MedShoppe

Pharmacy Management System App

I. Purpose

Many people don't know if the medicine is available in pharmacies or not, so they waste their time and effort trying to reach many pharmacies to ask about the specific medicine. The goal of this project is to create a mobile app, which we called "MedShoppe", that provides a system that interconnect many pharmacies and list all available medicines within each pharmacy to facilitate searching for all medicines.

II. Requirements

1) User Requirements:

- **Functional user requirements:**
  1. The pharmacy app must provide authentication for different users (customer, pharmacist, admin).
  2. The pharmacy app customer can make order.
  3. The pharmacy app customer/pharmacist can edit their profiles.
  4. The pharmacy app admin can display/manage pharmacies.
  5. The pharmacy app admin can create/edit pharmacist.
  6. The pharmacy app pharmacist can manage pharmacy's medicines.
  7. The pharmacy app must provide search algorithm.
  8. The pharmacy app can display pharmacies that own a certain medicine.

- **Non-functional requirements:**
  1. Passwords must be secured (Security)
  2. Easy to use the app (Understandability)
  3. Adapt to changes (Maintainability)

2) System Requirements:

- **Functional system requirements:**
  1. The pharmacy app should have login and register screen.

  2.1. The pharmacy app customer can search for required medicine.

  2.2. The pharmacy app customer can add/remove an item into a cart.

  3.1. The pharmacy app customer/pharmacist can change their name, password and phone.

3.2. The pharmacy app customer/pharmacist must update database with these modifications.

4.1. The pharmacy app admin can add pharmacies.

4.2. The pharmacy app admin can remove pharmacies.

4.3. The pharmacy app admin can edit pharmacies.

5.1. The pharmacy app admin can add pharmacist.

5.2. The pharmacy app admin can remove pharmacist.

5.3. The pharmacy app admin can edit pharmacist.

6.1. The pharmacy app pharmacist can increase quantity of certain medicines.

6.2. The pharmacy app pharmacist can decrease quantity of certain medicines.

7.1 The pharmacy app customer can search for an item.

7.2. The pharmacy app customer can search for a pharmacy.

7.3. The pharmacy app admin can search for a pharmacy.

7.4. The pharmacy app admin can search for a pharmacist.

7.5. The pharmacy app can suggest results upon typing in search.

8. When customer select an item the app shows all pharmacies that have this item

- **Non-functional:**

  1. Passwords authentication is handled by the Firebase (Confidentiality).

  2. User-friendly GUI.

  3. develop using agile.

  4. Reusability when components usually used.

III. Software Process

1) Suggested type of software process:

Agile approach (incremental delivery).

2) Division of Phases:

1. Login and Registration pages.

2. Customer page.
   - 2.1. Home Screen in the Nav Bar.
     - 2.1.1. Display available pharmacies and medicines.
   - 2.2. Pharmacies Screen in the Nav Bar.
     - 2.2.1. Display available pharmacies.
   - 2.3. Profile Screen in the Nav Bar.
     - 2.3.1. Update personal data.
     - 2.3.2. Sign Out.
   - 2.4. Search Screen
     - 2.4.1. Customer can search for a medicine or a pharmacy by its name and the system provides suggestions upon typing in the search bar.
   - 2.5. Display pharmacies selling a certain medicine when a customer chooses that medicine.
   - 2.6. Pharmacy Screen (when customer selects a pharmacy from which he/she can make order)
     - 2.6.1. Display medicines in the selected pharmacy with their prices and quantities.
     - 2.6.2. Add selected items to cart.
     - 2.6.3. Remove from cart.
   - 2.7. Order Screen
     - 2.7.1. Show ordered items in cart and display the total price.
   - 2.8. Checkout Screen
     - 2.8.1. Get delivery date and address from the customer.

3. Admin page.
   - 3.1. Pharmacists Screen in the Nav Bar.
     - 3.1.1. Display pharmacists.
     - 3.1.2. Edit pharmacist.
     - 3.1.3. Add new pharmacist.
     - 3.1.4. Delete pharmacist.

        3.1.5. Search for pharmacy by its name with the aid of system suggestions upon typing in the search bar.

    3.2. Pharmacies Screen in the Nav Bar.
        3.2.1. Display pharmacies.
        3.2.2. Edit pharmacy.
        3.2.3. Add new pharmacy.
        3.2.4. Delete pharmacy.
        3.2.5. Search for pharmacist by his/her name with the aid of system suggestions upon typing in the search bar.

    3.3. Sign Out in the Nav Bar.

4. Pharmacist page.
    4.1. Pharmacy Medicines Screen in the Nav Bar.
        4.1.1. Display medicines in the pharmacy in addition to their prices and quantities.
        4.1.2. Edit medicine.
        4.1.3. Add new medicine.

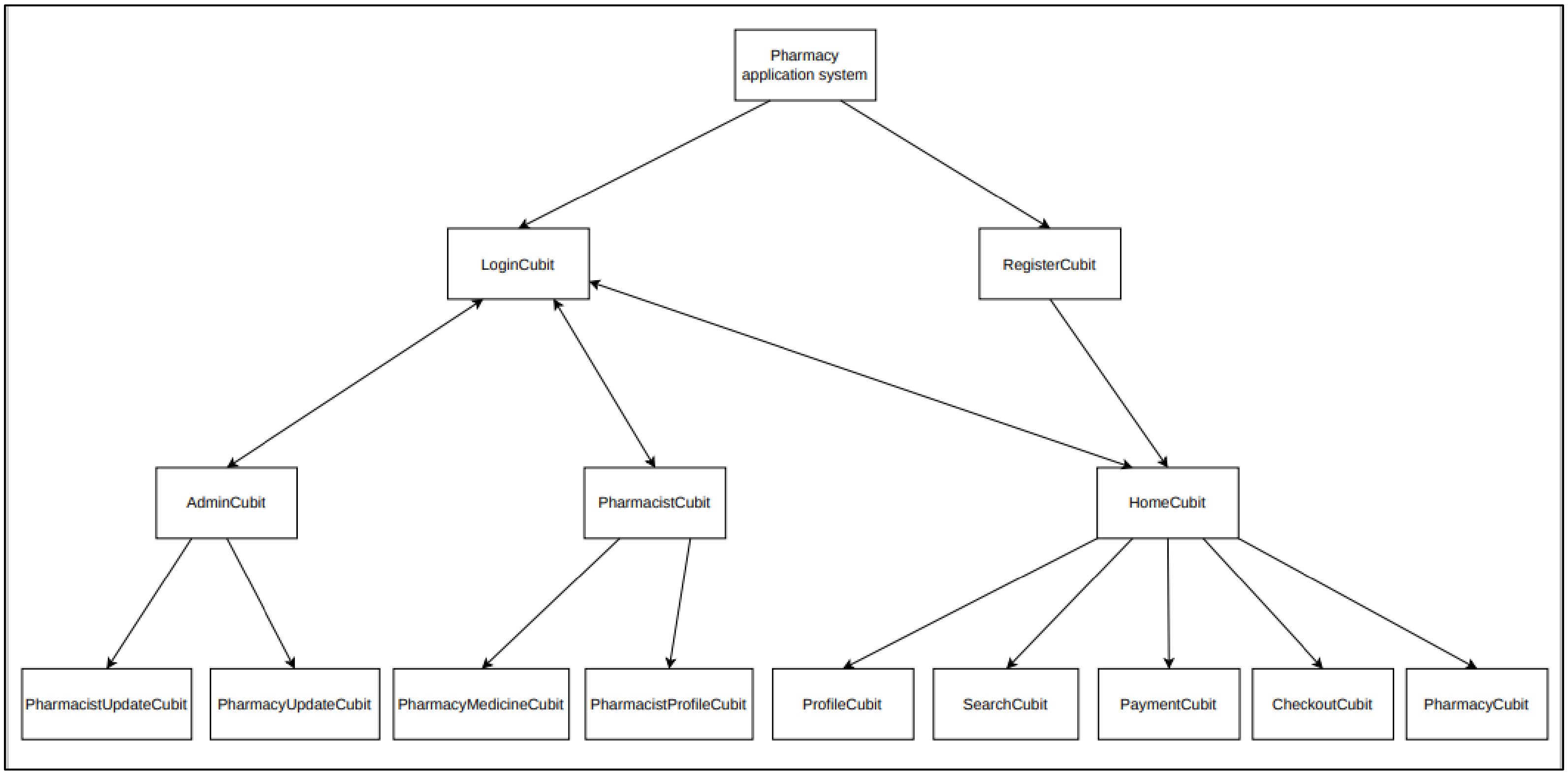
    4.2. Orders Screen in the Nav Bar.
        4.2.1. Display orders made by customers.
        4.2.2. Accept order.

