Cairo University **Faculty of Computers and Artificial Intelligence**

Software design specification document

2022

Project Team

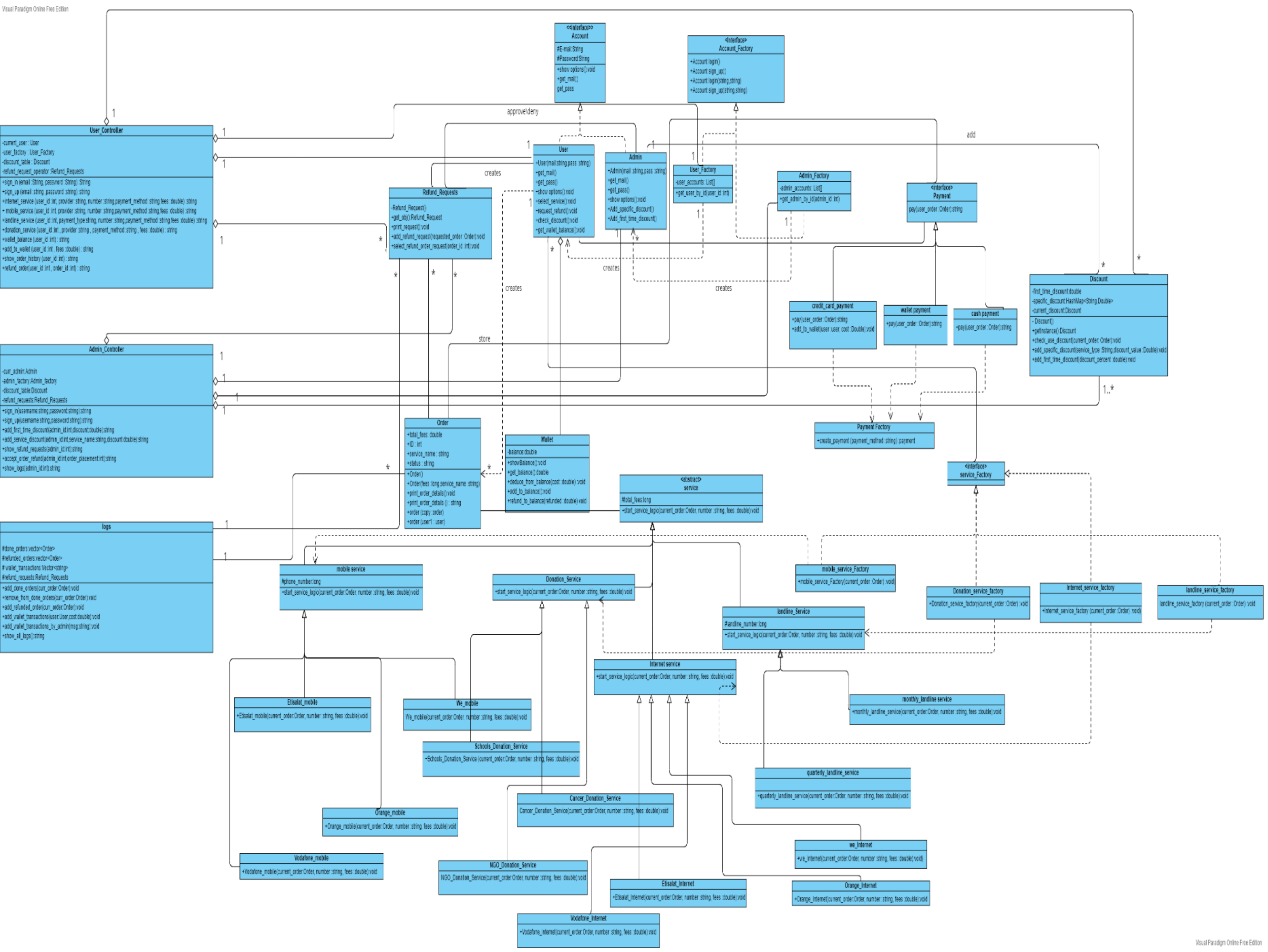
ID	Name	Email
20210612	Omar Hussien Ibrahim Ibrahim	omarhus98@gmail.com
20201152	Mohamed Ezzat Awad Badawy	moezza39@gmail.com
20200060	Ahmed Waleed Shawky	theassassin103@gmail.com
20200343	Omar ayman hussien salem	11410120200343@fci-cu.edu.eg

CS352: Sprint SDS– Team Name, Proj Name

SDS document

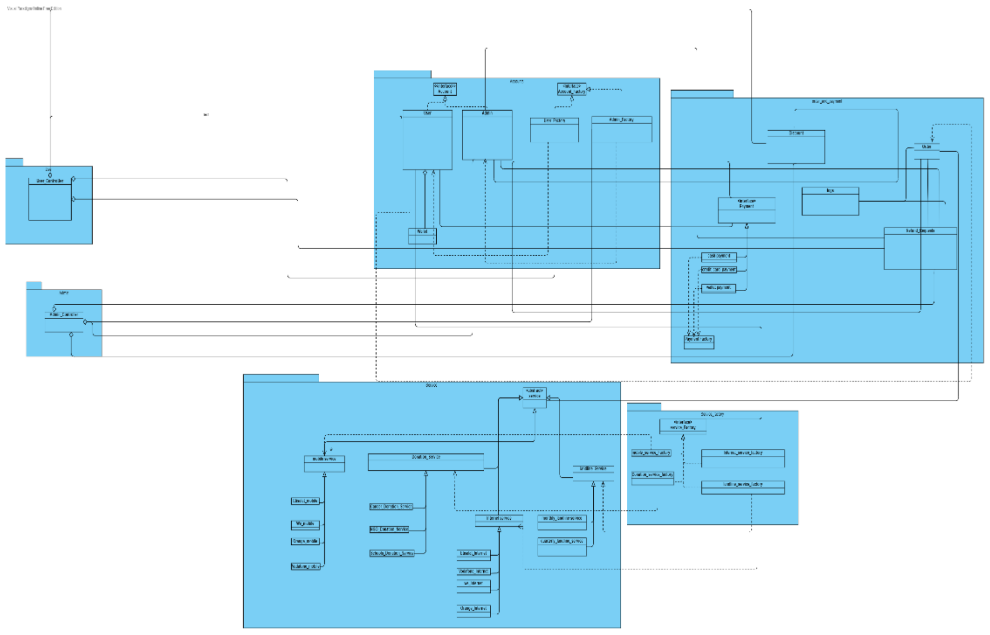
Contents

Github repository link : https://github.com/Ahmed-Waleed-01/Advanced_Software_Phase_2/tree/master



