Cairo University Faculty of Computers and Artificial Intelligence



**Software design specification document**

**2022**

**Project Team**

|  |  |  |
| --- | --- | --- |
| **ID** | **Name** | **Email** |
| 20206148 | Abdulrhman Emad Kamel Ismail | [ab.em1298@gmail.com](mailto:ab.em1298@gmail.com) |
| 20206120 | Ahmed Elsayed Moein | [ahmed33elsayed22@gmail.com](mailto:ahmed33elsayed22@gmail.com) |
| 20206136 | Ahmed Sami Darwish | ahmedsami1423@gmail.com |
| 20206074 | Mootaz Medhat Ezzat | mootazmwahab@gmail.com |

Contents

[Instructions[To be removed] **Error! Bookmark not defined.**](#_Toc120811426)

[Class diagram design 2](#_Toc120811427)

[Class diagram Explanation 2](#_Toc120811428)

[Sequence diagram design 3](#_Toc120811429)

[Github repository link 5](#_Toc120811430)

# Class diagram design

**Diagram, schematic

Description automatically generated**

**For a better view please click here:** [**https://drive.google.com/file/d/1tVYl4UyvBcfQ3l5xPG\_laV2iPClr9vtd/view?usp=sharing**](https://drive.google.com/file/d/1tVYl4UyvBcfQ3l5xPG_laV2iPClr9vtd/view?usp=sharing)

Open the link above then click on ***open with*** and choose ***diagrams.net***.

# <https://drive.google.com/file/d/1tVYl4UyvBcfQ3l5xPG_laV2iPClr9vtd/view?usp=sharing>

# Class diagram Explanation

# we have a service class that combine between two patterns

# - Abstract factory

# - Strategy pattern

# it helps us to avoid class explosion and easily control the objects

# The abstract factory pattern:

# we have two services that are factories – mobile recharge - internet payment

# they control to produce products -> service providers:

# - We

# - Orange

# - Vodafone

# - Etisalat

# Every service provider controls the forms if it’s a mobile recharge or internet payment

# The service providers have a various code families related with products

# Example of a concrete product (we Internet payment)

# Each concrete product depends on interface product

# Also, abstract factory helps us to add a new service provider ass an abstract product without any changes on the main code

# The strategy pattern:

# At the donation and land line we used a strategy pattern for each

# The strategy pattern helps us in these tow services that we have the same need of creating form but every form has its own algorithm

# - At land line each QuartlyLL – MonthlyLL has its own algorithm for creating a form

# - At donations each hospital – schools – NGOs has its own algorithm for creating a form

# That different algorithm depends on interface services donations and land line

# also, the strategy pattern helps us for adding any new algorithm without any changes on the code

# just adding the new code

# At the end:

# There is a service control class that control with every class at the package and control the communication between other components

# With help from service interface class

# The Decorator pattern:

We used the decorator pattern to decorate the bill with the bill with 2 types of discounts overall discount and service discount as both need to alter the behavior of the bill without a change in structure. This is implemented by using Bill <<interface>> which is a common interface for both BillDecorator and ConcreteBill. The Concrete bill has the basic behavior which will be altered by the decorator. Then the BillDecorator implements one or both of the Discounts Available through concrete Decorators OverallDiscount and ServiceDiscount