    4.3. Profile Screen in the Nav Bar.
        4.3.1. Update personal data.
        4.3.2. Sign Out.

IV. Architectural Design

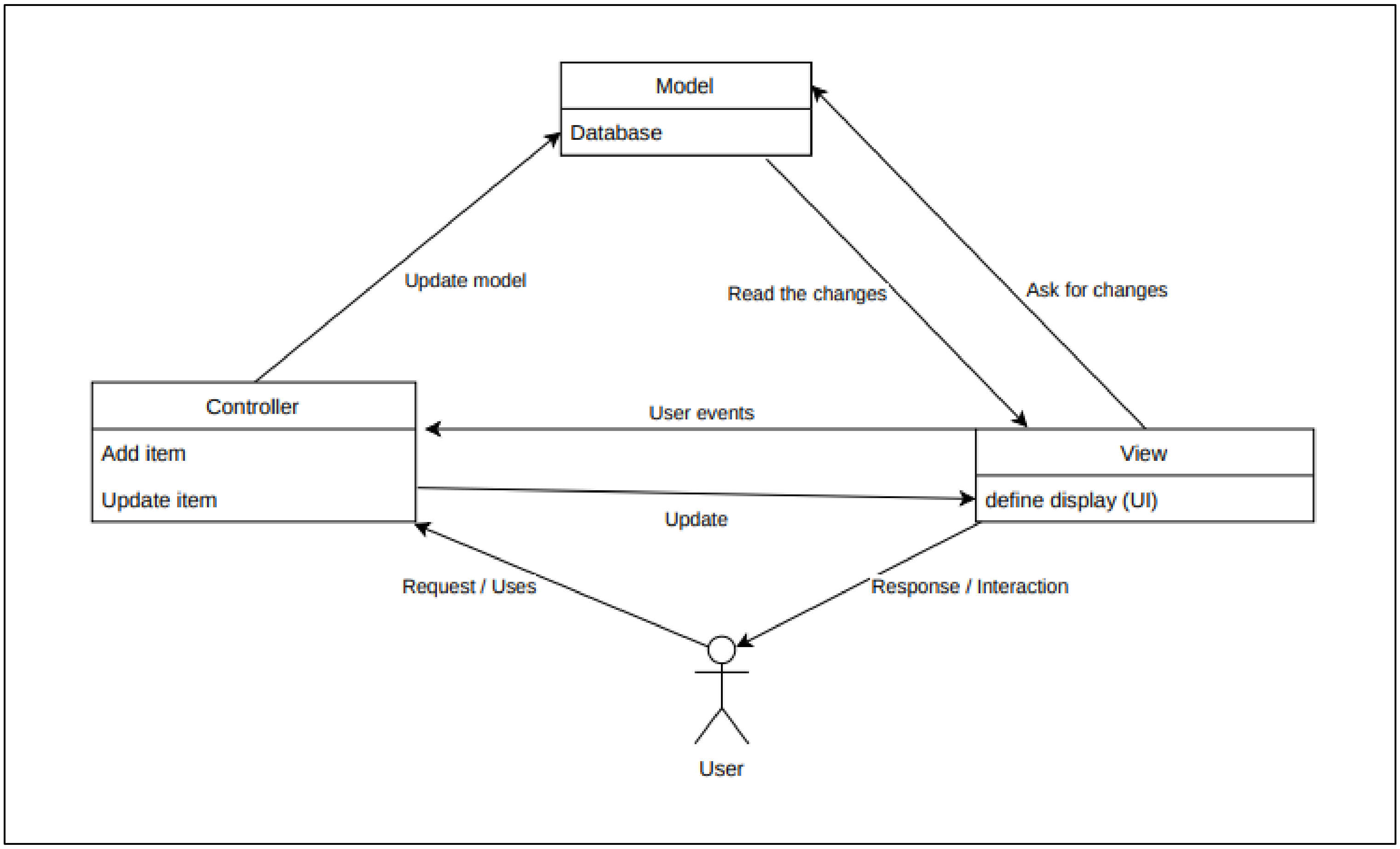
1) Suggested System Architecture:

- Block Diagram.



2) <u>Suggested Application Architecture:</u>

- MVC (Model – View - Controller) architecture pattern.



V. Backlog

- **ToDo List** (priority is listed from highest to lowest, where highest priority = 1)

Story	Estimation	Priority
As a user, I want to be able to login and register.	3	1
As a customer, I want to be able to display available medicines and pharmacies.	1	2
As a customer, I want to be able to make an order (choose medicine and its quantity) from the selected pharmacy.	2	3
As a customer, I want to be able to view my cart and proceed to checkout.	1	4
As a customer, I want to be able to select a medicine and show all pharmacies that sell this medicine.	1	5
As a customer, I want to be able to search for a medicine or pharmacy by name.	2	6
As a customer, I want to be able to edit my profile.	2	7
As an admin, I want to be able to display available pharmacies and pharmacists.	1	8
As an admin, I want to be able to add, edit or delete pharmacy.	4	9
As an admin, I want to be able to add, edit or delete pharmacist.	4	10
As a pharmacist, I want to be able to display medicines in my pharmacy.	1	11
As a pharmacist, I want to be able to edit medicine.	2	12
As a pharmacist, I want to be able to display all orders made by customers.	1	13
As a pharmacist, I want to be able to accept orders.	1	14
As a pharmacist, I want to be able to edit my profile.	2	15

As a user, I want to be able to sign out.	1	16
Total	**29**	

VI. Design & Implementation

1) Design Description:

➢ UML diagrams are used for designing our pharmacy application:

1. **Structure Diagram:**
   - **Class Diagram**

     Shows the structural hierarchy of our application divided into number of classes:

     Class 'User' is an abstract class which is inherited by three class representing our system users (Customer, Pharmacist, Admin). Each user has classes representing the screens that he/she can display. Class 'Order' represents orders made by customers, which have delivery information represented by Class 'Delivery'. Class 'Item' represents medicines. Class 'Pharmacy' represents the attributes of pharmacies in our system.

