Cairo University Faculty of Computers and Artificial Intelligence



**Advanced SE**

**Software design specification document**

**2022**

**Project Team**

|  |  |  |
| --- | --- | --- |
| **ID** | **Name** | **Email** |
| 20200234 | سميه محمد يوسف محمد | [sooma3mohammed@gmail.com](mailto:sooma3mohammed@gmail.com) |
| 20200501 | محمود عبدالراضي جادالرب | [mahmoudten55@gmail.com](mailto:mahmoudten55@gmail.com) |
| 20201038 | ايمان ابراهيم شعبان جاد | [emanelkaser@gmail.com](mailto:emanelkaser@gmail.com) |
| 20201061 | دينا احمد عبدالراضي | [da449393@gmail.com](mailto:da449393@gmail.com) |

Contents

[Class diagram design 2](#_Toc120811427)

[Class diagram Explanation 3](#_Toc120811428)

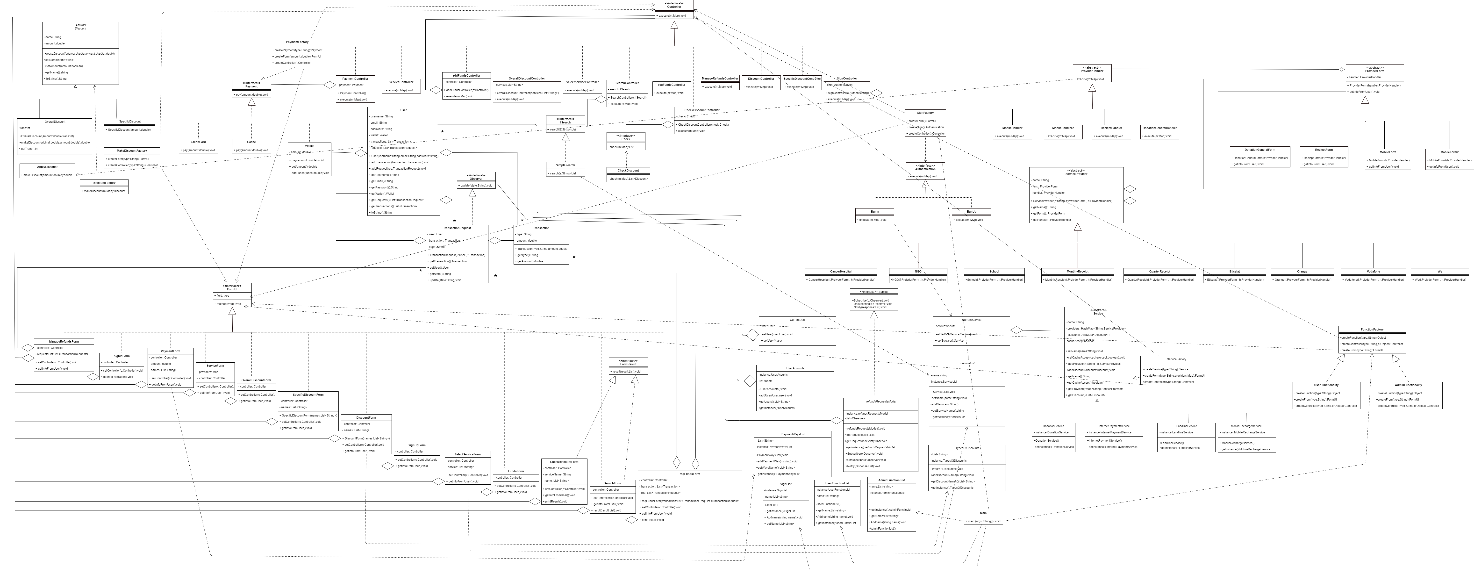
[Sequence diagram design 4](#_Toc120811429)

[Github repository link 9](#_Toc120811430)

**Read ME**

* To access admin functionality please use the following account: “email: admin, password: 123”
* If you want to enter to user account you can use sign up to make account or use the following account: “email: user, password: 123”

# Class diagram design

****

**To see class diagram clearly look at the picture or use drawio file in the folder, or use this link:**

[**https://drive.google.com/file/d/1UVrf2MQMMaj3R9SD0lvNix2\_BDkyh3gH/view?usp=share\_link**](https://drive.google.com/file/d/1UVrf2MQMMaj3R9SD0lvNix2_BDkyh3gH/view?usp=share_link)

# 

# Class diagram Explanation

* Singleton, we used it to manage a shared resources and to restrict the instantiation of our classes and ensures that only one instance of our classes exists.