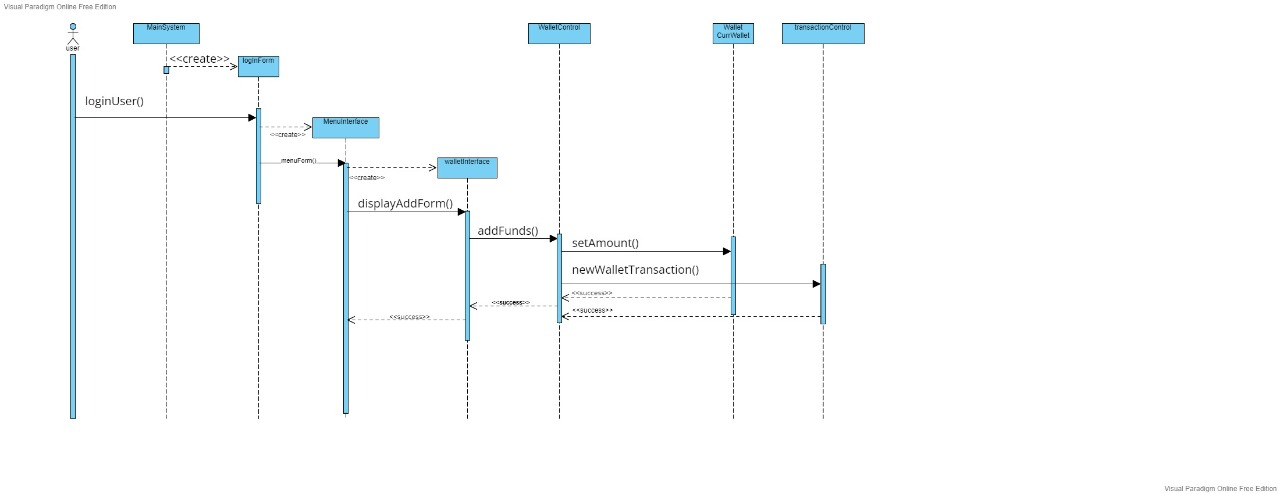
# The Builder pattern:

for the user component we applied the builder pattern to avoid the large constructor of the user class ... by adding IUserBuilder as the builder to build the NormalUser and the SuperUser classes to the client which is the UserControl