Visual Paradigm Online Free Edition
<<interface>> Account
#E-mail:String
#Password:String
+show options():void
+get_mail()
get_pass
<interface> Account_Factory
+Account login()
+Account sign_up()
+Account login(string,string)
+Account sign_up(string,string)
User_Controller
-current_user : User
-user_factory : User_Factory
-discount_table : Discount
-refund_request_operator :Refund_Requests
+sign_in (email :String, password :String) :String
+sign_up (email :string ,password :string) :string
+internet_service (user_id :int, provider :string, number :string,payment_method :string,fees :double) :string
+ mobile_service (user_id :int, provider :string, number :string,payment_method :string,fees :double) :string
+landline_service (user_id :int, payment_type:string, number :string,payment_method :string,fees :double) :string
+donation_service (user_id :int , provider :string , payment_method :string , fees :double) : string
+wallet_balance (user_id :int) : string
+add_to_wallet (user_id :int , fees :double) : string
+show_order_history (user_id :int) : string
+refund_order(user_id :int , order_id :int) : string
approve\deny
creates
store
add
Refund_Requests
-Refund_Request()
+get_obj():Refund_Request
+print_request():void
+add_refund_request(requested_order :Order):void
+select_refund_order_request(order_id :int):void
User
+User(mail:string,pass :string)
+get_mail()
+get_pass()
+show options():void
+select_service():void
+request_refund():void
+check_discount():void
+get_wallet_balance():void
Admin
+Admin(mail :string,pass :string)
+get_mail()
+get_pass()
+show options():void
+Add_specific_discount()
+Add_first_time_discount()
User_Factory
-user_accounts: List[]
+get_user_by_id(user_id :int)
Admin_Factory
-admin_accounts: List[]
+get_admin_by_id(admin_id :int)
<interface> Payment
pay(user_order :Order):string
credit_card_payment
+pay(user_order :Order):string
+add_to_wallet(user :user, cost :Double):void
wallet payment
+pay(user_order :Order):string
cash payment
+pay(user_order :Order):string
Discount
-first_time_discount:double
-specific_discount:HashMap<String,Double>
-current_discount:Discount
- Discount()
+getInstance():Discount
+check_use_discount(current_order: Order):void
+add_specific_discount(service_type :String,discount_value :Double):void
+add_first_time_discount(discount_percent :double):void
Payment Factory
+create_payment (payment_method :string) :payment
<interface> service_Factory
Admin_Controller
-curr_admin:Admin
-admin_factory:Admin_factory
-discount_table:Discount
-refund_requests:Refund_Requests
+sign_in(username:string,password:string):string
+sign_up(username:string,password:string):string
+add_first_time_discount(admin_id:int,discount:double):string
+add_service_discount(admin_id:int,service_name:string,discount:double):string
+show_refund_requests(admin_id:int):string
+accept_order_refund(admin_id:int,order_placement:int):string
+show_logs(admin_id:int):string
logs
#done_orders:vector<Order>
#refunded_orders:vector<Order>
# wallet_transactions:Vector<string>
#refund_requests:Refund_Requests
+add_done_orders(curr_order:Order):void
+remove_from_done_orders(curr_order:Order):void
+add_refunded_order(curr_order:Order):void
+add_wallet_transactions(user:User,cost:double):void
+add_wallet_transactions_by_admin(msg:string):void
+show_all_logs():string
Order
+total_fees: double
+ID : int
+service_name : string
+status : string
+Order()
+Order(fees :long,service_name :string)
+print_order_details():void
+print_order_details () : string
+order (copy :order)
+order (user1 : user)
Wallet
-balance:double
+showBalance():void
+get_balance():double
+deduce_from_balance(cost :double) :void
+add_to_balance():void
+refund_to_balance(refunded :double):void
<abstract> service
#total_fees:long
+start_service_logic(current_order:Order, number :string, fees :double):void
mobile service
#phone_number:long
+start_service_logic(current_order:Order, number :string, fees :double):void
Donation_Service
+start_service_logic(current_order:Order, number :string, fees :double):void
mobile_service_Factory
+mobile_service_Factory(current_order :Order) :void)
Donation_service_factory
+Donation_service_factory(current_order :Order) :void
Internet_service_factory
+Internet_service_factory (current_order :Order) :void)
landline_service_factory
landline_service_factory (current_order :Order) :void
landline_Service
#landline_number:long
+start_service_logic(current_order:Order, number :string, fees :double):void
Internet service
+start_service_logic(current_order:Order, number :string, fees :double):void
monthly_landline service
+monthly_landline_service(current_order:Order, number :string, fees :double):void
Etisalat_mobile
+Etisalat_mobile(current_order:Order, number :string, fees :double):void
We_mobile
We_mobile(current_order:Order, number :string, fees :double):void
Schools_Donation_Service
+Schools_Donation_Service (current_order:Order, number :string, fees :double):void
quarterly_landline_service
+quarterly_landline_service(current_order:Order, number :string, fees :double):void
Orange_mobile
+Orange_mobile(current_order:Order, number :string, fees :double):void
Cancer_Donation_Service
Cancer_Donation_Service(current_order:Order, number :string, fees :double):void
we_Internet
+we_Internet(current_order:Order, number :string, fees :double):void)
Vodafone_mobile
+Vodafone_mobile(current_order:Order, number :string, fees :double):void
NGO_Donation_Service
NGO_Donation_Service(current_order:Order, number :string, fees :double):void
Etisalat_Internet
+Etisalat_Internet(current_order:Order, number :string, fees :double):void
Orange_Internet
+Orange_Internet(current_order:Order, number :string, fees :double):void
Vodafone_Internet
+Vodafone_internet(current_order:Order, number :string, fees :double):void
Visual Paradigm Online Free Edition

