Cairo University Faculty of Computers and Artificial Intelligence



**Software design specification document**

**2022**

**Project Team**

|  |  |  |
| --- | --- | --- |
| **ID** | **Name** | **Email** |
| 20201115 | Abdallah mohammed abdul-rhem | body5401@gmail.com |
| 20200203 | Zeyad farag mohamed | asdanazeyadandali545@gmail.com |
| 20200050 | Ahmed Mohammed Hany | ahmedhany20200050@gmail.com |
| 20200298 | Abdel-salam saeed abdel-salam | abdelalamsaeed01@gmail.com |
|  |  |  |

Contents

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

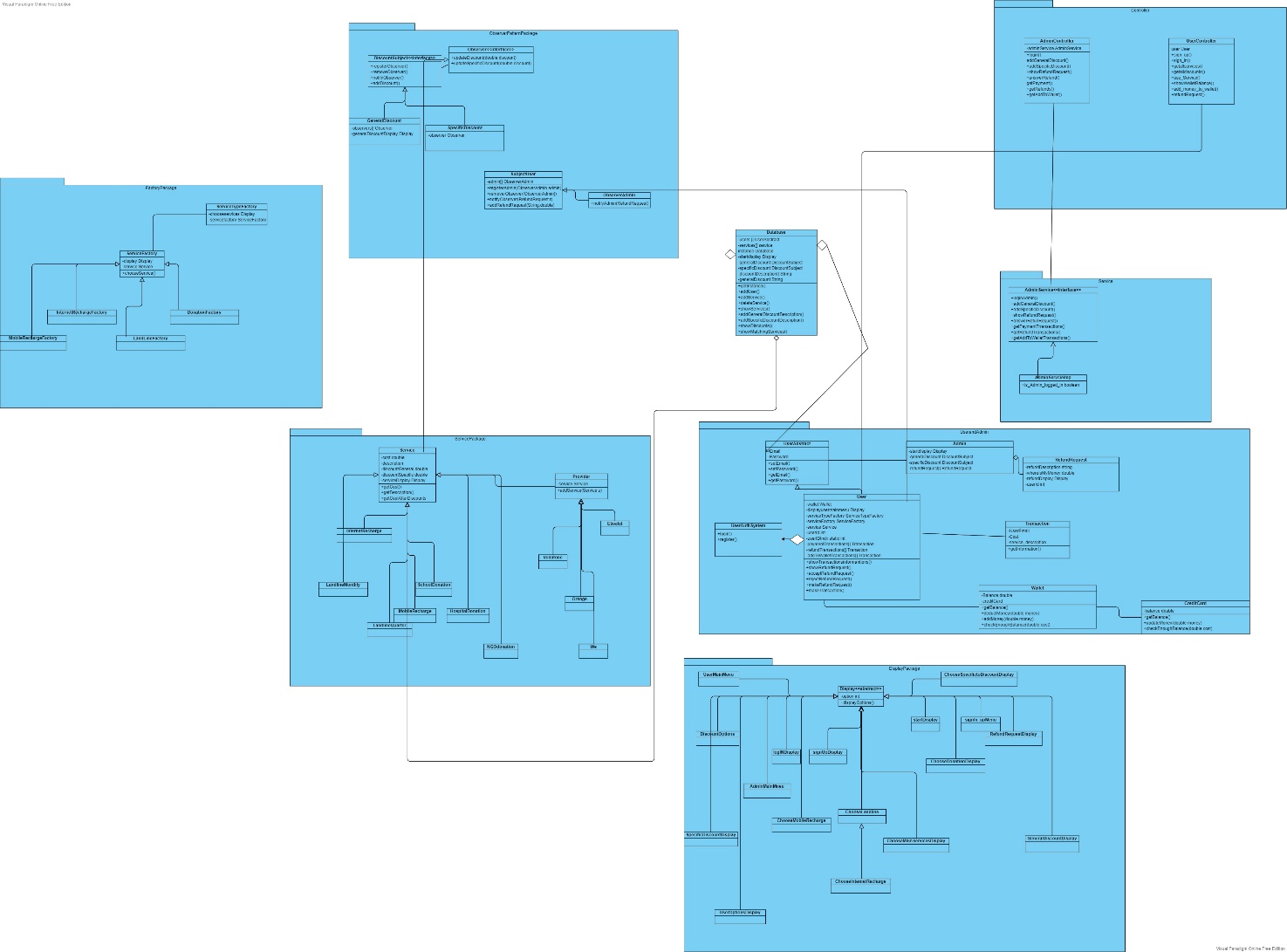
[Class diagram design 2](#_Toc120811427)