2. **Behavior Diagram:**
   - **Use Case Diagram**

     Describes the high-level functions and scope of a system where:
     Customer can login, register, show available pharmacies and medicines, search for a medicine or pharmacy, make order, show cart, checkout, edit profile and sign out.
     Pharmacist can login, show available medicines, add/edit medicine, display orders, accept order, edit profile and sign out.
     Admin can login, show available pharmacies and pharmacists, add/edit pharmacy/pharmacist and sign out.

- **Activity Diagram**

  Flow Chart which represents the flow from one activity to another activity where:

  The diagram is partitioned into three partitions representing our three users (admin, pharmacist and customer) where each of them has his own flow of activities. E.x: Customer can register/login then search for medicine/ pharmacy then make order.

  Pharmacist can login then display orders/medicines then add/edit medicines.

  Admin can login then display pharmacies/pharmacists then add/edit them.

- **State Machine Diagram**

  Describes the states, which the user can attain as well as the transitions between those states. First, the user is either in Logged In state or not. If the user is in the Logged In state, then he/she can change states by navigating through the application pages and doing certain actions (such as clicking on buttons). For each action, a transition from one state to another takes place depending on the user currently logged in whether he/she is an admin or pharmacist or customer.

3. **<u>Behavior / Interaction Diagram:</u>**
   - Sequence Diagram

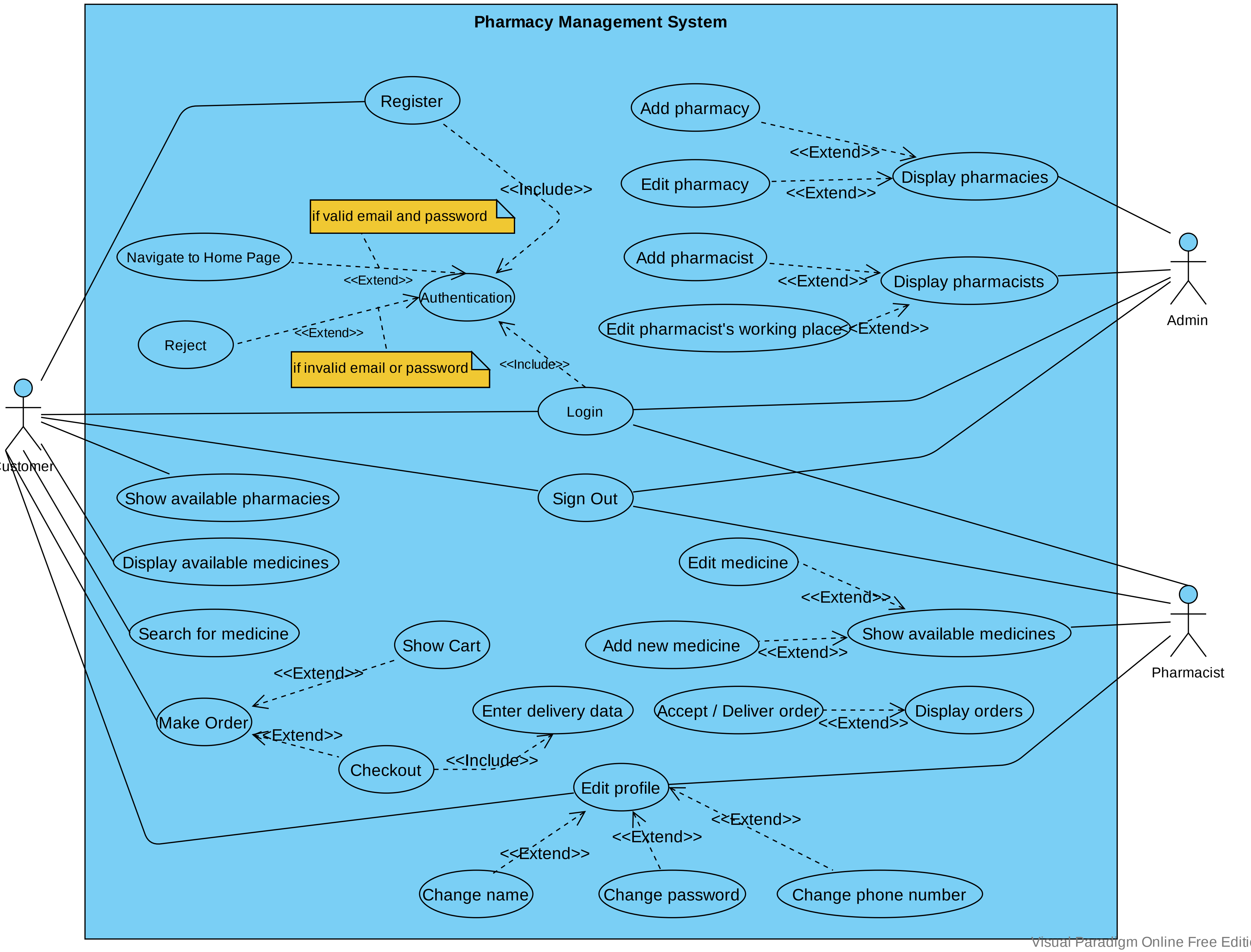
     Type of interaction diagram because it describes how and in what order a group of objects works together. We implemented three sequence diagrams, one for each user (customer, admin and pharmacist) separately, showing the sequence of actions that occurs when the user interacts with the system.