Classes: UserAccounts, PaymentWaysList, ServiceList, SignList, refundsRequestsModel, UserFunctionList, AdminFunctionList, TypeOfDiscounts, DonationService, InternetPaymentService, LandlineService and MobileRechargeService.

* Factory, we used it to allow the sub-classes in our system to choose the type of objects to create which enable us to initiate our classes in a separate class.

Classes: SignFactory, PaymentFctory, MakeDiscountFactory, ServiceFactory, DiscountFactory, FunctionFactory, UserFunctionFactory and AdminFunctionFactory

* Observer, to make the transaction request subscribe to the refunds request model when the user wants to send it to the admin. When rejecting or accepting the request, the request state will be updated and then the request will unsubscribe from this model.

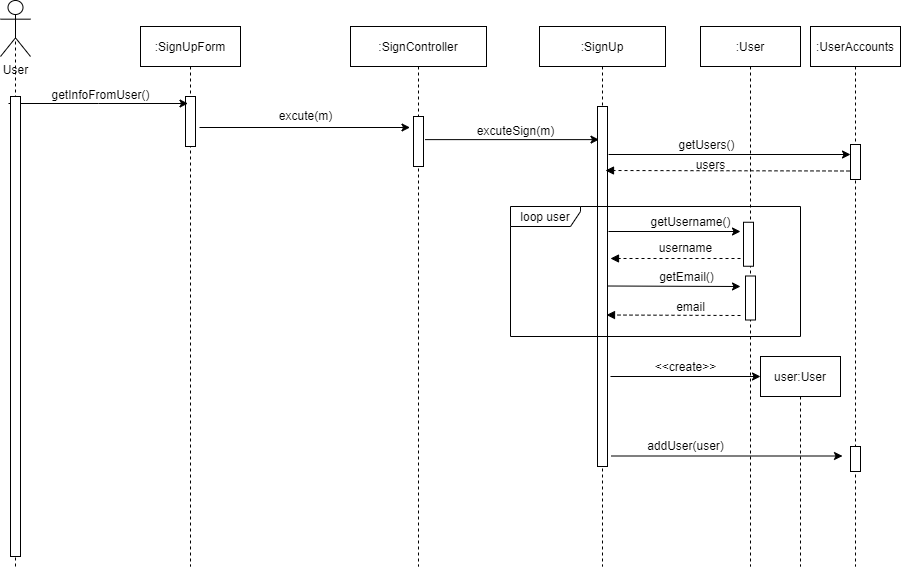
Classes: refundsRequestsModel and TransactionRequest

* Strategy pattern: This pattern enables us to select a specific algorithm at run time, so we used it because we have multiple algorithms for specific classes in our system

