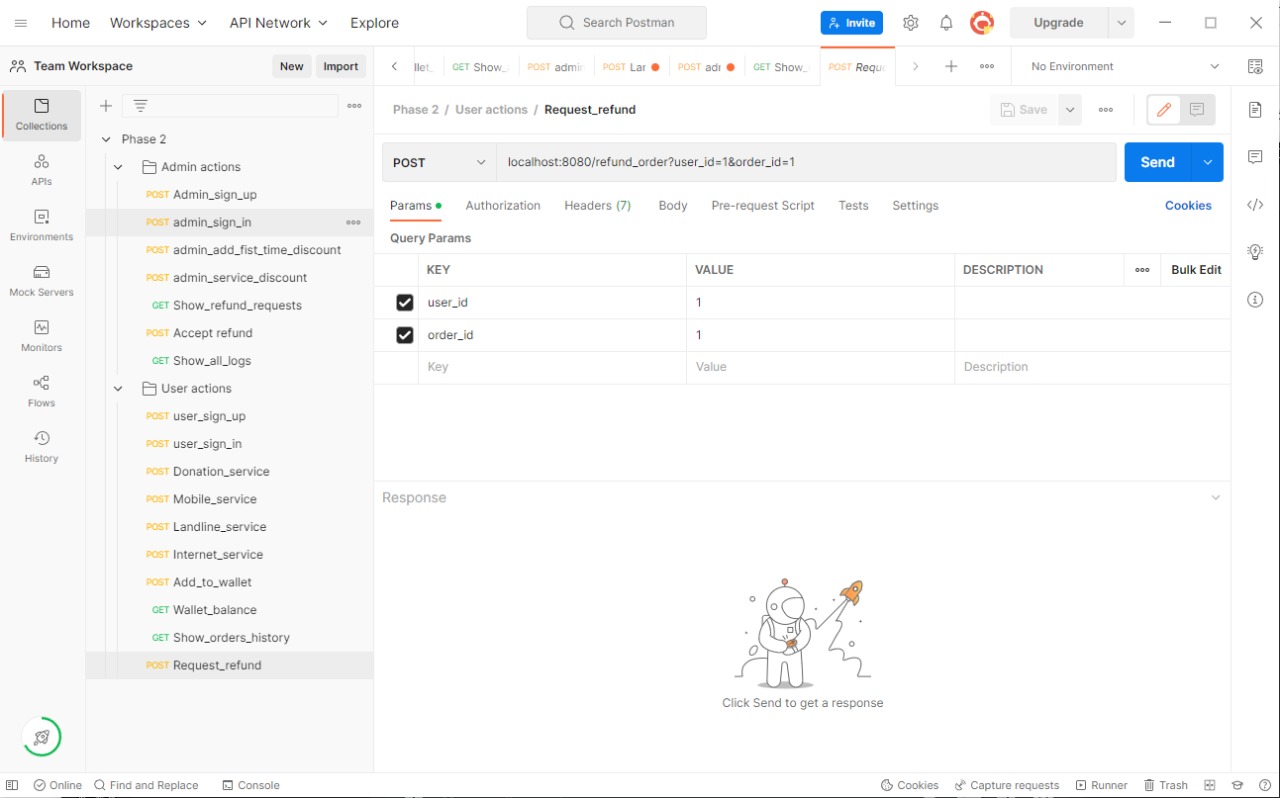
# 

# 

# 

# Requirements Exposure as Web Service API

**Part 1: Exposed Postman Collection**

****

**Link:**

<https://martian-star-992501.postman.co/workspace/Team-Workspace~f8e02b87-d796-4383-8440-ac65da6fee57/collection/25101763-713d161e-4e23-43ba-8239-b4b85ab26d95?action=share&creator=25101763>

**Part 2:**

**Explain here the exact mapping between every single requirement and its corresponding web service API operation. A sample example is provided to better explain the concept.**

|  |  |
| --- | --- |
| Requirement | Exposed API |
| The system should check if the username or the email is registered before. | Post/ localhost:8080/admin\_sign\_up?email=bro5@mail&password=24ufaz  The service checks if the user exists or not and the service returns a message that describes the sign up attempt if it was successful or unsuccessfully |
| 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 | Post/ localhost:8080/admin\_sign\_in?email=bro5@mail&password=24ufaz  The API takes e-mail and password as parameters in url query  The service return if the user exists or not in a string return type. |
| 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. | Post/ localhost:8080/donation\_service?user\_id=1&provider=schools&payment\_method=cash&fees=500  localhost:8080/mobile\_service?user\_id=1&provider=etisalat&number=01235678&payment\_method=wallet&fees=80  localhost:8080/landline\_service?user\_id=1&payment\_type=quarter&number=01235678&payment\_method=cash&fees=500  localhost:8080/internet\_service?user\_id=1&provider=vodafone&number=01235678&payment\_method=cash&fees=500  this service API takes user ID and service provider as well as payment method and payment amount and takes phone number if needed in different services.  it returns a string message that describes the operation and it’s status. |
| The user can ask for a refund for any complete transaction to any given service. | Post  localhost:8080/refund\_order?user\_id=1&order\_id=1  API takes user id and order id as query parameters and returns a message that states if the order refund request is complete. |
| The user should be able to add any funds to the wallet. Adding funds to the wallet should be done via credit card. | POST  localhost:8080/add\_to\_wallet?user\_id=1&fees=87  API takes user id and fees as query parameters  It returns a message that states the request status. |
| The user should be able to check  funds in the wallet | GET localhost:8080/wallet\_balance?user\_id=1  API takes user id as query parameters  It returns the user’s wallet balance |
| The admin should be able to add discounts to the system ex: first time orders. | POST  localhost:8080/add\_first\_time\_discount?admin\_id=1&discount=22  API takes admin id and discount percentage and  returns message that show that discount now is applied for first time discounts. |
| The admin should be able to add discounts to the system  ex:Specific discount to service | POST localhost:8080/add\_service\_discount?admin\_id=1&service=Mobile&discount=20   API takes admin id and service name that the discount will be applied to and discount percentage   it returns a string message with the status if it succeeds or not. |
| The admin should be able to list all user transactions | GET localhost:8080/show\_all\_logs?admin\_id=1  API takes admin id and then it returns a string message that lists all different transactions. |
| The admin should be able to list all refund requests. | GET  localhost:8080/show\_refund\_requests?admin\_id=1  API takes admin id and then it returns a string message that lists all refund requests. |
| he admin should be able to accept or reject any refund request | POST  localhost:8080/accept\_order\_refund?admin\_id=1&order\_placement=1  API takes admin id and order placement in refund requests. then it returns a string message that contains order fees as it was added to user’s wallet. |

# Github repository link

* **https://github.com/Ahmed-Waleed-01/Advanced\_Software\_Phase\_2**