Cairo University Faculty of Computers and Artificial Intelligence



**Software design specification document**

**2022**

**Project Team**

|  |  |  |
| --- | --- | --- |
| **ID** | **Name** | **Email** |
| 20200520 | Mariam Ashraf Amin | mariamashrafamin@gmail.com |
| 20200521 | Mariam Saeid Shawky | mariamsaeid142@gmail.com |
| 20200186 | Reem Ayman Abdel-Fattah | reem.ayman52002@gmail.com |

Contents

[Class diagram design 3](#_Toc123419201)

[Class diagram Explanation 4](#_Toc123419202)

[Sequence diagram design 6](#_Toc123419203)

[Requirements Exposure as Web Service API 13](#_Toc123419204)

[Github repository link 18](#_Toc123419205)

# Diagram, schematic Description automatically generatedClass diagram design

# Class diagram Explanation

**Factory method pattern:**

* **Classes:**

Services as product class.

MobileRechargeServices, InternetPaymentServices, LandLiveServices and Donation as concrete product class.

ServiceProvider as creator class.

Vodafone, Etisalat, Orange and We as concrete creator class.

* Using factory method pattern to let sub-classes to choose the type of objects to create.
* We can see that Donation can be done by alternative ways (Vodafone-We-Etisalat-Orange)

**Decorator pattern:**

* **Classes**:

Services as component class.

MobileRechargeServices, InternetPaymentServices, LandLiveServices and Donation as concrete component class.

Discounts as base decorator class.

OverAllDiscount and SpecificDiscount as concrete decorator class.

* Using decorator pattern to let inserting new behaviors to objects by placing these objects inside one or some wrapper objects that contain the behaviors.
* We can see that in OverAllDiscount and SpecificDiscount we can use them to add new behavior in services’ subclasses, one of them or mix of two decorator.

**Singleton pattern**

* **Class:** InitialData
* Using singleton pattern to create one instance which is updated not overwritten.
* getinistance returns updated data after adding.

**Strategy pattern**

* **Classes:**

User as context.

Transaction as strategy class.

PaymentTransaction, RefundTransaction and AddToWalletTransaction as a concrete strategy classes.

* Using strategy pattern to support alternative behaviors.
* User class communicates to paymentTransaction only through Transaction interface which contain what is common between concrete classes.

# Sequence diagram design

Diagram, schematic

Description automatically generated1) admin list all refund requsets Sequence

Diagram, schematic, timeline

Description automatically generated2) signIn – siginUp Sequence

Diagram, timeline

Description automatically generated3) search Sequence

Diagram

Description automatically generated4) choose service and service provider Sequence

Diagram, schematic

Description automatically generated5) pay Sequence

Diagram

Description automatically generated6) ask for refund Sequence

7) add funds Sequence

Diagram

Description automatically generated

# Requirements Exposure as Web Service API

**Part 1: Exposed Postman Collection**

[**https://elements.getpostman.com/redirect?entityId=25072478-96bd662b-1892-497b-adb4-491ef54d0c6d&entityType=collection**](https://elements.getpostman.com/redirect?entityId=25072478-96bd662b-1892-497b-adb4-491ef54d0c6d&entityType=collection)

**Part 2:**

|  |  |
| --- | --- |
| **Requirements** | **Exposed API** |
| 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 and use any of the system functionalities | 1. POST/singin   Allow user to login with his into system through entering email and password.  And system checks if that email is exit.  Input: email and password.  Example:  {  “email”:”ahmed@gmail.com”,  “password”:”123456”  } |
| 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 | 1. POST/singup   Allow users to create a new account with unique username and email if these are found error message is shown else signup process will be completed successfully.  Input: username, email and password.  Example:  {  “username”: “ahmed”,  “email”:”ahmed@gmail.com”,  “password”:”12345678”  } |
| 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. | 1. GET/search/{name}   Allow users to search for services if he singin,  As list of matching services will be return.  Input: name  Example:  /search/mobile |
| 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. | 1. GET/showservice   First check if user is singin or not, show services to choose num to pay for  **then**  POST/chooseservice  user enter num of service he wants  Input: number  example: 1  **then**  GET/showserviceprovider  show service providers for selected service  **then**  POST/chooseserviceprovider  user enter char of service provider he wants  Input: service provider char  example: e  **then**  GET/showform  It returns form for entering data by user  **then**  POST/handleform  user fill the form  Input: form field and answer for it  Example:  {  "amount":"100",  "mobile number":"01235633"  }  **then**  GET/showpaymenu  It returns different payment method.  **then**  POST /pay  For entering number of payment method, it will display error message in case there is not enough money. And another one for invalid payment method.  Input: number  example: 1 |
| 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. If the admin approves the refund then the refund process should complete successfully. | 1. GET/showtransaction   First check if user is singin or not, then system let user to show transactions to choose num to refund for  **then**  POST/askforrefund  entering number of transaction to refund.  Input: transaction number  Example: 2 |
| 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. | 1. POST/addfunds   First check if the user is signin or not, then user enter amount of money to add in wallet, system checks if credit card has enough money to add in wallet if not error message will be appeared.  Input : money  Example: 100 |
| The user should be able to check any discount for any service in the system. | 1. GET/showdiscounts   After signing in user can check the exiting discounts. |
| The admin should be able to list all user transactions. | 1. GET/admin/listusertransaction   First it checks if there is a user in system that has transactions to view if not error message will be appeared. |
| The admin should be able to list all refund requests. Each refund request should contain unique name, the related service and the amount to be refunded. 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. | 1. GET/admin/listrefundrequest   First it checks if there is a user in system with refund requests to view if not error message will be appeared.  **then**  POST/admin/chooserefundrequest  After viewing refund requests, admin will choose refund to reply to it as entering refund’s unique name(request num+username) and he accept it or not.  Input: name, flag  Example:  {  “name”: “1ahmed”,  “flag”:”y”  } |
| A service to check if the user exists or not. This service returns all user info if exists | 1. GET/user/check   It returns all information for user through entering his email and password, if such user is not found error message will be appeared  Input: email and password  Example:  {  “email”: “ahmed@gmail.com”,  “password”:”12345678”  } |

# Github repository link

* **https://github.com/Mariam-Ashraf-510/AdvancedSoftwareProject**