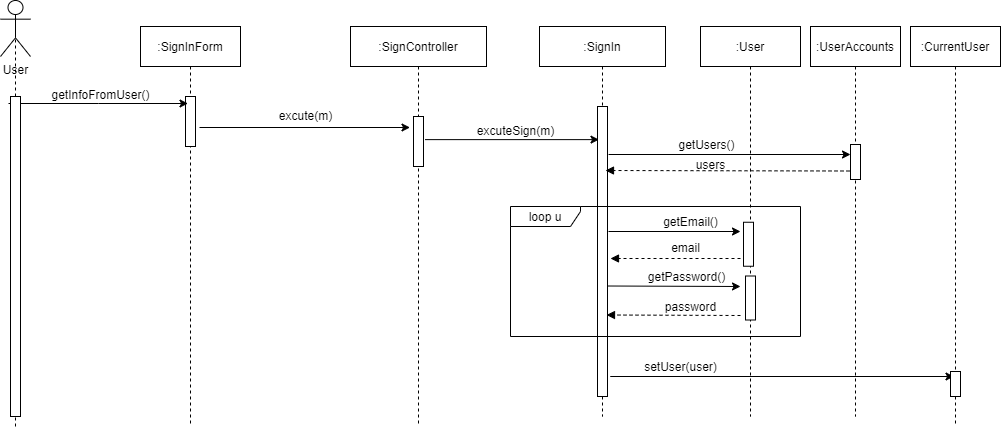
Classes (the name of abstract/interface classes): FormUI, Controller, Discount, Authentication, ISearch, Check and Payment