SDS document



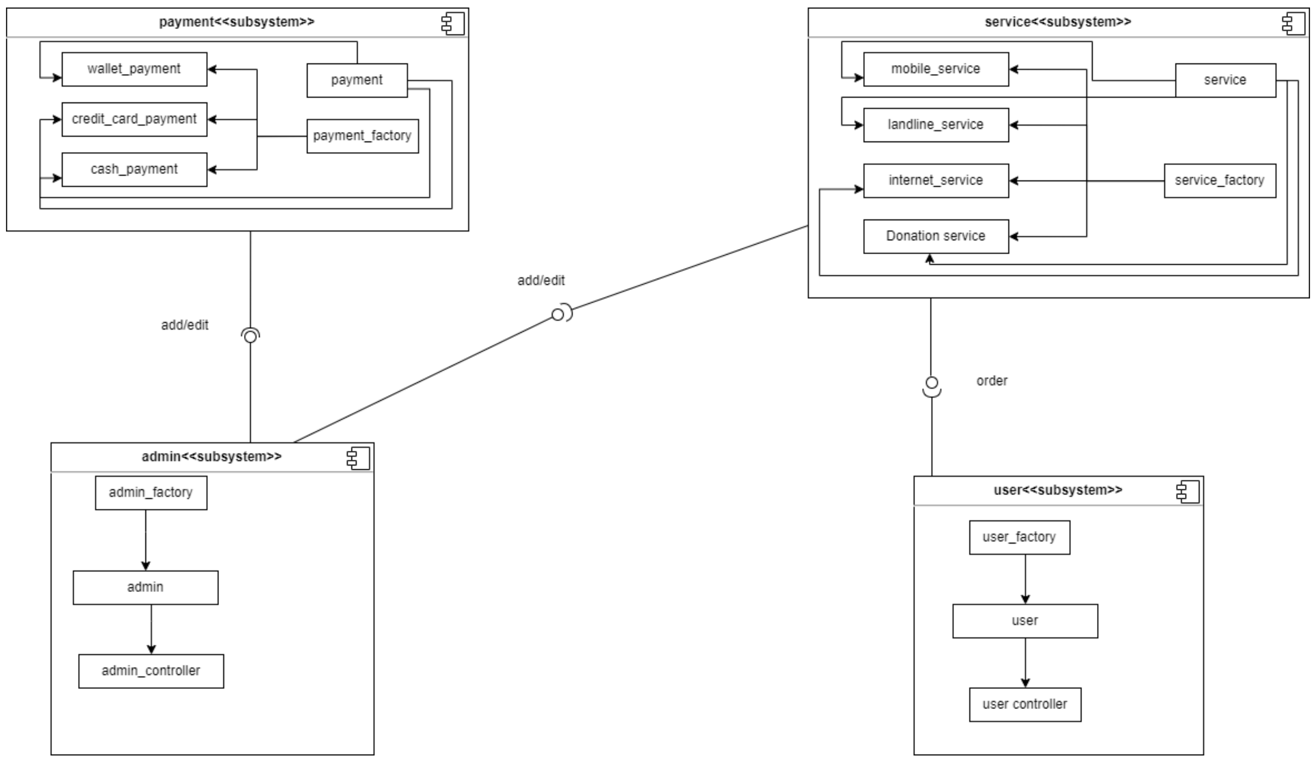
Note: class diagrams here are not clear so you should open it as an image from the project file, I provide two versions of class diagram one before packaging and one after packaging

SDS document

Class diagram Explanation

1) First things first, we used "strategy" design pattern as shown in "payment" class as it reflects different ways of paying for the service. This will allow to provide new payment methods without updating existing class, in another words, saving OCP
2) moving forward, we used "factory" design pattern in order to be able to create a new object every time we desire to do (e.g whenever a user comes to request a service, a new service is created with its handler to manage the request). It also provides the flexibility for the modification of the system without interrupting the previously working classes, which brings us back to preserving the OCP again
3) Finally, we used "single-tone" The Singleton's purpose is to control object creation, limiting the number to one but allowing the flexibility to create more objects if the situation changes. Since there is only one Singleton instance, any instance fields of a Singleton will occur only once per class, just like static fields. As shown in the provided class diagram, we can see that it's used in the following classes: logs, refund requests

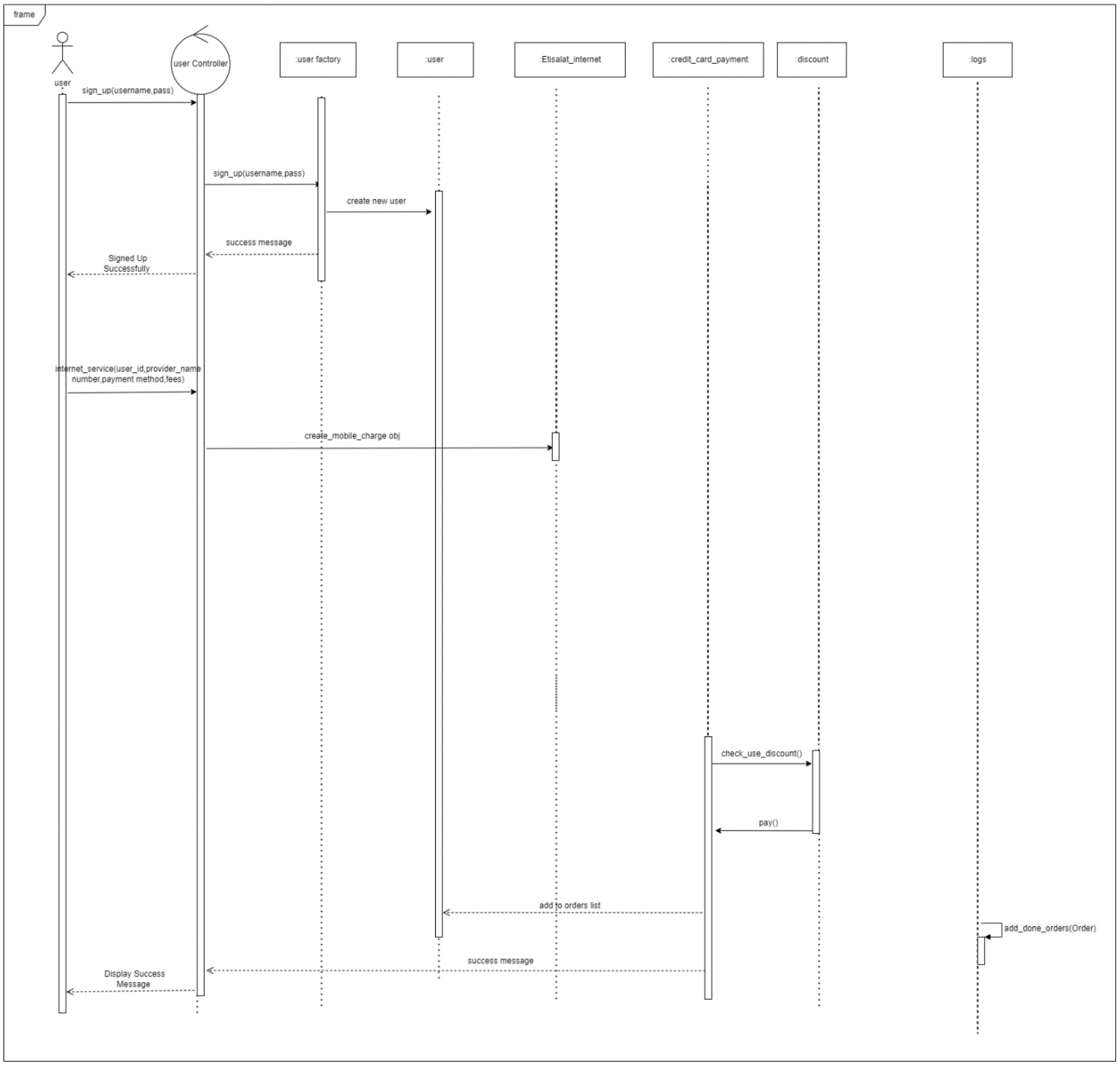
Decomposition diagram:



SDS document

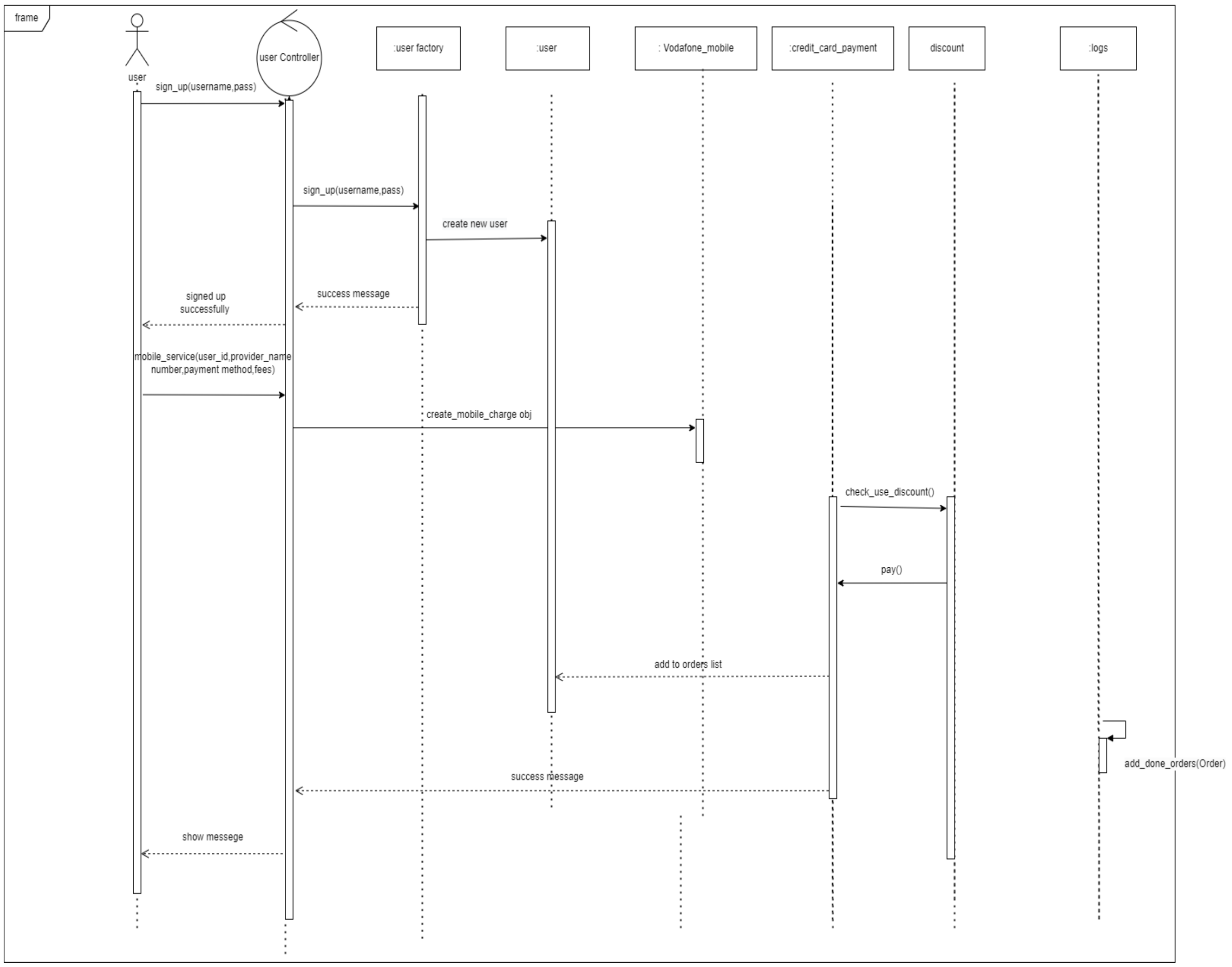
Sequence diagram design

Sequence digaram for booking internet service from etisalat (new_user)



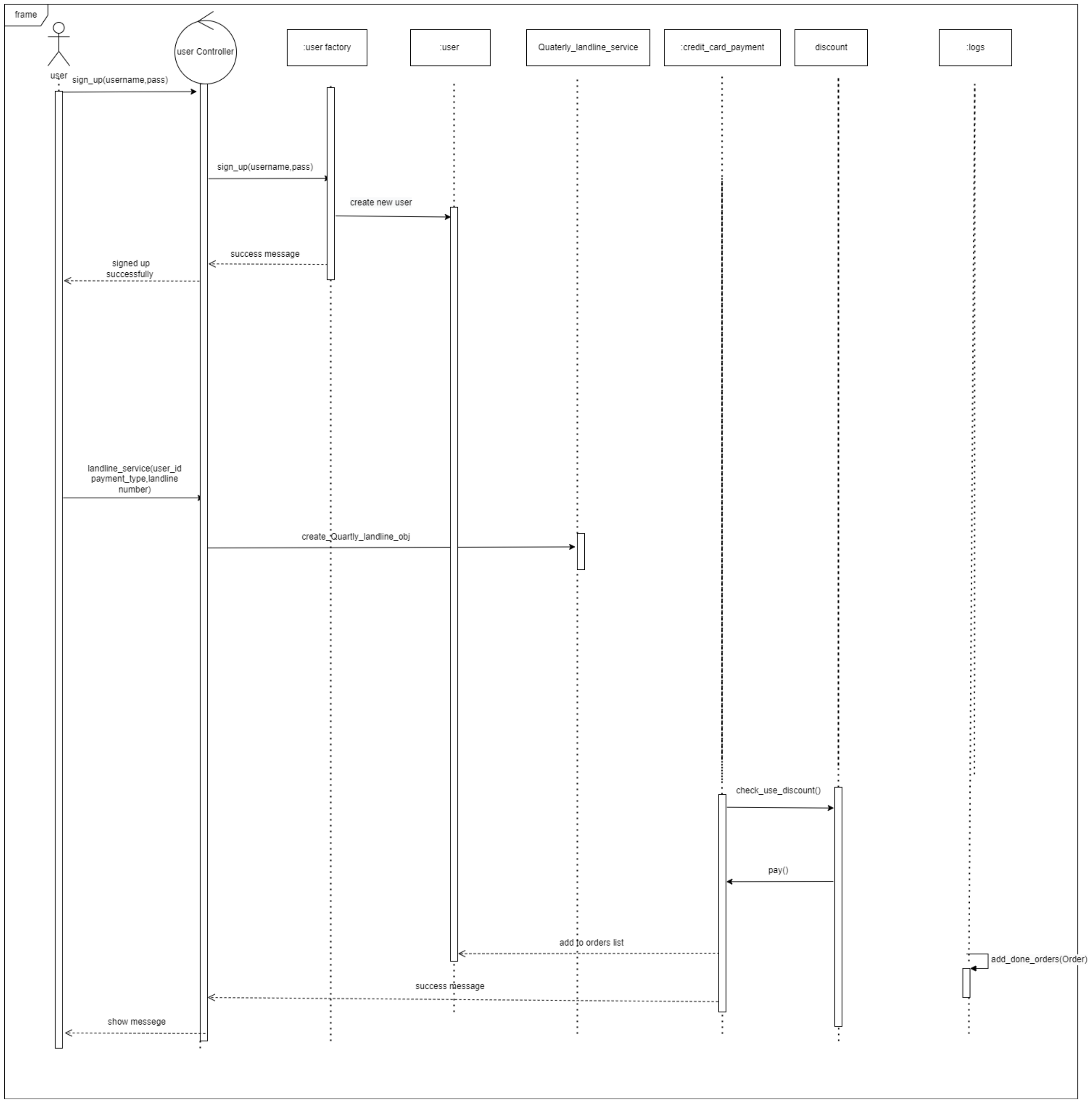
SDS document

Sequence digaram for boooking mobile recharge service from Vodafone (new_user)



