Movie Rental store

We propose a Movie Rental Store giving our End users the ability to Rent Whatever videos they want.

This is a web application.

We have two main Roles:

Admin: that we suppose he has the right to add, modify and delete any of the recommended Movies.

customer: who will assign to our application renting movies.

And we use in this project three concepts.

Movies: customer rent it

1 – Object Orient Programming

Many programming languages contain the idea of an object. These objects contain information and procedures. Objects can be tied to classes, a category or template that makes objects similar or different from one another.

There are four main principals of OOP:

- Abstraction: The process that makes each object different from one another.

- Encapsulation: Classes contain their own attributes and procedures. These aspects aren't shared by other classes. This is also known as data hiding.
- Inheritance: Some objects can inherit procedures or attributes from other classes.
- Polymorphism: An object's data type or class means that the programming language will have to process the object differently. For instance, a string is treated differently than an integer in many languages.

2- Model View Controller

Unlike OOP, MVC is what frameworks are built upon.

- Model: The template for each instance of a class.
- View: What the user sees. The client-facing instrumentation of a program.
- Controller: The routing protocols of a program.

3- Clean Code Principles

We use this summary on the internet with lectures to get clean code :

General rules

1. Follow standard conventions.
2. Keep it simple stupid. Simpler is always better. Reduce complexity as much as possible.
3. Boy scout rule. Leave the campground cleaner than you found it.
4. Always find root cause. Always look for the root cause of a problem.

Design rules

1. Keep configurable data at high levels.
2. Prefer polymorphism to if/else or switch/case.
3. Separate multi-threading code.
4. Prevent over-configurability.
5. Use dependency injection.
6. Follow Law of Demeter. A class should know only its direct dependencies.

Understandability tips

1. Be consistent. If you do something a certain way, do all similar things in the same way.
2. Use explanatory variables.
3. Encapsulate boundary conditions. Boundary conditions are hard to keep track of. Put the processing for them in one place.
4. Prefer dedicated value objects to primitive type.
5. Avoid logical dependency. Don't write methods which works correctly depending on something else in the same class.
6. Avoid negative conditionals.

Names rules

1. Choose descriptive and unambiguous names.
2. Make meaningful distinction.
3. Use pronounceable names.
4. Use searchable names.
5. Replace magic numbers with named constants.
6. Avoid encodings. Don't append prefixes or type information.

Functions rules

1. Small.
2. Do one thing.
3. Use descriptive names.
4. Prefer fewer arguments.
5. Have no side effects.
6. Don't use flag arguments. Split method into several independent methods that can be called from the client without the flag.

Comments rules

1. Always try to explain yourself in code.
2. Don't be redundant.
3. Don't add obvious noise.
4. Don't use closing brace comments.
5. Don't comment out code. Just remove.
6. Use as explanation of intent.
7. Use as clarification of code.
8. Use as warning of consequences.

Source code structure

1. Separate concepts vertically.
2. Related code should appear vertically dense.
3. Declare variables close to their usage.
4. Dependent functions should be close.
5. Similar functions should be close.
6. Place functions in the downward direction.
7. Keep lines short.

8. Don't use horizontal alignment.
9. Use white space to associate related things and disassociate weakly related.
10. Don't break indentation.

Objects and data structures

1. Hide internal structure.
2. Prefer data structures.
3. Avoid hybrids structures (half object and half data).
4. Should be small.
5. Do one thing.
6. Small number of instance variables.
7. Base class should know nothing about their derivatives.
8. Better to have many functions than to pass some code into a function to select a behavior.
9. Prefer non-static methods to static methods.

Tests

1. One assert per test.
2. Readable.
3. Fast.
4. Independent.
5. Repeatable.

Code smells

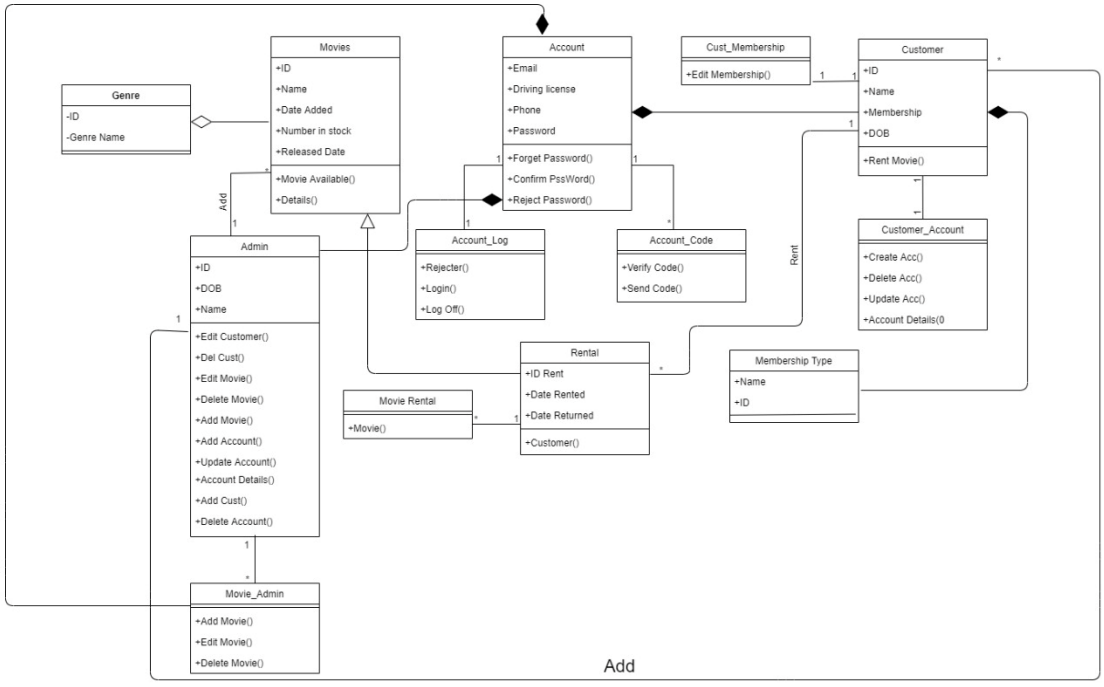
1. Rigidity. The software is difficult to change. A small change causes a cascade of subsequent changes.

2. Fragility. The software breaks in many places due to a single change.
3. Immobility. You cannot reuse parts of the code in other projects because of involved risks and high effort.
4. Needless Complexity.
5. Needless Repetition.
6. Opacity. The code is hard to understand.

SOLID principles :

1- Single Responsibility principle:

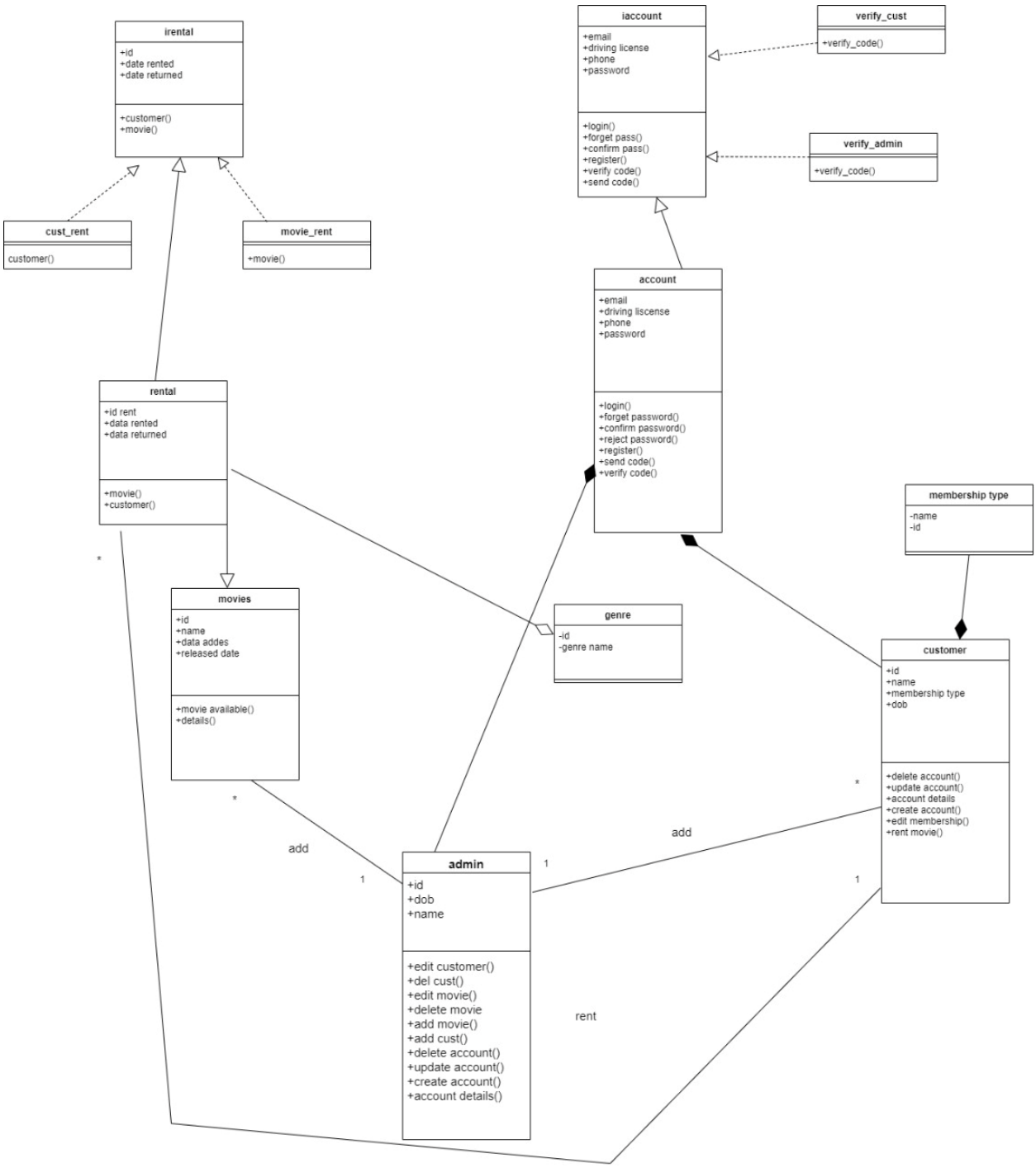
The idea behind the SRP is that every class, module, or function in a program should have one responsibility/purpose in a program. As a commonly used definition, "every class should have only one reason to change.