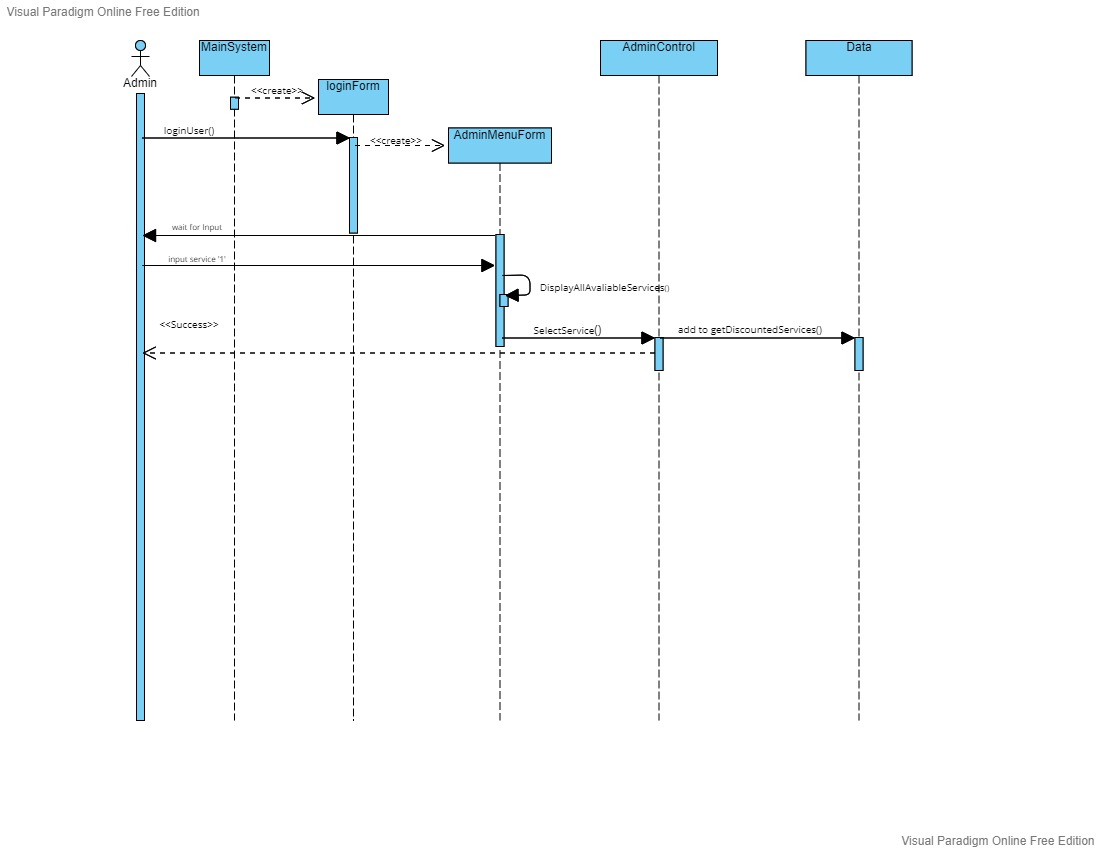
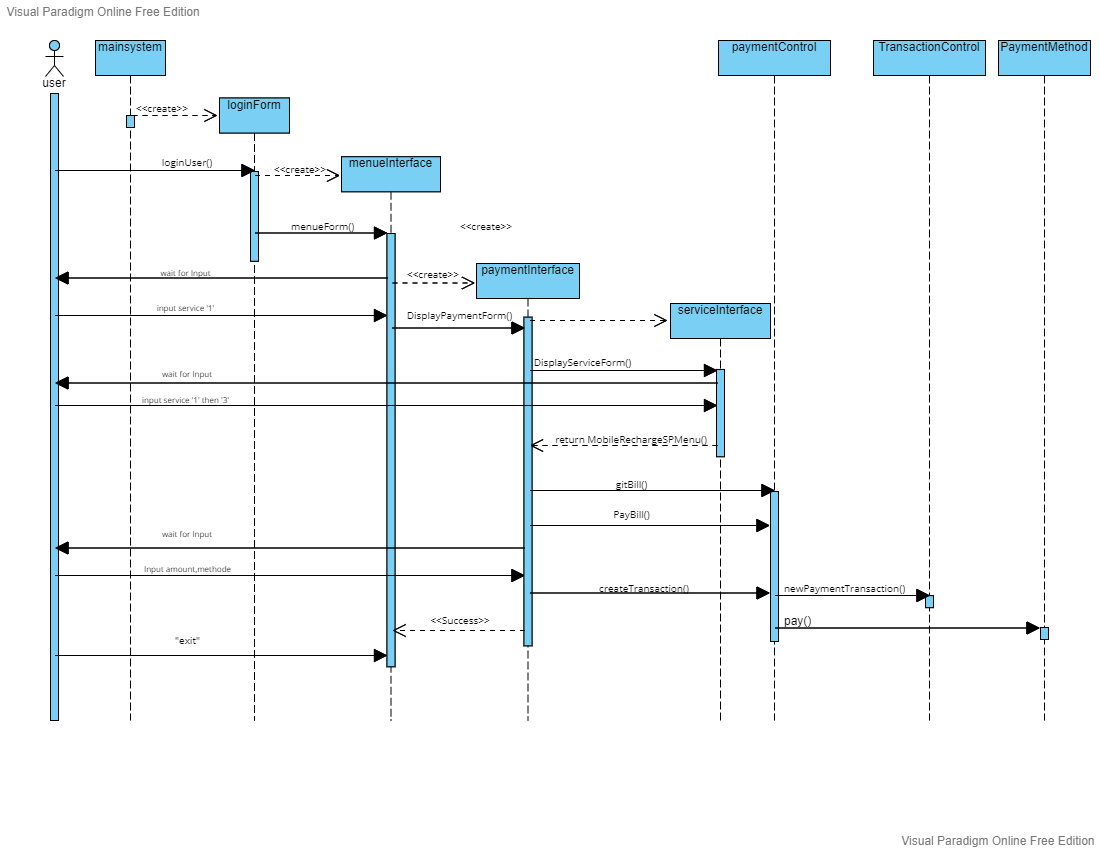
# The Singleton pattern:

for the Data component we used the singleton pattern to the class data because we need one instance from the data class in the whole system by privately create the constructor of data and add a public method called getInstance()