2) <u>Implementation:</u>

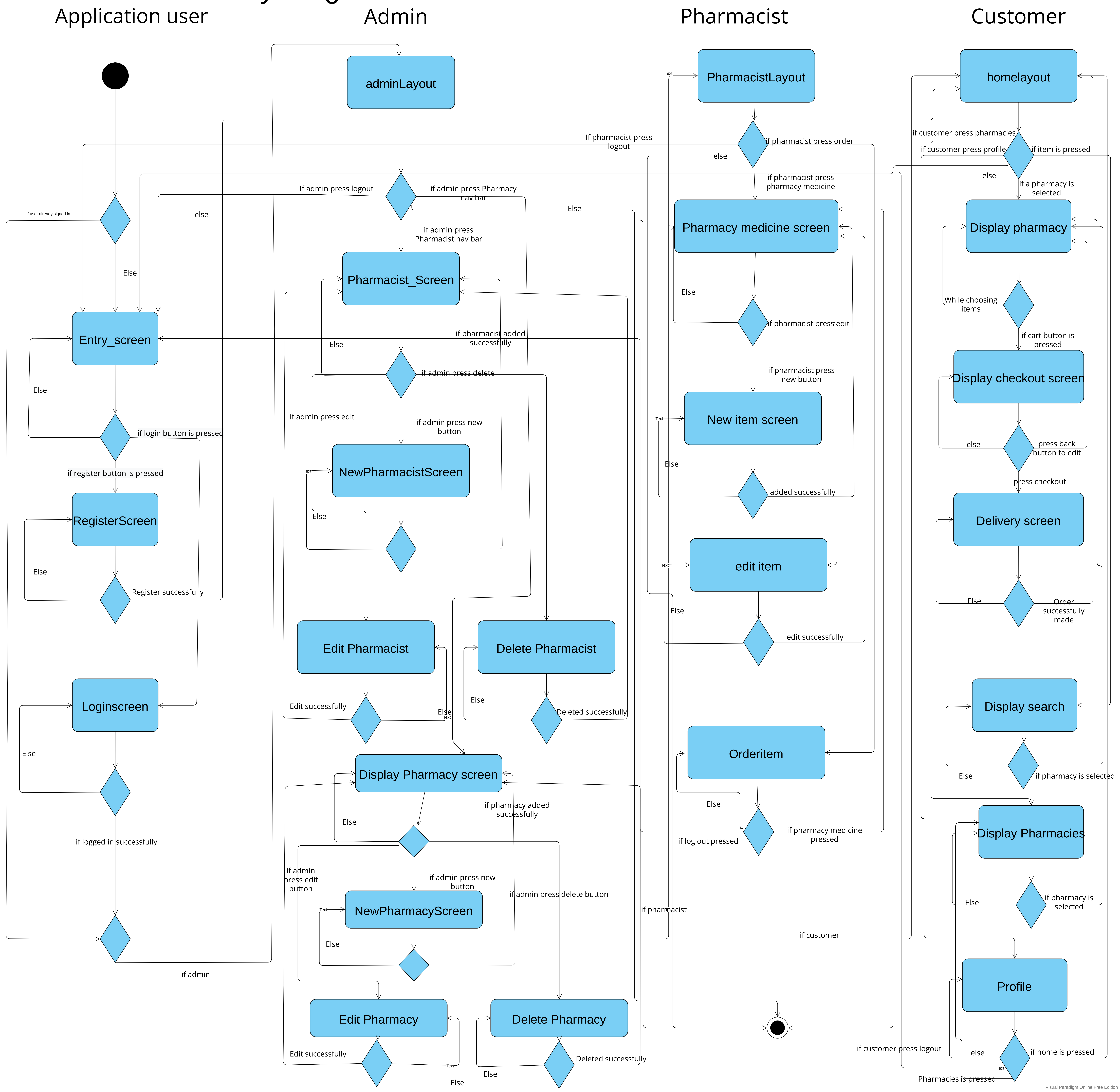
➢ **<u>Development environment & Coding:</u>**
- Flutter SDK (uses Dart programming language).

➢ **<u>Note:</u> (Modification to the design written in proposal)**
- In order to achieve practicality and ease of use, we decided to implement a mobile app instead of a website because almost all users commonly have mobile phones, which helps to easily access the app and make their orders (in case of customer) or manage the pharmacy system (in case of pharmacist and admin).