2- Open/Closed principle:

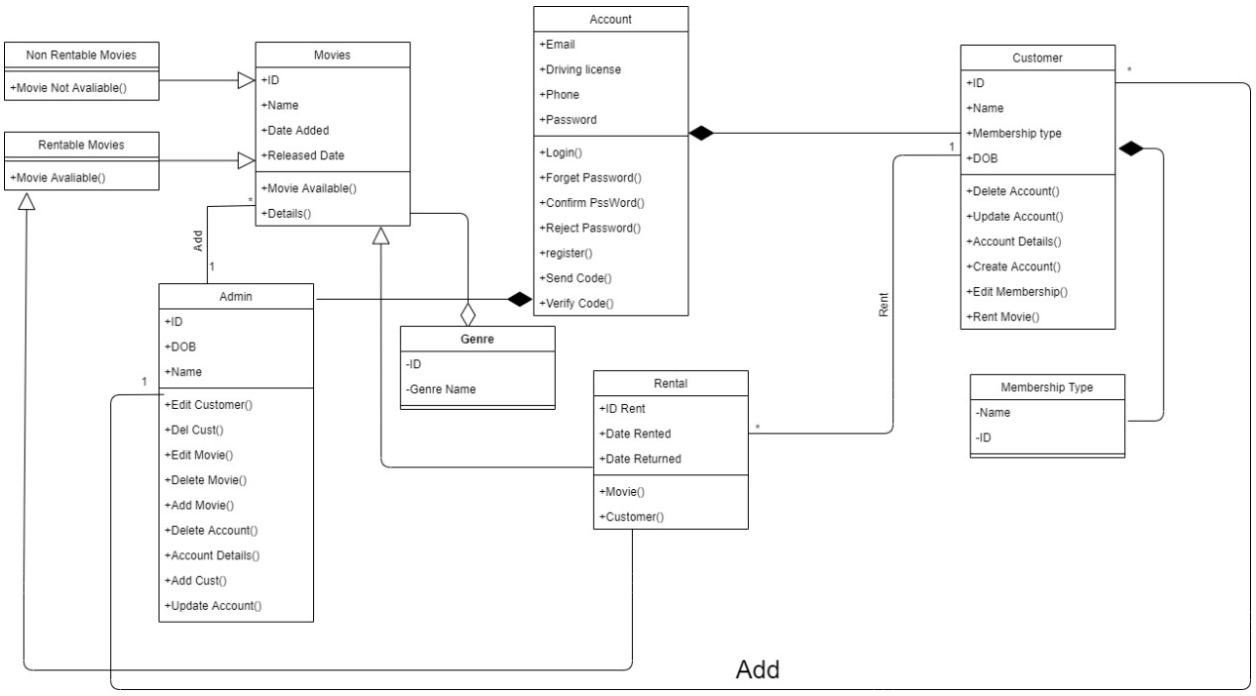
The open-closed principle states that software entities should be open for extension, but closed for modification.

This implies that such entities – classes, functions, and so on – should be created in a way that their core functionalities can be extended to other entities without altering the initial entity's source code.


irental
+id
+date rented
+date returned
+customer()
+movie()
iaccount
+email
+driving license
+phone
+password
+login()
+forget pass()
+confirm pass()
+register()
+verify code()
+send code()
verify_cust
+verify_code()
verify_admin
+verify_code()
cust_rent
customer()
movie_rent
+movie()
account
+email
+driving liscense
+phone
+password
+login()
+forget password()
+confirm password()
+reject password()
+register()
+send code()
+verify code()
rental
+id rent
+data rented
+data returned
+movie()
+customer()
membership type
-name
-id
movies
+id
+name
+data addes
+released date
+movie available()
+details()
genre
-id
-genre name
customer
+id
+name
+membership type
+dob
+delete account()
+update account()
+account details
+create account()
+edit membership()
+rent movie()
admin
+id
+dob
+name
+edit customer()
+del cust()
+edit movie()
+delete movie
+add movie()
+add cust()
+delete account()
+update account()
+create account()
+account details()
add
add
rent
*
*
*
1
1
1

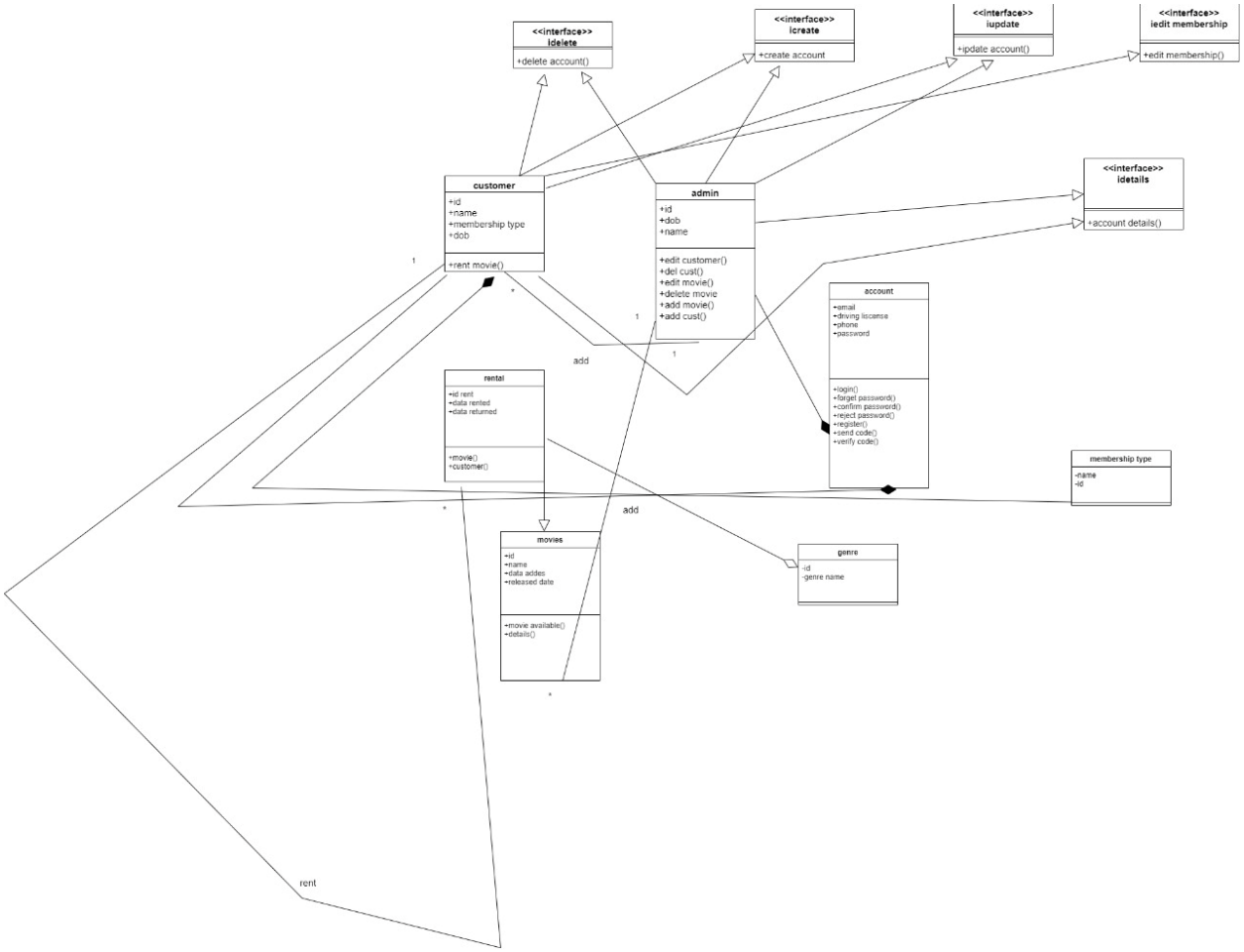
3- Liskove principle:

The Liskov substitution principle simply implies that when an instance of a class is passed/extended to another class, the inheriting class should have a use case for all the properties and behavior of the inherited class.



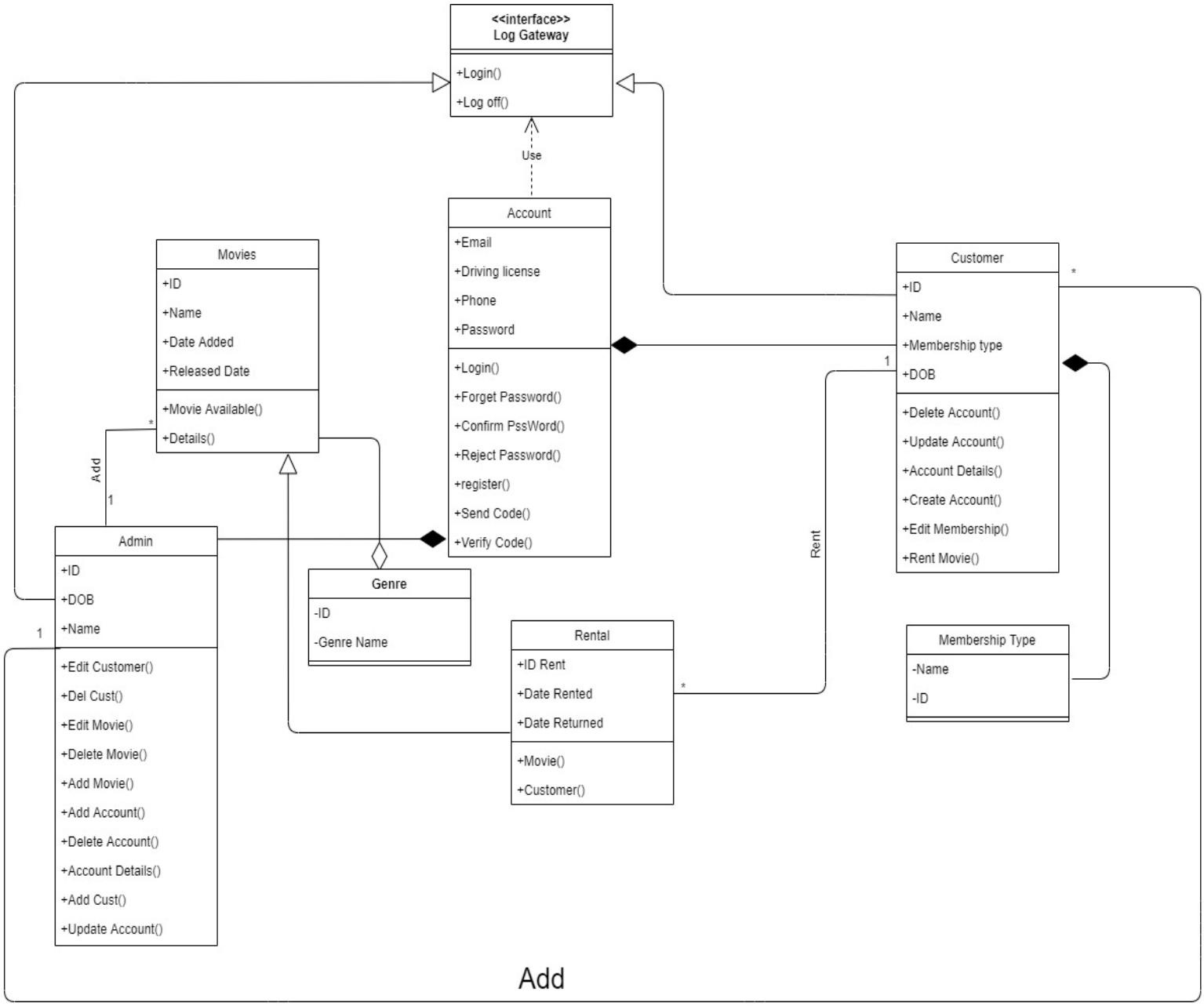
4- Interface Segregation principle:

