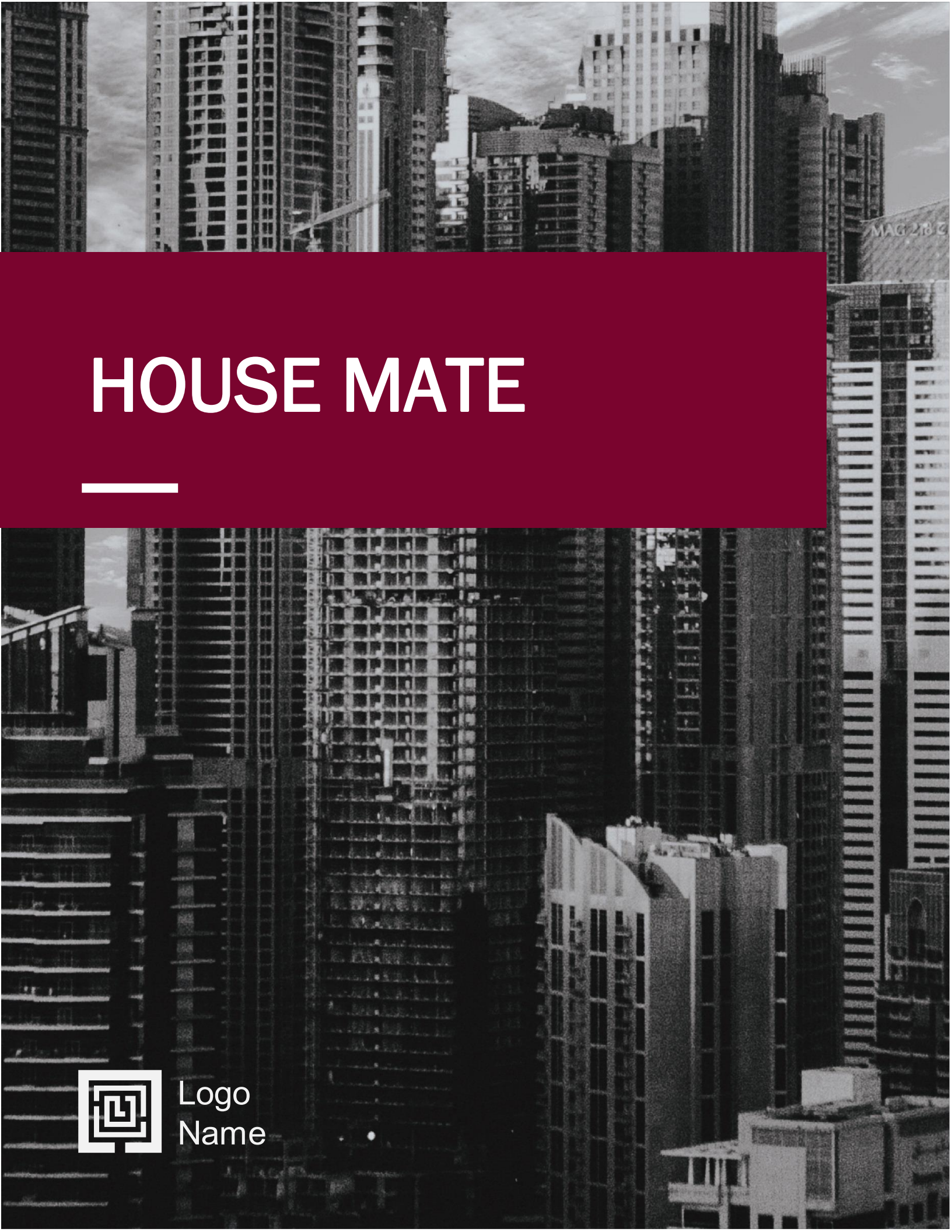


HOUSE MATE

Logo
Name

Team members

Name	ID
ياسمين عادل نصر غانم	201900959
ابراهيم محمود مكي حسين	201900012
كاترين حبيب جورج ناداب	201900573
غيداء الطاهر احمد محمد	201900547
فريدة أحمد توكو	201900568
عماد علي عماد حافظ	201900496

House Mate Web Application

About the project

The project aim is to ease the process of looking for a house mate. By creating an account and posting about what they are looking for.

Goals :

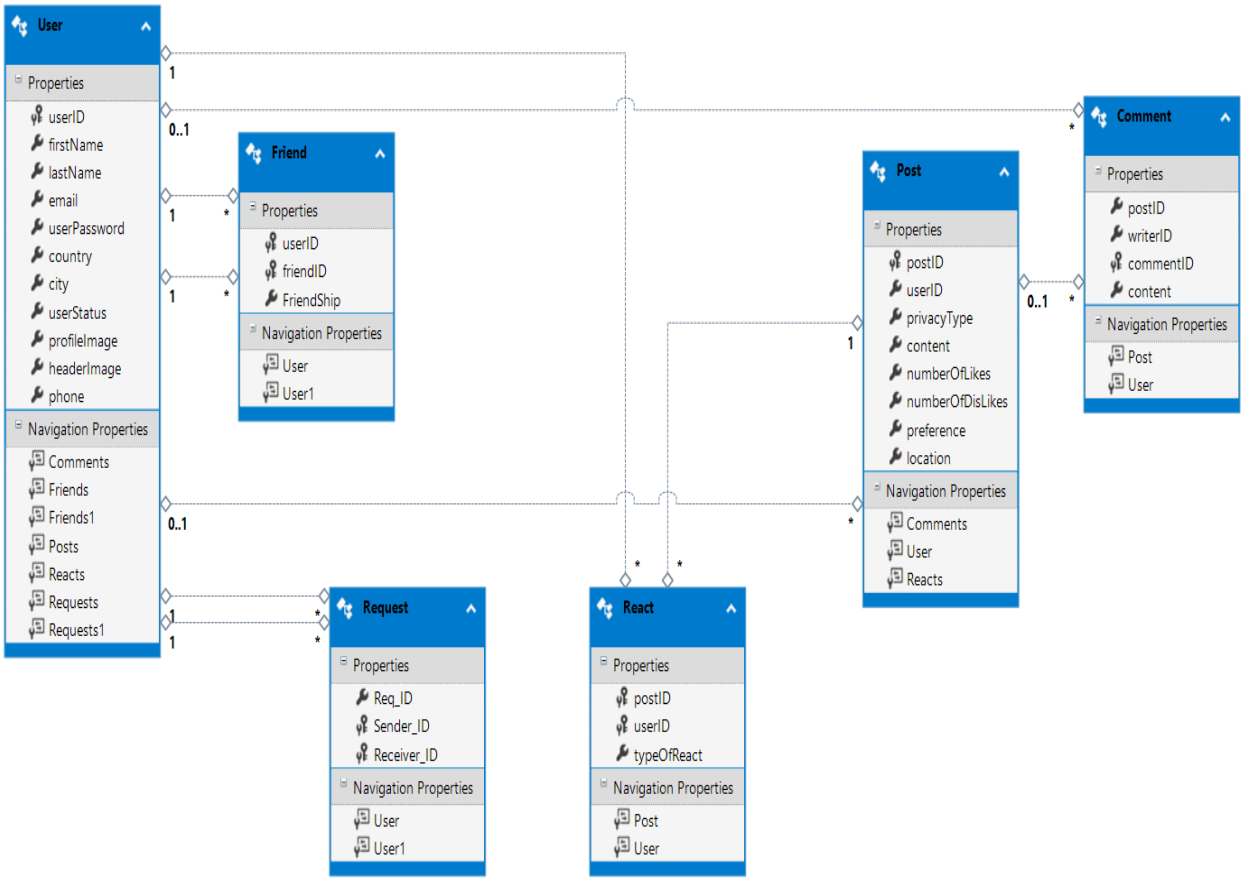
Ease is the process of finding a housemate
It reduce the time spending Looking for a house or a mate
Finding people at the same major or at the same faculty
Have the option to know their mate before joining the house

DESCRIBSION

This Application will allow the user to :

