Cairo University Faculty of Computers and Artificial Intelligence



**Software design specification document**

**2022**

**Project Team**

|  |  |  |
| --- | --- | --- |
| **ID** | **Name** | **Email** |
| 20200161 | Hussien Mostafa | Zeeka32@gmail.com |
| 20201118 | Ali Safwat | as20201118@gmail.com |
| 20200800 | Adel Tarek | Adeltarek202@gmail.com |
| 20201127 | Amr Riyad | spyvip123@gmail.com |

Contents

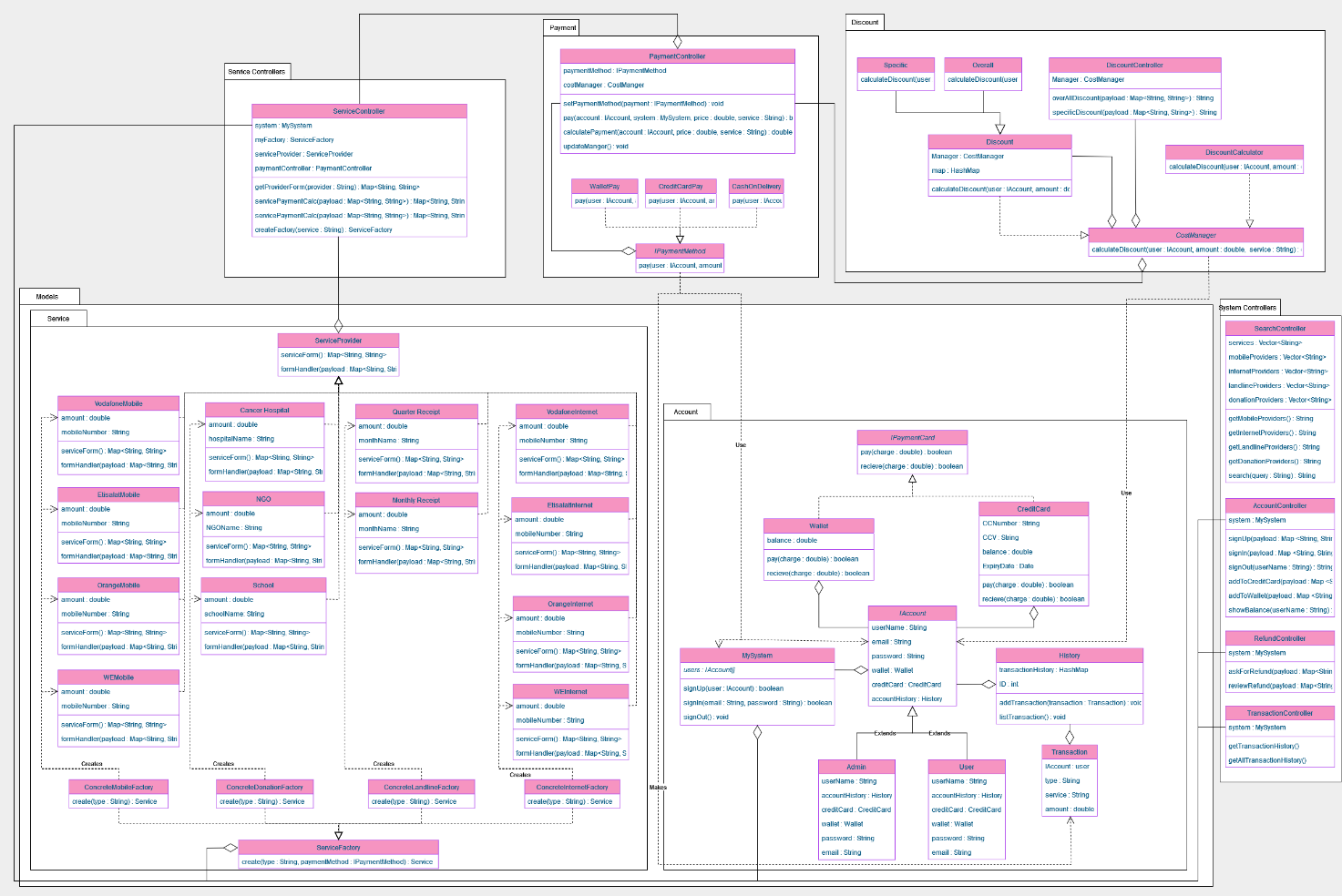
[Class diagram design 2](#_Toc120811427)