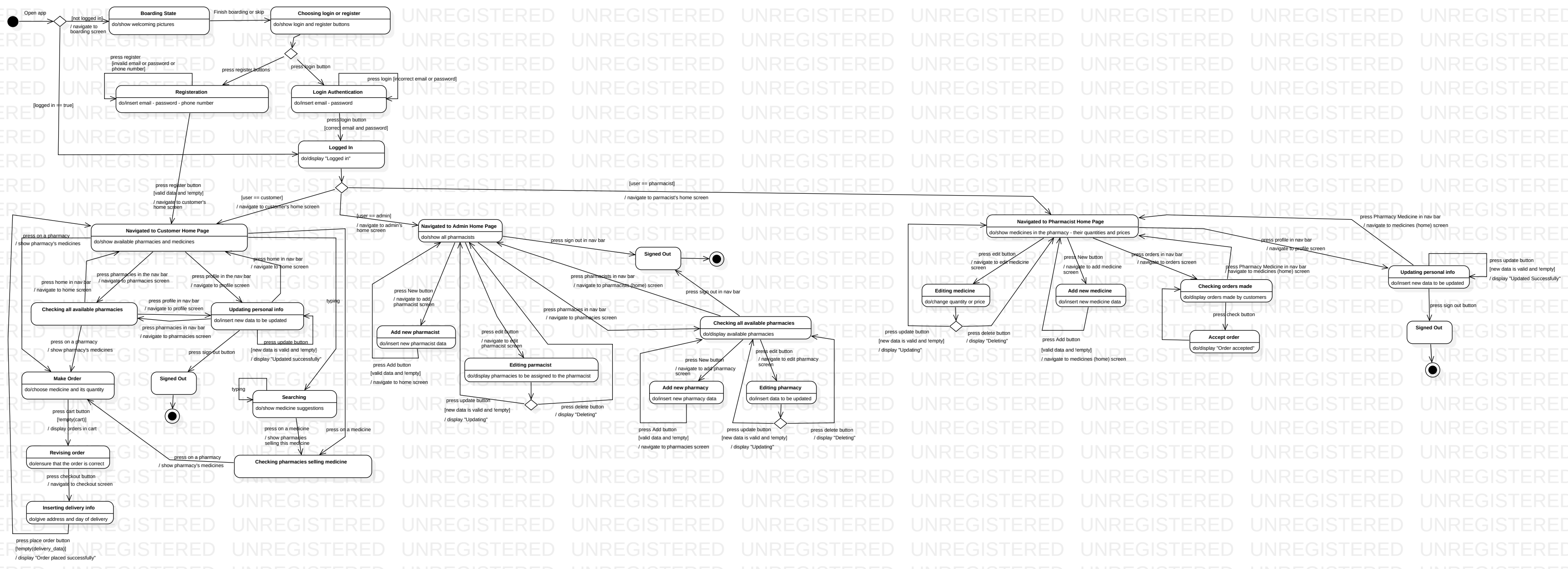
Use Case Diagram



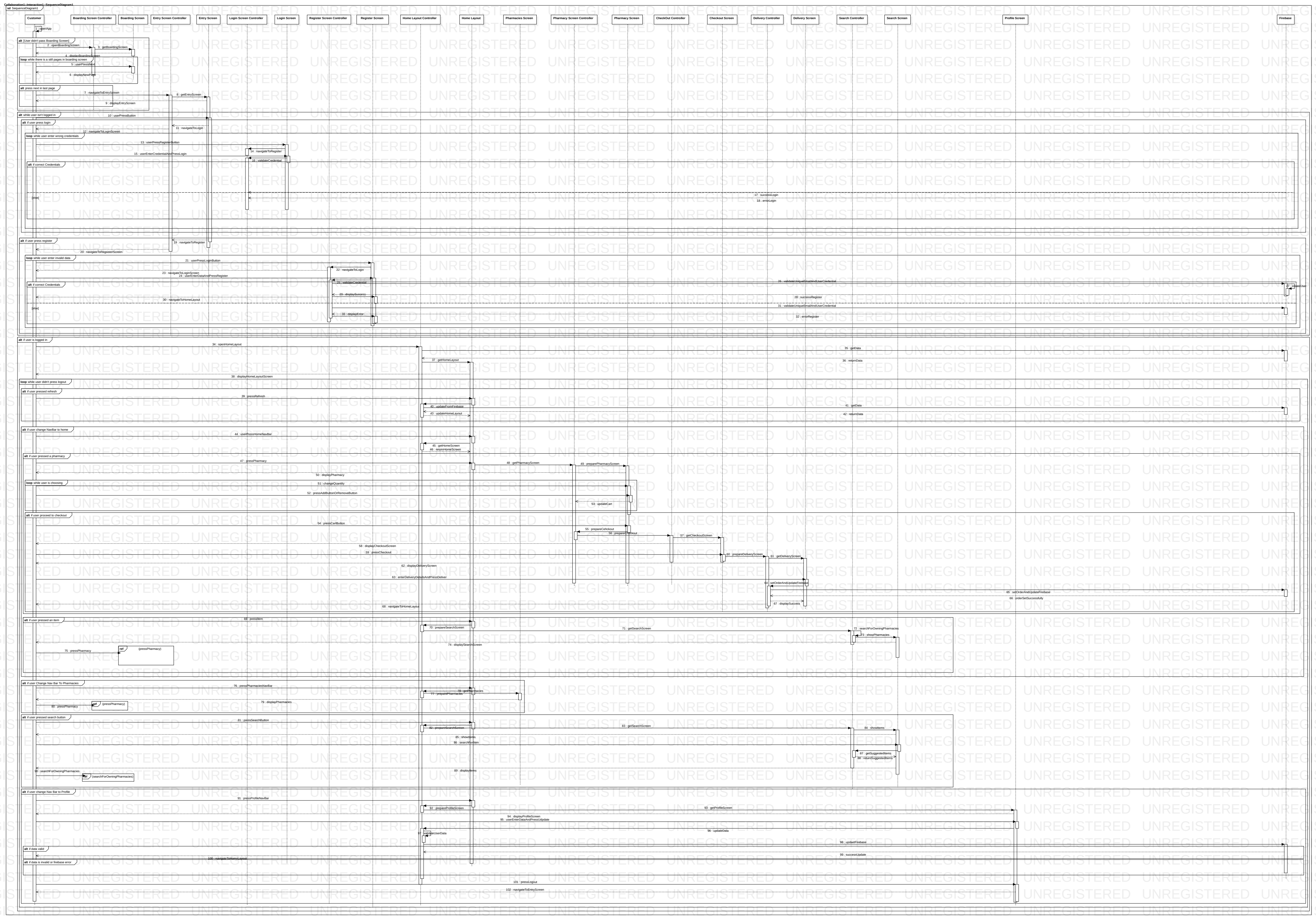
Activity Diagram



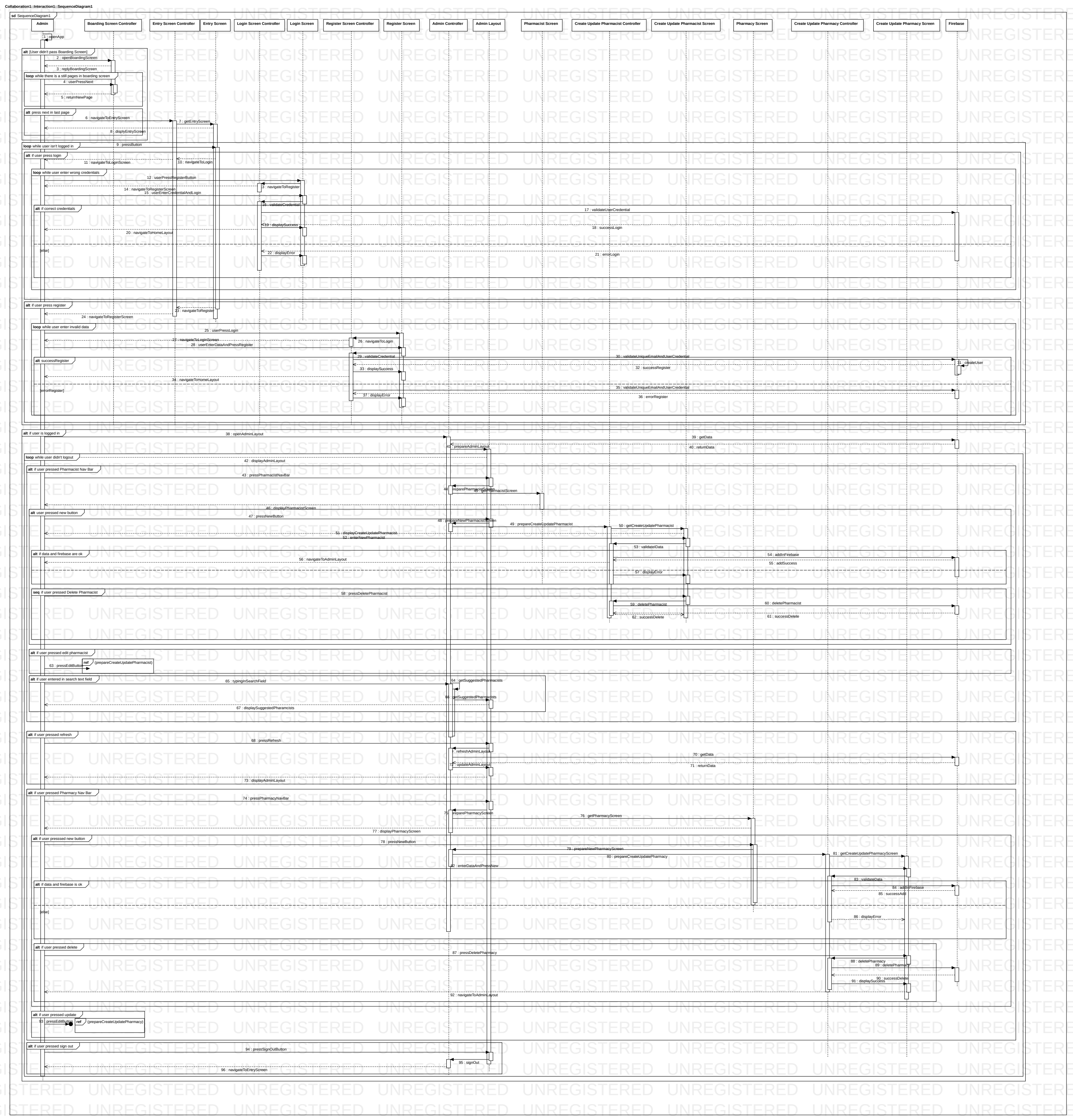
State Machine Diagram


Open app
[not logged in]
/ navigate to boarding screen
Boarding State
do/show welcoming pictures
Finish boarding or skip
Choosing login or register
do/show login and register buttons
press register buttons
press login button
press register
[invalid email or password or phone number]
Registeration
do/insert email - password - phone number
Login Authentication
do/insert email - password
press login [incorrect email or password]
[logged in == true]
press login button
[correct email and password]
Logged In
do/display "Logged in"
press register button
[valid data and !empty]
/ navigate to customer's home screen
[user == customer]
/ navigate to customer's home screen
[user == admin]
/ navigate to admin's home screen
[user == pharmacist]
/ navigate to parmacist's home screen
Navigated to Customer Home Page
do/show available pharmacies and medicines
press on a pharmacy
/ show pharmacy's medicines
press home in nav bar
/ navigate to home screen
press pharmacies in the nav bar
/ navigate to pharmacies screen
press profile in the nav bar
/ navigate to profile screen
press home in nav bar
/ navigate to home screen
typing
Checking all available pharmacies
press profile in nav bar
/ navigate to profile screen
press pharmacies in nav bar
/ navigate to pharmacies screen
Updating personal info
do/insert new data to be updated
press update button
[new data is valid and !empty]
/ display "Updated successfully"
press on a pharmacy
/ show pharmacy's medicines
press sign out button
Make Order
do/choose medicine and its quantity
Signed Out
typing
Searching
do/show medicine suggestions
press cart button
[!empty(cart)]
/ display orders in cart
press on a medicine
/ show pharmacies selling this medicine
press on a medicine
Revising order
do/ensure that the order is correct
press on a pharmacy
/ show pharmacy's medicines
Checking pharmacies selling medicine
press checkout button
/ navigate to checkout screen
Inserting delivery info
do/give address and day of delivery
press place order button