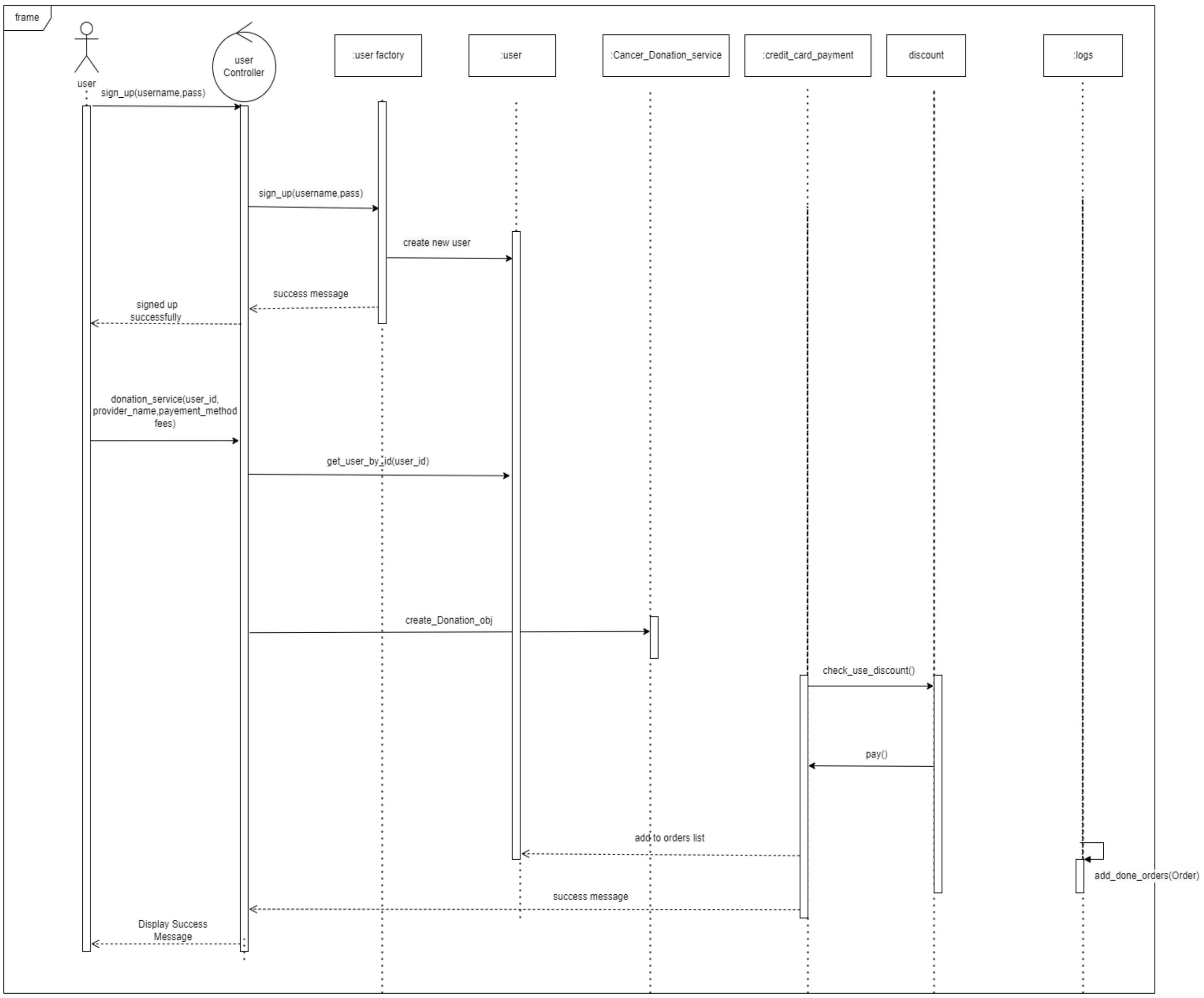
SDS document

Sequence digaram for booking Quartly landline service (new_user)