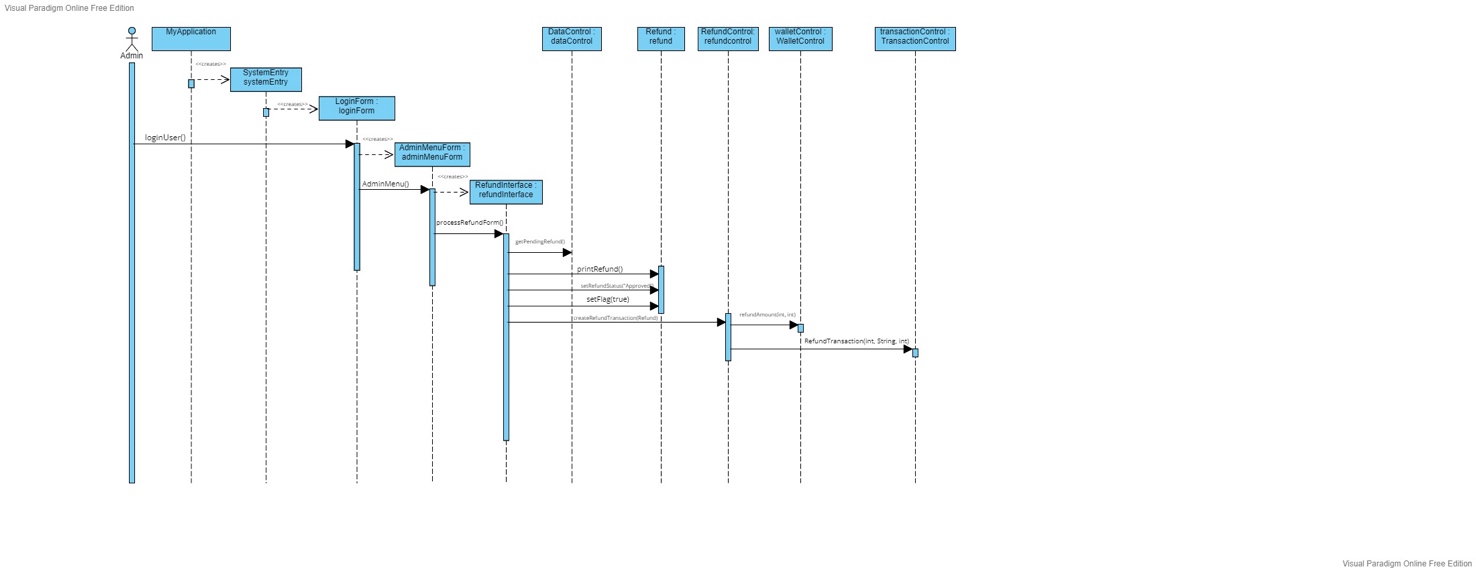
# Sequence diagram design

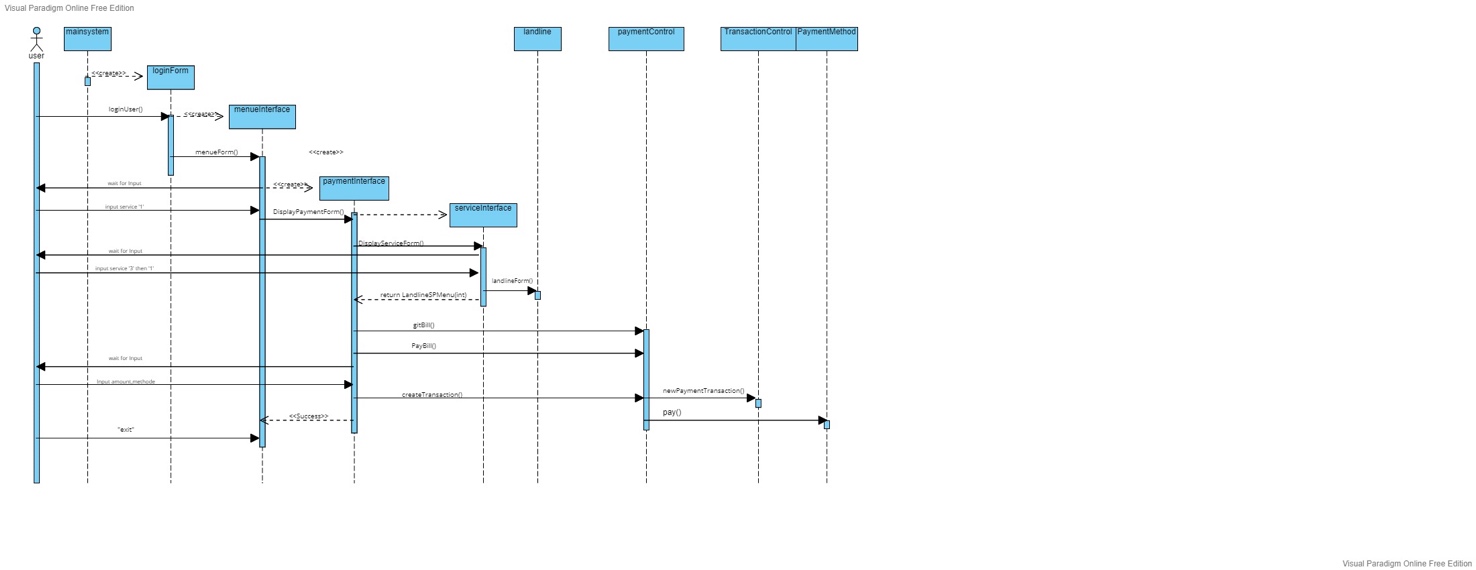


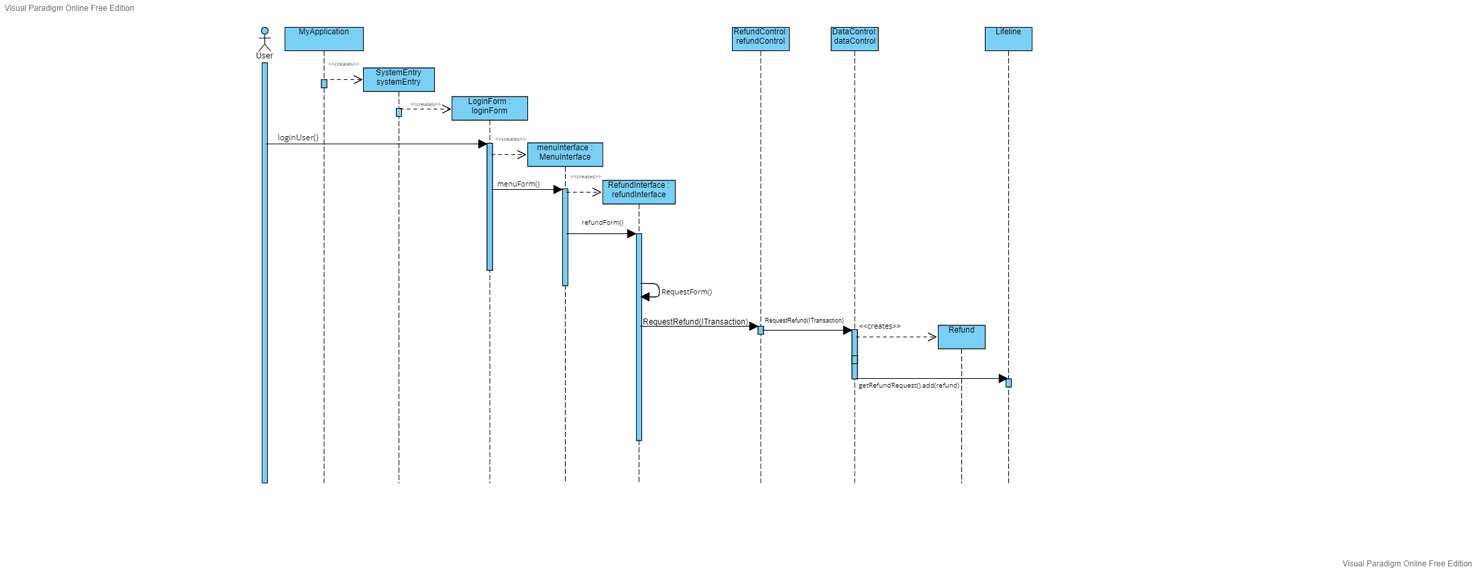
# 

****

# 







# Requirements Exposure as Web Service API

**Part 1: Exposed Postman Collection**

https://ae-coding.postman.co/workspace/My-Workspace~1a7d7998-e7b1-4939-8efd-6eb1106fa809/collection/25121317-c3a1b2d0-af56-4fac-b436-96220e88088f?action=share&creator=25121317