[Class diagram Explanation 3](#_Toc120811428)

[Sequence diagram design 4](#_Toc120811429)

[Github repository link 14](#_Toc120811430)

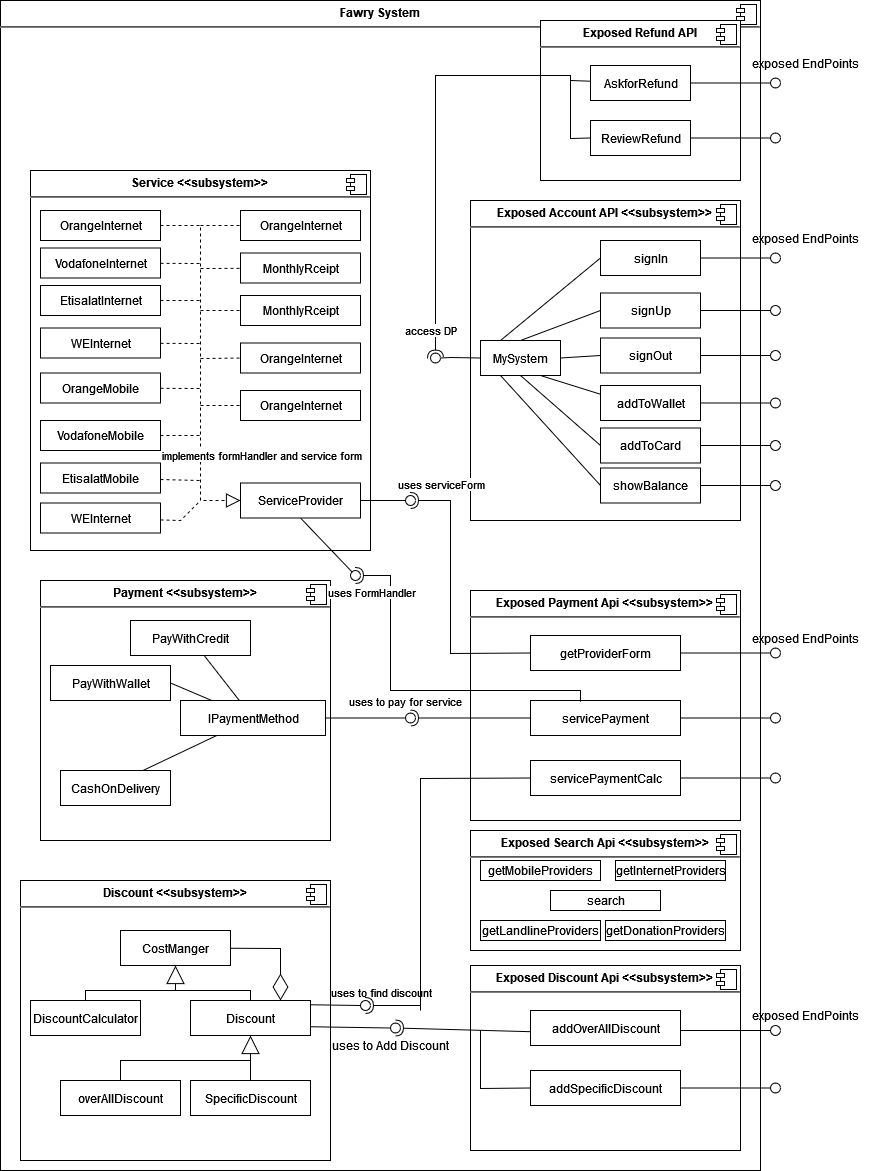
# Class diagram design



# Class diagram Explanation

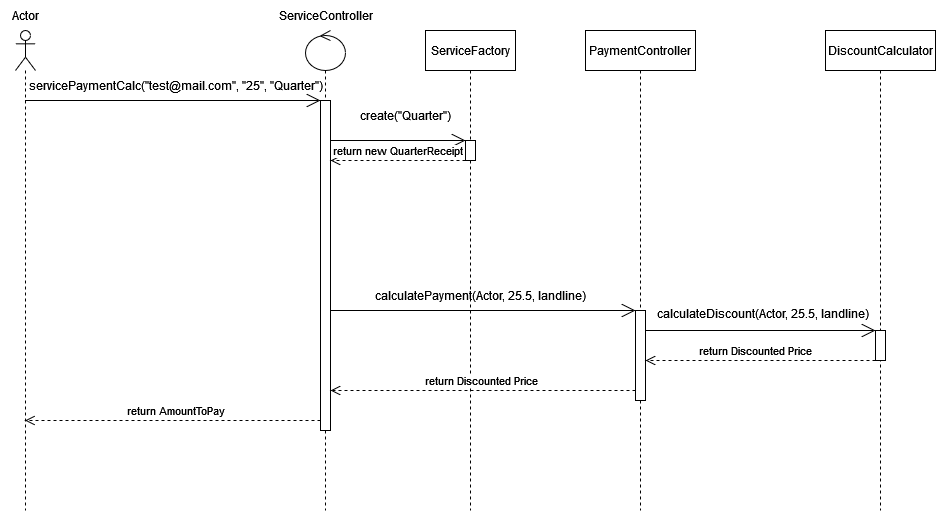
* **setters and getters were not included in the class diagram, however, all needed setters and getters were used.**
* **Strategy Pattern: this pattern was used because it lets you define a family of algorithms, put each of them into a separate class, and make their objects interchangeable. The family of algorithms in question are, payment with wallet, payment with credit card, and payment on delivery. Classes that use this pattern are: IPaymentMethod, PayWithWallet, PayWithCreditCards, and CashOnDelivery.**
* **Factory Pattern: we create objects without exposing the creation logic to the client and refer to newly created objects using a common interface, we use this pattern to create new service providers without having the classes using these services be affected by the creation process. Classes that use this pattern are: ServiceFactory, ConcreteMobileFactory, ConcreteDonationFactory, ConcreteInternetFactory, ConcreteLandlineFactory, and these factories create all the service providers.**
* **Decorator Pattern: we used this pattern to apply different discounts to different users and services, classes that use this pattern are: CostManager, CalculatePrice, Discount, Specific, and Overall.**
* **Singleton Patterns: this pattern was used to ensure 1 object of our database which is the System is present to all classes and to help with the decorator pattern. Classes that use this pattern are: CostManager, MySystem.**