[Class diagram Explanation 3](#_Toc120811428)

[Sequence diagram design 3](#_Toc120811429)

[Github repository link 4](#_Toc120811430)

# Class diagram design



# Class diagram Explanation

**Singleton pattern:**

1. **We needed a single database for system**

**Participating classes: database**

1. **We needed a single object for each service**

**EtislatMobRechareg,VodaMobRecharge,WeMobRecharge,InternetOrange,InternetWe,OrangeMobRecharge,InternetEtisalat,LandlineQuarte,LandlineMonthly…etc**

* **Strategy Pattern**

**Used in displays classes in the system, as every display has the same function but can diverse depending on where it will be used (admin menu, user main menu, choose services display, etc).**

* **Observer Design pattern**

**Used to help us add discounts to the services, as we register the services to the discount subject (for example GeneralDiscountSubject class) then when we add a discount the subject will notify the services and update the value of discount in them.**

* **Factory Design pattern**

**We used it to help us selecting the service the user wants to use depending on the options he chooses, as we have at first a service type factory which selects one of the following factories (Mobile Recharge factory, Donation Factory, etc), and the factory we select it will return specific service the user want to use (for example, in mobile recharge factory, the user will choose Etisalat recharge, we recharge, Vodafone recharge and Orange recharge).**

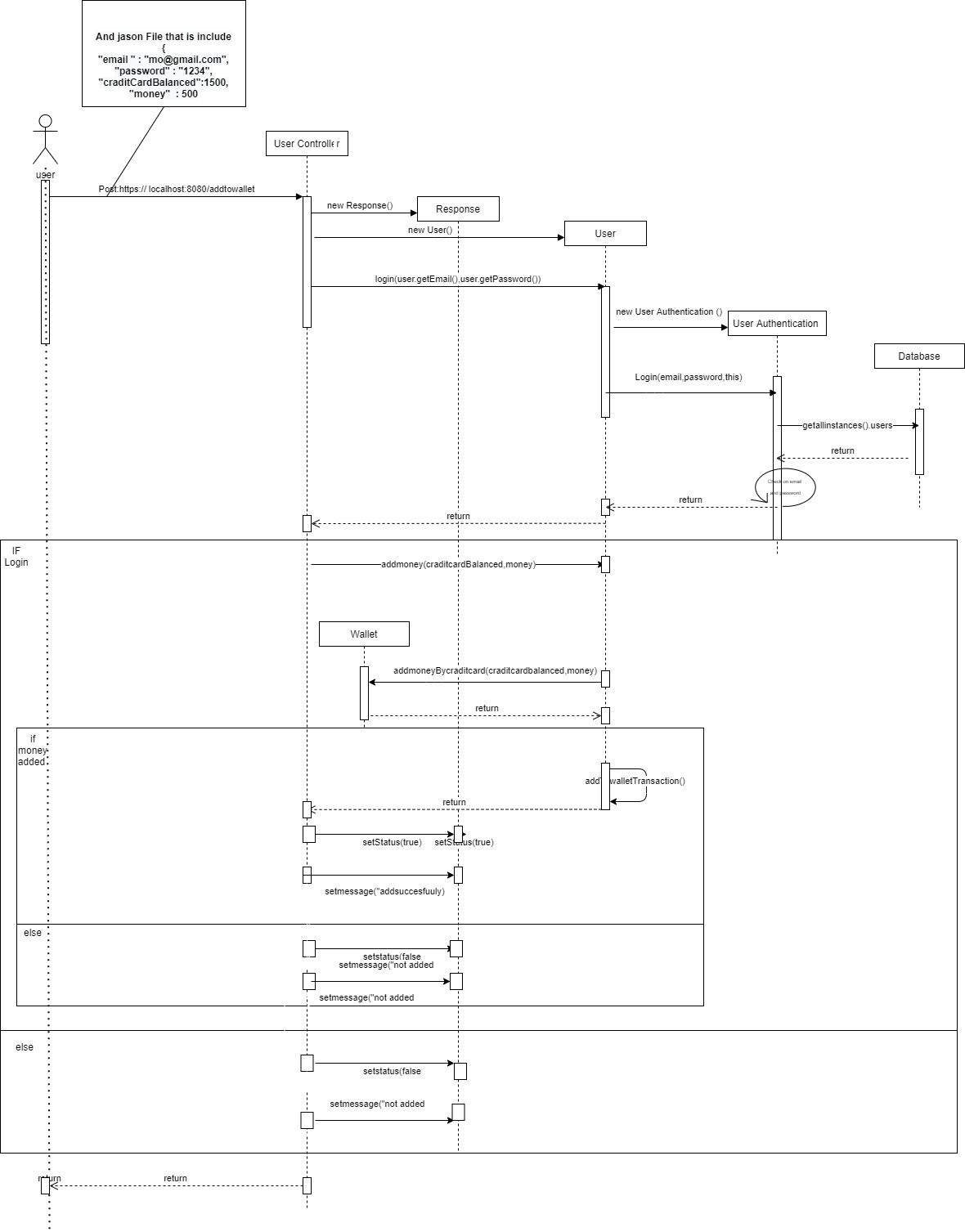
* **Decorator Design pattern**

**We used it to create the services . we use a provider (which is considered a service) and we put an actual service inside it.**