[!empty(delivery_data)]
/ display "Order placed successfully"
Navigated to Admin Home Page
do/show all pharmacists
press sign out in nav bar
Signed Out
press New button
/ navigate to add pharmacist screen
Add new pharmacist
do/insert new pharmacist data
press Add button
[valid data and !empty]
/ navigate to home screen
press edit button
/ navigate to edit pharmacist screen
Editing parmacist
do/display pharmacies to be assigned to the pharmacist
press update button
[new data is valid and !empty]
/ display "Updating"
press delete button
/ display "Deleting"
press pharmacists in nav bar
/ navigate to pharmacists (home) screen
press pharmacies in nav bar
/ navigate to pharmacies screen
press sign out in nav bar
Checking all available pharmacies
do/display available pharmacies
press New button
/ navigate to add pharmacy screen
press edit button
/ navigate to edit pharmacy screen
Add new pharmacy
do/insert new pharmacy data
Editing pharmacy
do/insert data to be updated
press Add button
[valid data and !empty]
/ navigate to pharmacies screen
press update button
[new data is valid and !empty]
/ display "Updating"
press delete button
/ display "Deleting"
Navigated to Pharmacist Home Page
do/show medicines in the pharmacy - their quantities and prices
press edit button
/ navigate to edit medicine screen
press New button
/ navigate to add medicine screen
press orders in nav bar
/ navigate to orders screen
press Pharmacy Medicine in nav bar
/ navigate to medicines (home) screen
press profile in nav bar
/ navigate to profile screen
press Pharmacy Medicine in nav bar
/ navigate to medicines (home) screen
Editing medicine
do/change quantity or price
Add new medicine
do/insert new medicine data
Checking orders made
do/display orders made by customers
press check button
Accept order
do/display "Order accepted"
press update button
[new data is valid and !empty]
/ display "Updating"
press delete button
/ display "Deleting"
press Add button
[valid data and !empty]
/ navigate to medicines (home) screen
Updating personal info
do/insert new data to be updated
press update button
[new data is valid and !empty]
/ display "Updated Successfully"
press sign out button
Signed Out