# Sequence diagram design

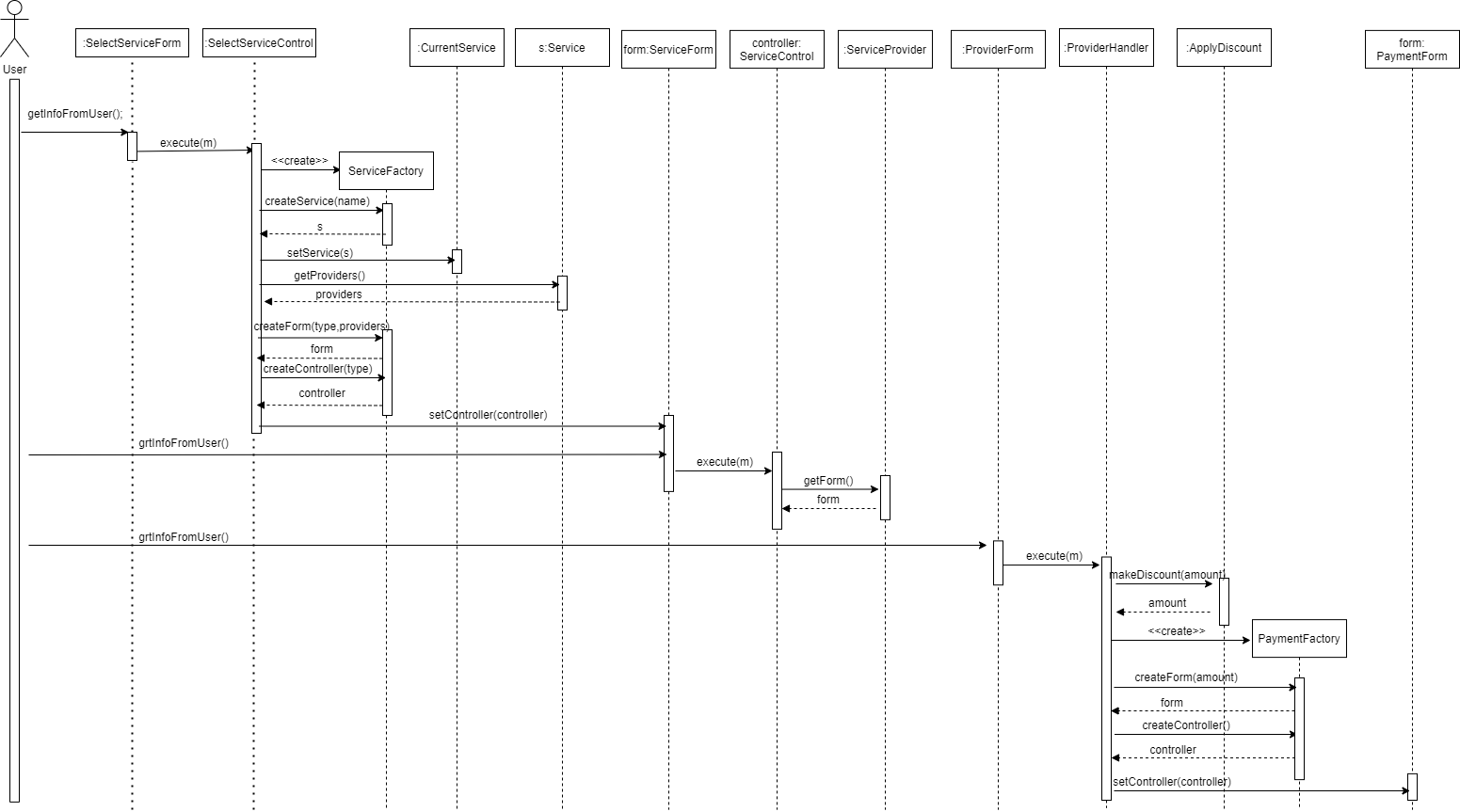
1. Sign up with no exception/error

****

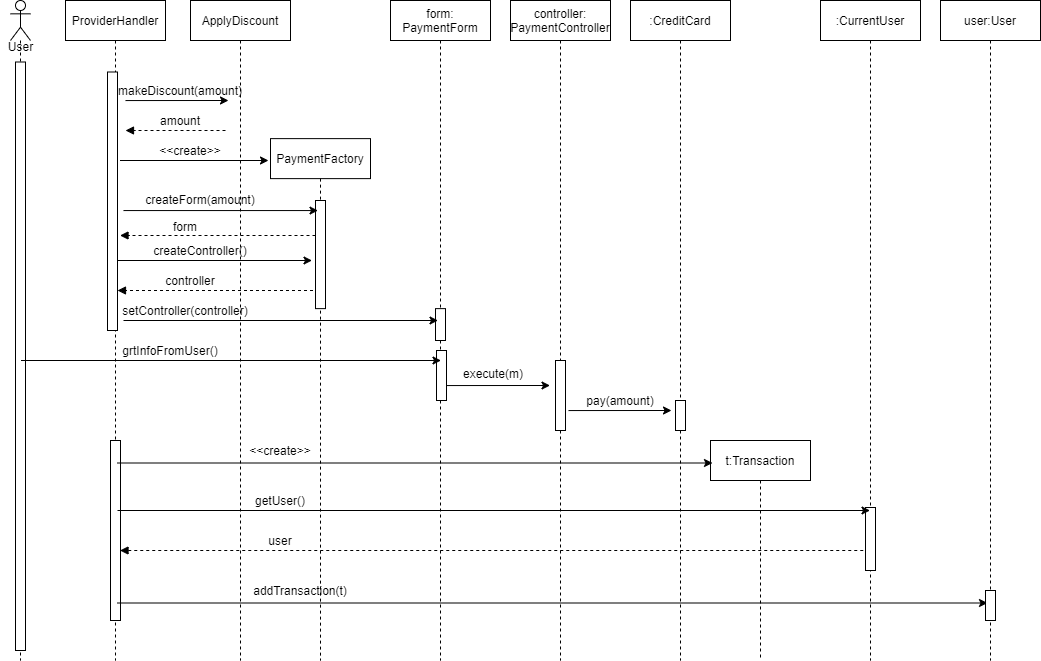
1. Sign in with no exception/error



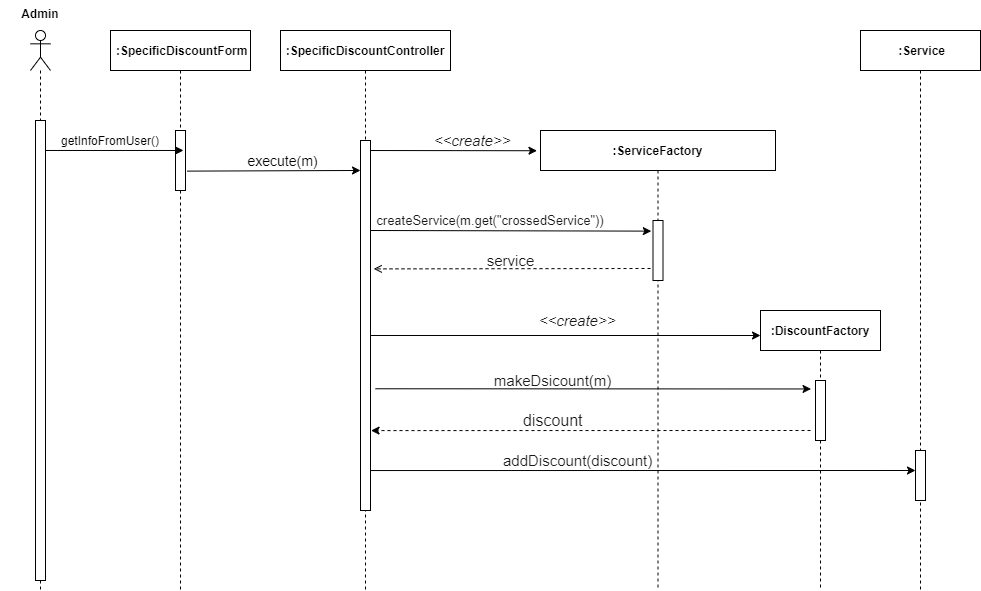