**Example: Vodafone class can be used with Mobile Recharge class with decorator pattern**

# Sequence diagram design

1-Add to Wallet

****

# 

2- user payment sequence diagram

Diagram

Description automatically generated

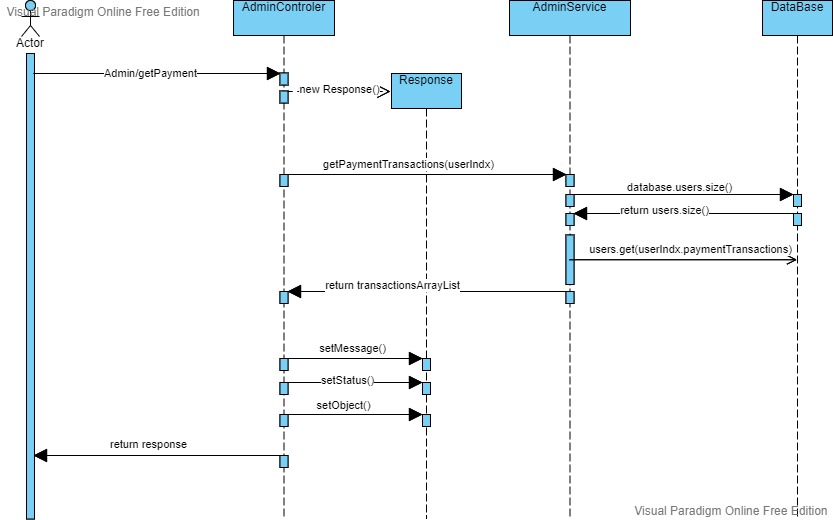
3- user refund request diagramDiagram

Description automatically generated