# Subsystem diagram design

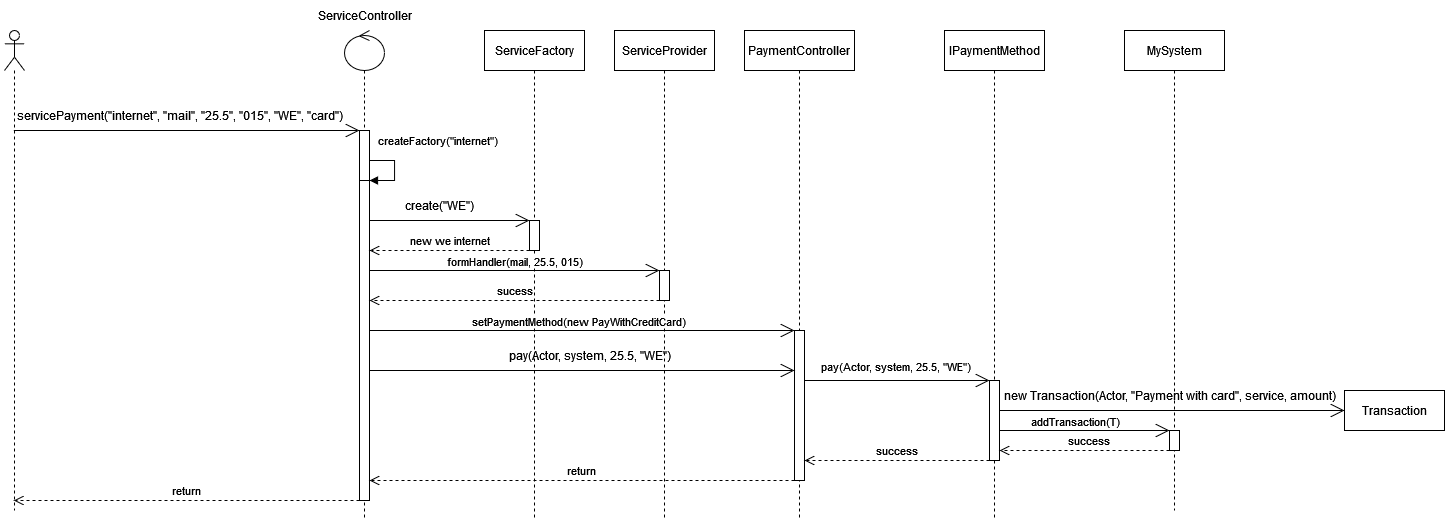


# Sequence diagram design

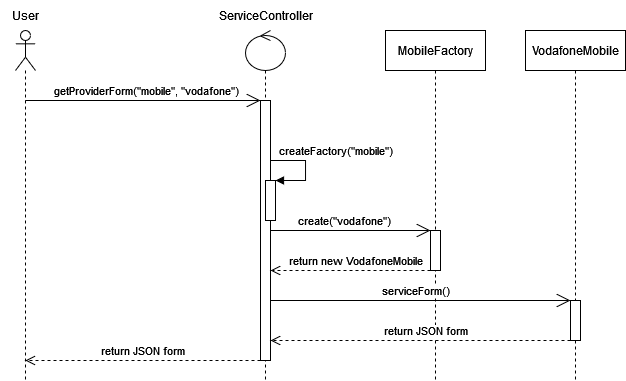
1- calculate discount



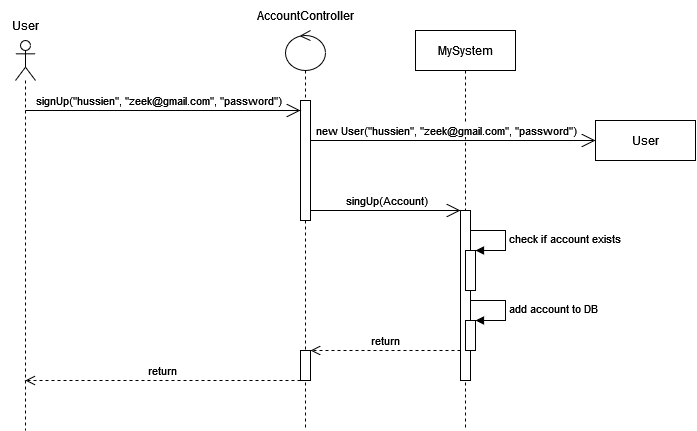
2- pay for service

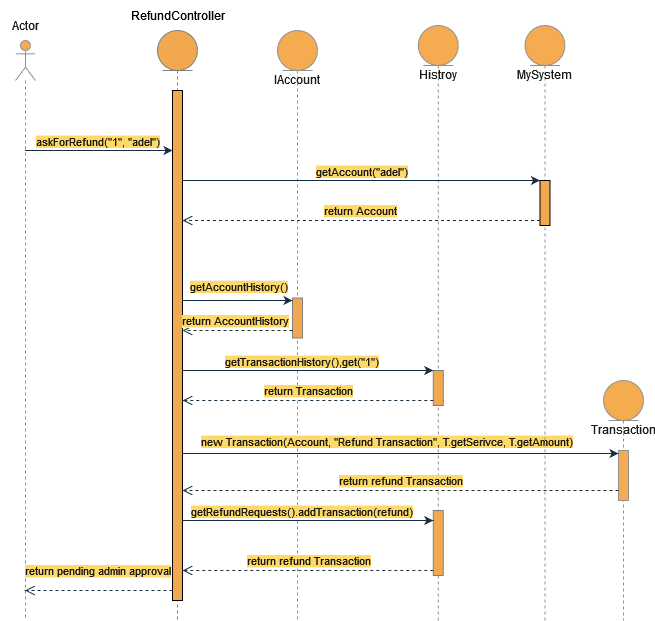


3- Get Service Form

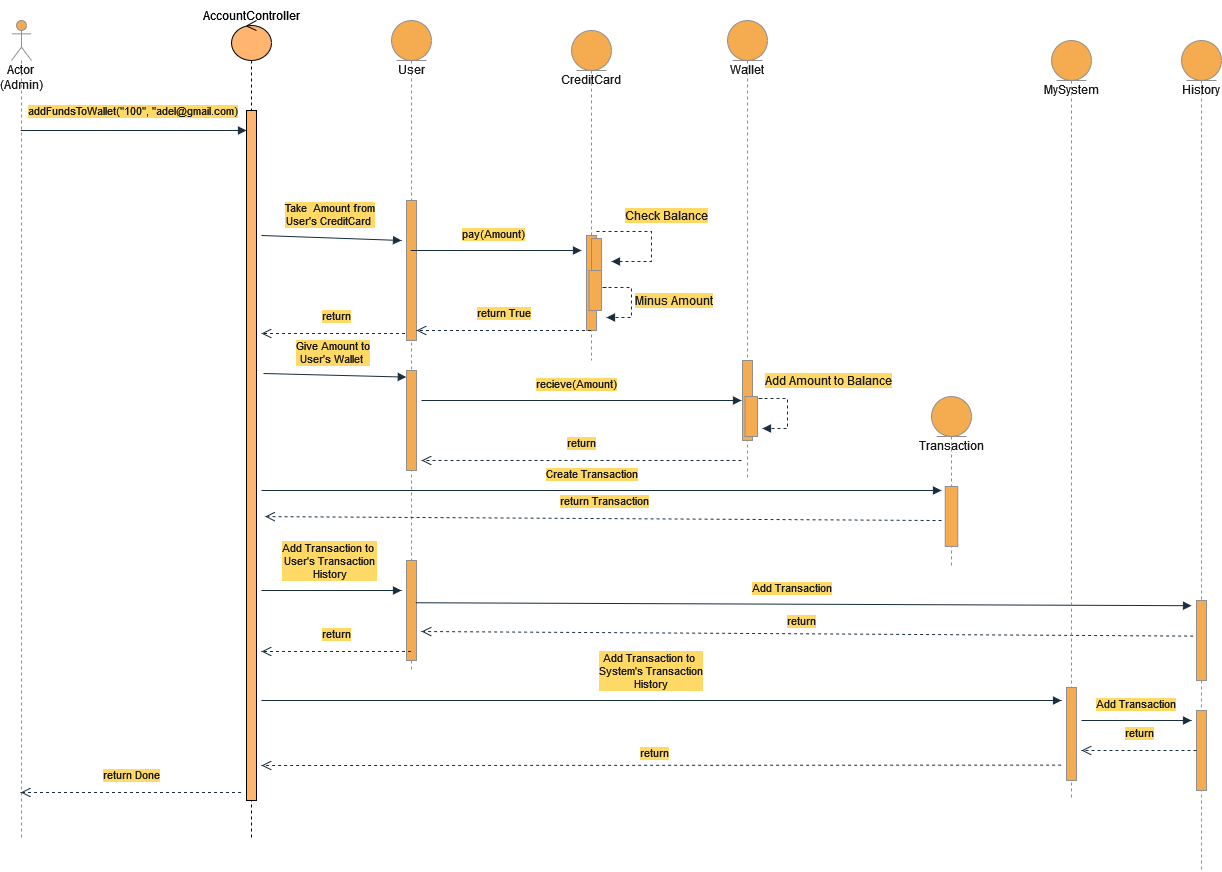


4- sign up a new account to the system

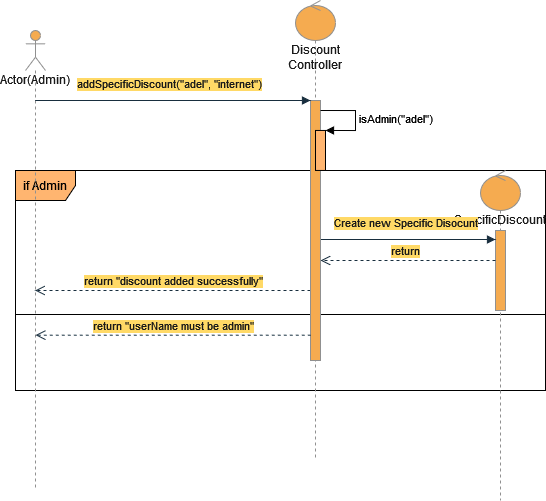


5- ask for refund

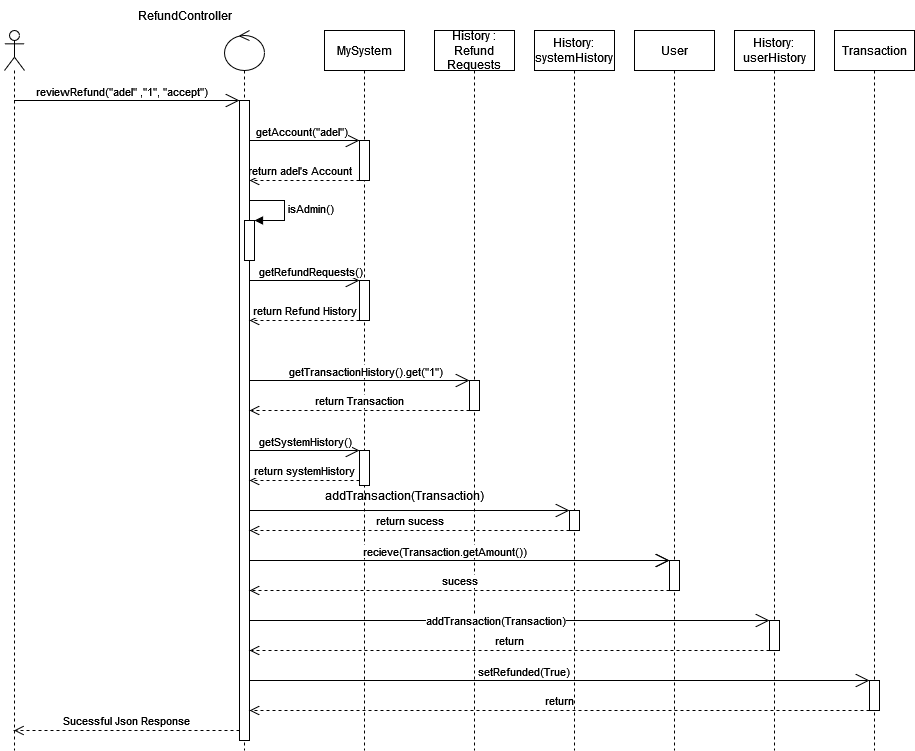
6- add funds to wallet sequence diagram



7- add specific discount



8- approve refund diagram



# Requirements Exposure as Web Service API

**Part 1: Exposed Postman Collection (to copy this file enable editing first then drag it onto desktop then import it using Postman, or download it from the github)**

****

**Part 2:**

|  |  |  |
| --- | --- | --- |
| Requirement | Example | Exposed API |
