Cairo University Faculty of Computers and Artificial Intelligence



**Software design specification document**

**2022**

**Project Team**

|  |  |  |
| --- | --- | --- |
| **ID** | **Name** | **Email** |
| 20200215 | Sara Ashraf Ali | sara.ashraf1820@gmail.com |
| 20200574 | Merahan Soliman Mohamed | merahansolimann@gmail.com |
| 20201048 | Toka Hamdy Mohamed | tokaahamdy.21@gmail.com |
| 20200674 | Youssef Nasser Abdel Hafeez | y556217@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

**A picture containing diagram

Description automatically generated**

# Class diagram Explanation

* Singleton pattern:
  + participating classes -> SystemData, FawrySystem.

By using only one static object and use it in the whole system.

* Observer Pattern:
* Participating classes -> (Subject: Admin, Observable: Service) and (Subject:Admin, Observable: User).

Admin notifies the services when adding a specific discount.

Admin notifies the user when adding an overall discount.

* Strategy Pattern:
* Participating classes -> Payment, Creidtcard, Cash, Wallet.

The payment method has different behaviors, so we separated it to change the payment method easily

* Factory Pattern:
* Participating classes -> Provider, We, Orange, Etisalat, Vodafone, Service, MobileRecharge, InternetPayment, Landline, Donation.

Abstract Factory -> Service.

Concrete Factory -> MobileRecharge, InternetPayment, Landline, Donation.

Abstract Product -> Provider.

Products -> We, Orange, Etisalat, Vodafone.

The abstract Service is the interface that defines how to make a family of related products.

# 

# Sequence diagram design

**Diagram

Description automatically generated with medium confidence**

# 

Diagram

Description automatically generated

# Diagram Description automatically generated

# Diagram Description automatically generated

# Diagram Description automatically generated

# Diagram Description automatically generated

# A picture containing diagram Description automatically generated

# Diagram Description automatically generated

# Subsystem decomposition Component diagram

# Graphical user interface, diagram, text, application Description automatically generated

# Requirements Exposure as Web Service API

**Part 1: Exposed Postman Collection**

<https://www.postman.com/crimson-star-936809/workspace/fa958cc4-6fd8-4e7b-8b91-733d1d6e0ab6/collection/25080807-cdbf6a95-02dd-4bff-aea1-ab48ebddd659?action=share&creator=25080807>

**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.**

|  |  |  |
| --- | --- | --- |
| Requirement |  | Exposed API |
| The user should log in first. |  | 1-  -URl: /user/myInfo  - A service should return the information of the current user  -No input |
| The system should check if the username or the email is registered before. |  | 2-  - URL: /user/signup  - A service for user to make account in the system  Input: username, password, e-mail |
| The system should check if the username or the email is registered before. |  | 3-  -URL: /user/login  - A service for the user to log in by his account  - Input: username, password. |
| The user should be logged in first. |  | 4-  - URL: /user/logout  - A service for the user to log out from his account  -No input |
| The user should be logged in first and should have enough money to pay for the service. |  | 5-  -URL: /user/mobilerecharge/{provider}/{paymentMethod}  Where:   * provider: the provider of the service. * paymentMethod: the way of payment. * Ex: /user/mobilerecharge/WE/cash   - Mobile Recharge service for the user  - Input: name, amount |
| The user should be logged in first and should have enough money to pay for the service. |  | 6-  -URL: /user/internetpaymment/{provider}/{paymentMethod}  Where:   * provider: the provider of the service. * paymentMethod: the way of payment. * Ex: /user/internetpaymment/We/cash   - Internet payment service for the user  - Input: name, amount. |
| The user should be logged in first and should have enough money to pay for the service. |  | 7-  -URL: /user/donation/{place}/{paymentMethod}  Where:   * place: the place user wants to donate * paymentMethod: the way of payment. * Ex: /user/donation/School/cash   - Donation service for the user  - Input: name, amount. |
| The user should be logged in first and should have enough money to pay for the service. |  | 8-  -URL: /user/landline/{provider}/{paymentMethod}  Where:   * provider: the provider of the service. * paymentMethod: the way of payment. * Ex: /user/landline/monthly/cash   - Landline service for the user  - Input: name, amount. |
| The user should be logged in first. |  | 9-  -URL:/user/viewDiscounts  - A service for the user to view the discounts for all services  - No input. |
| The user should be logged in first. |  | 10-  -URL: /user/viewTransactions  - A service for the user to view his transactions it returns all the user transactions.  - No input |
| The user should be logged in first and made any completed transactions before. |  | 11-  -URL: /user/makerefund/{index}  Where:   * index: is the id of the transaction needed to request a refund for it. * Ex: /user/makerefund/1   -A service for the user to request to refund a completed transaction.  - No input |
| The user should log in first and should have enough money in his credit card. |  | 12-  URL: /user/chargewallet/{amount}  Where:   * amount: the amount of money that the user wants to charge the wallet. * Ex: /user/chargewallet/12   -A service for users to charge their wallet in the system.  - No input |
| The System validates the account. |  | 13-  -URL: /admin/login  - A service for the admin to log in by his account  - Input: username, password. |
| The admin should be logged in first. |  | 14-  -URL: /admin/logout  - A service for the admin to log out from his account  - No input |
| The admin should log in first. |  | 15-  URL: /admin/ addSpecificDiscount/{value}  Where:   * value: is the value of the discount. * Ex: /admin/addSpecificDiscount/mobile/10   - A service for the admin to add a discount for a specific service.  - No input. |
| The admin should log in first. |  | 16-  URL: /admin/addOvarallDiscount/{value}  Where:   * value: is the value of the discount. * Ex: /admin/addOvarallDiscount/5   -A service for the admin to add an overall discount for all users.  - No input. |
| The admin should log in first. |  | 17-  -URL: /admin/ListRefunds  - A service for the admin to list all the refund requests.  - No input. |
| The admin should log in first. |  | 18-  -URL: /admin/refundAction/{index}/{state}  Where:   * Index: is the id of the refund request chosen * state: 1-> to accept.   : 2-> to reject.   * Ex: /admin/refundAction/1/1   -A service for the admin to respond for the refund requested by the users.  - No input. |
| The admin should log in first. |  | 19-  URL: /admin/viewHistory  -A service for the admin to view the history of all users transaction.  -No input. |

# GitHub repository link

<https://github.com/youssefdevo/Software-Project.git>