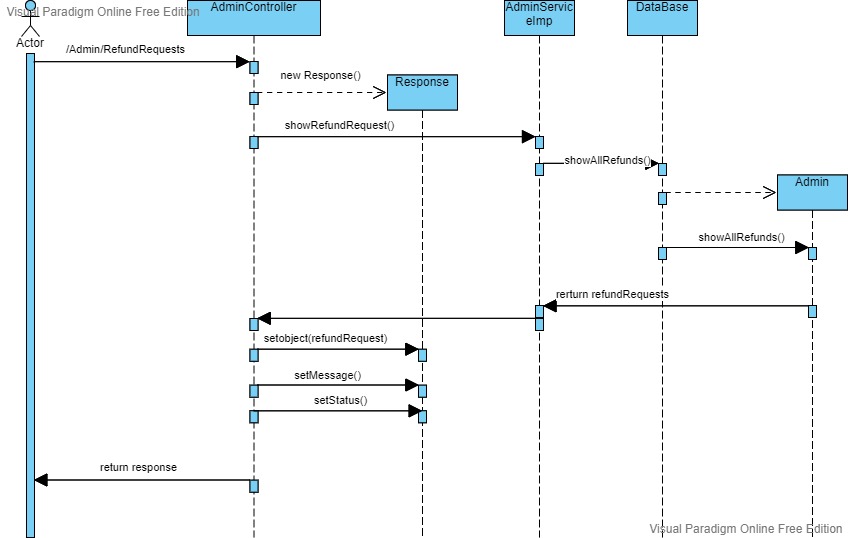
# 4-user search services diagram

# Diagram Description automatically generated

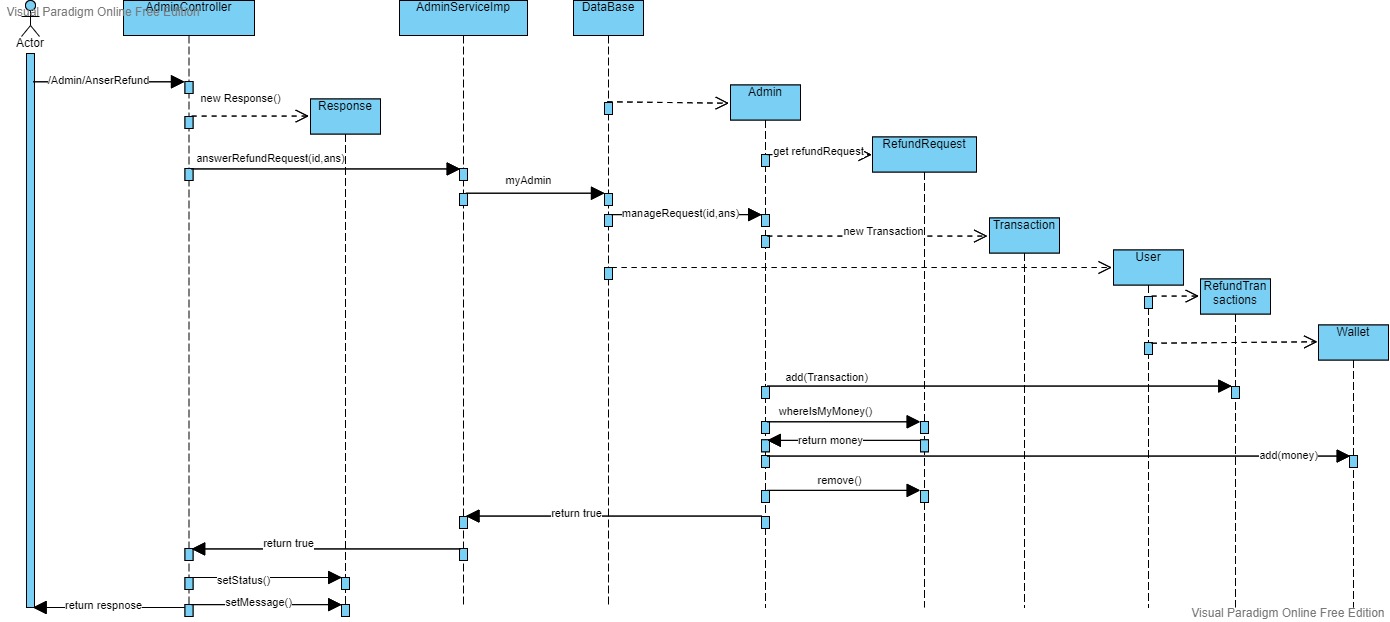
5- Admin want to get the payment transactions log of a certain user



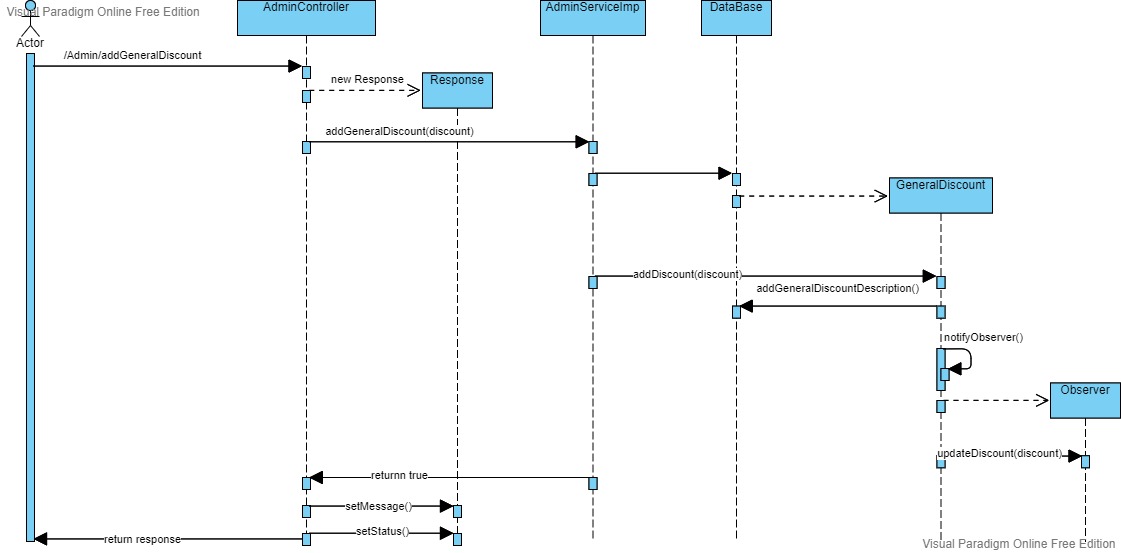
6- Admin lists all the refund requests in the system



