Cairo University Faculty of Computers and Artificial Intelligence



**Software design specification document**

**2022**

**Project Team**

|  |  |  |
| --- | --- | --- |
| **ID** | **Name** | **Email** |
| 20201122 | عمر أشرف مسعد على | Narutoomar12@gmail.com |
| 20200759 | نور الدين وليد مرسى حسن | noureldin20200759@gmail.com |
| 20201007 | احمد حسام احمد عبد الفتاح | ahmedhossam222333@gmail.com |
| 20201003 | احمد ثروت سيد احمد | ahmad.th26@gmail.com |

Contents

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

[Class diagram design 2](#_Toc120811427)

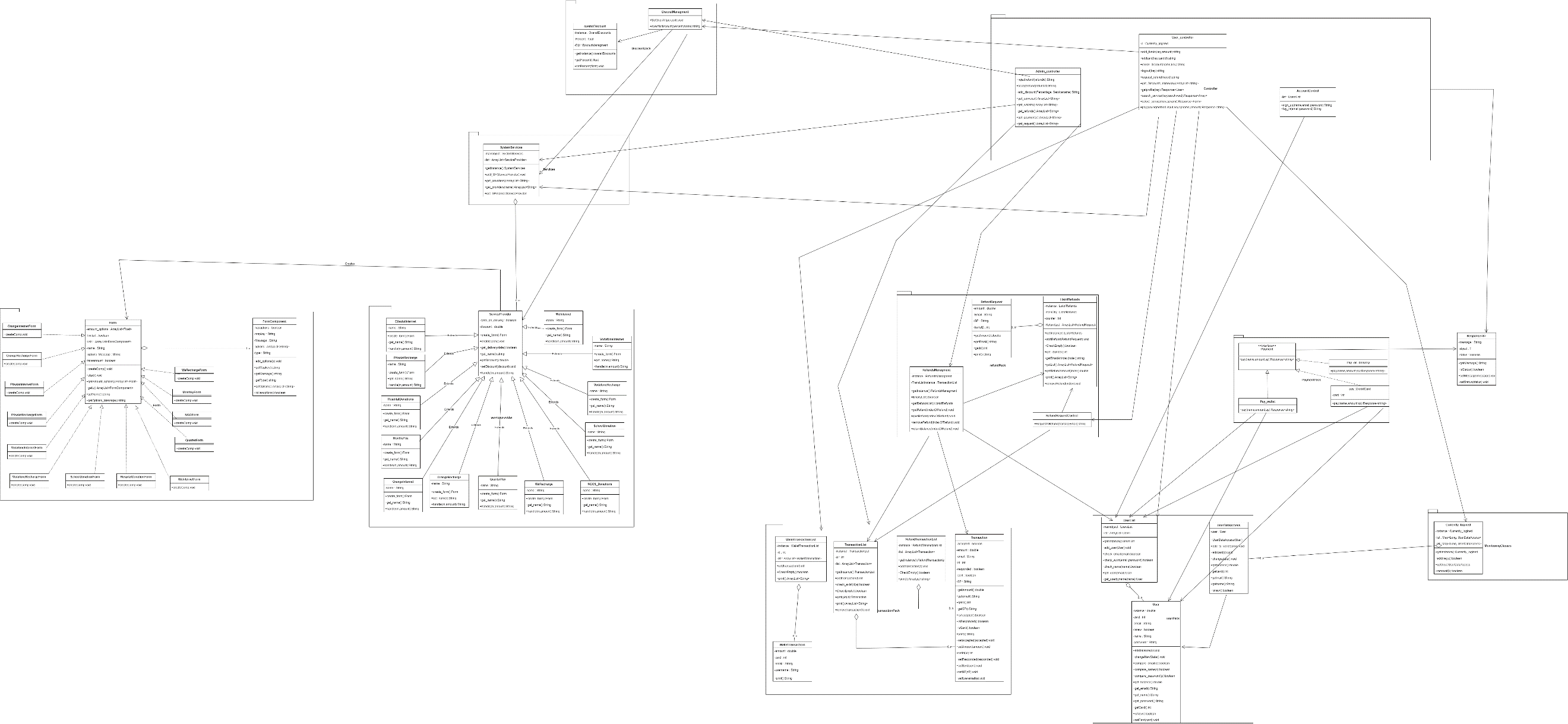
[Class diagram Explanation 3](#_Toc120811428)