Customer Sequence Diagram

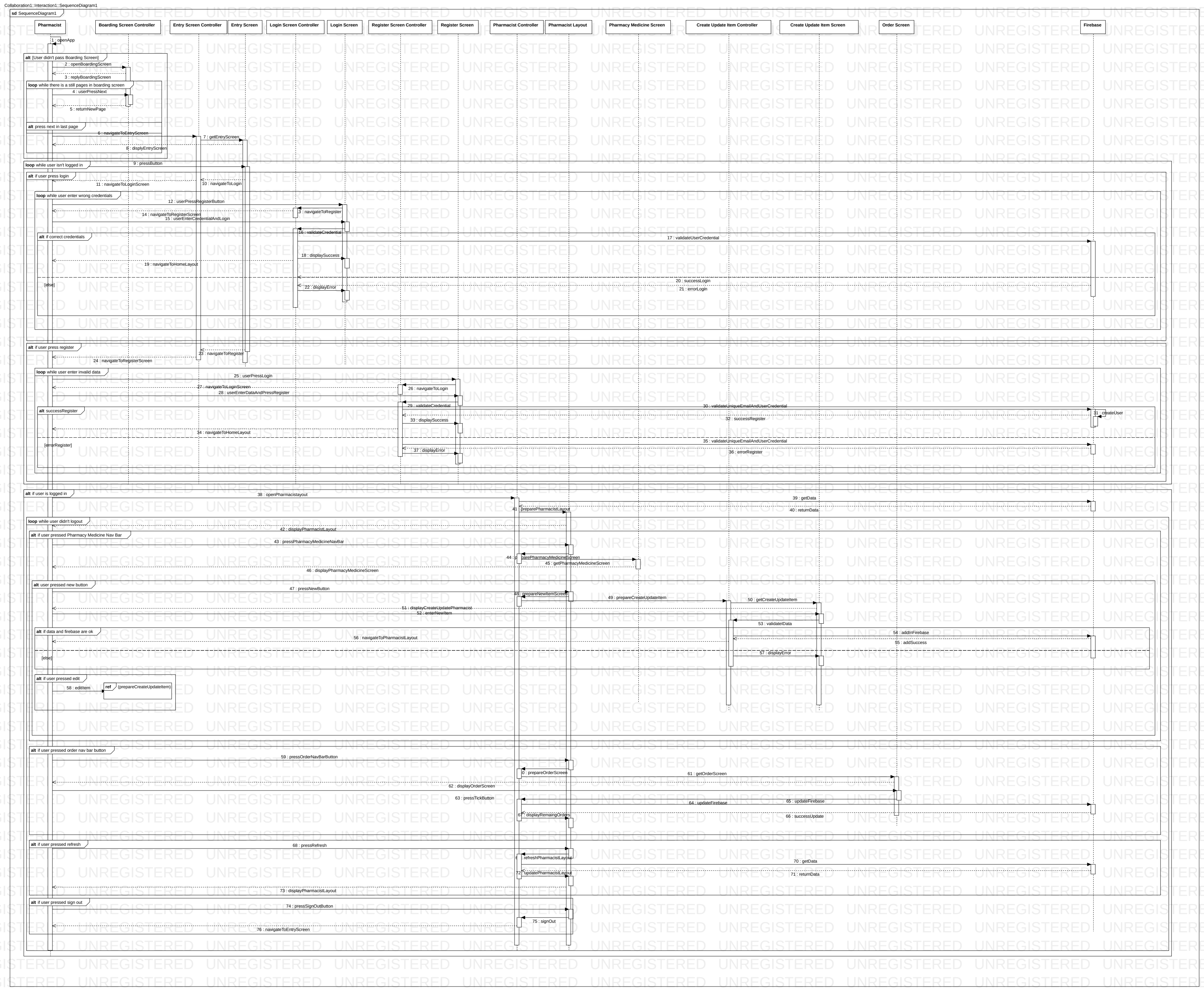


Admin Sequence Diagram



Collaboration1::Interaction1::SequenceDiagram1
sd SequenceDiagram1
Admin
Boarding Screen Controller
Entry Screen Controller
Entry Screen
Login Screen Controller
Login Screen
Register Screen Controller
Register Screen
Admin Controller
Admin Layout
Pharmacist Screen
Create Update Pharmacist Controller
Create Update Pharmacist Screen
Pharmacy Screen
Create Update Pharmacy Controller
Create Update Pharmacy Screen
Firebase
1 : openApp
alt [User didn't pass Boarding Screen]
2 : openBoardingScreen
3 : replyBoardingScreen
loop while there is a still pages in boarding screen
4 : userPressNext
5 : returnNewPage
alt press next in last page
6 : navigateToEntryScreen
7 : getEntryScreen
8 : displyEntryScreen
loop while user isn't logged in
9 : pressButton
alt if user press login
10 : navigateToLogin
11 : navigateToLoginScreen
loop while user enter wrong credentials
12 : userPressRegisterButton
13 : navigateToRegister
14 : navigateToRegisterScreen
15 : userEnterCredentialAndLogin
16 : validateCredential
alt if correct credentials
17 : validateUserCredential
18 : successLogin
19 : displaySuccess
20 : navigateToHomeLayout
[else]
21 : errorLogin
22 : displayError
alt if user press register
23 : navigateToRegister
24 : navigateToRegisterScreen
loop while user enter invalid data
25 : userPressLogin
26 : navigateToLogin
27 : navigateToLoginScreen
28 : userEnterDataAndPressRegister
29 : validateCredential
alt successRegister
30 : validateUniqueEmailAndUserCredential
31 : createUser
32 : successRegister
33 : displaySuccess
34 : navigateToHomeLayout
[errorRegister]
35 : validateUniqueEmailAndUserCredential
36 : errorRegister
37 : displayError
alt if user is logged in
38 : openAdminLayout
39 : getData
40 : returnData
41 : prepareAdminLayout
loop while user didn't logout
42 : displayAdminLayout
alt if user pressed Pharmacist Nav Bar
43 : pressPharmacistNavBar
44 : preparePharmacistScreen
45 : getPharmacistScreen
46 : displayPharmacistScreen
alt user pressed new button
47 : pressNewButton
48 : prepareNewPharmacistScreen
49 : prepareCreateUpdatePharmacist
50 : getCreateUpdatePharmacist
51 : displayCreateUpdatePharmacist
52 : enterNewPharmacist
53 : validatelData
alt if data and firebase are ok
54 : addInFirebase
55 : addSuccess
56 : navigateToAdminLayout
57 : displayError
seq if user pressed Delete Pharmacist
58 : pressDeletePharmacist
59 : deletePharmacist
60 : deletePharmacist
61 : successDelete
62 : successDelete
alt if user pressed edit pharmacist
63 : pressEditButton
ref (prepareCreateUpdatePharmacist)
alt if user entered in search text field
64 : getSuggestedPharmacists
65 : typingInSearchField
66 : getSuggestedPharmacists
67 : displaySuggestedPharamcists
alt if user pressed refresh
68 : pressRefresh
69 : refreshAdminLayout
70 : getData
71 : returnData
72 : updateAdminLayout
73 : displayAdminLayout
alt if user pressed Pharmacy Nav Bar
74 : pressPharmacyNavBar
75 : preparePharmacyScreen
76 : getPharmacyScreen
77 : displayPharmacyScreen
alt if user presssed new button
78 : pressNewButton
79 : prepareNewPharmacyScreen
80 : prepareCreateUpdatePharmacy
81 : getCreateUpdatePharmacyScreen
82 : enterDataAndPressNew
83 : validateData
alt if data and firebase is ok
84 : addInFirebase
85 : successAdd
[else]
86 : displayError
alt if user pressed delete
87 : pressDeletePharmacy
88 : deletePharmacy
89 : deletePharmacy
90 : successDelete
91 : displaySuccess
92 : navigateToAdminLayout
alt if user pressed update
93 : pressEditButton
ref (prepareCreateUpdatePharmacy)
alt if user pressed sign out
94 : pressSignOutButton
95 : signOut
96 : navigateToEntryScreen

Pharmacist Sequence Diagram



Collaboration1::Interaction1::SequenceDiagram1
sd SequenceDiagram1
Pharmacist
Boarding Screen Controller
Entry Screen Controller
Entry Screen
Login Screen Controller
Login Screen
Register Screen Controller
Register Screen
Pharmacist Controller
Pharmacist Layout
Pharmacy Medicine Screen
Create Update Item Controller
Create Update Item Screen
Order Screen
Firebase
1 : openApp
alt [User didn't pass Boarding Screen]
2 : openBoardingScreen
3 : replyBoardingScreen
loop while there is a still pages in boarding screen
4 : userPressNext
5 : returnNewPage
alt press next in last page
6 : navigateToEntryScreen
7 : getEntryScreen
8 : displyEntryScreen
loop while user isn't logged in
9 : pressButton
alt if user press login
10 : navigateToLogin
11 : navigateToLoginScreen
loop while user enter wrong credentials
12 : userPressRegisterButton
13 : navigateToRegister
14 : navigateToRegisterScreen
15 : userEnterCredentialAndLogin
16 : validateCredential
alt if correct credentials
17 : validateUserCredential
18 : displaySuccess
19 : navigateToHomeLayout
20 : successLogin
[else]
21 : errorLogin
22 : displayError
alt if user press register
23 : navigateToRegister
24 : navigateToRegisterScreen
loop while user enter invalid data
25 : userPressLogin
26 : navigateToLogin
27 : navigateToLoginScreen
28 : userEnterDataAndPressRegister
29 : validateCredential
alt successRegister
30 : validateUniqueEmailAndUserCredential
31 : createUser
32 : successRegister
33 : displaySuccess
34 : navigateToHomeLayout
[errorRegister]
35 : validateUniqueEmailAndUserCredential
36 : errorRegister
37 : displayError
alt if user is logged in
38 : openPharmacistayout
39 : getData
40 : returnData
41 : preparePharmacistLayout
loop while user didn't logout
42 : displayPharmacistLayout
alt if user pressed Pharmacy Medicine Nav Bar
43 : pressPharmacyMedicineNavBar
44 : preparePharmacyMedicineScreen
45 : getPharmacyMedicineScreen
46 : displayPharmacyMedicineScreen
alt user pressed new button
47 : pressNewButton
48 : prepareNewItemScreen
49 : prepareCreateUpdateItem
50 : getCreateUpdateItem
51 : displayCreateUpdatePharmacist
52 : enterNewItem
53 : validateIData
alt if data and firebase are ok
54 : addInFirebase
55 : addSuccess
56 : navigateToPharmacistLayout
[else]
57 : displayError
alt if user pressed edit
58 : editItem
ref (prepareCreateUpdateItem)
alt if user pressed order nav bar button
59 : pressOrderNavBarButton
60 : prepareOrderScreen
61 : getOrderScreen
62 : displayOrderScreen
63 : pressTickButton
64 : updateFirebase
65 : updateFirebase
66 : successUpdate
67 : displayRemaingOrders
alt if user pressed refresh
68 : pressRefresh
69 : refreshPharmacistLayout
70 : getData
71 : returnData
72 : updatePharmacistLayout
73 : displayPharmacistLayout
alt if user pressed sign out
74 : pressSignOutButton
75 : signOut
76 : navigateToEntryScreen