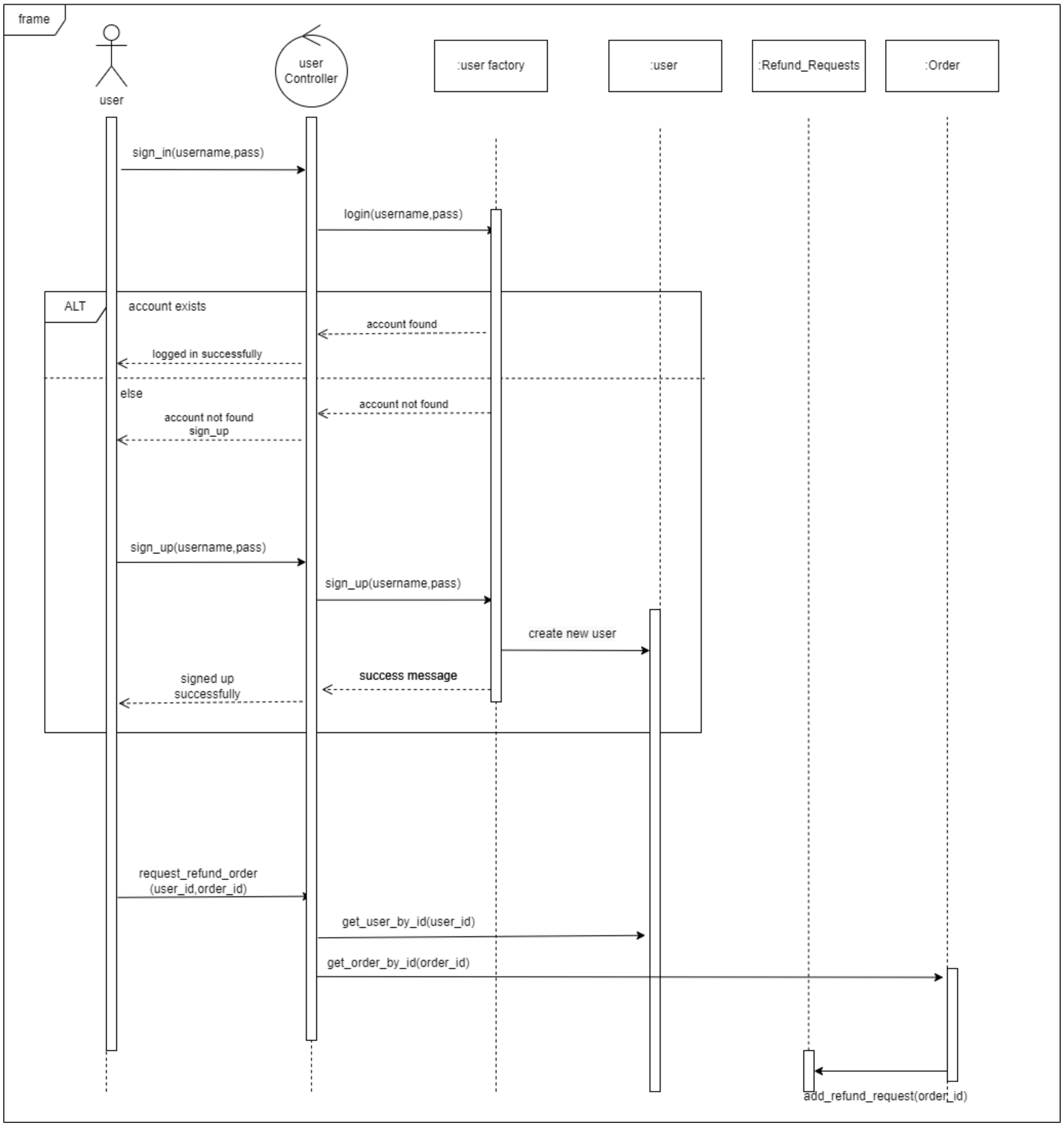
SDS document

Sequence digaram for school donation service (new_user)