7- Admin accepts a refund request for a user



8-The admin adds a general discount to the system



# Requirements Exposure as Web Service API

**Part 1:**

**User APIs Link:**

[**https://api.postman.com/collections/24960898-22f71ad3-e19f-4f07-bfed-746f1ada95f8?access\_key=PMAT-01GNPZZRVSF4A7WDBFSG8KMPHS**](https://api.postman.com/collections/24960898-22f71ad3-e19f-4f07-bfed-746f1ada95f8?access_key=PMAT-01GNPZZRVSF4A7WDBFSG8KMPHS)

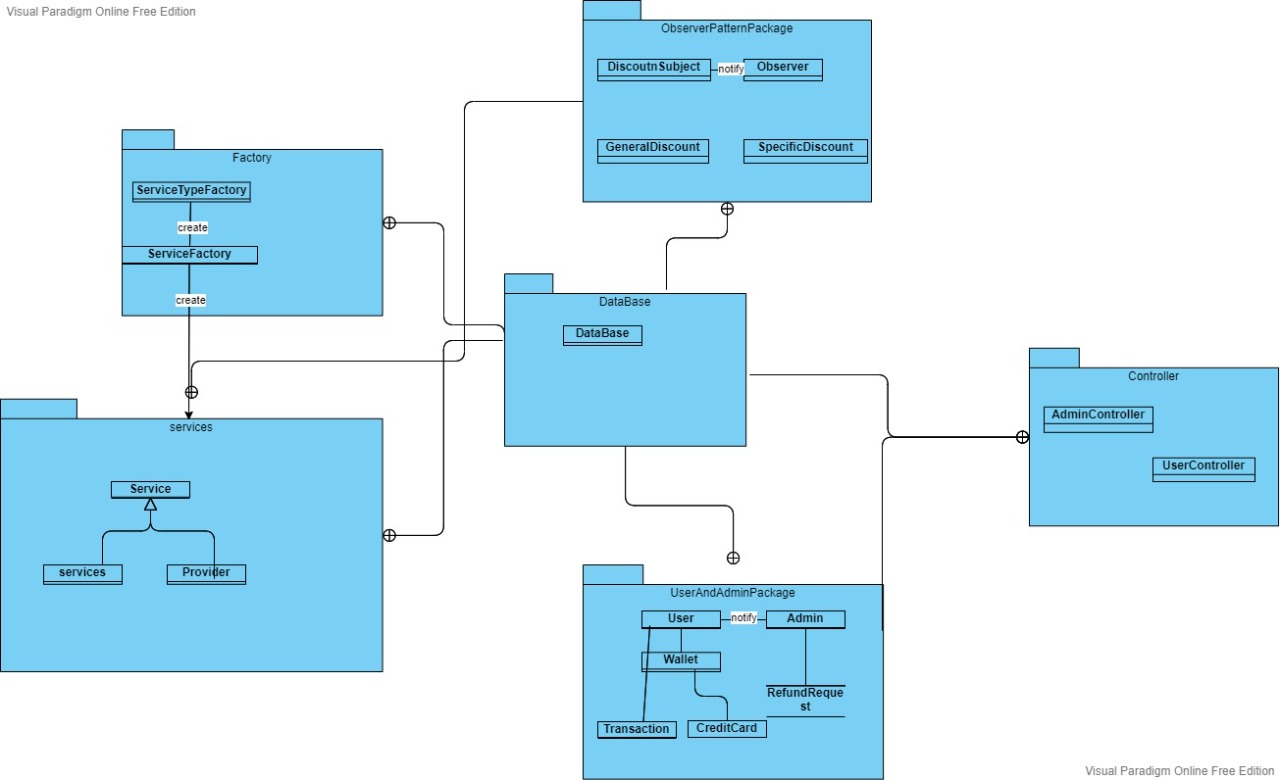
**Admin APIs Link:**

[**https://api.postman.com/collections/24960898-5c3fe8cd-2e41-4165-937a-d5fbb8d5bd4d?access\_key=PMAT-01GNPXSP3SPBP96X2W38QQBT43**](https://api.postman.com/collections/24960898-5c3fe8cd-2e41-4165-937a-d5fbb8d5bd4d?access_key=PMAT-01GNPXSP3SPBP96X2W38QQBT43)