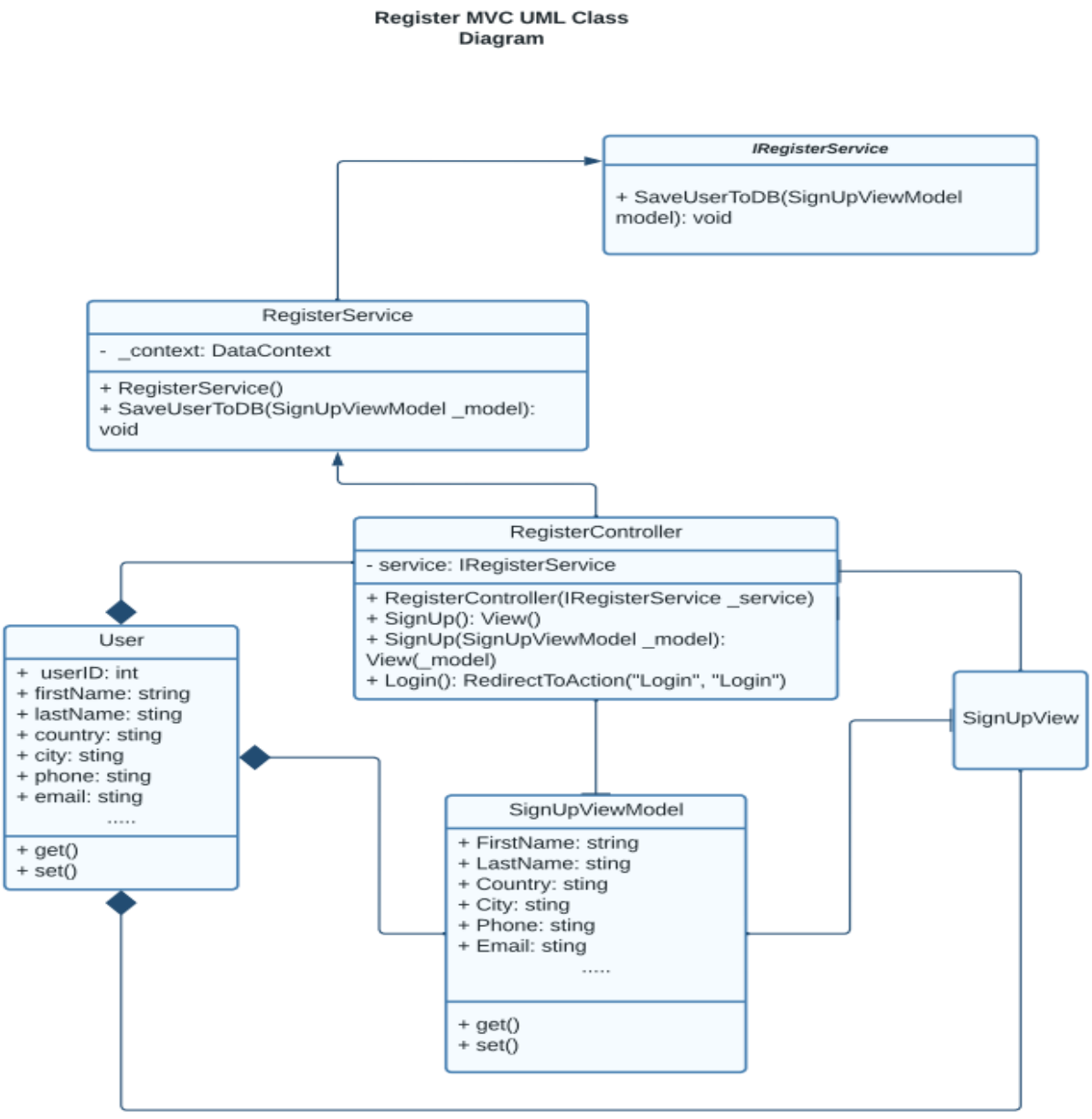
- create an account
- login to his/her account
- edit his/her profile (profile image, first name, last name, city, country, email, mobile).
- write posts in his/her profile.
- search for other users by email or mobile
- send friend requests to other users.
- accept or reject the friend request.
- access his/her friends' profiles and see their posts.
- like and dislike his/her friend's posts and also write comments.
- be able to restrict some of his own posts to not be seen by anyone.

DataBase Schema

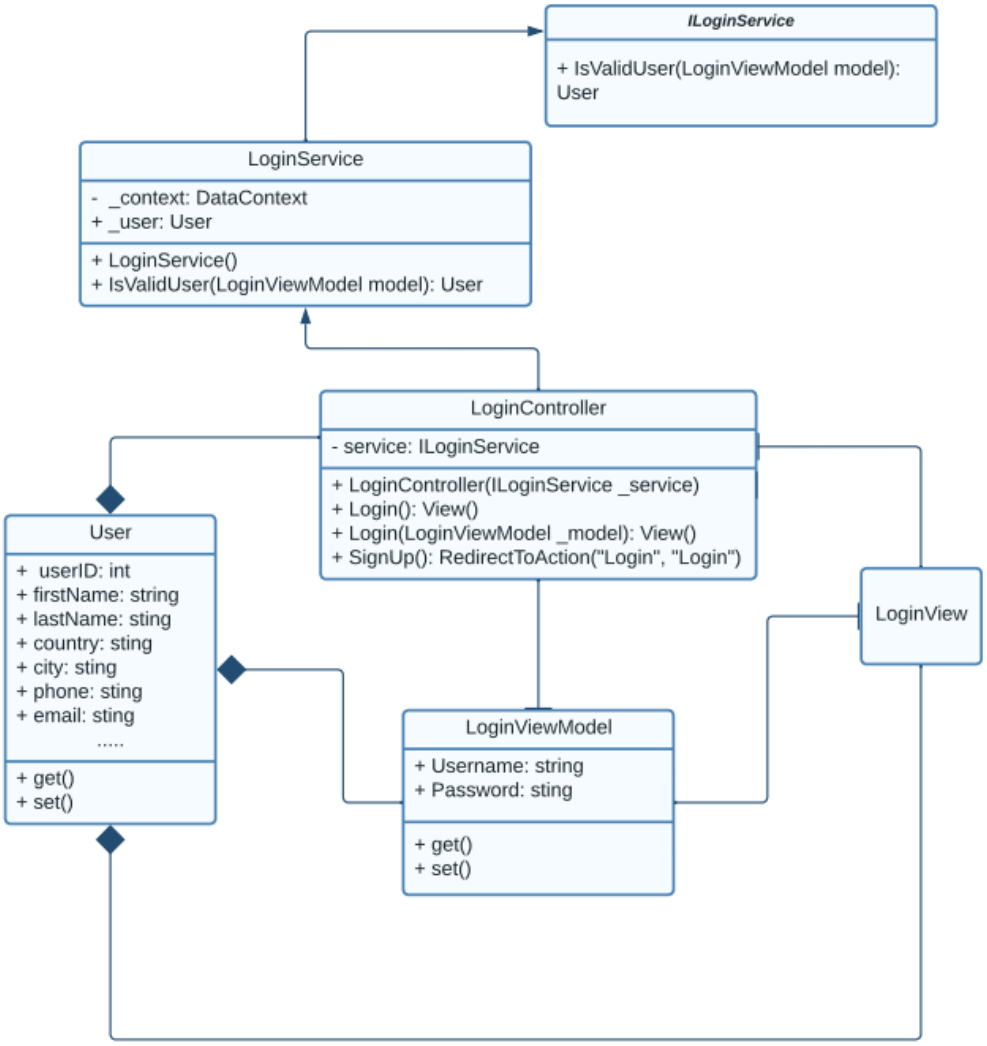


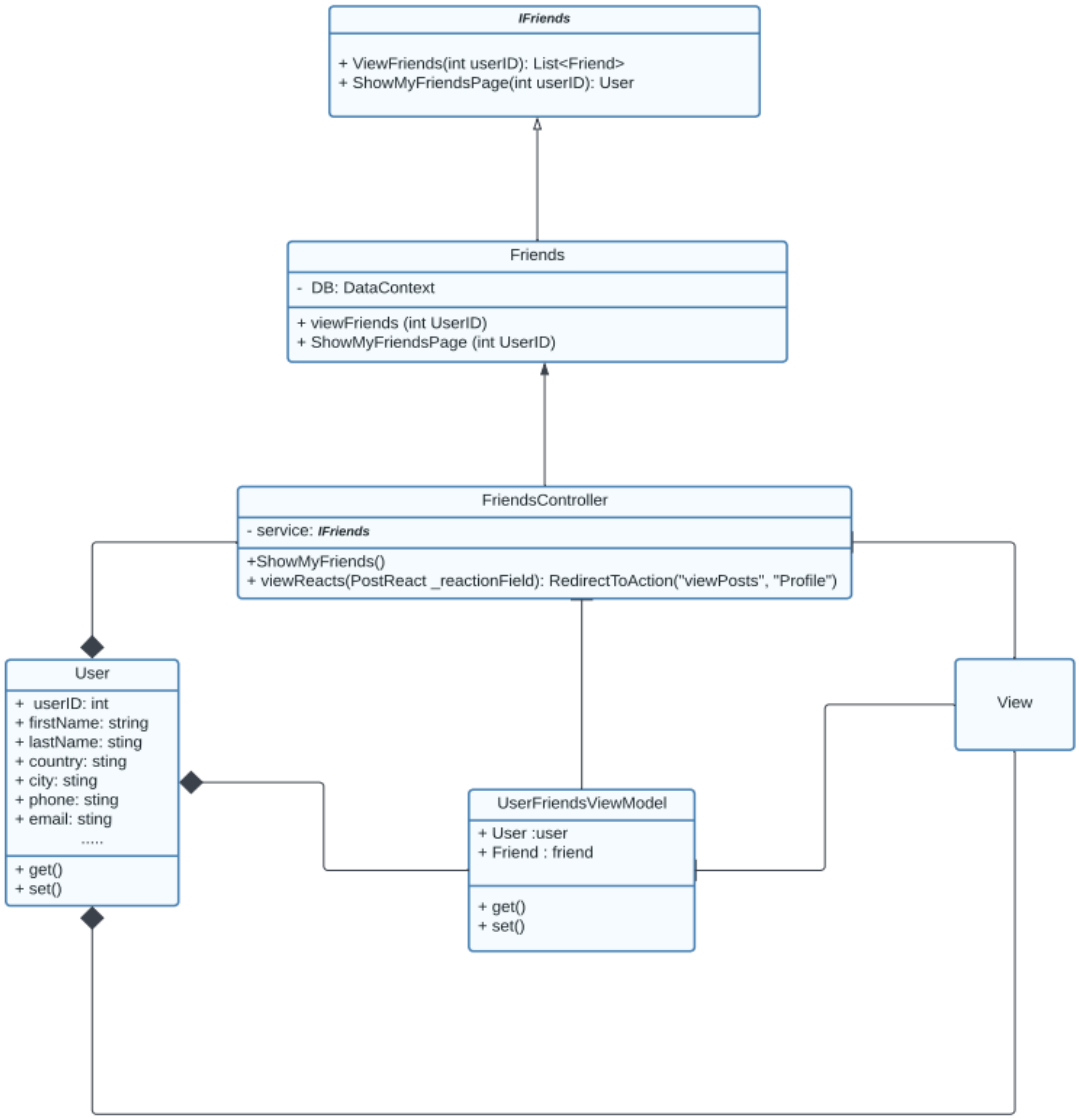
Class Diagrams

Some Samples

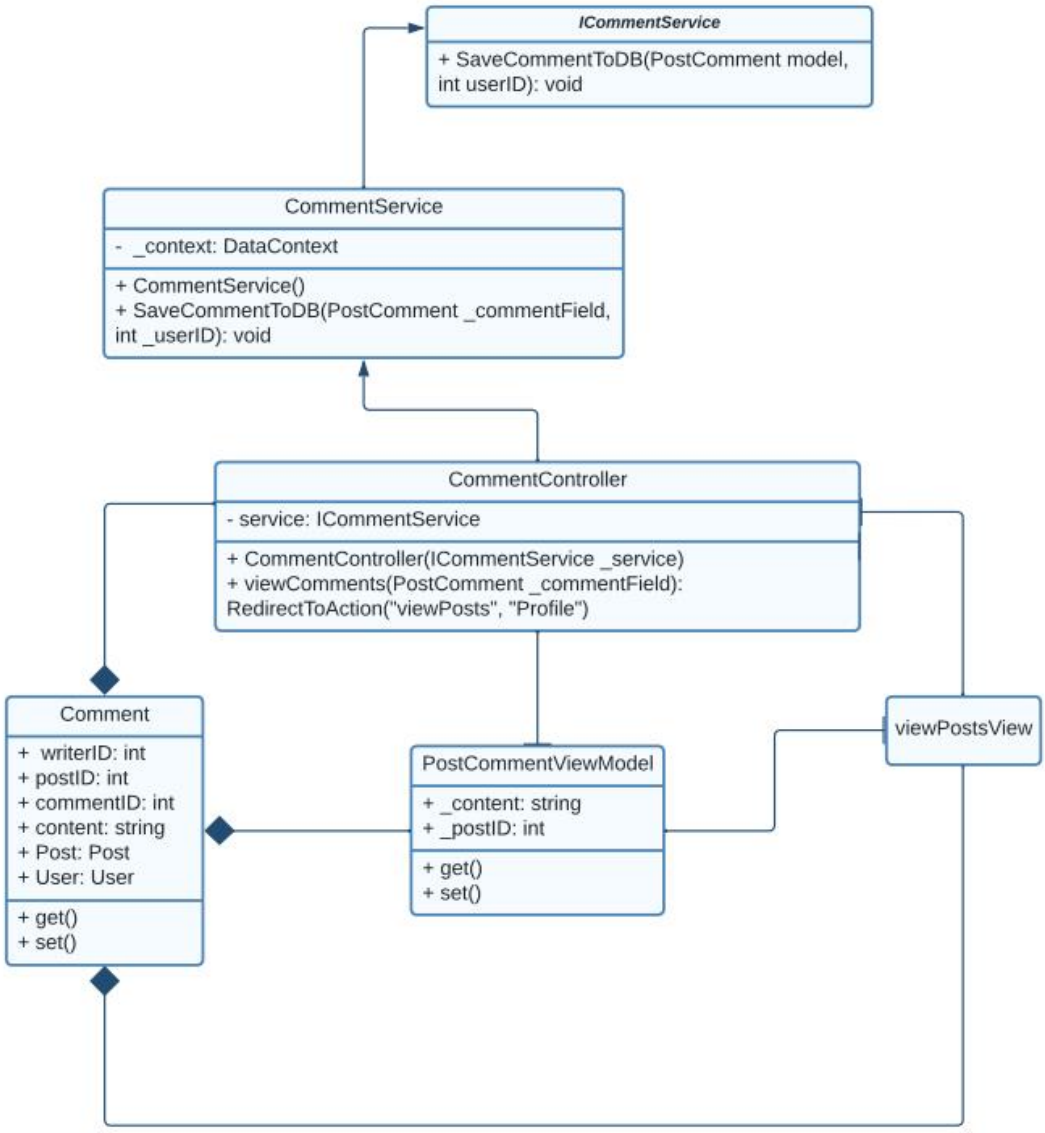


Login MVC UML Class Diagram

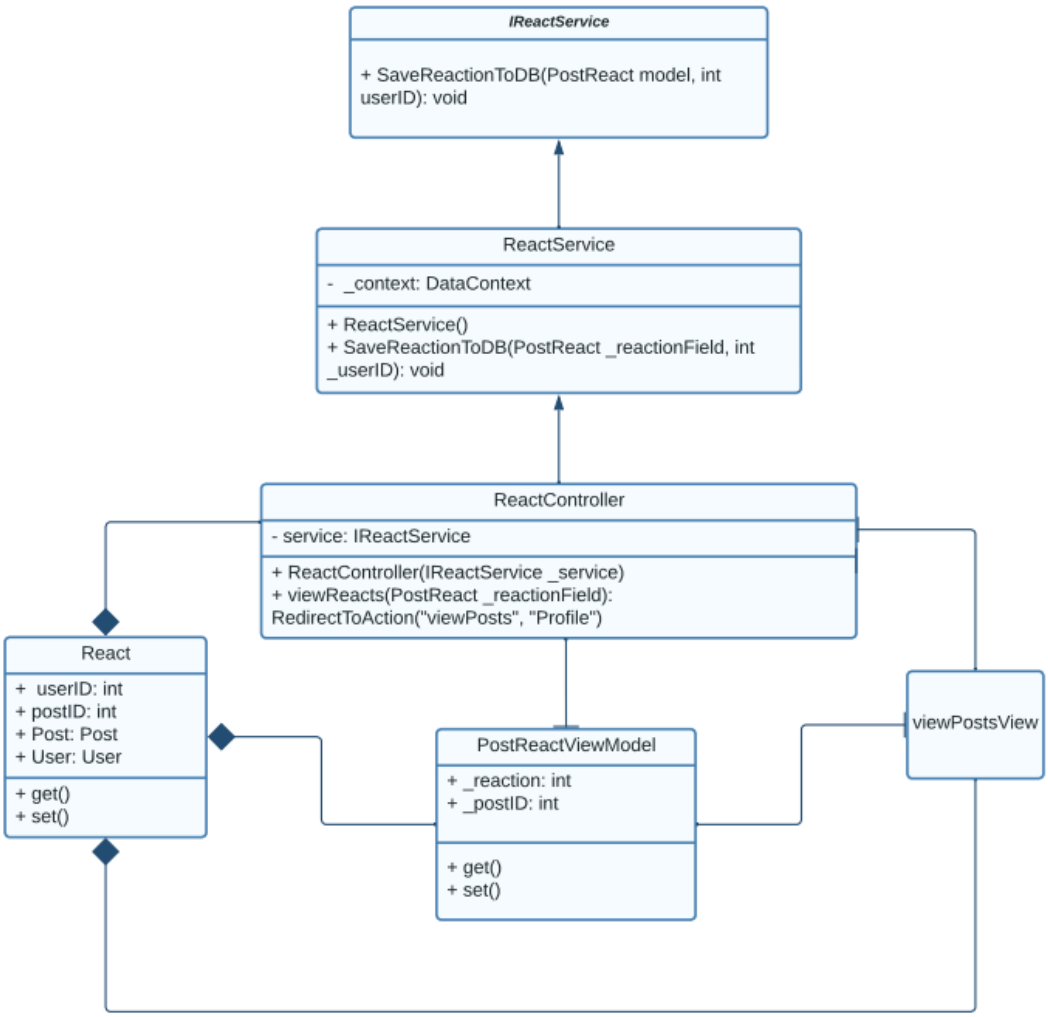


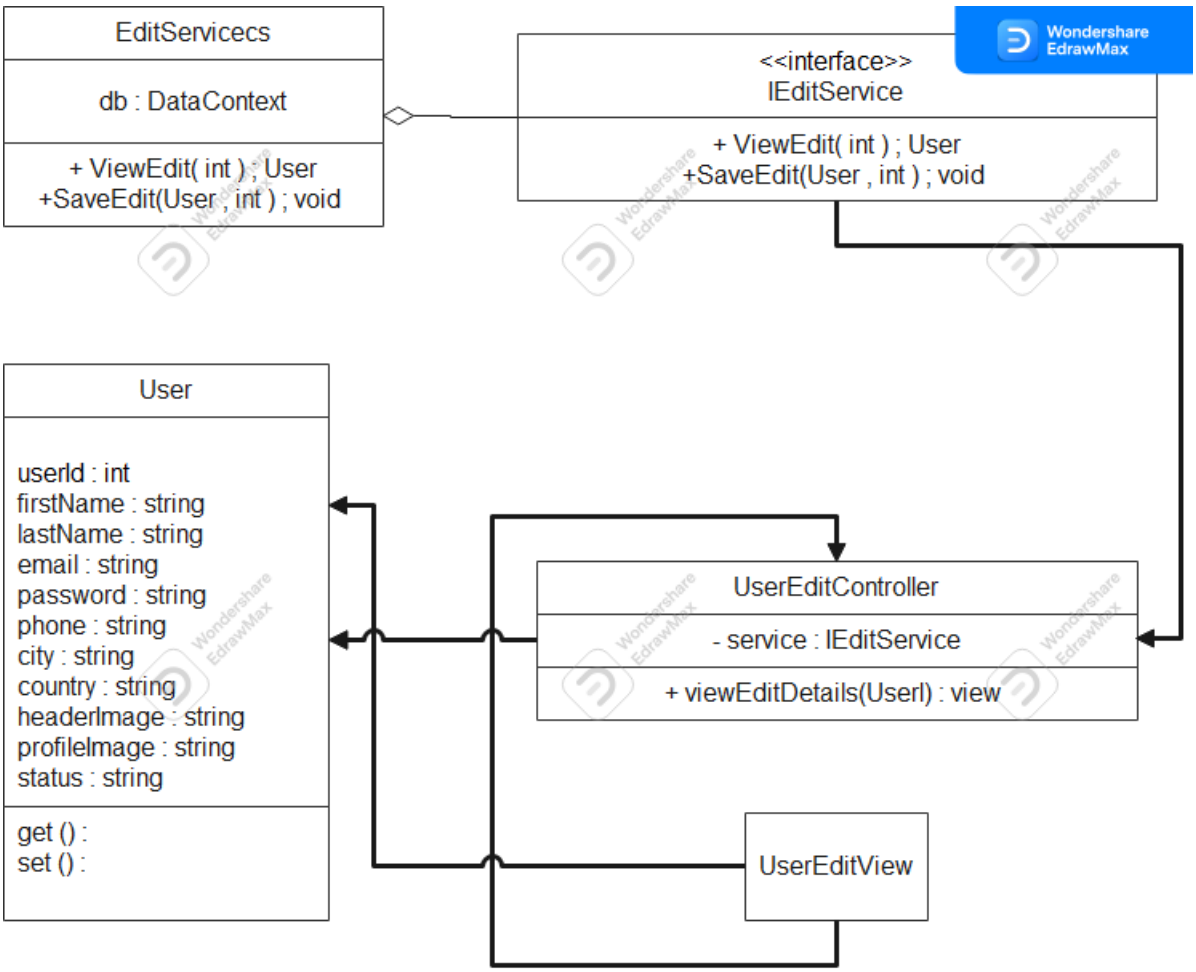

IFriends
+ ViewFriends(int userID): List<Friend>
+ ShowMyFriendsPage(int userID): User
Friends
- DB: DataContext
+ viewFriends (int UserID)
+ ShowMyFriendsPage (int UserID)
FriendsController
- service: IFriends
+ShowMyFriends()
+ viewReacts(PostReact _reactionField): RedirectToAction("viewPosts", "Profile")
User
+ userID: int
+ firstName: string
+ lastName: sting
+ country: sting
+ city: sting
+ phone: sting
+ email: sting
.....
+ get()
+ set()
UserFriendsViewModel
+ User :user
+ Friend : friend
+ get()
+ set()
View