1. Select service to pay with no exception\error (Continue for payment in seq#4)



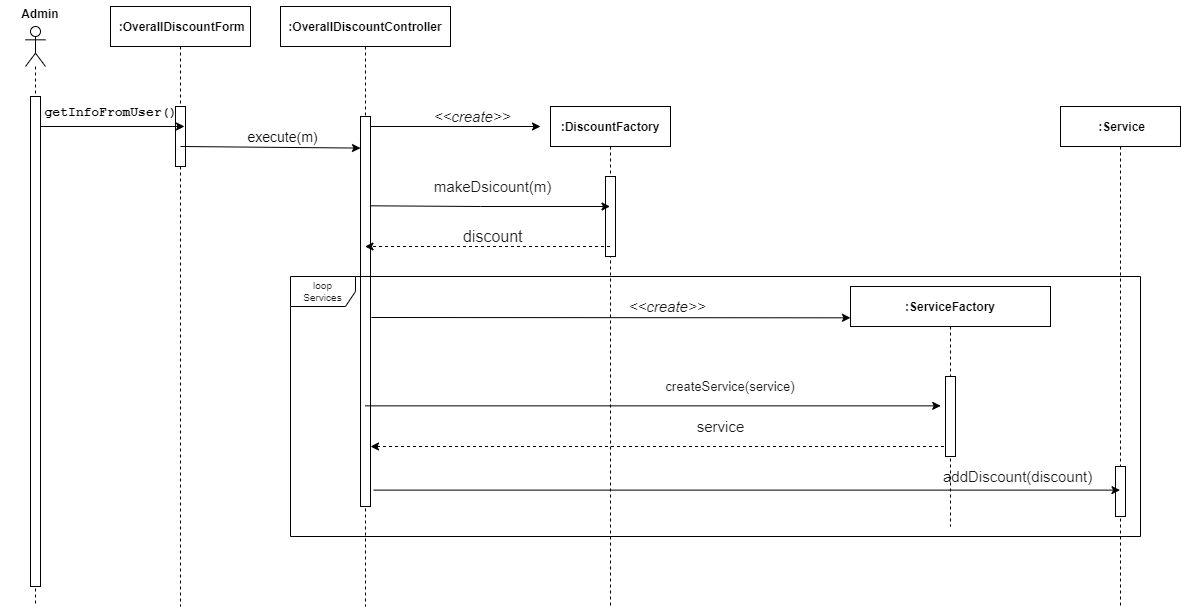
1. Payment for specific service by Credit Card with no exception\error (Part 1)