[Sequence diagram design 3](#_Toc120811429)

[Github repository link 4](#_Toc120811430)

# Class diagram design

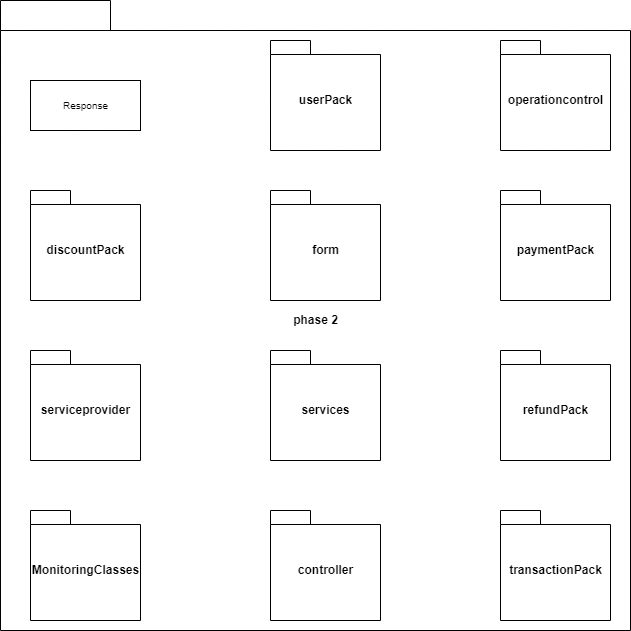
If you want to see a better class diagram picture we have added the source class diagram png



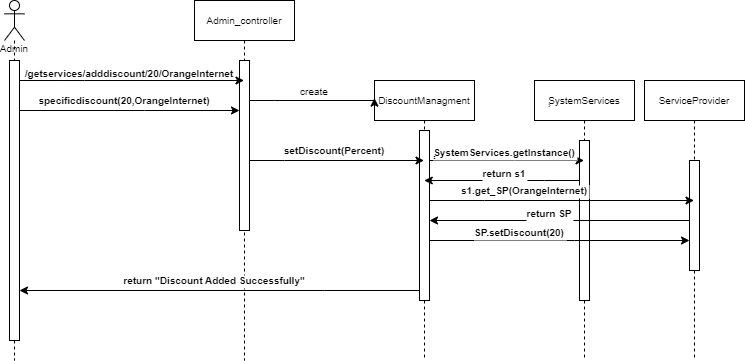
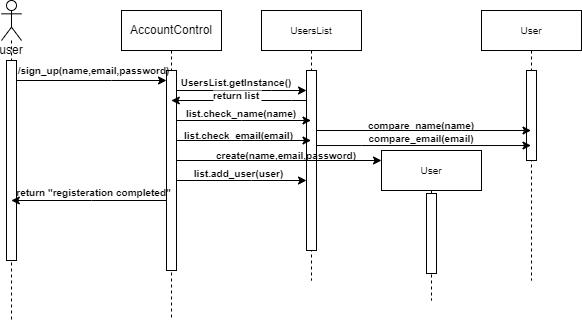
# Class diagram Explanation