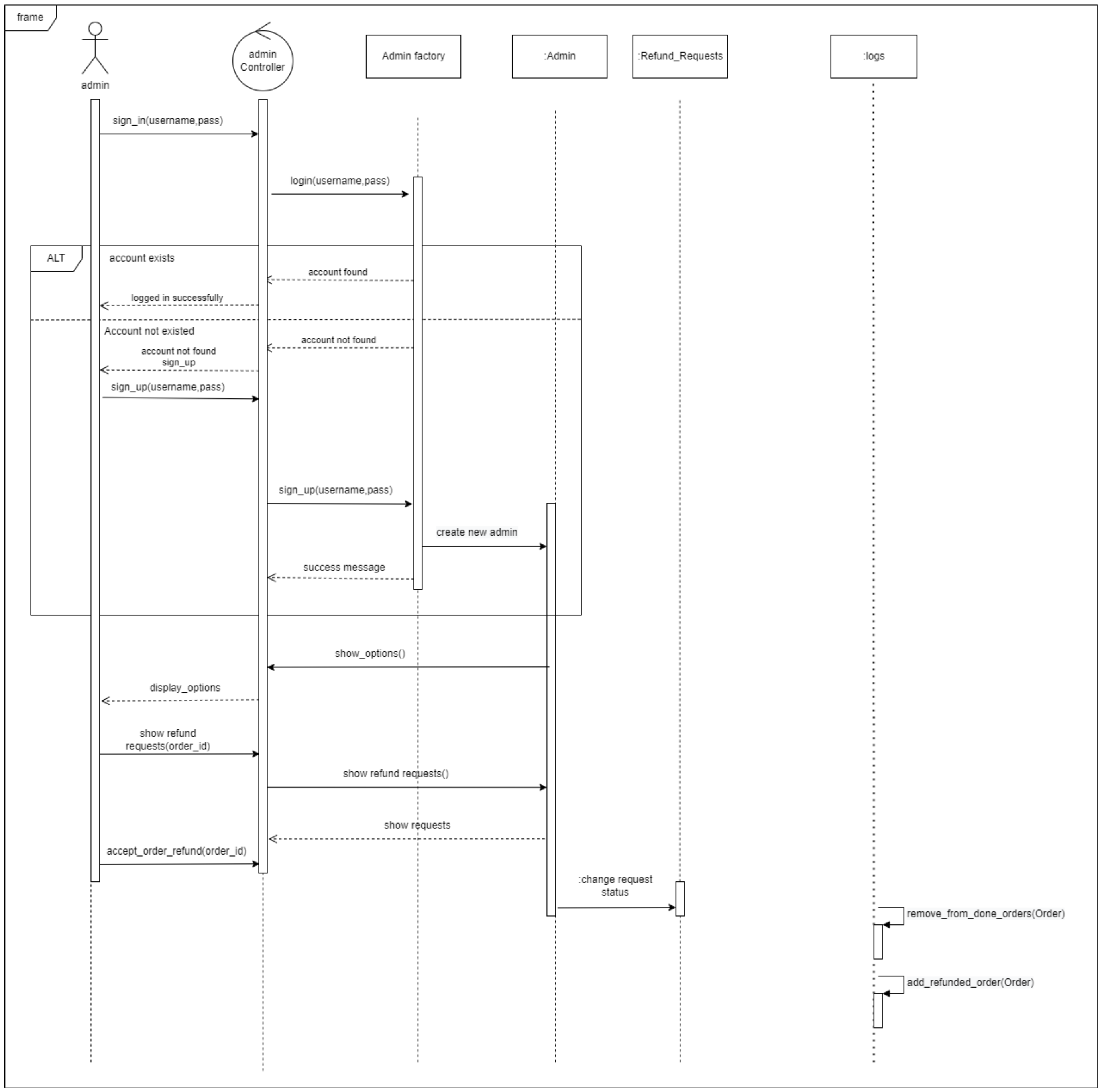
SDS document

Sequence digaram for a user requesting refund for an order



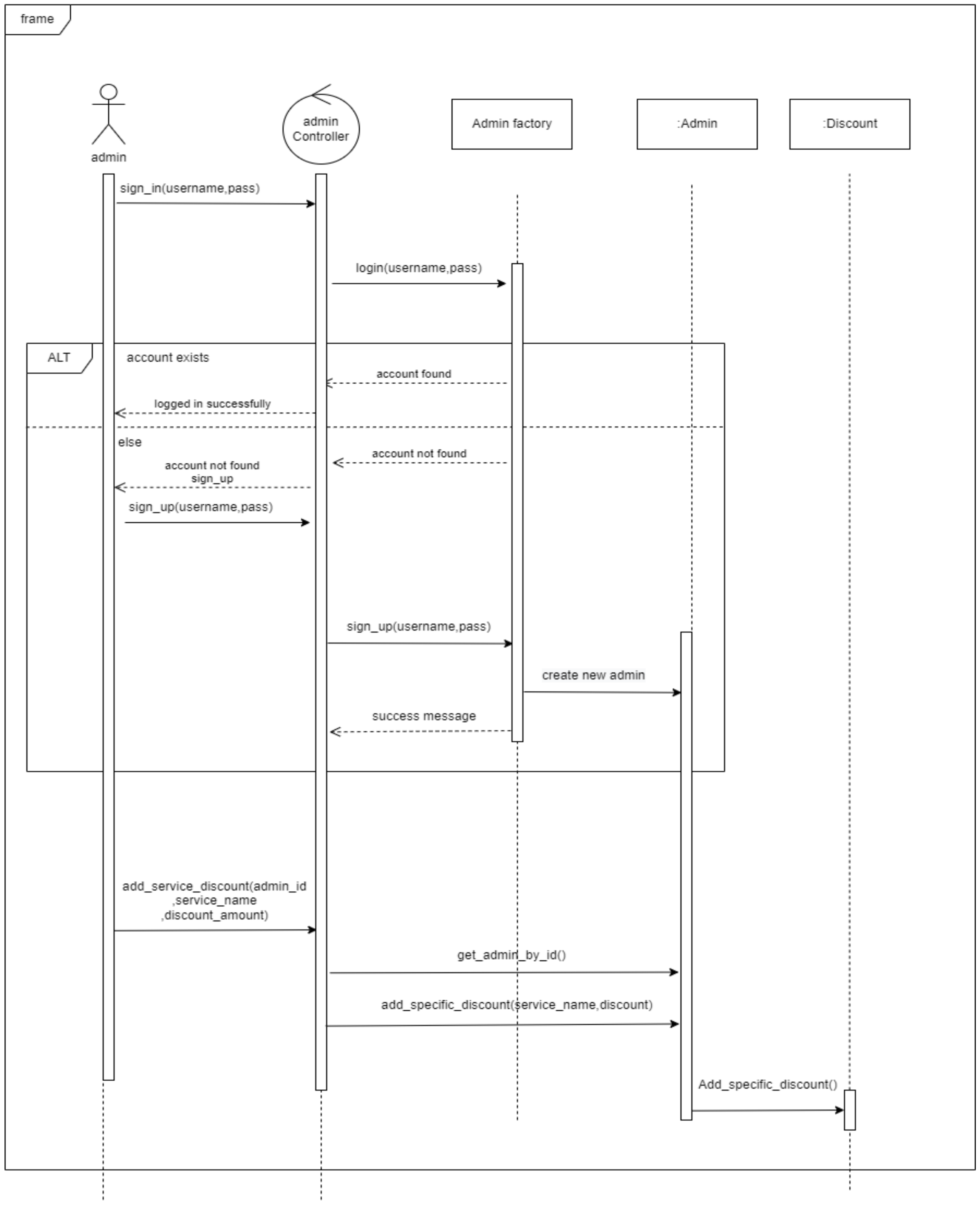
SDS document

Sequence digaram for a admin approving refund for an order



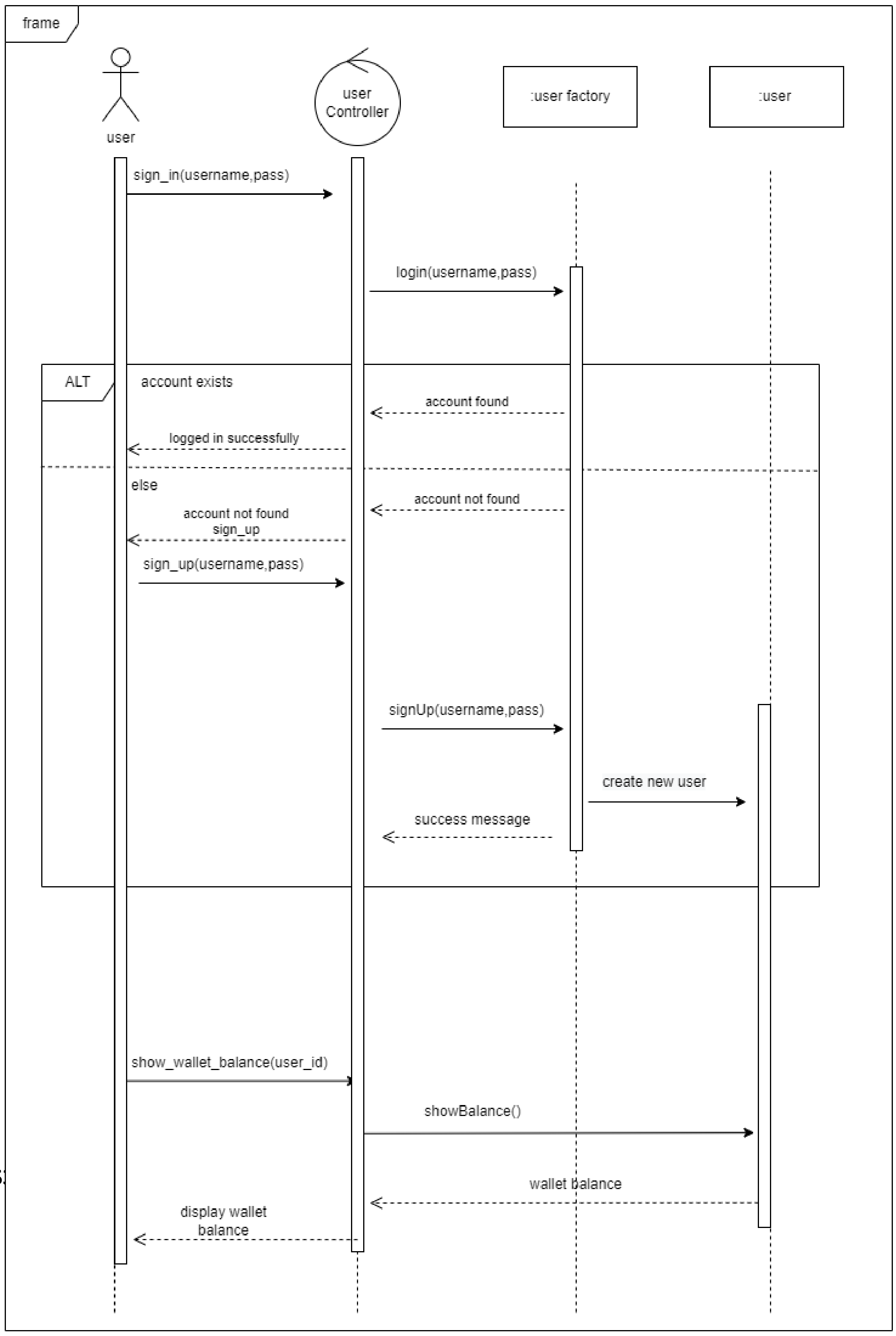
SDS document

Sequence digaram for a admin adding specific discount to specific service



SDS document

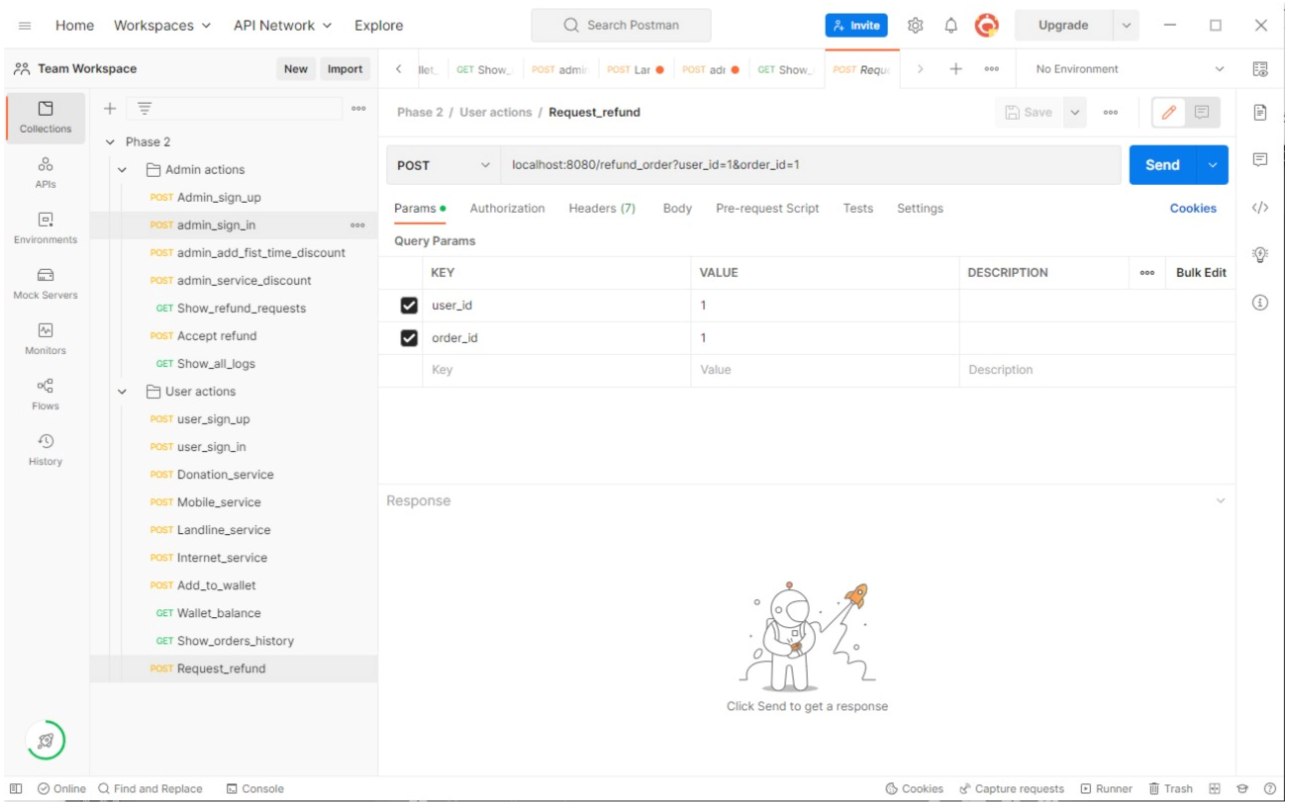
Sequence digaram for a user showing the balance of his wallet



SDS document

Requirements Exposure as Web Service API

Part 1: Exposed Postman Collection



POST localhost:8080/refund_order?user_id=1&order_id=1

	KEY	VALUE	DESCRIPTION
✓	user_id	1	
✓	order_id	1	
	Key	Value	Description

Link:

https://martian-star-992501.postman.co/workspace/Team-Workspace~f8e02b87-d796-4383-8440-ac65da6fee57/collection/25101763-713d161e-4e23-43ba-8239-b4b85ab26d95?action=share&creator=25101763

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 system should check if the username or the email is registered before.	Post/ localhost:8080/admin_sign_up?email=bro5@mail&password=24ufaz The service checks if the user exists or not and the service returns a message that describes the sign up attempt if it was successful or unsuccessfully
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	Post/ localhost:8080/admin_sign_in?email=bro5@mail&password=24ufaz The API takes e-mail and password as parameters in url query The service return if the user exists or not in a string return type.
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. The default way is to pay via credit card.	Post/ localhost:8080/donation_service?user_id=1&provider=schools&payment_method=cash&fees=500 localhost:8080/mobile_service?user_id=1&provider=etisalat&number=01235678&payment_method=wallet&fees=80 localhost:8080/landline_service?user_id=1&payment_type=quarter&number=01235678&payment_method=cash&fees=500 localhost:8080/internet_service?user_id=1&provider=vodafone&number=01235678&payment_method=cash&fees=500 this service API takes user ID and service provider as well as payment method and payment amount and takes phone number if needed in different services.