**Part 2:**

|  |  |  |
| --- | --- | --- |
| Requirement |  | Exposed API |
| No specific requirement(but we need it to be used before any Admin API) |  | 1-POST http://localhost:8080/Admin/login  a service to check if admin email and password is correct or not.  NOTE:  If yes you can access all admin functions  Input:  email  password |
| The admin should be able to add discounts to the system. There are two types of discounts. a. Overall discounts. |  | 2- POST http://localhost:8080/Admin/addGeneralDiscount  a service for the admin to add a general service  input:  discount (integer) |
| The admin should be able to add discounts to the system. There are two types of discounts.b. Specific discount. |  | 3- POST http://localhost:8080/Admin/addSpecificDiscount  a service for the admin to add a specific discount  input: discount (integer) + index of service (get it from "get all services" API in user APIs) |
| **The admin should be able to list all refund requests** |  | 4- GET http://localhost:8080/Admin/RefundRequests  getting all refund requests data (you will need the attribute "id" to answer the request in the next API (API number five))  **no** input. |
| 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. |  | 5-POST http://localhost:8080/Admin/AnswerRefund  answering a refund request requires refund id (check previous API number 4)  answering with 0 means no  answering with 1 means yes  input: id (of request) + answer |
| The admin should be able to list all user transactions. The transactions types area. Payment transaction |  | 6-POST http://localhost:8080/Admin/getPayment  **NOTE:**  You need the index of the user  (it is the attribute "This id" in User class)  Just call API number 4 (get all refund requests API and you will see user object which contains user id "This id" attached to the request)  Input: indx (index of user) |
| The admin should be able to list all user transactions. The transactions types areb. Add to wallet transaction. |  | 7-POST http://localhost:8080/Admin/getAddToWallet  **NOTE:**  You need the index of the user  (it is the attribute "This id" in User class)  Just call API number 4 (get all refund requests API and you will see user object which contains user id "This id" attached to the request)  Input: indx (index of user) |
| The admin should be able to list all user transactions. The transactions types arec. Refund transaction. |  | 8-POST http://localhost:8080/Admin/getAddToWallet  **NOTE:**  You need the index of the user  (it is the attribute "This id" in User class)  **just call API number 4 (get all refund requests API) and you will see user object which contains user id called "This id" attached to the request)**  Input: indx (index of user) |
| The user should be able to sign up to the system |  | 9-POST http://localhost:8080/User/sign\_up input: username + email + password |
| The user should be able to sign-in to the system |  | 10-POST http://localhost:8080/User/sign\_in  input: email + password |
| No specific requirement.(but we need it to be used before another API "payment API") |  | 11-GET http://localhost:8080/User/getallservices  **NOTE:**  This API is needed before the payment  because we need **the id of the service**  which the user will pay money to.  It is also used to know **the price before paying**.  No input |
| **The user should be able to check any discount for any service in the system.** |  | 12-GET http://localhost:8080/User/getalldiscounts  no input |
| The user can pay for any service in the system. |  | 13-POST http://localhost:8080/User/Useservice  payment is done here  **NOTE:**  We need service id which is given in API number 11 (get all services API).  Input: email + password + id (id of service) |
| The user should be able to search for any service in the system |  | 14-POST http://localhost:8080/User/search  search for User Services.  Input: word |
| The system maintain a wallet balance for each user |  | 15-POST http://localhost:8080/User/showWalletBalance  give it your email and password and it will return your wallet balance  input: email + password |
| **The user should be able to add any funds to the wallet. Adding funds to the wallet should be done via credit card.** |  | 16-POST http://localhost:8080/User/addtowallet  NOTE:  We assumed that the user can update his credit card balance while adding to wallet.  (this piece of information should be retrieved from a real bank but we cannot do that).  Input: email + password + money (money to be added to wallet)+ craditCardBalance  (current credit card balance which is retrieved from bank) |
| The user can ask for a refund |  | 17-POST http://localhost:8080/User/refundRequest  the user can describe the problem and type what amount of money he wants  (responsibility of admin).  Input: email + password +money+ description (user's description of the problem). |
|  |  |  |

# subsystem diagram



# Github repository link

* [abdulla54abdulrhem/SW\_temp (github.com)](https://github.com/abdulla54abdulrhem/SW_temp)