Comment MVC UML
Class Diagram



React MVC UML Class Diagram



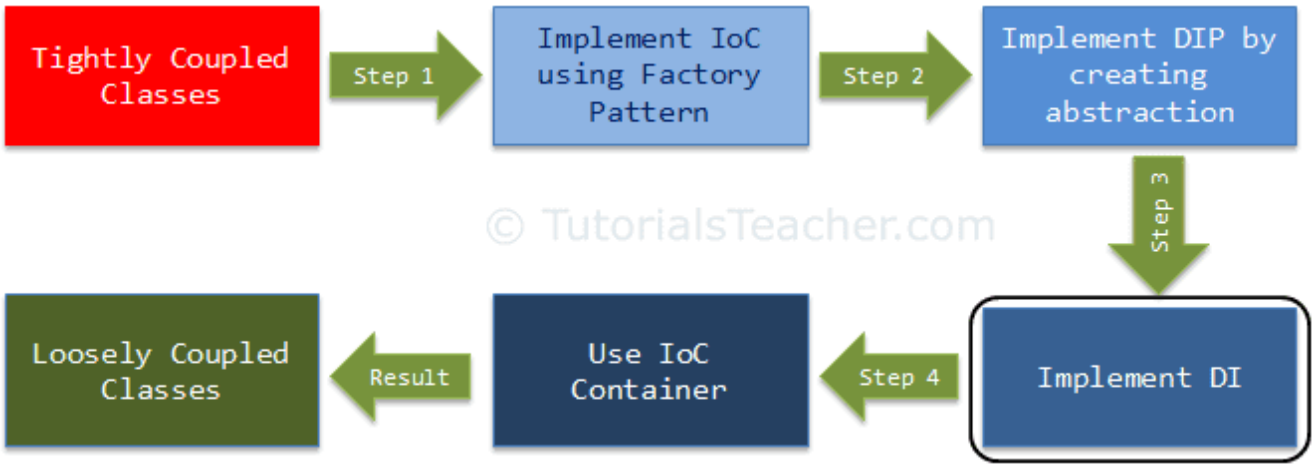

EditServicecs
db : DataContext
+ ViewEdit(int) ; User
+SaveEdit(User , int) ; void
<<interface>>
IEditService
+ ViewEdit(int) ; User
+SaveEdit(User , int) ; void
User
userId : int
firstName : string
lastName : string
email : string
password : string
phone : string
city : string
country : string
headerImage : string
profileImage : string
status : string
get () :
set () :
UserEditController
- service : IEditService
+ viewEditDetails(UserI) : view
UserEditView

Design Pattern

We tried to implement this project based on 'Dependency Injection' design pattern.

By moving dependency object creation completely out of the class. (loosely coupled classes).

❖ A Brief Definition To Dependency Injection Design Pattern:

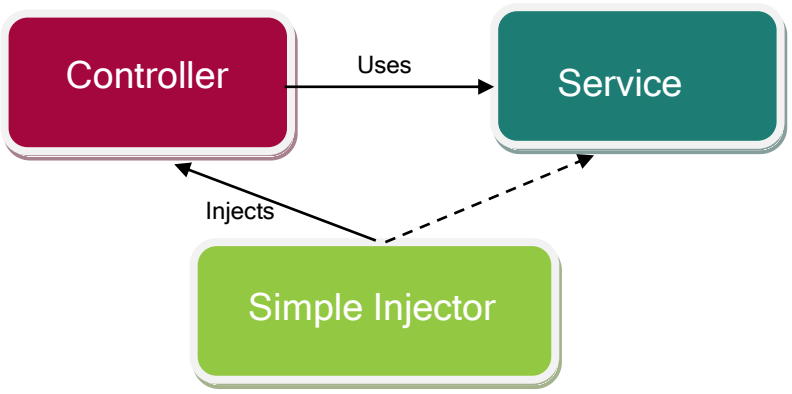


Dependency Injection (DI) is a design pattern used to implement IoC. It allows the creation of dependent objects outside of a class and provides those objects to a class through different ways. Using DI, we move the creation and binding of the dependent objects outside of the class that depends on them.

The Dependency Injection pattern involves 3 types of classes.

1. **Client Class:** The client class (dependent class) is a class which depends on the service class (*Controller classe in our case*)
2. **Service Class:** The service class (dependency) is a class that provides service to the client class (*Service classe*)
3. **Injector Class:** The injector class injects the service class object into the client class. (*Simple Injector Container*)

- This figure illustrates the relationship between these classes:



Types of Dependency Injection

As the injector class injects the service (dependency) to the client (dependent). The injector class injects dependencies broadly in three ways: through a constructor, through a property, or through a method.

Constructor Injection: In the constructor injection, the injector supplies the service (dependency) through the client class constructor.

Property Injection: In the property injection (aka the Setter Injection), the injector supplies the dependency through a public property of the client class.

Method Injection: In this type of injection, the client class implements an interface which declares the method(s) to supply the dependency and the injector uses this interface to supply the dependency to the client class.

_ In Our project we used Constructor Injection.

Thank You.