The interface segregation principle states that the interface of a program should be split in a way that the user/client would only have access to the necessary methods related to their needs.



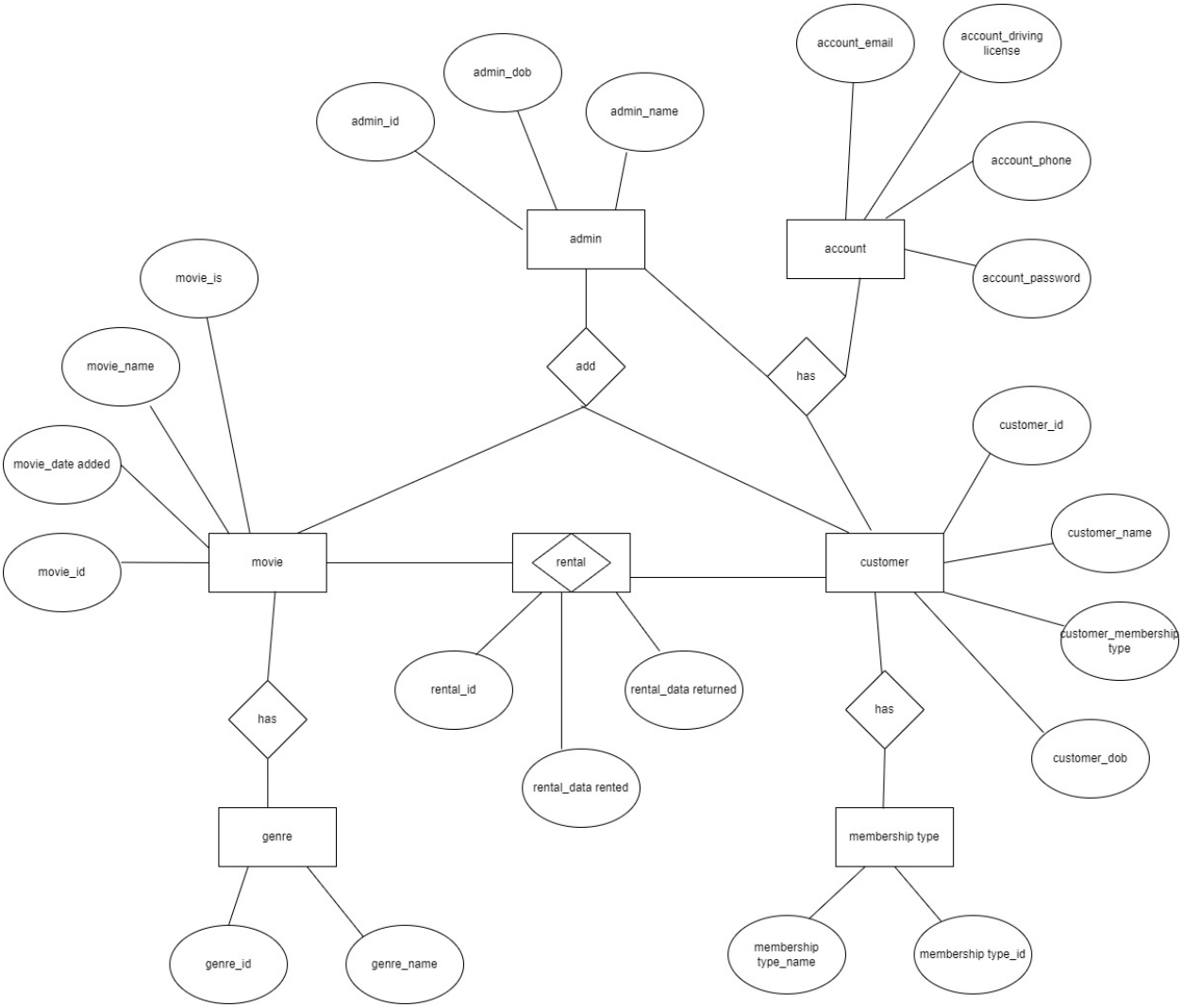
5- Dependancy principle:

High-level modules should not import anything from low-level modules. Both should depend on abstractions



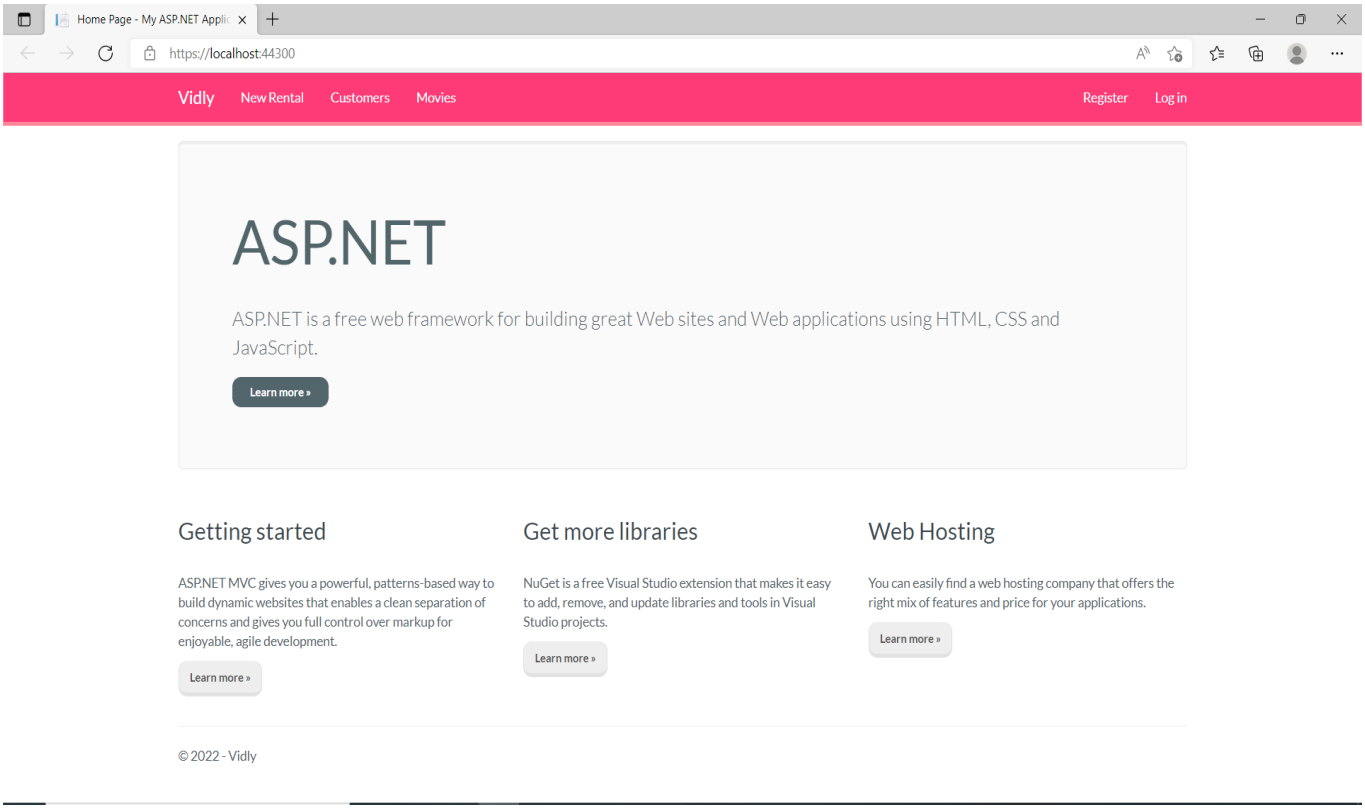
(e.g., interfaces).