**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.**

For the best experience display all the users firstly (GET /Admin/Display/AllUsers) and then set one of the users as the current user (POST /User/CurrentUser/{id})

|  |  |  |
| --- | --- | --- |
| Requirement |  | Exposed API |
| Display all users using id. |  | 1- GET /Data/Display/User/{id}  A service to retrieve the user with a specific id. Input: id |
| Display all the users on the system |  | 2- GET /Data/Display/AllUsers  This prints all the users saved on the system with their data  **No Input** |
| Display all the transactions completed |  | 3- GET /Data/Display/Transactions  This prints all the transactions saved on the system with their info  **No Input** |
| Display all the payment transactions |  | 4- GET /Data/Display/PaymentTransactions  This prints all the transactions saved on the system with their info but only the payment transactions  **No Input** |
| Display all Services with a discount |  | 5- GET /Data/Display/DiscountedService  This prints all the discounted services saved on the system  **No Input** |
| Display all the Users with a discount |  | 6- GET /Data/Display/DiscountedUsers  This prints all the discounted services saved on the system  **No Input** |
| Display the Refund Requests |  | 7- GET /Data/Display/RefundRequests  This prints all the Refund Requests saved on the system  **No Input** |
| Check if the user has discount |  | 8- GET /Data/Check/OverallDisount/{id}  Checks if the user has a discount and displays the result.  Input: UID |
| Chich if the service has discount |  | 9- GET /Data/Check/ServiceDisount/{Service}  Checks if the user has a discount and displays the result.  Input: ServiceName |
| Show the user’s wallet data |  | 10- GET /Data/Wallet/{id}  Displays the wallet data of the user (UID & amount in wallet).  Input: id |
| Add discount to a service |  | 11- GET /Data/Add/ServiceDiscount/{Service}  This adds a 20% discount to a service  Input: ServiceName |
| Add a discount to a user |  | 12- POST /Data/Add/UserDisount/{id}  This adds a 20% discount to a user  Input: UID |
| Checks if the user had signed up |  | 13- GET /Data/Check/User/SignUp  This takes a JSON input of a user and outputs whether the user have signed up or not  Input: User |
| Add a wallet |  | 14- POST /Data/Add/Wallet  This takes a wallet and saves it on the system  Input: Wallet |
| Registers a User to the system |  | 15- POST /Data/Add/User  This Registers the user and saves him on the system  Input: User (JSON) |
| Check If the user exists |  | 16- GET /Data/Check/User/Login/{userName} /{password}  This checks if the user is saved on this system using the username and the password.  Input: username, password |
| Display the Pending Refund Requests |  | 17- GET /Data/Display/PendingRefundRequest  This Displays all the pending refund requests saved on the system  No Input |
| Add a Transaction |  | 18- POST /Data/Add/Transaction  Adds the Transaction to the data saved on the system  Input : ITransaction (JSON) |
| Display User Refund Requests |  | 19- GET /Data/Display/PendingRefund/{id}  Displays all the pending refund requests of a specific user  Input: UID |
| Display User Transactions |  | 20- GET /Data/Display/UserTransactions  Displays all the transactions of the Current user  No Input |
| Add a payment Transaction |  | 21- POST /Data/Add/PaymentTransaction/{Service} /{amount}  Adds a payment transaction to the data saved on the system  Input: ServiceName, amount |
| Display User payment Transactions |  | 22- GET /Data/Display/UserPayTransactions  Displays all the payment transactions of the Current user  No Input |