	it returns a string message that describes the operation and it's status.
The user can ask for a refund for any complete transaction to any given service.	Post localhost:8080/refund_order?user_id=1&order_id=1 API takes user id and order id as query parameters and returns a message that states if the order refund request is complete.
The user should be able to add any funds to the wallet. Adding funds to the wallet should be done via credit card.	POST localhost:8080/add_to_wallet?user_id=1&fees=87 API takes user id and fees as query parameters It returns a message that states the request status.
The user should be able to check funds in the wallet	GET localhost:8080/wallet_balance?user_id=1 API takes user id as query parameters It returns the user's wallet balance
The admin should be able to add discounts to the system ex: first time orders.	POST localhost:8080/add_first_time_discount?admin_id=1&discount=22 API takes admin id and discount percentage and returns message that show that discount now is applied for first time discounts.
The admin should be able to add discounts to the system ex:Specific discount to service	POST localhost:8080/add_service_discount?admin_id=1&service=Mobile&discount=20 API takes admin id and service name that the discount will be applied to and discount percentage it returns a string message with the status if it succeeds or not.
The admin should be able to list all user transactions	GET localhost:8080/show_all_logs?admin_id=1 API takes admin id and then it returns a string message that lists all different transactions.
The admin should be able to list all refund requests.	GET localhost:8080/show_refund_requests?admin_id=1 API takes admin id and then it returns a string message that lists all refund requests.
he admin should be able to accept or reject any refund request	POST localhost:8080/accept_order_refund?admin_id=1&order_placement=1 API takes admin id and order placement in refund requests.

SDS document

	then it returns a string message that contains order fees as it was added to user's wallet.

Github repository link

- **https://github.com/Ahmed-Waleed-01/Advanced_Software_Phase_2**