Class Diagram

User
-name: String
-email: String
-password: String
-id: String
-phone: String
+login(String, String): Boolean
+setName(String): void
+getName(): String
+setEmail(String): void
+getEmail(): String
+setPassword(String): void
+getPassword(): String
+setId(String): void
+getId(): String
+setPhone(String): void
+getPhone(): String
Customer
-orders[]: List<Order>
+register(String, String): Boolean
+makeOrder(String, String, List<Item>, String, String, String, LocalDateTime): Order
+addToOrders(Order): void
+navigate(): void
+searchForMedicine(String): Item
+selectPharmacy(Pharmacy): void
+selectMedicine(Item): void
+signOut(): void
+setOrders(List<Order>): void
+getOrders(List<Order>): List<Order>
Pharmacist
-pharmacyId: String
+navigate(): void
+updateMedicinePrice(float): void
+updateMedicineQuantity(int): void
+addNewMedicine(Item): void
+removeMedicine(Item): void
+setPharmacyId(String): void
+getPharmacyId(): String
Admin
+addNewPharmacy(Pharmacy) void
+deletePharmacy(Pharmacy): void
+editPharmacy(Pharmacy): void
+addNewPharmacist(Pharmacist): void
+deletePharmacist(Pharmacist): void
+editPharmacist(Pharmacist): void
+signOut(): void
uses
uses
uses
displays
CustomerPageLayout
-screenIndex: int
+navigate(int): CustomerPageLayout
+setScreenIndex(int): void
+getScreenIndex(): int
Order
-orderId: String
-pharmacyId: String
-medicines: Hashtable<String, String>
-delveryInfo: Delivery
-customerId: String
+setOrderId(String): void
+getOrderId(): String
+setPharmacyId(String): void
+getPharmacyId(): String
+setOrderMedicines(Hashtable): void
+getOrderMedicines(): Hashtable
+setDeliveryInfo(Delivery): void
+getDeliveryInfo(): Delivery
+setCustomerId(String): void
+getCustomerId(): String
PharmacistPageLayout
-screenIndex: int
+navigate(int): PharmacistPageLayout
+setScreenIndex(int): void
+getScreenIndex(): int
AdminPageLayout
-screenIndex: int
+navigate(int): AdminPageLayout
+setScreenIndex(int): void
+getScreenIndex(): int
views
OrdersScreen
-orders: List<Hashtable>
+displayOrders: List<Hashtable>
+setOrders(List<Hashtable>): void
PharmacyMedicinesScreen
-pharmacy: Pharmacy
+displayMedicines(): List<Item>
+setPharmacy(Pharmacy): void
+getPharmacy(): Pharmacy
PharmacistProfileScreen
-name: String
-currentPassword: String
-newPassword: String
-phone: String
+update(Pharmacist): Boolean
+setName(String)
+getName(): String
+setCurrPass(String): void
+getCurrPass(): String
+setNewPass(String): void
+getNewPass(): String
+setPhone(String): void
+getPhone(): String
PharmacistsScreen
-pharmacists: List<Pharmacist>
+displayPharmacists: List<Pharmacist>
+setPharmacists(List<Pharmacist>): void
PharmaciesScreen
-pharmacies[]: List<Pharmacy>
+displayPharmacies(): List<Pharmacy>
+setPharmacies(List<Pharmacy>): void
CustomerHomePage
-pharmacies: List<Pharmacy>
-medicines: List<Item>
+displayPharmacies: List<Pharmacy>
+displayMedicines: List<Item>
+setPharmacies(List<Pharmacy>): void
+setMedicines(List<Item>): void
CustomerPharmaciesScreen
-pharmacies: List<Pharmacy>
+displayPharmacies: List<Pharmacy>
+setPharmacies: List<Pharmacy>
CustomerProfileScreen
-name: String
-currentPassword: String
-newPassword: String
-phone: String
+update(Customer): Boolean
+signOut(): void
+setName(String): void
+getName(): String
+setCurrPass(String): void
+getCurrPass(): String
+setNewPass(String): void
+getNewPass(): String
+setPhone(String): void
+getPhone(): String
Item
-itemId: String
-name: String
-description: String
-image: String
+setItemId(String): void
+getItemId(): String
+setName(String): void
+getName(): String
+setDescription(String): void
+getDescription(): String
+setImage(String): void
+getImage(): String
Delivery
-adress: String
-buildingNumber: String
-flatNumber: String
-deliveryDate: LocalDateTime
+setAddress(String): void
+getAddress(): String
+setBuildingNumber(String): void
+getBuildingNumber(): String
+setFlatNumber(String): void
+getFlatNumber(): String
+setDeliveryDate(LocalDateTime): void
+getDeliveryDate(): LocalDateTime
displays
delivered from
displays
displays
displays
Pharmacy
-pharmacyId: String
-name: String
-description: String
-address: String
-medicines[]: List<Hashtable>
-image: String
+setPharmacyId(String): void
+getPharmacyId(): String
+setName(String): void
+getName(): String
+setDescription(String): void
+getDescription(): String
+setAddress(String): void
+getAddress(): String
+setMedicines(List<Hashtable>): void
+getMedicines(): List<Hashtable>
+setImage(String): void
+getIamge(): String
views

VII. Testing

1) Development Testing:

- ➢ Unit testing is used for implementing an automated test-driven approach.

- ➢ **Test Cases:**

1. Login testing
   - 1.1. Email tests
     - 1.1.1. Correct email should give no error.
     - 1.1.2. Empty email field should return error
       → Expected: “Email field is required !”.
     - 1.1.3. Incorrect email by removing “@” or “.com” should return error
       → Expected: “Email is incorrect !”.
   - 1.2. Password test
     - 1.2.1. Empty password filed should return error
       → Expected: “Password field is required !”.
2. Shopping Cart testing
   - 2.1. Adding an item to shopping cart should increment its length by 1.
     → Expected: newCartLength = oldCartLength + 1
   - 2.2. Removing an item from shopping cart should deccrement its length by 1.
     → Expected: newCartLength = oldCartLength - 1
   - 2.3. Testing sum of quantities in cart (added two medicines: one with quantity = 5 and other with quantity = 2)
     → Expected: totalQuantities = 7

2) Release Testing:

- ➢ Added unit tests to the code.
- ➢ Implement the code and keep adapting it until all test cases pass in order to achieve automated test-driven approach.