| Display User Refund Requests |  | 23- GET /Data/Display/UserRefund  Displays all the refund requests of the Current user  No Input |
| Add Refund Request |  | 24- POST /Data/Add/Refund  Adds a refund request to the data saved on the system  No Input |
| Search for a service |  | 25- GET /Data/Service/Search/{Service}  Search for a service on the system by taking the service name or a part of the name ie: Voda shows Vodafone Mobile Recharge and Vodafone Internet payment.  Input: ServiceName |
| Display All Available Services |  | 26- GET /Data/Service/Display  Displays all the available services on the system.  No Input |
| Select a Service |  | 27- GET /Admin/SelectService/{num}  Allow the user to select a service by inputting a numer  Input: num |
| Add Service Discount |  | 28- POST /Admin/Add/ServiceDiscount/{num}  Add a 20% discount to the service with the given the number of the service in the menu  Input: num |
| Add User Discount |  | 29- POST /Admin/Add/UserDiscount/{id}  Add a 20% discount to the user with the given UID  Input: UID |
| Display All Users |  | 30- GET /Admin/Display/AllUsers  Displays all the users in the system  No Input |
| Display all available Services |  | 31- GET /Admin/Display/AllServices  Displays all the available services  No Input |
| Build a User |  | 32- POST /User/Build  Builds a user using the builder pattern.  No Input |
| Register a user |  | 33- POST /User/Register  Adds the user to the system  Input: User (JSON) |
| Check if user exists |  | 34- GET /User/Check//{userame} /{password}  Input: username, password |
| Check if user is admin |  | 35- GET /User/CheckAdmin/{userame} /{password}  Checks if the user is admin or not  Input: username, password |
| Set User as Current User |  | 36- POST /User/CurrentUser/{id}  Makes the user become the current user of the system  Input: id |
| Get User Wallet |  | 37- GET /Wallet/Get/{id} |
| Add Funds To wallet |  | 38- POST /Wallet/AddFuds/{id}/{amount} |
| Add Refund To wallet |  | 39- POST /Wallet/Refund/{id}/{amount} |
| Display Wallet Balance |  | 40- GET /Wallet/Balance/{id} |
| Pay Using Wallet |  | 41- POST /Wallet/Pay/{id}/{amount} |
| Start Payment |  | 42- POST /Payment/Start/{id}/{service} /{amount} |
| Get Bill with the Discount |  | 43- GET /Payment/Bill/{id}/{service} /{amount} |
| Pay Using Cash |  | 44- POST /Payment/Cash/{amount} |
| Pay Using Credit Card |  | 45- POST /Payment/Credit/{amount} |
| Pay Using Wallet |  | 46- POST /Payment/Wallet/{amount} |
| Create Transaction |  | 47- POST /Payment/Transaction/{id}/{service} /{amount} |
| Get Current User id |  | 48- GET /Service/User/ID |
| Pay For Service |  | 49- POST /Service/Pay/{id}/{service} /{amount} |
| Mobile Recharge Service |  | 50- POST /Service/MobileRecharge/{id} |
| Internet payment service |  | 51- POST /Service/InternetPayment/{id} |
| Landline service |  | 52- POST /Service/Landline/{id} |
| Donation Service |  | 53- POST /Service/Donations/{id} |
| Get User ID for Transaction |  | 54- GET /Transaction/GetUID |
| Print the Transaction |  | 55- GET /Transaction/Print |
| Save Transaction |  | 56- POST /Transaction/Save |
| Get A transaction |  | 57- GET /Transaction/Get/{id} |
| Display All Transactions |  | 58- GET /Transaction/GetAll |
| New Payment Transaction |  | 59- POST /Transaction/Payment/{service} /{amount} |
| New Wallet Transaction |  | 60- POST /Transaction/Wallet/{amount} |
| New Refund Transaction |  | 61- POST /Transaction/Refund/{id}/{amount} |
| Display Payment Transaction |  | 62- GET /Transaction/GetPay |

# Github repository link

https://github.com/aemoein/MyServices