| 1- 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 http://localhost:8080/signUp Body  {      "username": "hussien",      "email": "zeek@gmail.com",      "password": "test",      "type": "admin"  }  Returns  account added successfully  or  account already exists | POST /signUP  A service which takes a json in its body containing the user’s username, email and password, for this project’s demonstration purposes it also takes “type” of user since there exists no database. Returns a string with the status of whether the account completion was successfully or not. |
| 2- The user should be able to sign-in to the system. Given the user’s email and password, the user can login to the system and use any of the system functionalities. | POST http://localhost:8080/signIn Body  {      "email": "zeek@gmail.com",      "password": "test"  }  Returns  successfully signed in  or  this account is already logged in to the system  or  incorrect credentials | POST /signIn Takes a json containing the email and password of the account, returns successfully message if the user exists and is not already logged in to the system |

|  |  |  |
| --- | --- | --- |
| 3- 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 http://localhost:8080/search?query=mobile**  Returns  1- Mobile Recharge Services  **GET http://localhost:8080/search?query=ser**  Returns  1-Mobile Recharge Services  2-Internet Services  3-LandLine Services  4-Donation Services | GET /search?query=This service takes a query in its parameter and returns all matching services that contains said query Note: the account needs to be logged in to use this feature |
| 4- The user can pay for any service in the system, the system should allow the user to consume from wallet, or cash on delivery if the provider supports it. | POST http://localhost:8080/internet/pay Body  {      "email": "zeek@gmail.com",      "amount": "25.5",      "number": "01551384493",      "provider":"WE",      "paymentType": "card"  }  Returns JSON Containing metadata regarding the request  {      "amount": "25.5",      "user": "hussien",      "status": "transaction complete"  }    Or  {      "amount": "25.5",      "error": "not enough funds",      "user": "hussien",      "status": "transaction failed"  } | POST /{service}/pay This service takes a json in its body containing the amount to pay and the provider to pay to and returns successful payment if the payment was complete, service is replaced with any service in the system such as “mobile”, “internet”, “landline”, or “donation”  The entered email is that of the user that wishes to pay for this service  Note: the account needs to be logged in to use this feature |

|  |  |  |
| --- | --- | --- |
| 4 - The system should prompt the user to the payment form | GET /mobile/form?provider=Vodafone Returns  {      "number": "",      "amount": "",      "paymentType": ""  }  **GET http://localhost:8080/donation/form?provider=CancerHospital**  Returns  {      "amount": "",      "hospitalName": "",      "paymentType": ""  } | GET /{service}/form?provider= Service is replaced with any service’s name and provider’s name is sent in the url as well, this service returns the form fields as JSON  Note: the account needs to be logged in to use this feature |
| 5- 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. | POST http://localhost:8080/refund/request Body  {      "ID":"1",      "username":"hussien"  }  Returns Refund pending if it was accepted or invalid transaction if it not feasible. | POST /refund/request This service takes in its body the ID of the transaction to be refunded and the username of the account that wants to refund this transaction. If the transaction is valid it is sent to the admin.  Valid transactions are ones that are not “cash on delivery” or ones that have not been requests for refund before nor ones that don’t exist meaning their ID is not in the system.  Note: the username in the body is that of the user that needs to refund the request  Note: the account needs to be logged in to use this feature |

|  |  |  |
| --- | --- | --- |
| 6- The system maintains a wallet balance for each user. The user should be able to add any funds to the wallet. Adding funds to the wallet should be done via credit card. | POST http://localhost:8080/addToWallet Body  {      "email": "zeek@gmail.com",      "amount": "100.0"  }  Returns successfully message if the amount can be transferred from the account’s credit card. | POST /addToWallet Takes JSON in its body containing the amount to transfer from the account’s creditCard and the account’s email  Note: the account needs to be logged in to use this feature |
| 7- The user should be able to check any discount for any service in the system. | POST http://localhost:8080/internet/calculatePay Body  {      "email": "zeek@gmail.com",      "amount": "25.5",      "provider":"WE"  }  Returns if the internet service has a specific discount  {      "amountToPay": "20.66666"  } | POST /{service}/calculatePay Takes JSON in its body containing the amount to pay, the provider and the email of the user. Returns the amount at a discounted price if a discount exists in the system.  Returns discounted price |
| Admin  2- The admin should be able to add overall discounts | GET http://localhost:8080/discount/overall?userName=hussien Adds an overall discount to all users for the first purchase regardless of service | GET /discount/overall Input: userName  Adds discount if the username is an admin, else displays an error |
| Admin  2- The admin should be able to add specific discounts | GET http://localhost:8080/discount/specific?userName=hussien Adds a specific discount to the service entered in its body | GET /discount/specific Input: userName, service  Adds discount if the username is an admin to the specified service, displays an error otherwise. |

|  |  |  |
| --- | --- | --- |
| Admin 3- The admin should be able to list all user transactions. | GET http://localhost:8080/transaction/all?userName=hussien Returns for example  [      {          "Transaction ID": "1",          "userName": "hussien",          "Transaction Type": "Add to Wallet Transaction",          "amount": "100.0",          "service": "Bank System"      },      {          "Transaction ID": "2",          "userName": "hussien",          "Transaction Type": "Payment Transaction with Credit Card",          "amount": "25.5",          "service": "WE"      }  ] | GET /transaction/all?userName= Input: userName  Returns a json array containing all transactions in the system  Note: the entered username must be that of an Admin else displays an error |
| Admin 4- The admin should be able to list all refund requests. | GET http://localhost:8080/transaction/refundRequests?userName=hussien Returns for example  [      {          "Transaction ID": "1",          "userName": "hussien",          "Transaction Type": "Refund Transaction",          "amount": "100.0",          "service": "Bank System"      }  ] | GET /transactions/refundRequsts? userName=  takes a userName as input and returns all refund requests if the username is that of an Admin |

|  |  |  |
| --- | --- | --- |
| Admin 4- The admin should be able to accept  or reject any refund request | POST http://localhost:8080/refund/review Body  {      "ID":"1",      "username": "hussien",      "Action":"accept"  }  Returns  {      "Response": "Refund Approved"  }  If the refund ID exists  {      "Response": "Invalid Request"  }  If the ID is invalid | POST /refund/review Takes transaction ID, username and the action the admin has taken towards this refund  Note: the username in the body needs to be that of an Admin |

# Github repository link

* **https://github.com/Zeeka32/Software-Engineering-2-Project**