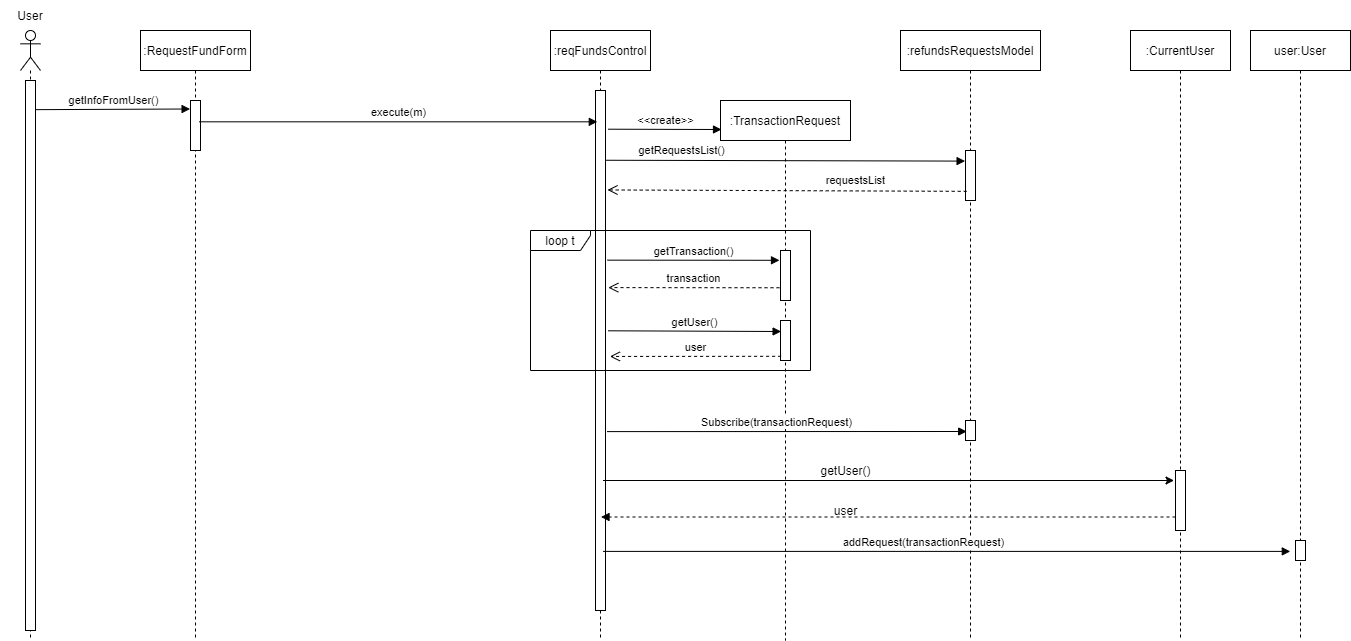
1. Make Specific Discount



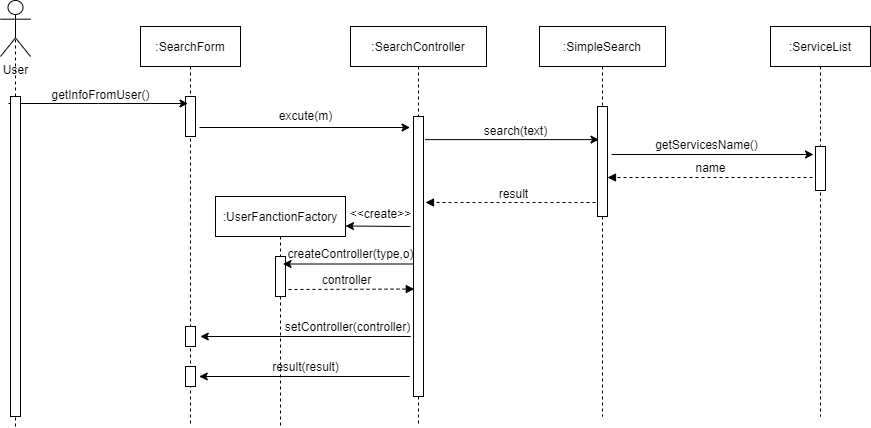
1. Make Overall Discount



1. User ask for a refund for any complete transaction



1. Search for any service in the system



# Github repository link

* <https://github.com/Sooma-M/PaymentSystem>
* If the link didn’t work, please contact with anyone of us via email or in our section