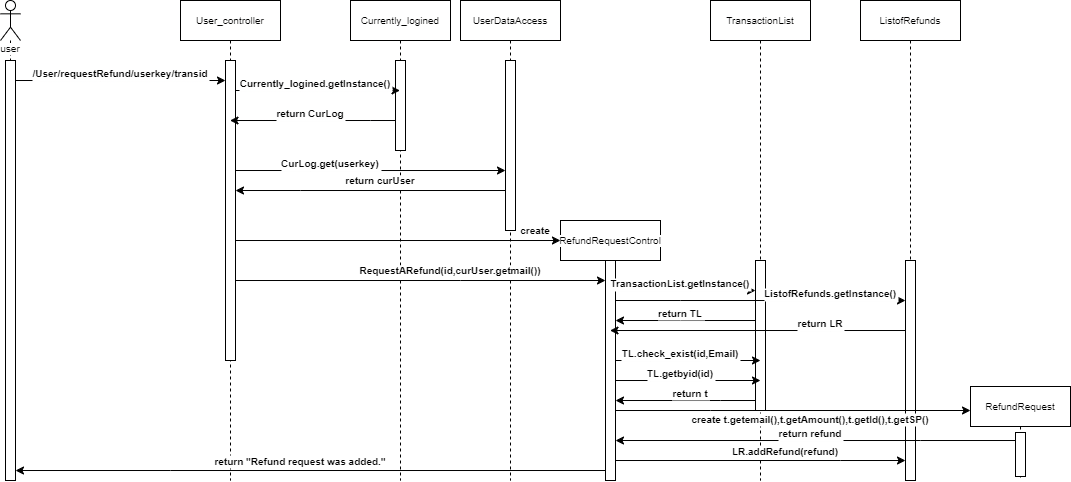
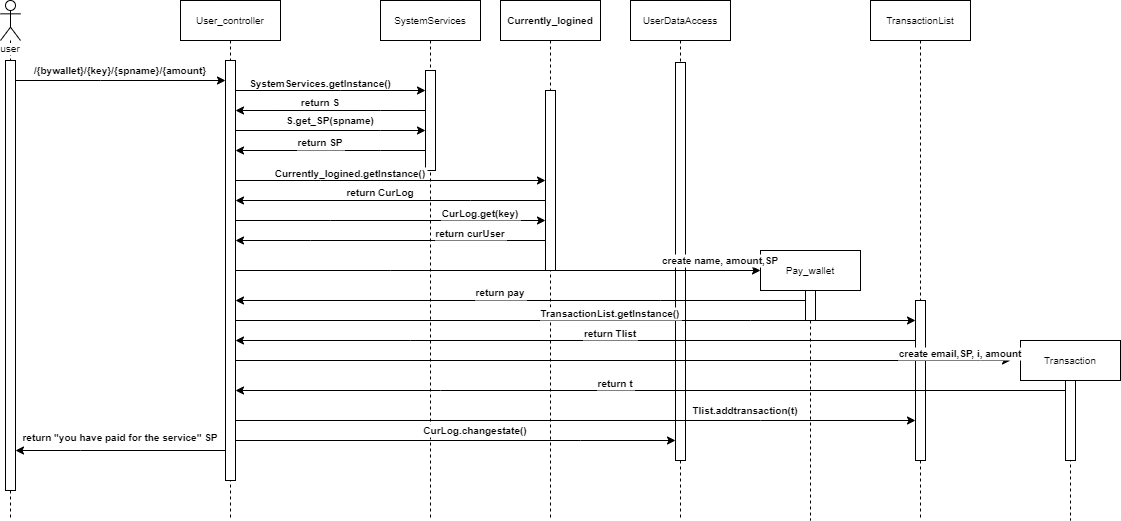
* **Singleton:Used to get one instance from all objects used to store a list of other objects and wanted to be used by many control objects.**
* **Used in: UsersList,SystemServices,list of refunds.**
* **Factory: Used with service provider objects to create a concrete form representing each object.**
* **Used with all objects implementing ServiceProvider interface.**
* **Strategy: used with payment object to be able to change the payment method at run time.**
* **Used in PaymentControl and payment interface.**

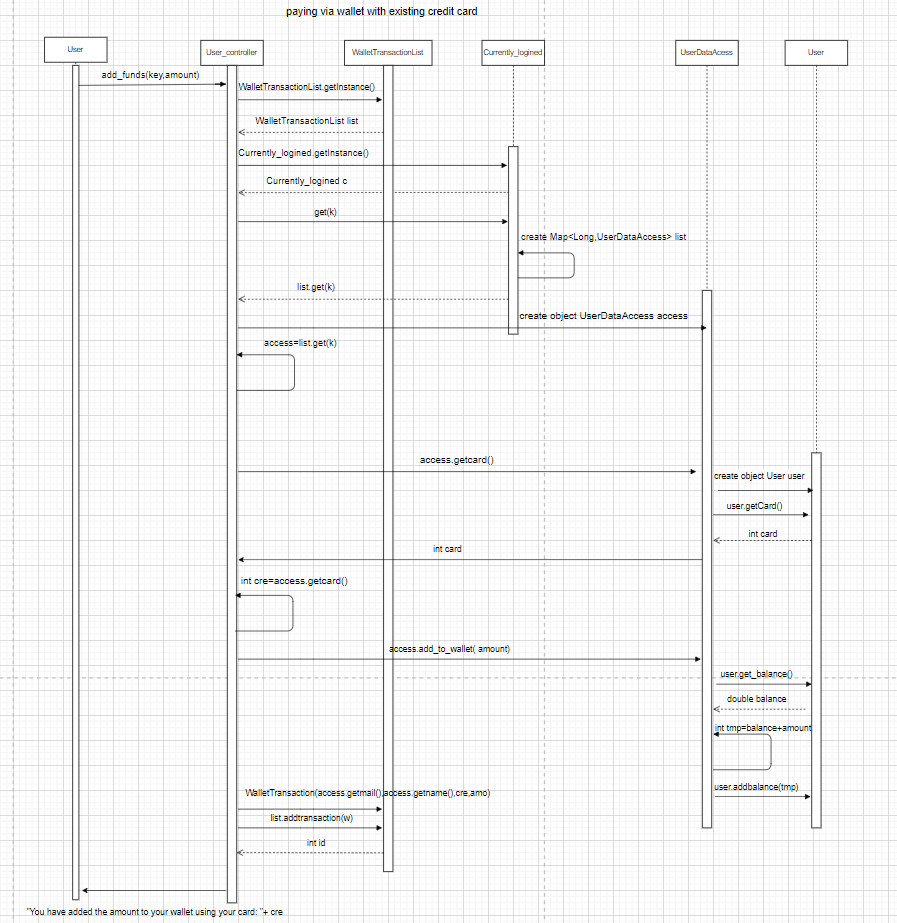
**Subsystem Decomposition Diagram:**

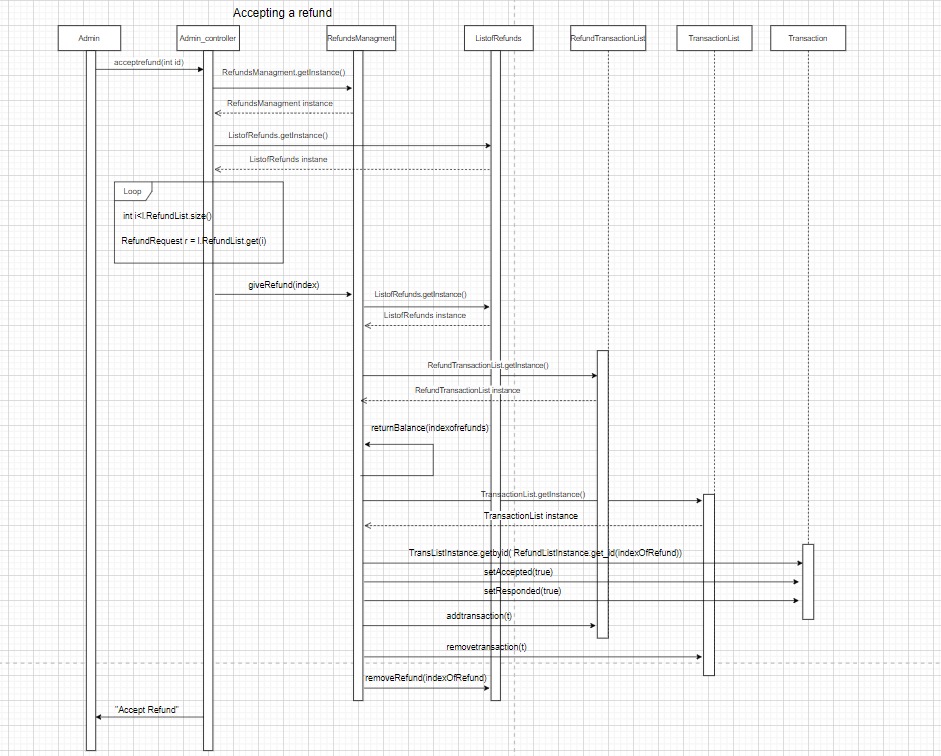
****

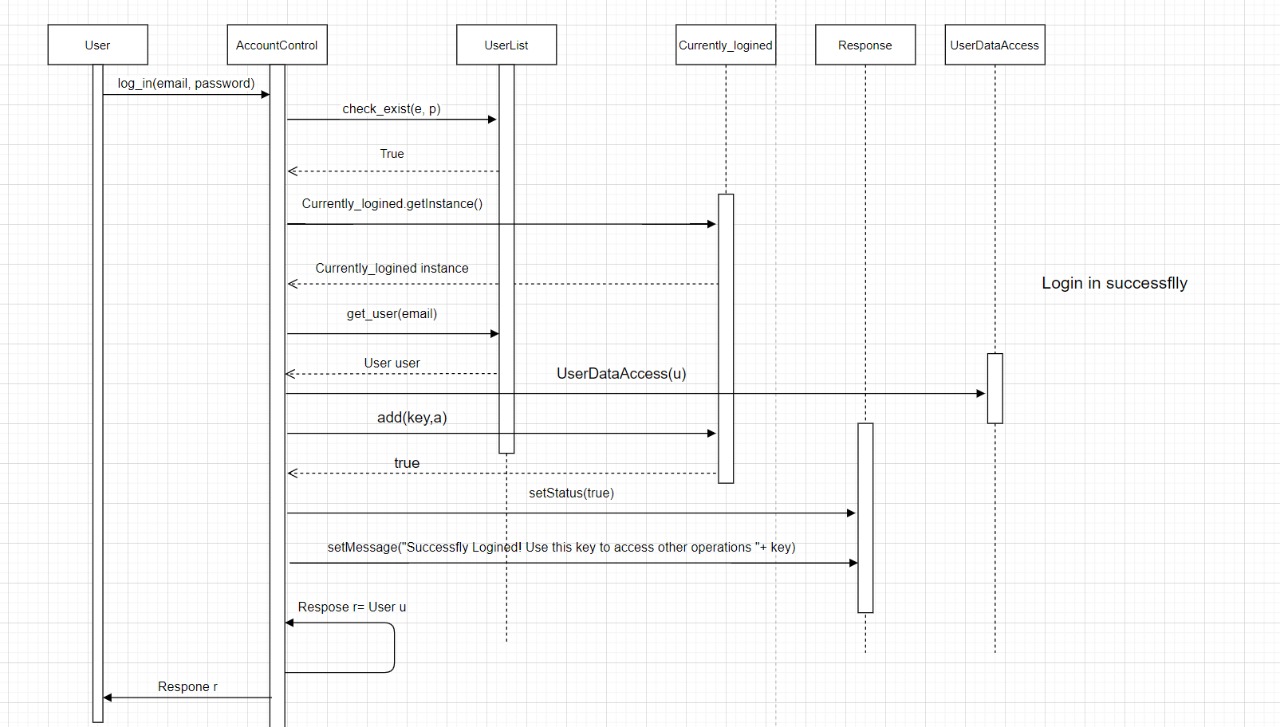
# Sequence diagram design

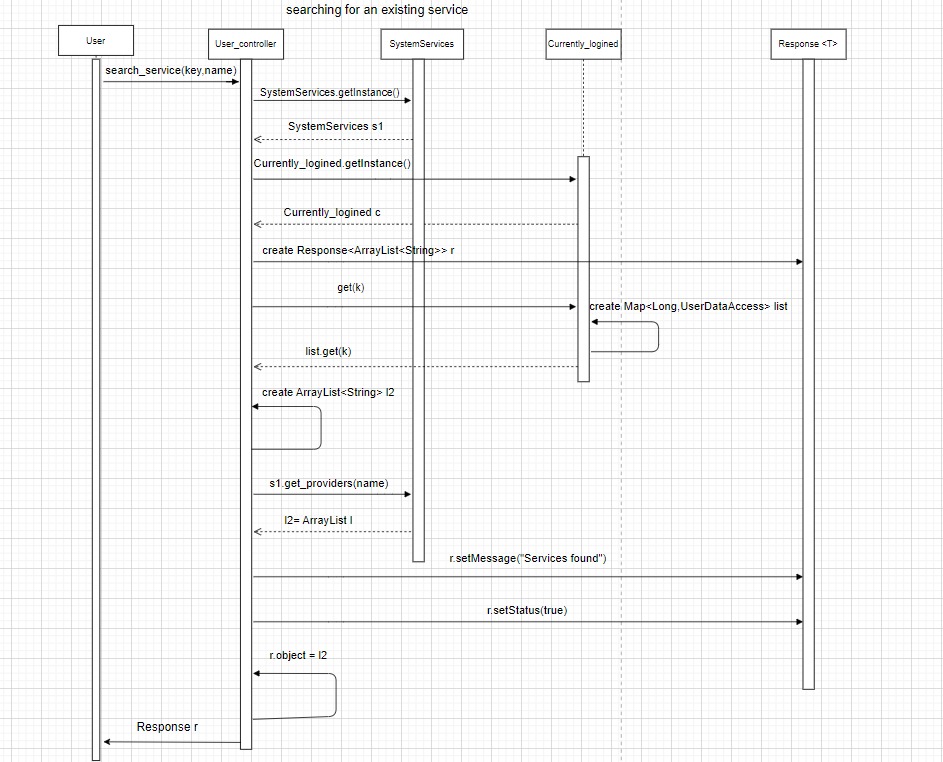
****

****

****

****

****

****

# Requirements Exposure as Web Service API

1-Post man collection

**Import the post man collection using this link note:Some urls requires you to update them with a new authentication key for a user after you login this user**

**https://api.postman.com/collections/25120965-e91f3c4b-6521-4f26-a72d-f356b822f3e8?access\_key=PMAT-01GNPNBMV178ZBHEJZ4RS1F81C**

|  |  |  |
| --- | --- | --- |
| Requirement |  | Exposed API |
| The user should be able to sign-in to the system. Given the user’s email and apassword, the user can login to the system and use any of the system functionalities. Example: GET /Users/login/ahmed/ahmed123  Input: email and password |  | 1-GET/Users/login/{email}/{pass} Service used to log in users this api returns user data and an authentication key if the inputs were correct the authenitication key is then used for the user to access other system services The authentication key is randomly generated and is valid for a user until he logs out of the system then he is given a new key when he logins again  The authentication key is then used in all other User functions it is located at the end of the message section of the returned response object if the login was successful  Also note if you logined for a user you cant login again unless you logout this user the logout service will be mentioned later in this table and when you logout a user his old key will not be valid any more and you have to login him again to get a new key for this user  When a user logins the system creates a key and store the key and user data in a map it uses later to check the key when accessing other functions  A user cant get more than one key when he logins to get a new key for a user you have to logout this user then you can login him again and get a new key  To get a user profile data there is a service mentioned later in this table displays the api used to get the data of the user using his authentication key that is given after login |
| 2- The user should be able to sign up to the system. The user should provide hisusername, email and password. The system should check if the username or the emailis registered before, if they are not registered before then the signup process shouldcomplete successfully, if not, the system will show an error to the user example:POST /Users/signup?name=ahmed&email=ahmed&password=ahmed123  input:name,email and password |  | 2-Post/Users/signup?name=value&email=value&password=value Used to sign up a new user accepts email and name and password if the user name or email exists before it returns an error message else it creates a new user and returns a success message |
| The user should be able to search for any service in the system. The user can type theservice name and the system will return all services that match the user query. Example:GET /User/search/1672499518264/internet  Input: user authentication key and search word |  | 3-GET/ User/search/{key}/{searchword} Service uses authentication key of the logined user and the word and returns all services names matching this word |
| NOTE: you can set the search word to overall to get all services names in the system |  |  |
| The user can pay for any service in the system. The system should prompt the user tothe payment form when the user asks to pay for any service. The default way is to payvia credit card. The system should allow the user to consume from the wallet (checkReq. 6) for this payment. If the service that should to receive the payment accepts cacheon delivery, then this option should be visible too. Example:  POST /User/pay/bywallet/1672499871994/OrangeRecharge/500  {  "Number": "01005164789"  }  Returns a transaction id the user can use later if he want to ask for a refund for this payment operation  Note the “Number” key is not constant each service provider has its own set of keys that it needs to perform its operation successfully  To get the names of keys use  GET /User/select/1672499871994/OrangeRecharge |  | 4-POST User/pay/{payingMethod}/{key}/{spname}/{amount}{“KEY1”: “VALUE”, “KEY2”: “VALUE” } NOTES: Input  KEY:Must be a valid logined user authentication key  SPname:Must be a valid service provider name  Very important:  The body of the requsest must contain a valid key name and value that the sp accepts to get the names of the key the sp accepts use  GET /User /select/{key}/{spname}  Paying methods names:  bycard:requires that the user add his card id before paying adding card is performed using :POST/User/AddCard/{key}/{cardid}  bywallet: requires that the user has sufficient funds in his wallet  cash: requires that the sp accepts cash on delivery  note the systems apply discounts automatically if it exists for these services and if admin add an overall discount it applies to the first payment transactions of new users you can test it if you like |
| The user can ask for a refund for any complete transaction to any given service. Therefund request will be issued by the user and sent to the admin. If the admin approvesthe refund then the refund process should complete successfully. Example:  POST s/User/requestRefund/1672499871994/1 |  | 5- POST /USER /requestRefund/{key}/{id}  Notes:  Key must be a valid logined user authentication key  Id must be a valid payment id that belongs to the user with the key  The refund request will not be sent again represinting the same payment transaction if it was rejected or if it has been sent before but not yet responded to |
| The system maintain 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. EXAMPLE: POST/User/addwallet/1672499871994/5000  Returns an error message if the user didn’t add a credit card number to his profile |  | 6-GET /USER /addwallet/{key}/{amount}NOTES: Key must be a valid logined user authentication key  The user must have added his card id to his profile before using this operation  To add a credit card to profile use:  POST/User/AddCard/{key}/{cardid} |
| The user should be able to check any discount for any service in the system. Discountscould be added by the admin (this will be discussed later). GET /User/checkdiscount/1672499871994/HospitalDonations |  | It is done using two apis: 1-GET/USER/getdiscountnames/{key}  The key is as shown above is the authentication key a user gets when logs in  Displays all services name that the user can select one name from to send a request to get discount amount of this service  2-GET/User/checkdiscount/{key}/{name}  Returns the discount value of the service name if it exists return error other wise  Note send overall in the name section to get the overall discount value |
| Some useful services we added |  |  |
|  |  | 1-GET/User /profile/{key}Returns all user data related to this key if the key is a valid login key else returns and error message |
|  |  | 2-POST/User/AddCard/{key}/{cardid} Add a card number to the user account to be used in payment and adding funds to wallet operations |
|  |  | 3-POST/User /logout/{key} Logut a user from the system meaning that his authentication key will not be valid any more and if a user want a new key he must login again and use the returned key |
| The admin should be able to add discounts to the system. There are two types ofdiscounts.a. Overall discounts. For example the user should have 10% discount for the firsttransaction (regardless the service)b. Specific discount.. For example the admin can apply 20% discount for all mobilerecharge services.For any given service. All overall discounts and specific discounts for this service shouldapply. Example:  POST/Admin/adddiscount/30/HospitalDonations |  | 1-GET /Admin/getservices Returns all system services names he can use to add a request to a specific service using its correct name  2-POST /Admin/adddiscount/{percentage}/{Servicename}  Service name must be a valid correct service name and percentage must be lower than 100 and higher than 0 |
| The admin should be able to list all refund requests. Each refund request should containthe related service and the amount to be refunded. EXAMPLE:  GET /Admin/getRefunds  **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.**  **Example:**  **POST** /Admin/acceptrefund/1  Here 1 was a refund request id that existed |  | 1-GET/Admin /getRefunds Takes no arguments and returns the refund requests details (if exist).the most important detail is its id because the admin will use to accept or reject the refund  1-POST/Admin/acceptrefund/{refunded}  2-POST/Admin/rejectrefund/{refundid}  Accept or reject a refund using the refund id passed in the url note that the id must be valid you can check ids using the service above |
| 3. The admin should be able to list all user transactions. The transactions types area. Payment transaction.b. Add to wallet transaction.c. Refund transaction. EXAMPLE:  GET /Admin/getTransaction/wallet |  | 1-GET/Admin/getTransaction/{name} There are three name types  payment: returns all payment transactions  refund: returns all accepted refund transactions  wallet: returns all add to wallet transactions |

# Github repository link

* **https://github.com/TheNight1234/SEProject.git**