DATABASE SCHEMA:

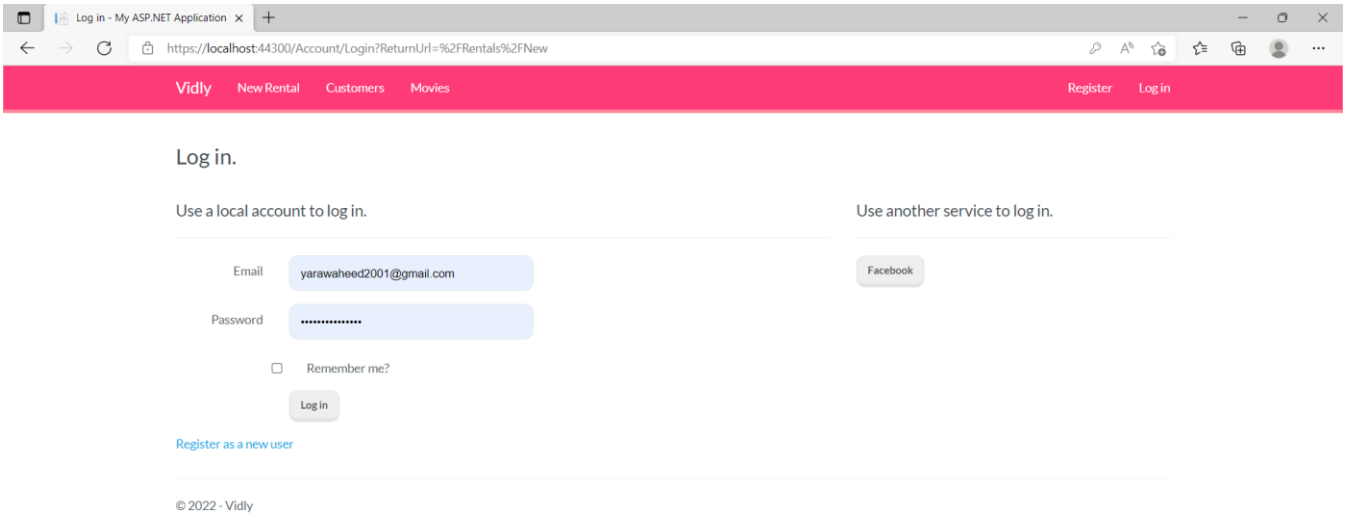


Pictures of our project:

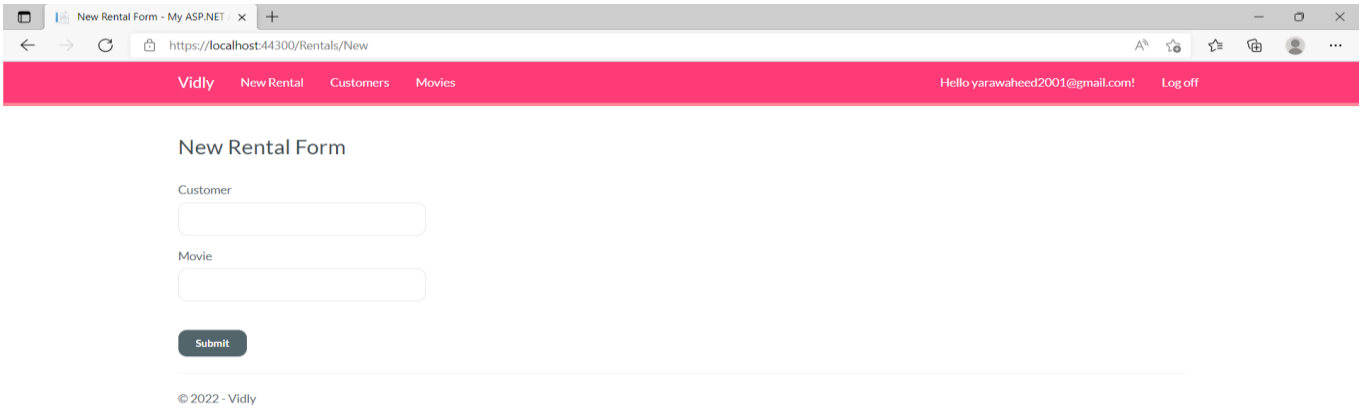
Home:


Home Page - My ASP.NET Applic
https://localhost:44300
Vidly
New Rental
Customers
Movies
Register
Log in
ASP.NET
ASP.NET is a free web framework for building great Web sites and Web applications using HTML, CSS and JavaScript.
Learn more »
Getting started
ASP.NET MVC gives you a powerful, patterns-based way to build dynamic websites that enables a clean separation of concerns and gives you full control over markup for enjoyable, agile development.
Learn more »
Get more libraries
NuGet is a free Visual Studio extension that makes it easy to add, remove, and update libraries and tools in Visual Studio projects.
Learn more »
Web Hosting
You can easily find a web hosting company that offers the right mix of features and price for your applications.
Learn more »
© 2022 - Vidly

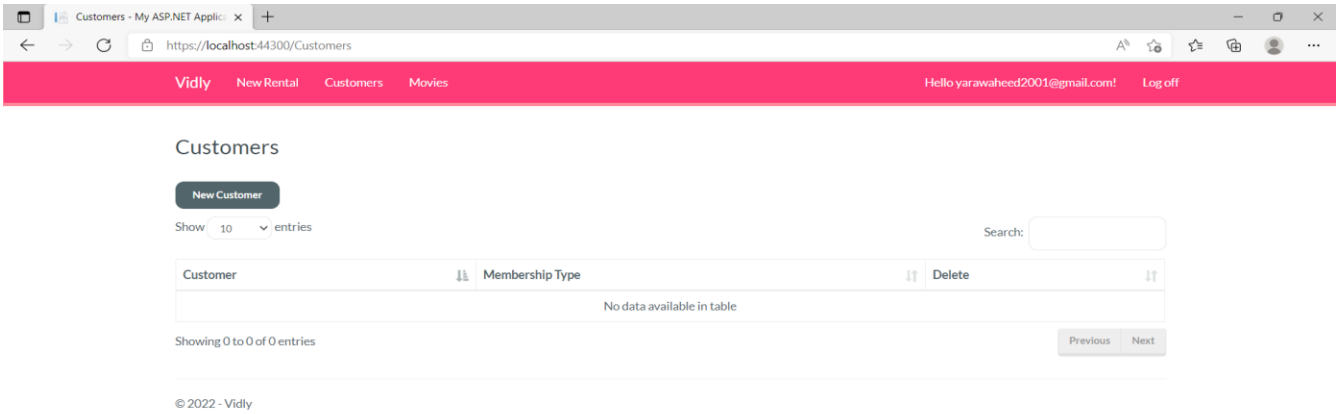
Login:


Log in - My ASP.NET Application
https://localhost:44300/Account/Login?ReturnUrl=%2FRentals%2FNew
Vidly
New Rental
Customers
Movies
Register
Log in
Log in.
Use a local account to log in.
Email
yarawaheed2001@gmail.com
Password
Remember me?
Log in
Register as a new user
Use another service to log in.
Facebook
© 2022 - Vidly

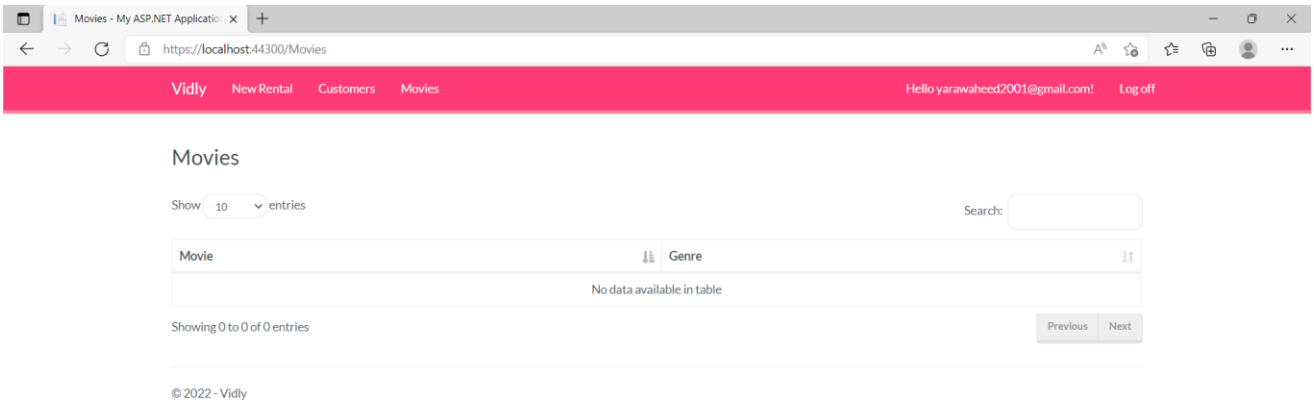
New Rental:



Customers:

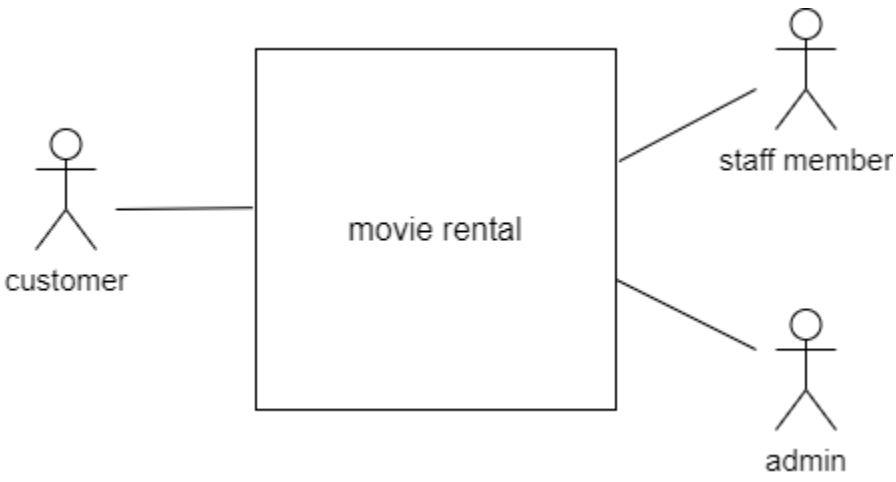


Movies:

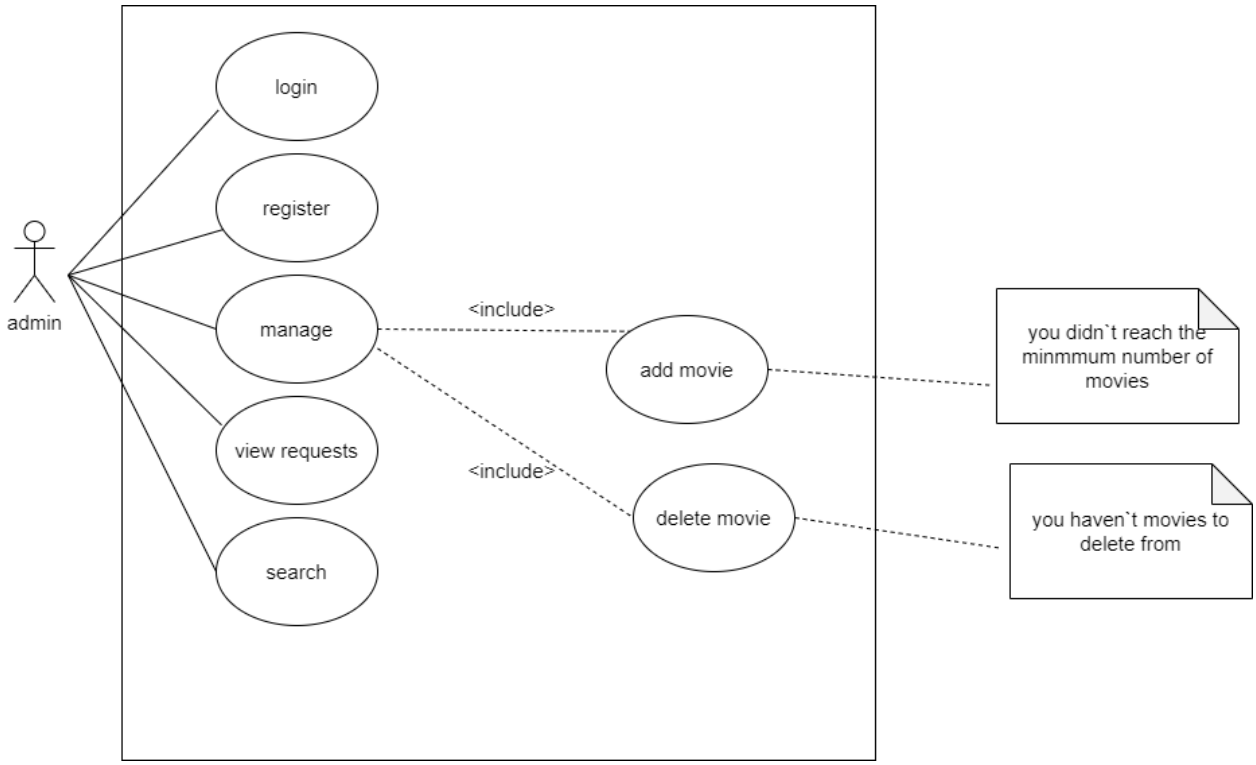


Use case diagrams:

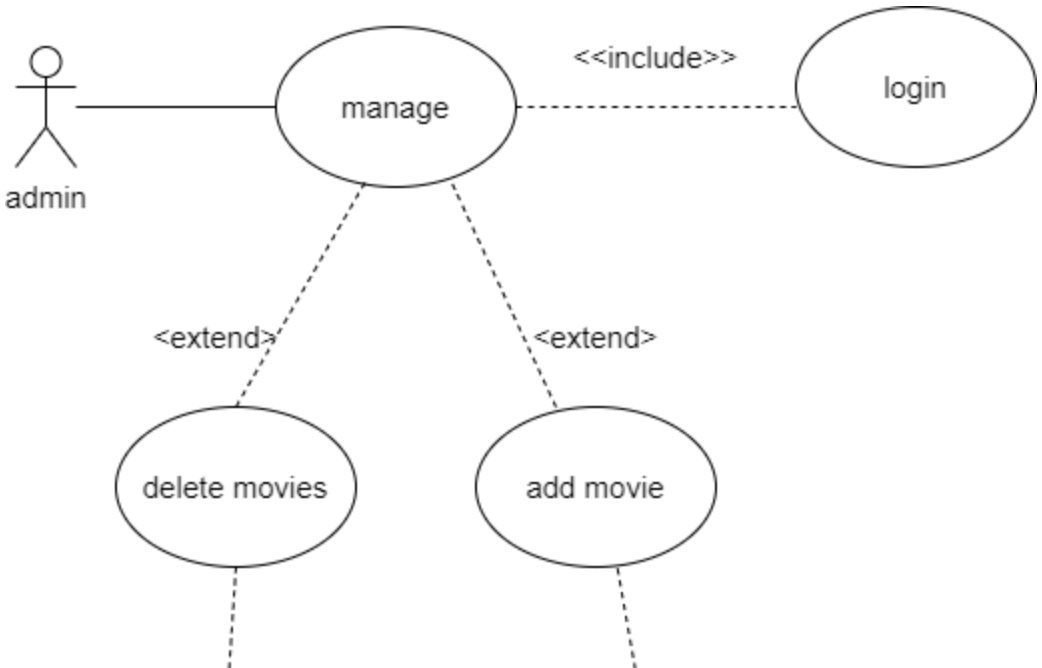
Environment:



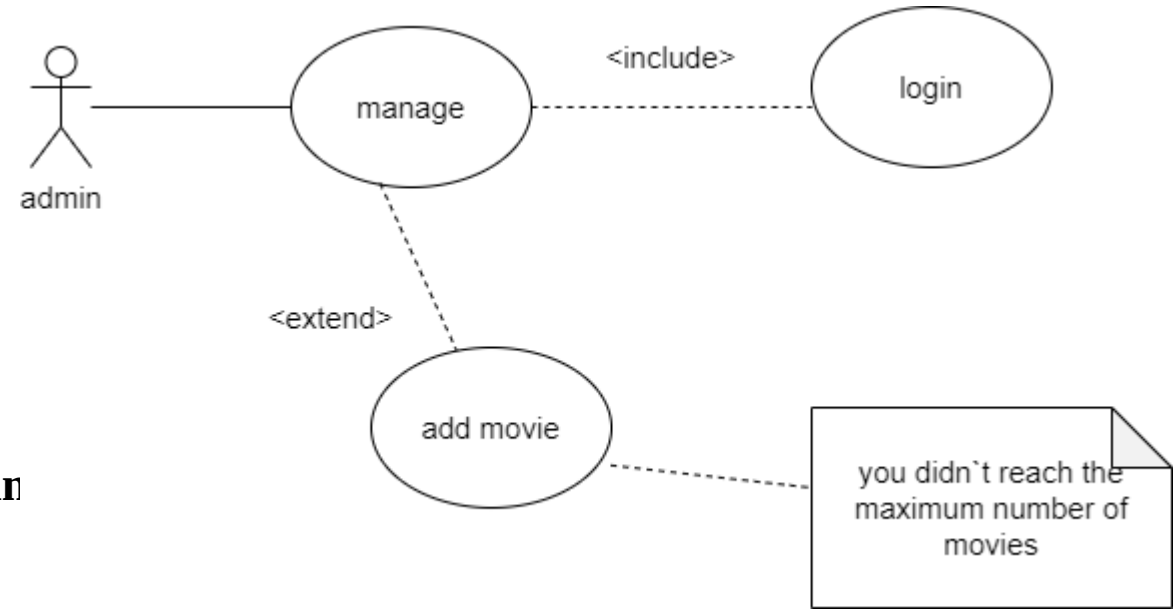
Admin:



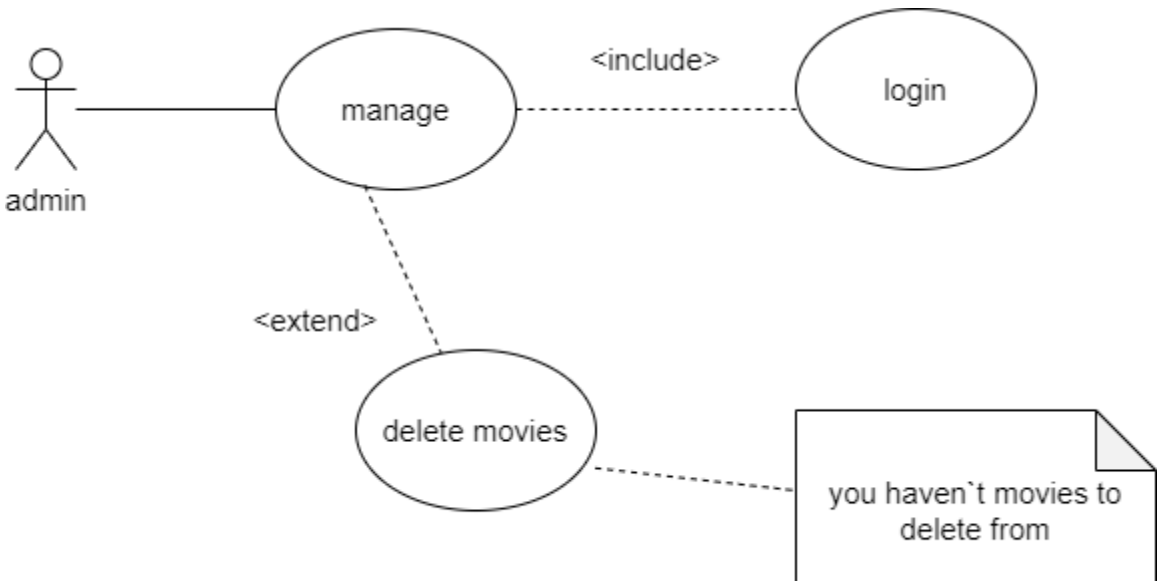
admin manage:



Admin add movie:

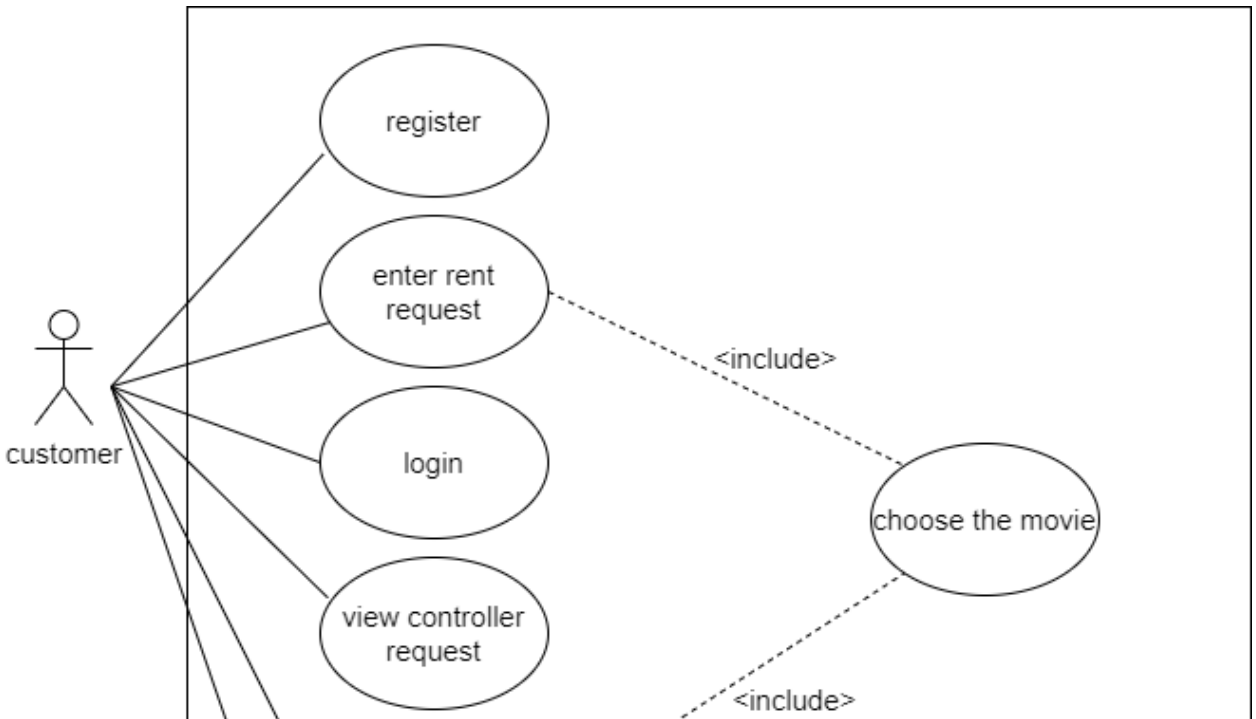


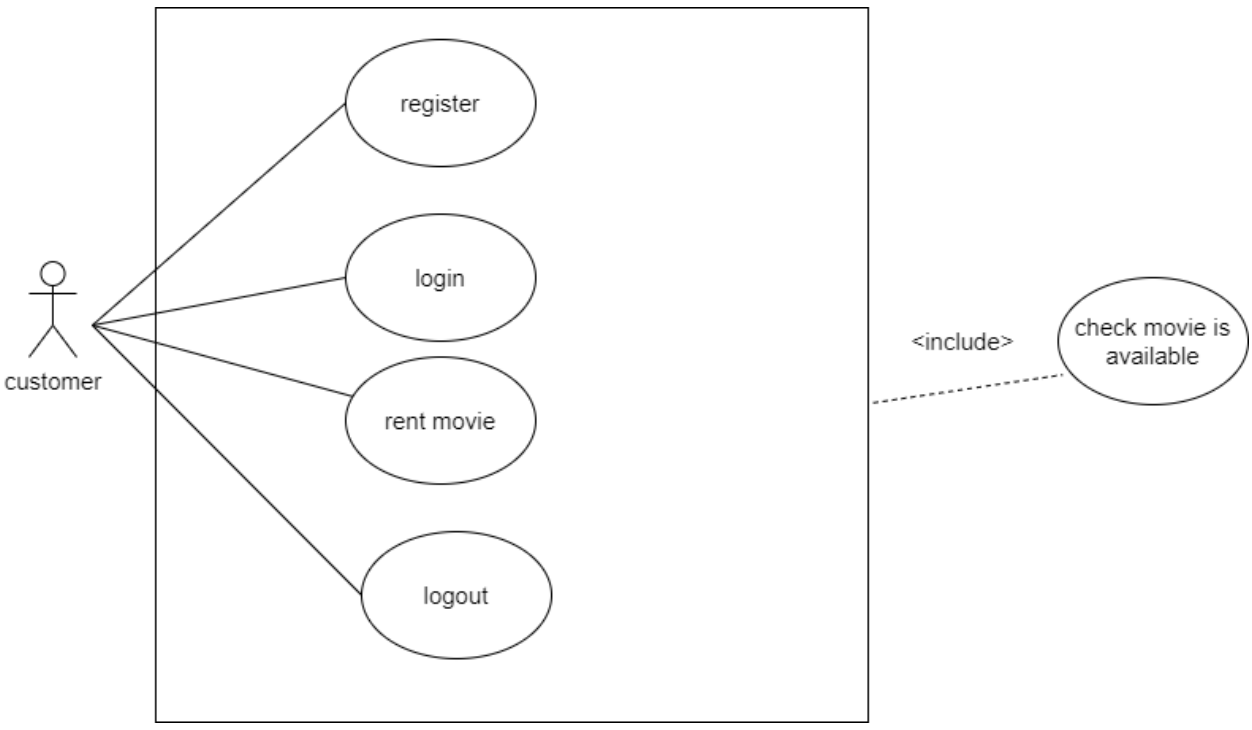
adr



Admin view reports:

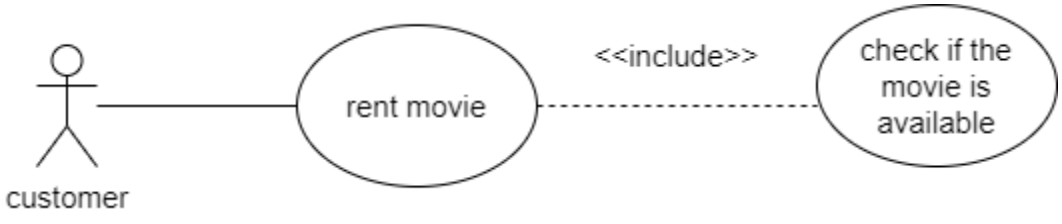
User:

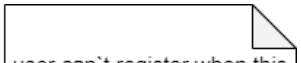



register
login
rent movie
logout
customer
<include>
check movie is available

User manage profile:

User rent movie:



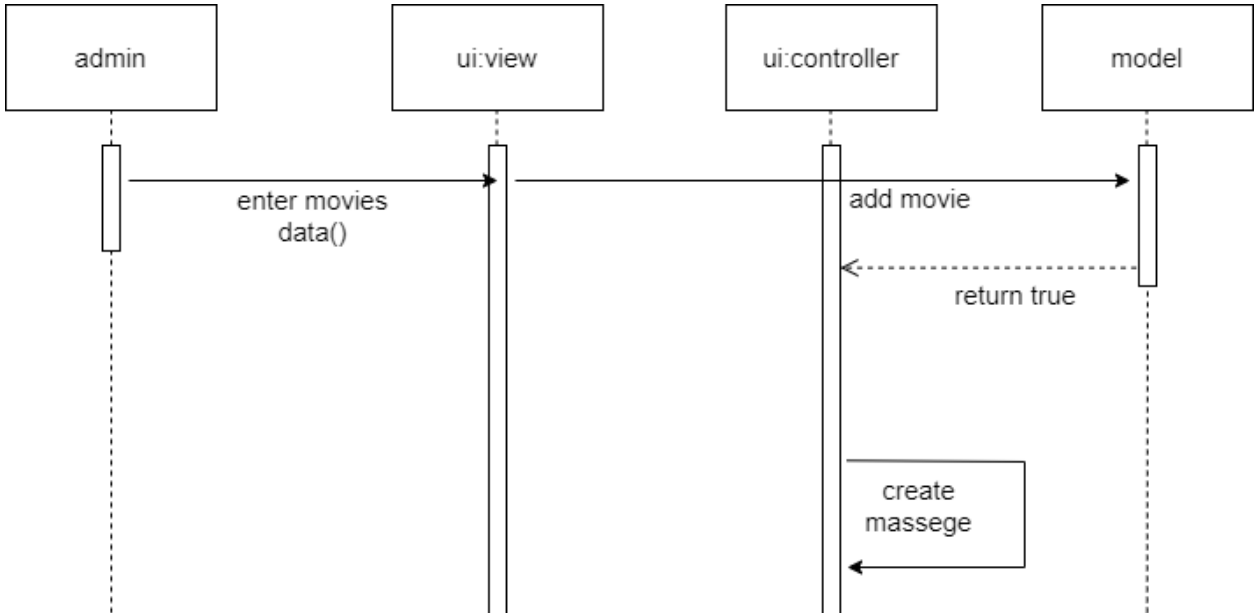


User register:

User login:

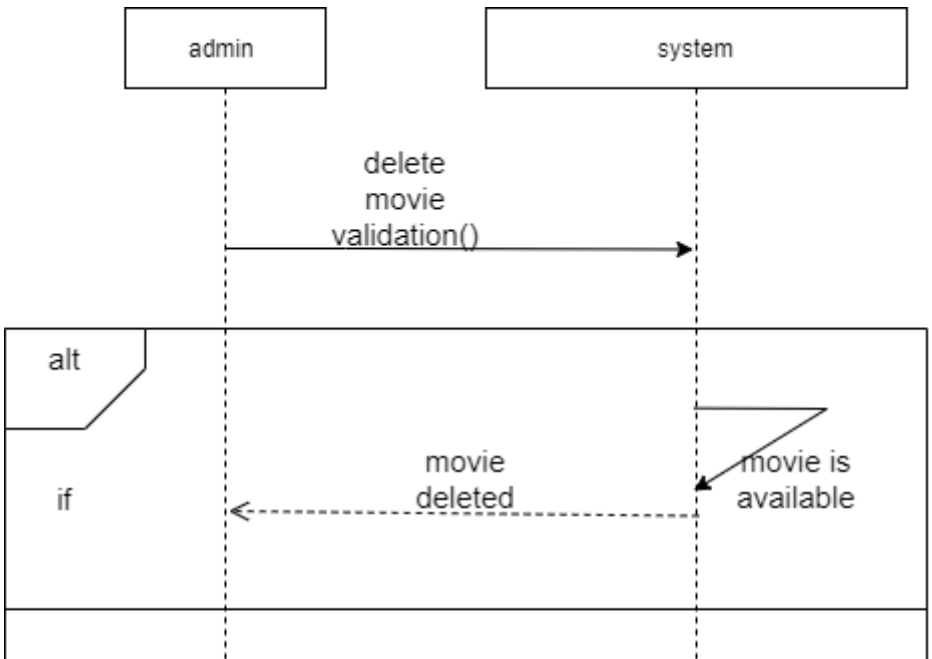
Sequence diagrams:

Admin add movie:



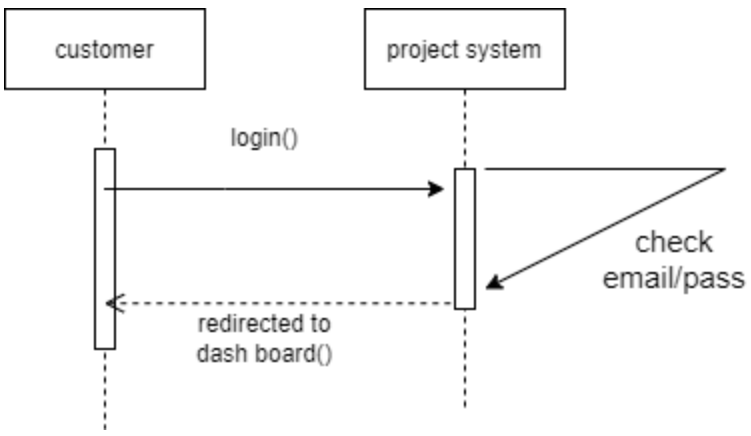
Admin view reports:

admin delete movie:

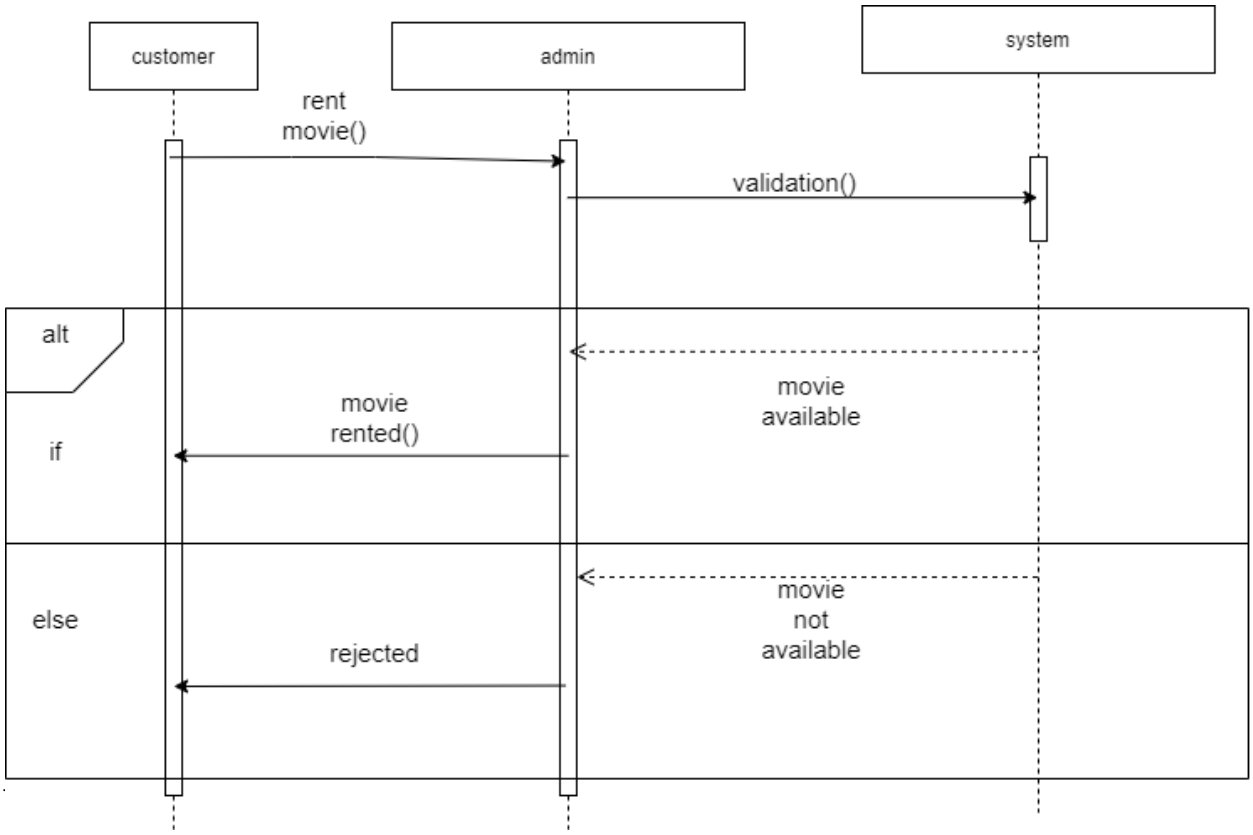


Customer:

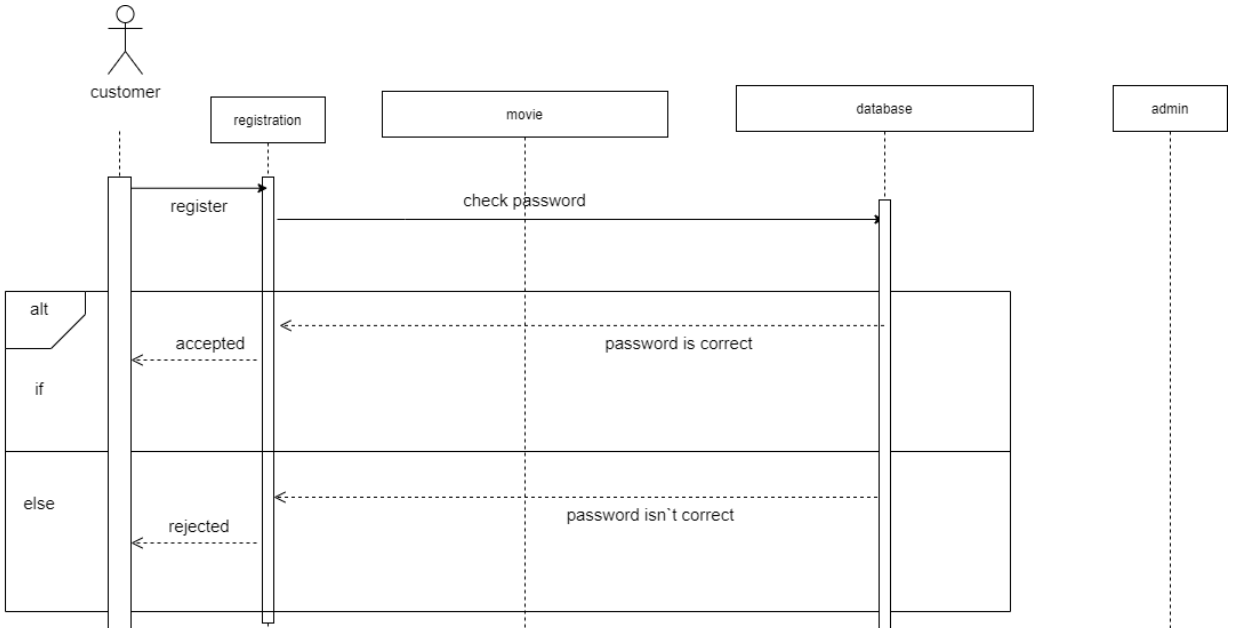
Customer login:



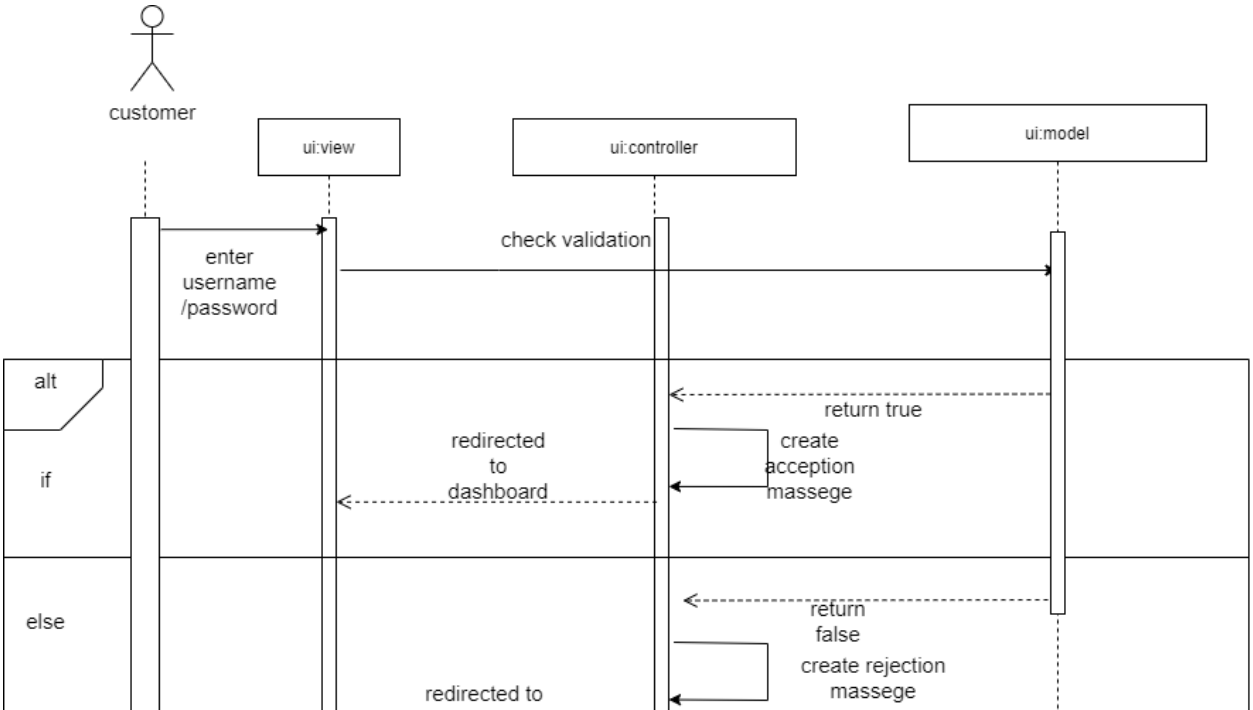
Customer rent movies:



C



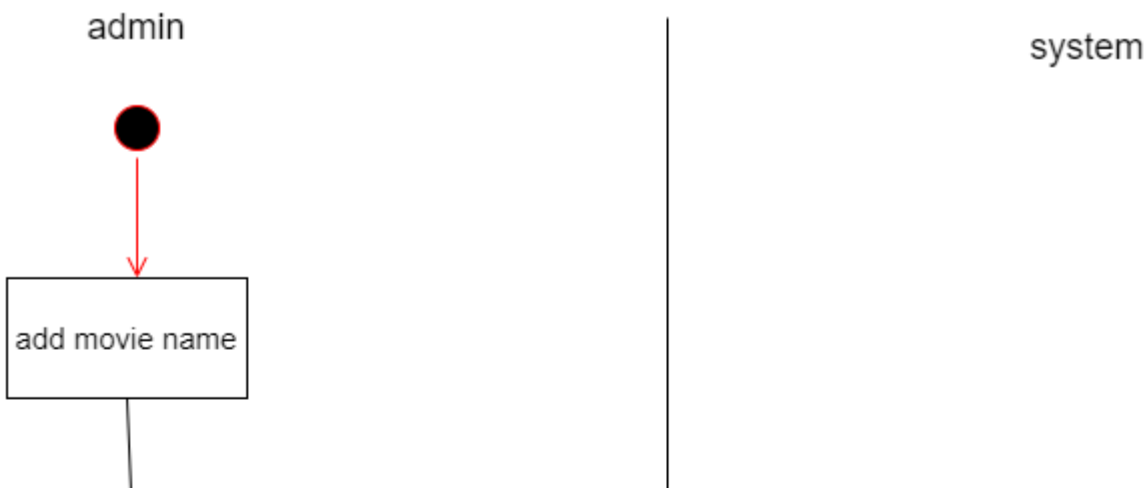
login customer/admin:



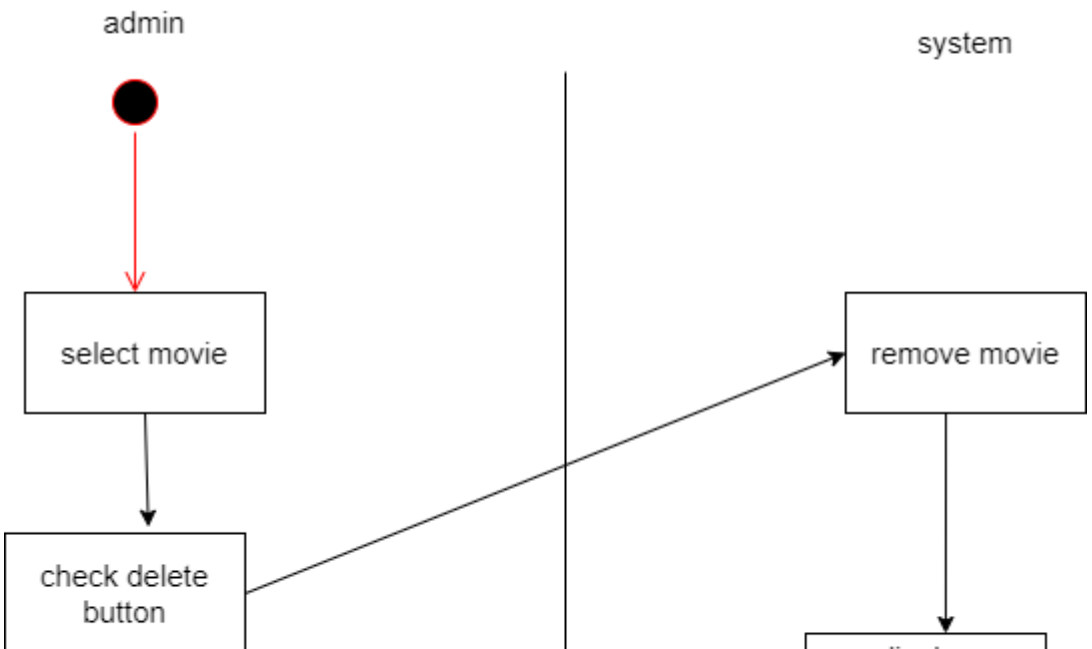
Activity diagram:

Admin:

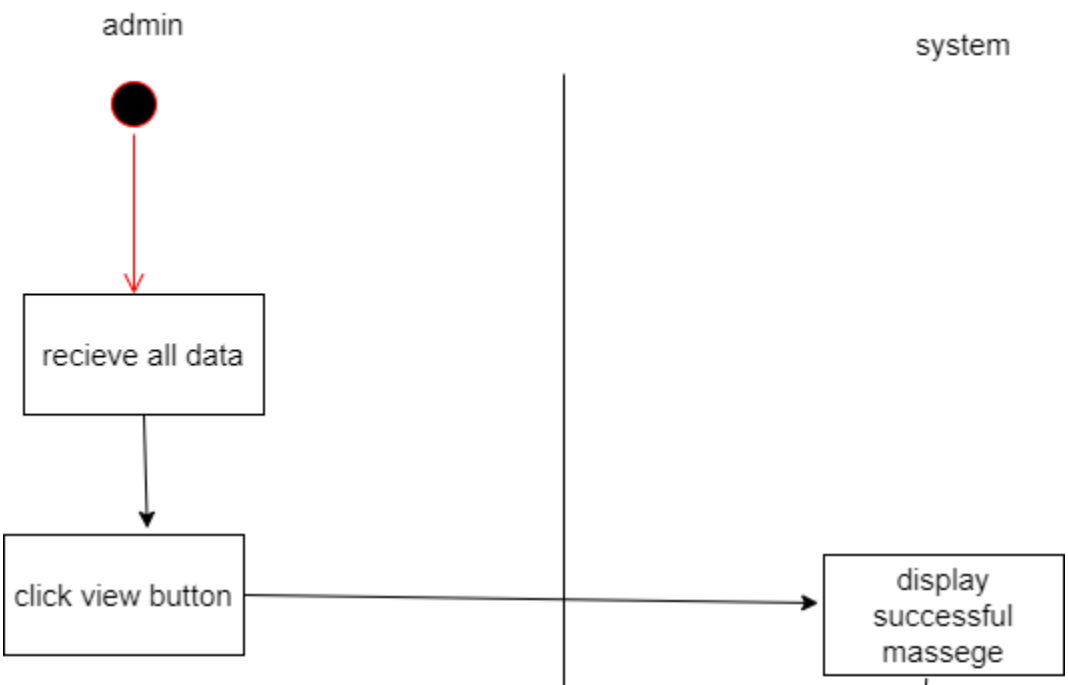
Admin add movie:



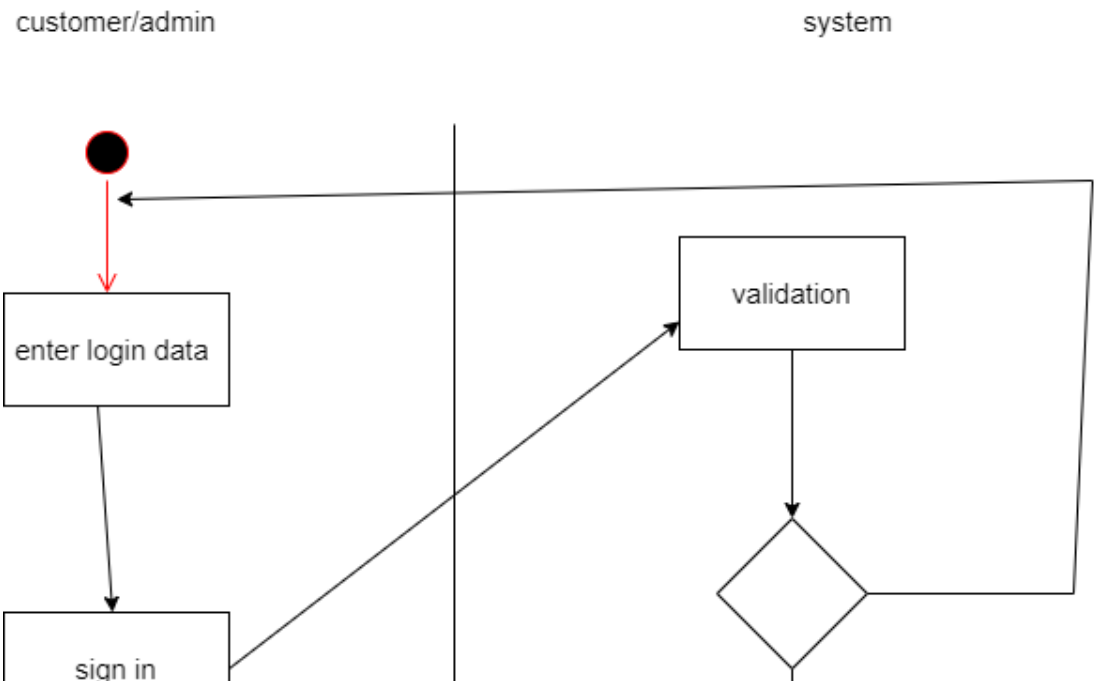
admin delete movie:



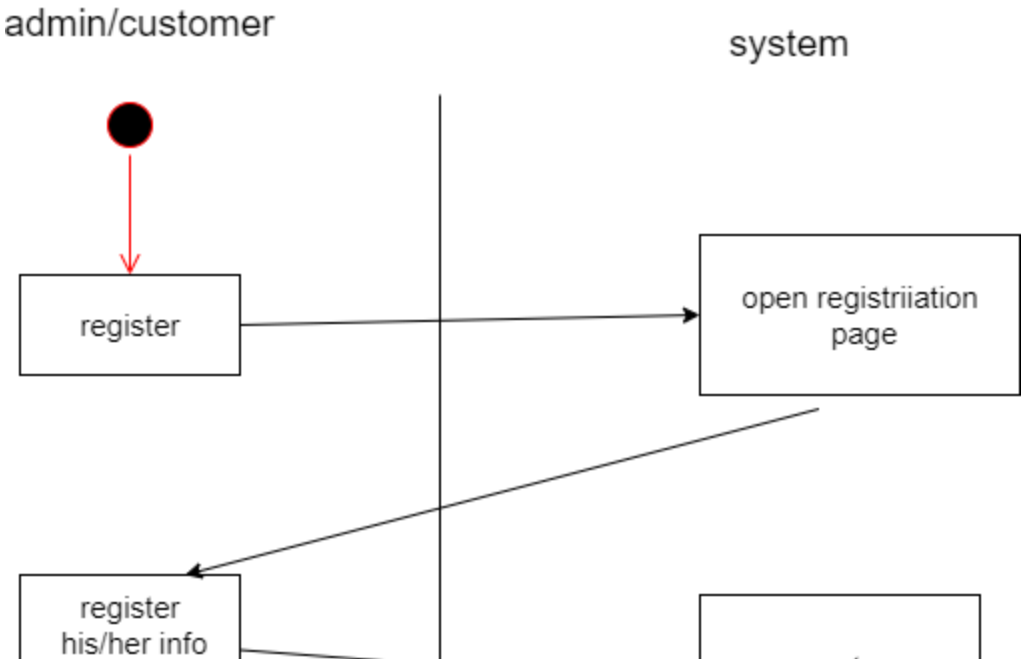
admin view reports:



Login customer/admin:



Register customer/admin:



Class diagram: