CS4125 Project Specification

group Team Leftovers.

Cinema System

Group Memebrs:

* Kristof Flaks 15169081
* Brandon Schahill 16177797
* Daniel Kennedy 14167077
* Oisin Hassett 15182304
* Tushar Mittal 16123921

***Table of Contents***

1. Business Scenario Page 3
2. List of Requirements Page 4
3. Software lifecycle discussion Page 5
4. Lightweight project plan Page 6
5. Requirements
   1. Use case diagrams Page 7 - 10
   2. Use case descriptions Page 11 - 15
   3. Non-Functional Requirements Page 16
   4. Tactics for supporting quality attributes Page 16
6. System Architecture
   1. Package Diagram Page 17
   2. Architectural discussion Page 17
7. Design
   1. List of candidate objects Page 18
   2. Analysis time class diagram Page 19
8. Coding Fragments Page 20
9. Workload Page 21&22
10. Added Value Page 23
11. Recovered Architecture blueprints Page 24
12. Critique of analysis time class diagram Page 25

***The Business Scenario.***

Our group is focusing on a primarily online based cinema company, who allows for booking of seats for their movies online with discounts if they apply as our business scenario. We chose this scenario because we think it is interesting to explore the different workings of a cinema and how the system would be built for it.

The scenario of business looks at how the system works for the manager of the company and the consumer side of the system which is what the costumers will interact with through the website of the company. The manager needs to be able to log into the system which is tailored to specifically be interacted with by the manager of the cinema. He must be able to order movie viewing licences from the movie distributers and to be able to show it to the customers. Once the manager gets a hold of the rights to show the movie and the movie itself, he must be able to select the viewing times that would be suitable for the movie that is ordered. The price of the tickets is also determined at this stage by the manager. He may also take recommendation from the distributers about the viewing time. The manager also decides how long the movie will be available on the listings, and the system should be able to automatically remove the movie once the outlined period is over.

For the customer, the system will interact with them entirely different to how it interacts with the manager. The customers should be able to log on as well, but into their own specific customer portal. In there, they should be able to view all the movies that are listed in the cinema and their viewing times respectively. The customer can then select the movie time which they want to attend, and from there be able to book tickets to that movie time slot. While the customer is making their booking, if they qualified for one, they shall receive a discount which will be automatically deducted from the tickets total value.

**List of requirements**

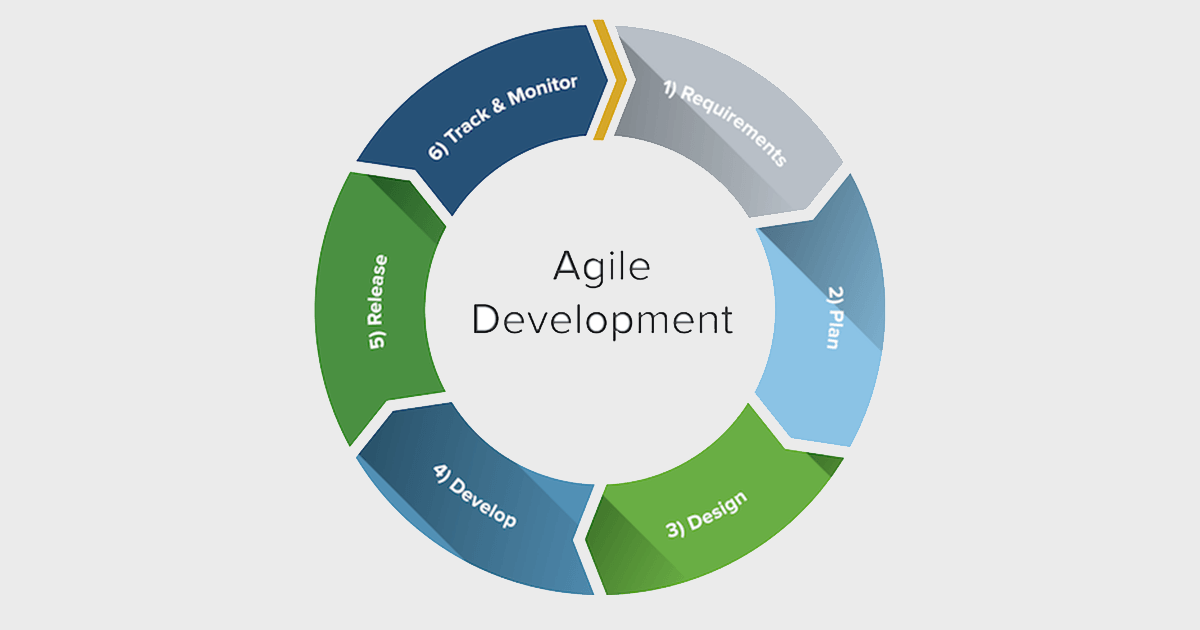
1. User can login/create an account.
2. User can select a movie for a specific time and date.
3. User can book a specific seat for a movie.
4. User can cancel their booking and request a refund.
   1. Must be within 24 hours of the movie showing.
5. Manger can login to program.
6. Manager can see a list of refunds requested and decide whether to approve them.
7. Manager can add/remove movies from the System.

**Agile Development**

Agile focuses on adapting the development process to match changing requirements.

Customer collaboration is vital to the development of the project. It focuses on the development stage. It puts working software above implementation. It is adaptive to change. It also encourages face to face communication as developers.

**2.**

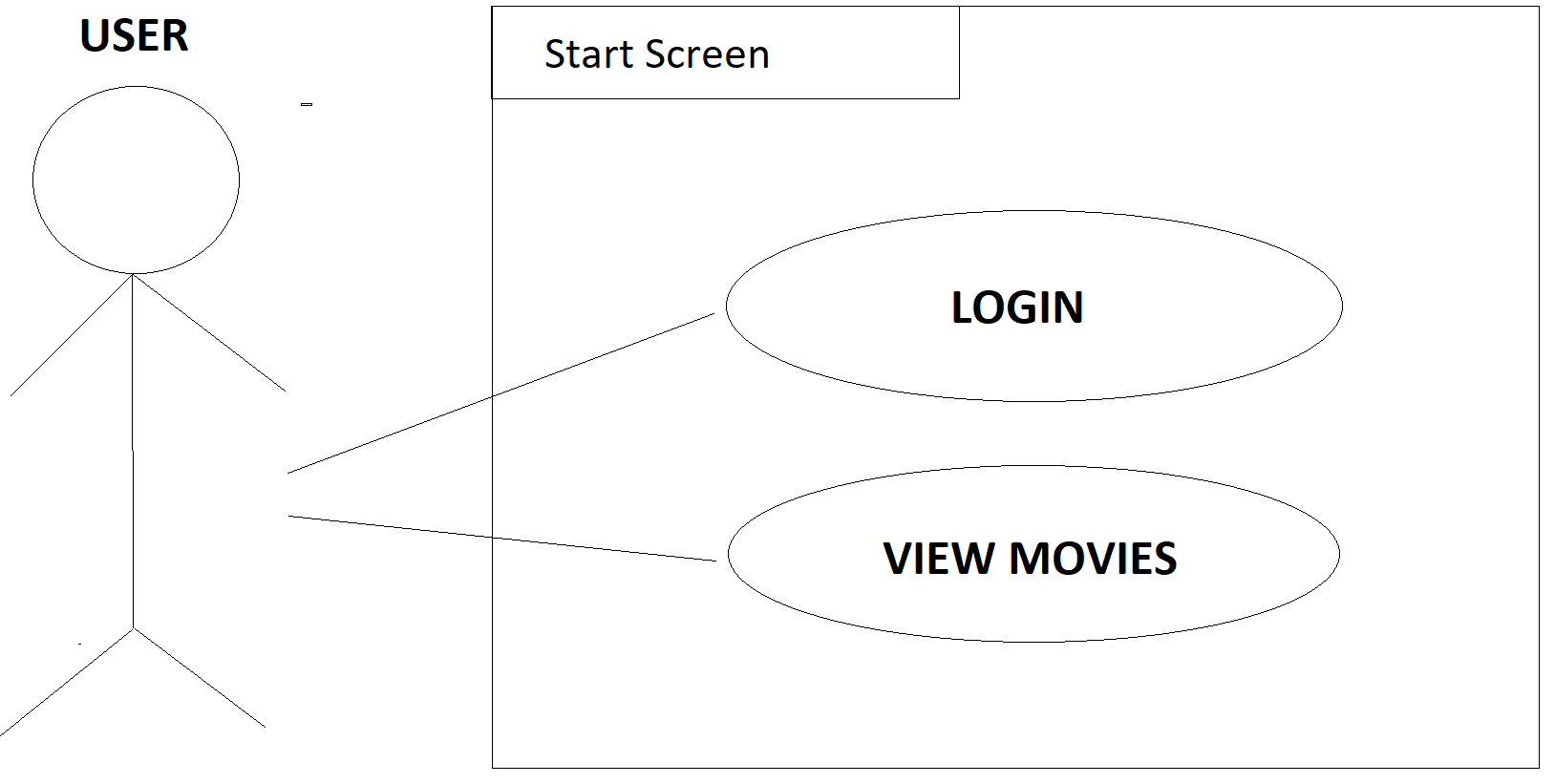
****

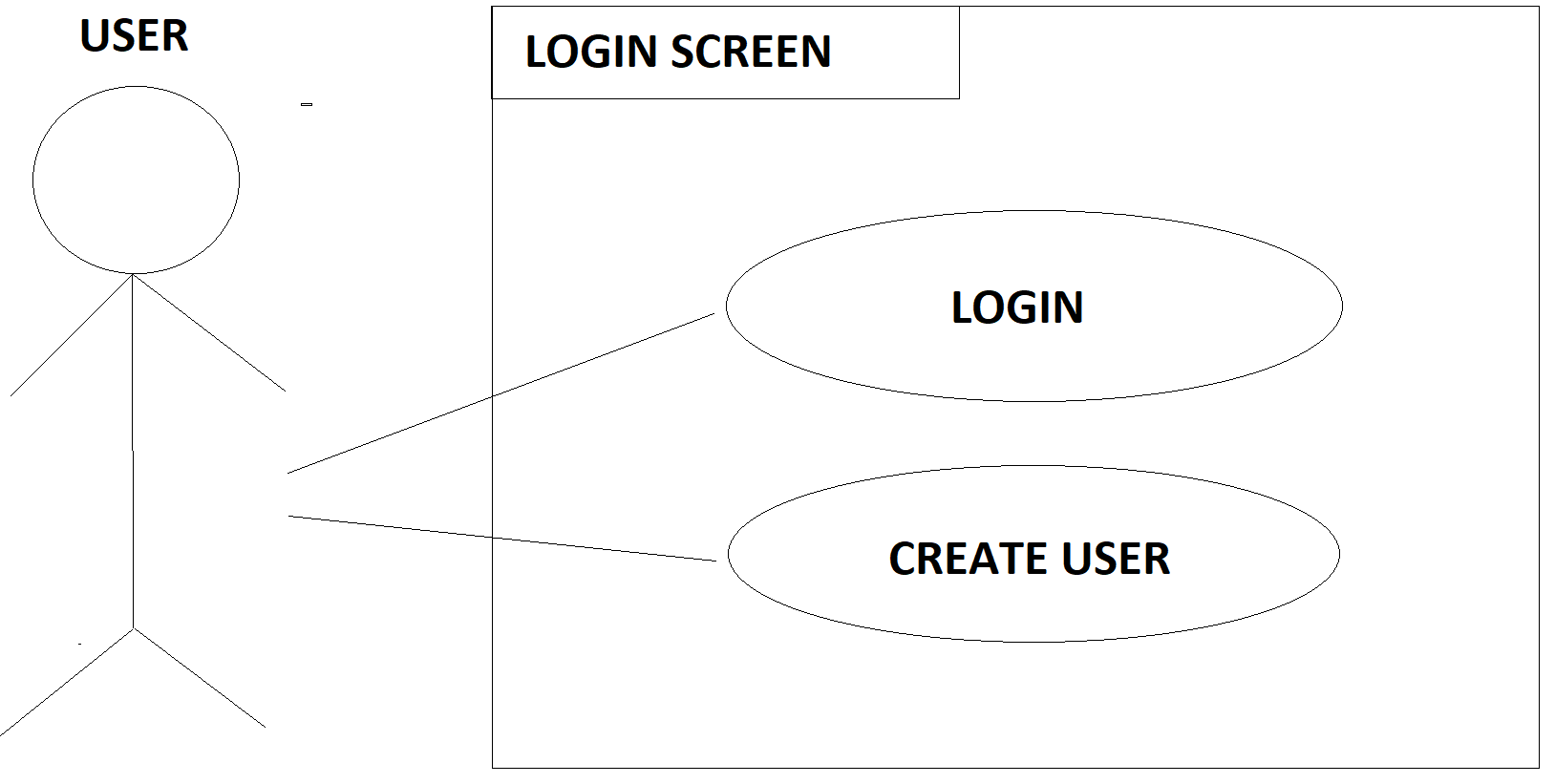
**Project Plan**

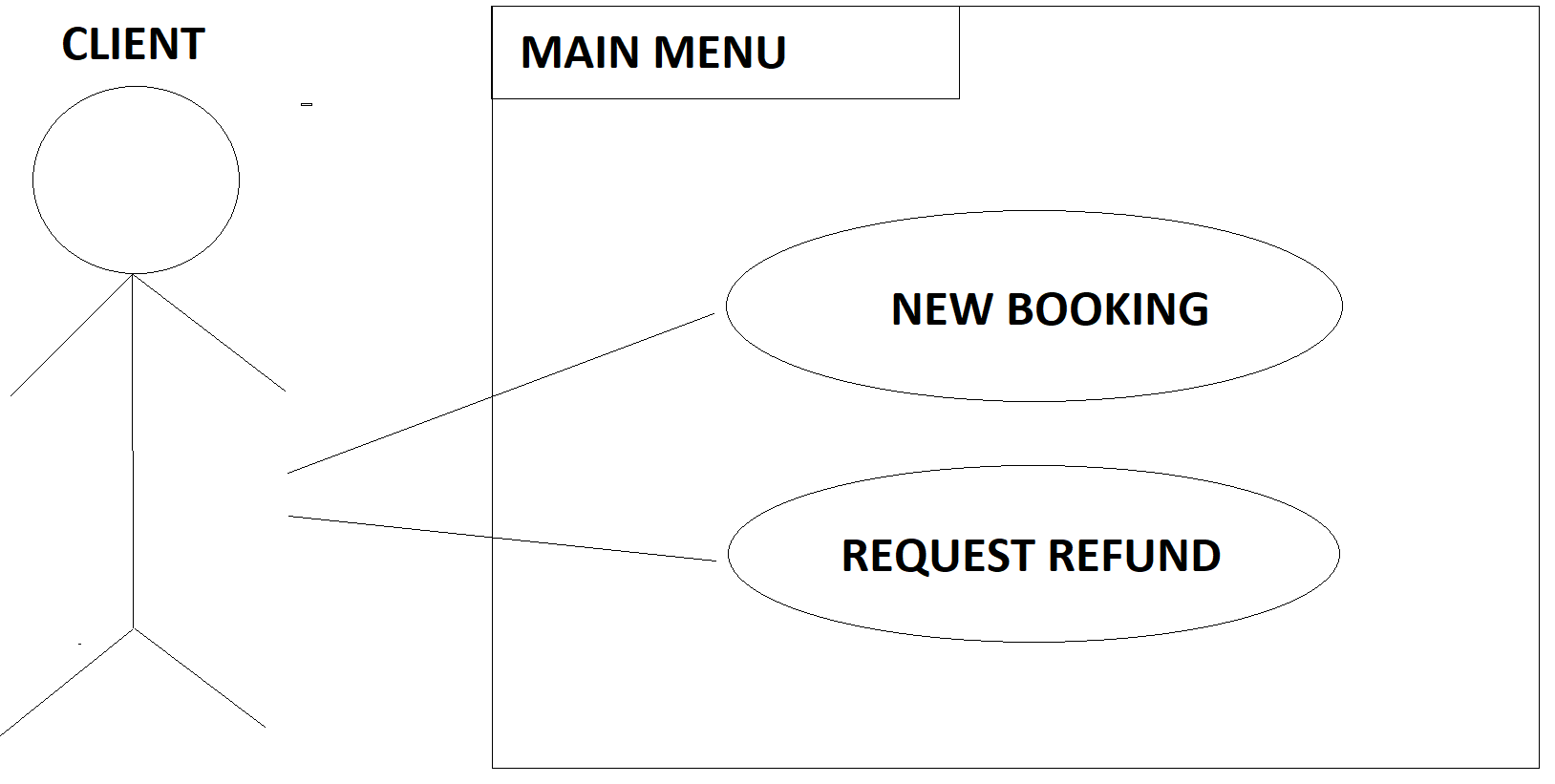
|  |  |  |  |
| --- | --- | --- | --- |
|  | **Role** | **Description** | **Responsible team member** |
| **1** | **Project Manager** |  | **Oisin Hasset** |
| **2** | **Documentation manager** |  | **Kristof Flaks** |
| **3** | **Business Analyst** |  | **Brandon Schahill** |
| **4** | **Architect** |  | **Daniel Kennedy** |
| **5** | **Systems Analysts** |  | **Tushar Mittal** |
| **6** | **Designer** |  | **Tushar Mittal** |
| **7** | **Technical Lead** |  | **Daniel Kennedy** |
| **8** | **Programmers** |  | **All** |
| **9** | **Tester** |  | **Oisin Hasset/Brandon Schahill** |
| **10** | **Dev Op** |  | **Kristof Flaks** |

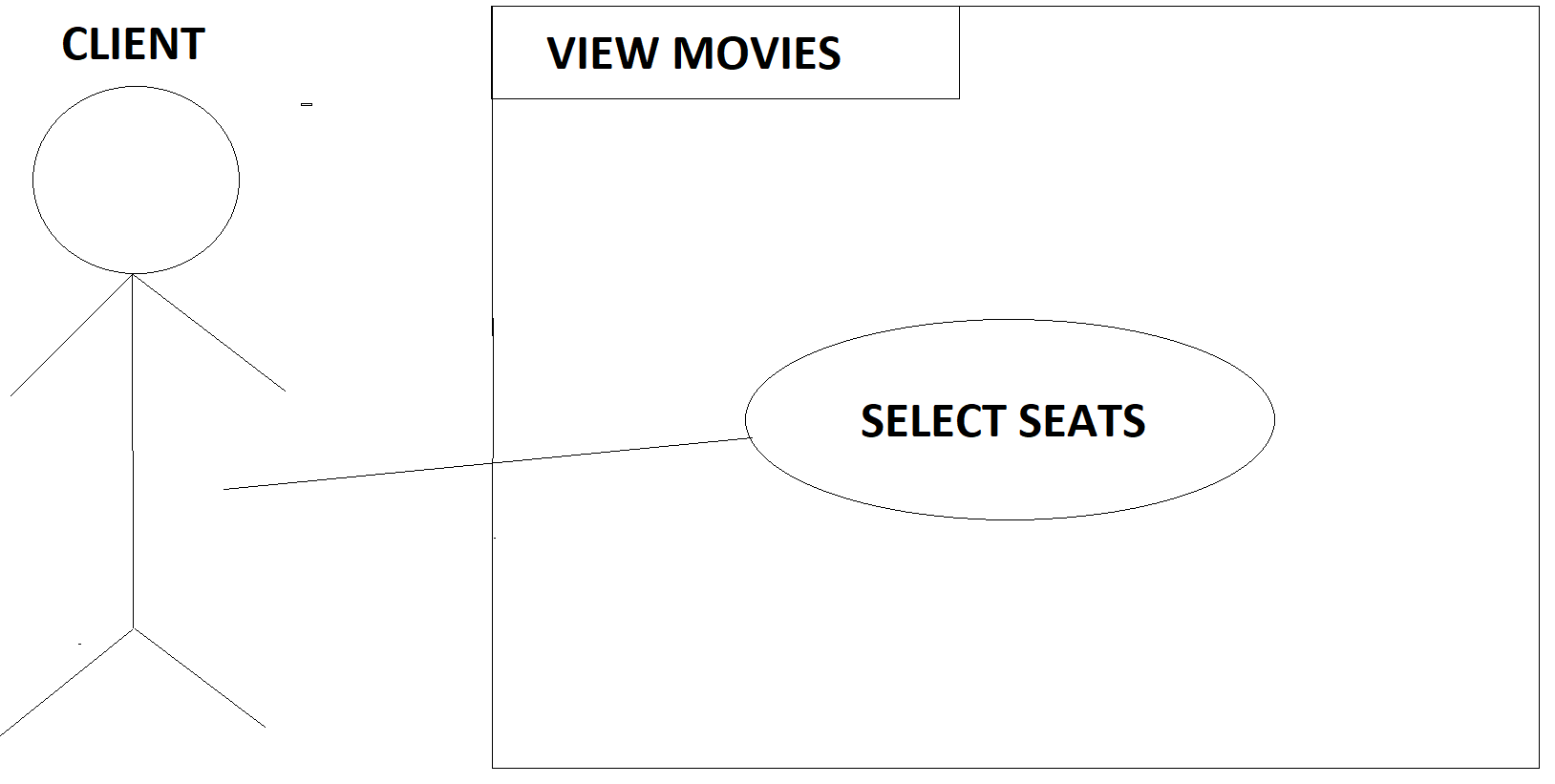
**Requirements**

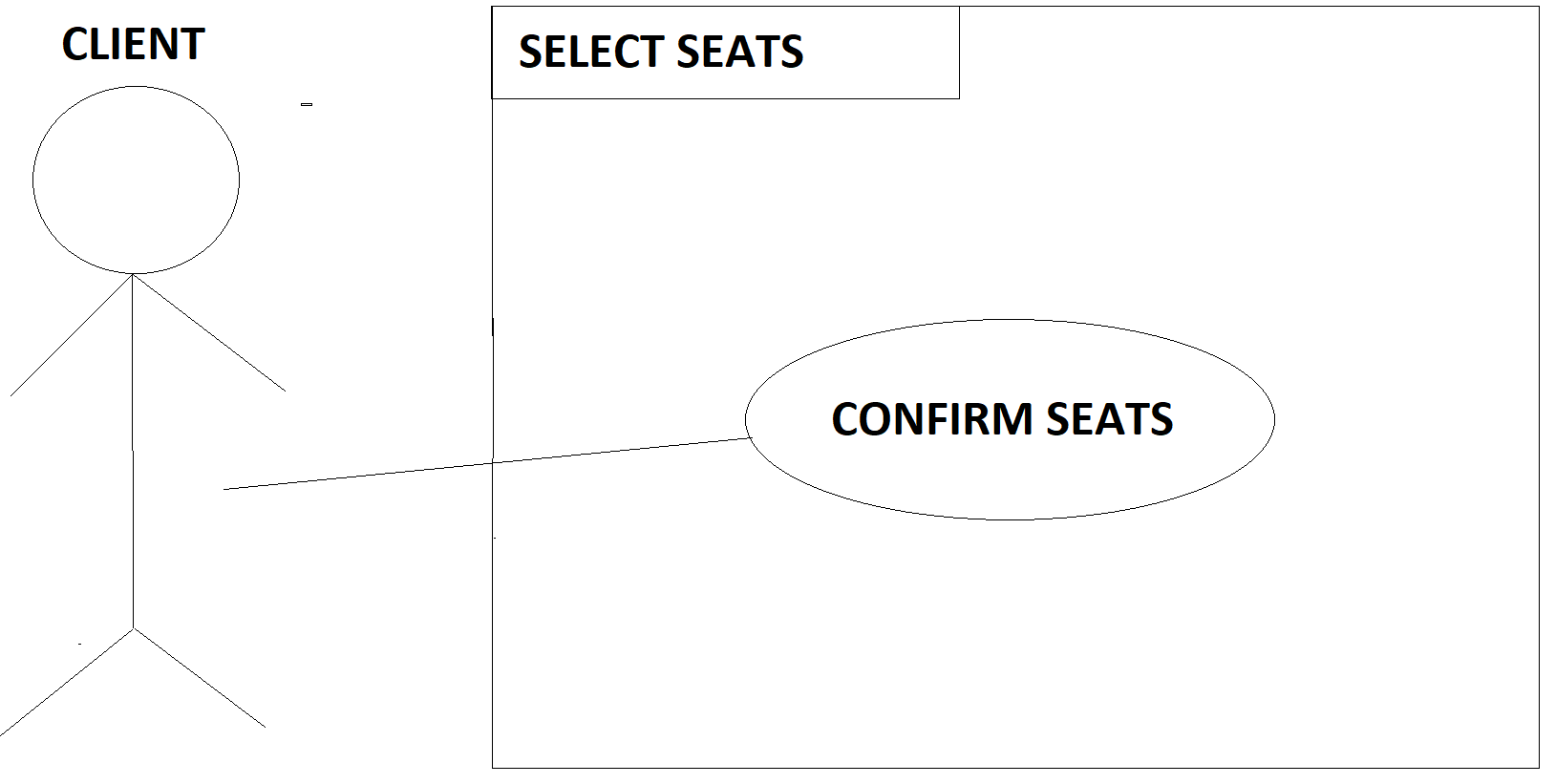
**Use case Diagrams**

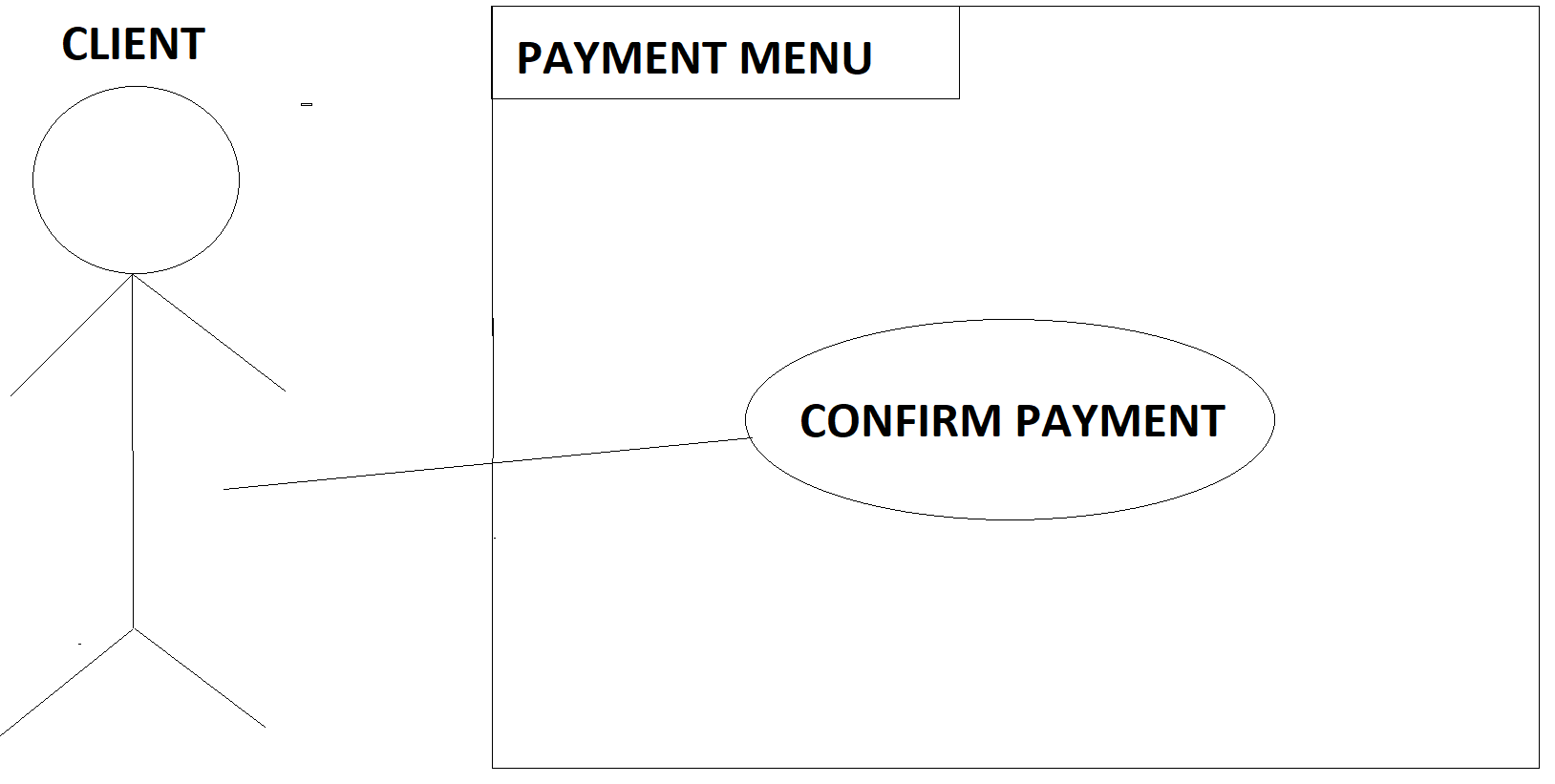


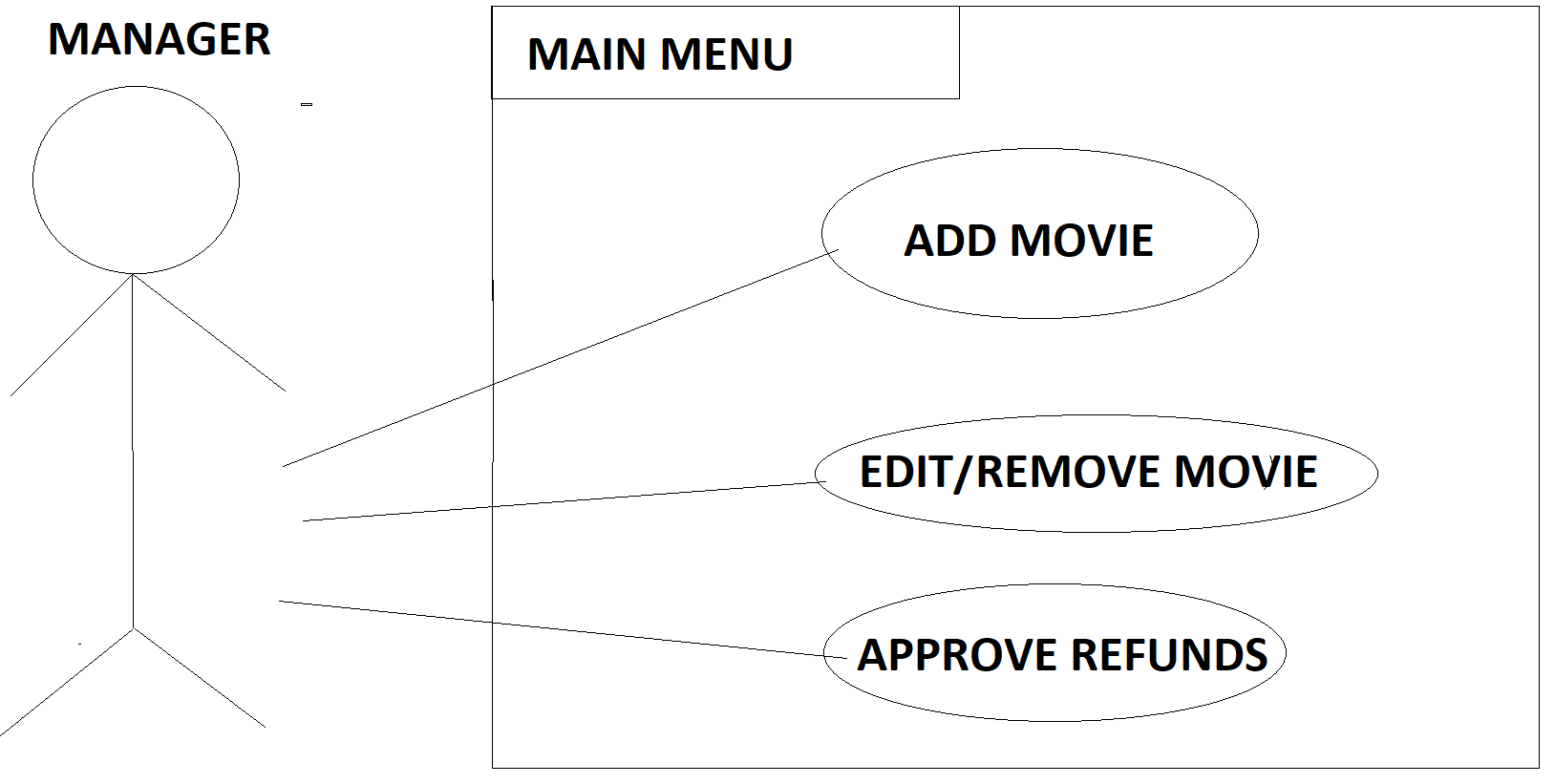


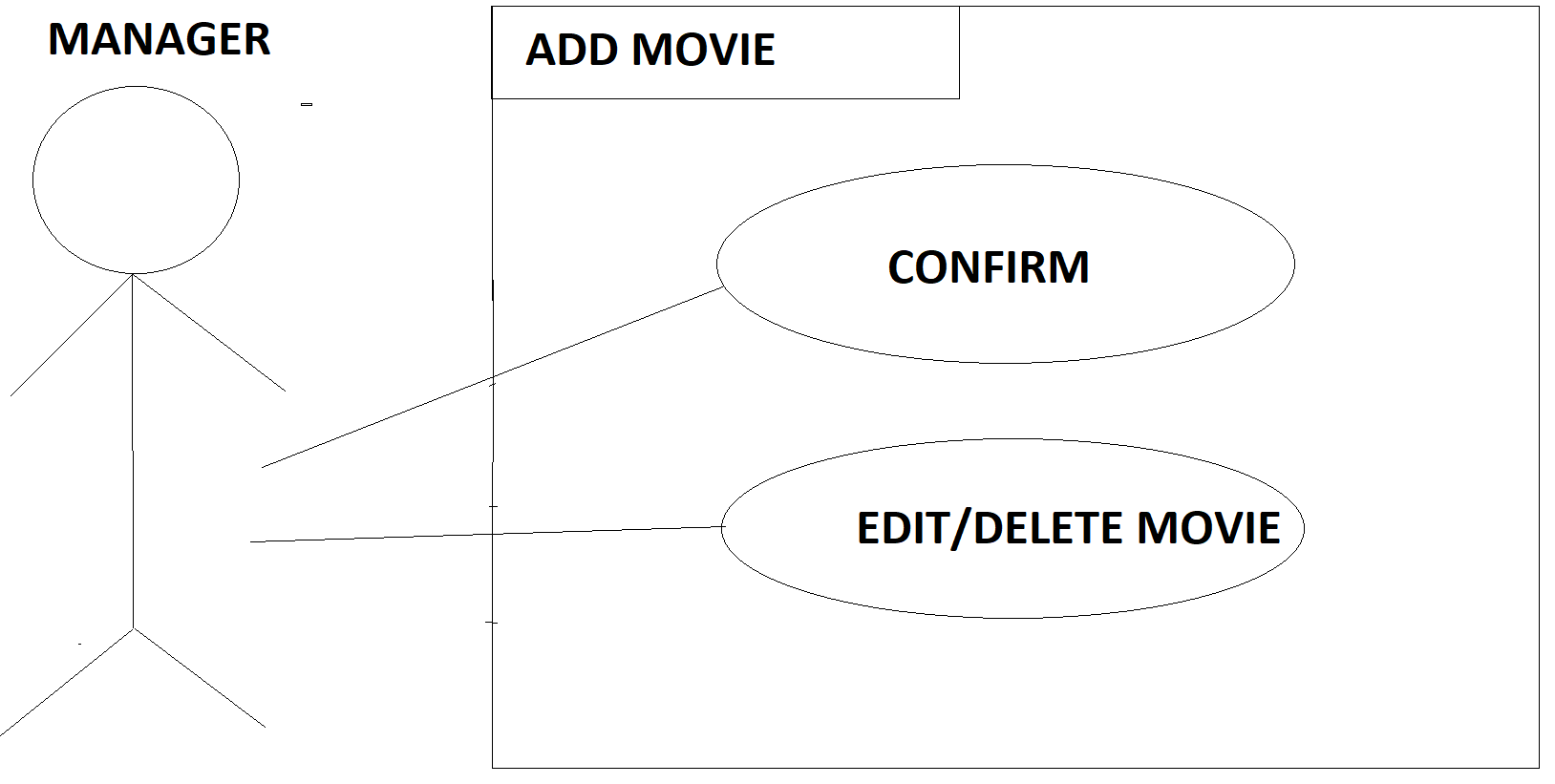


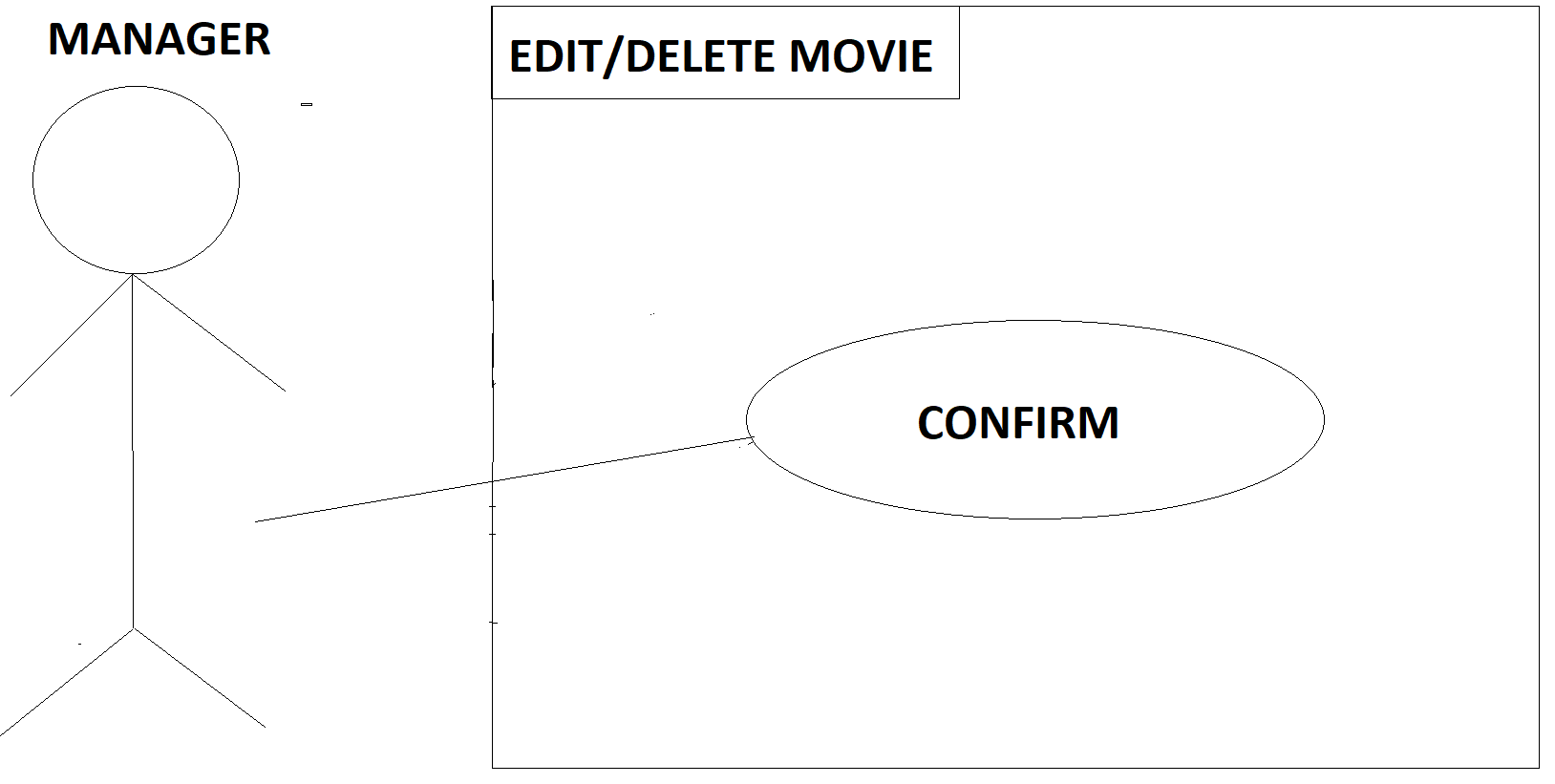


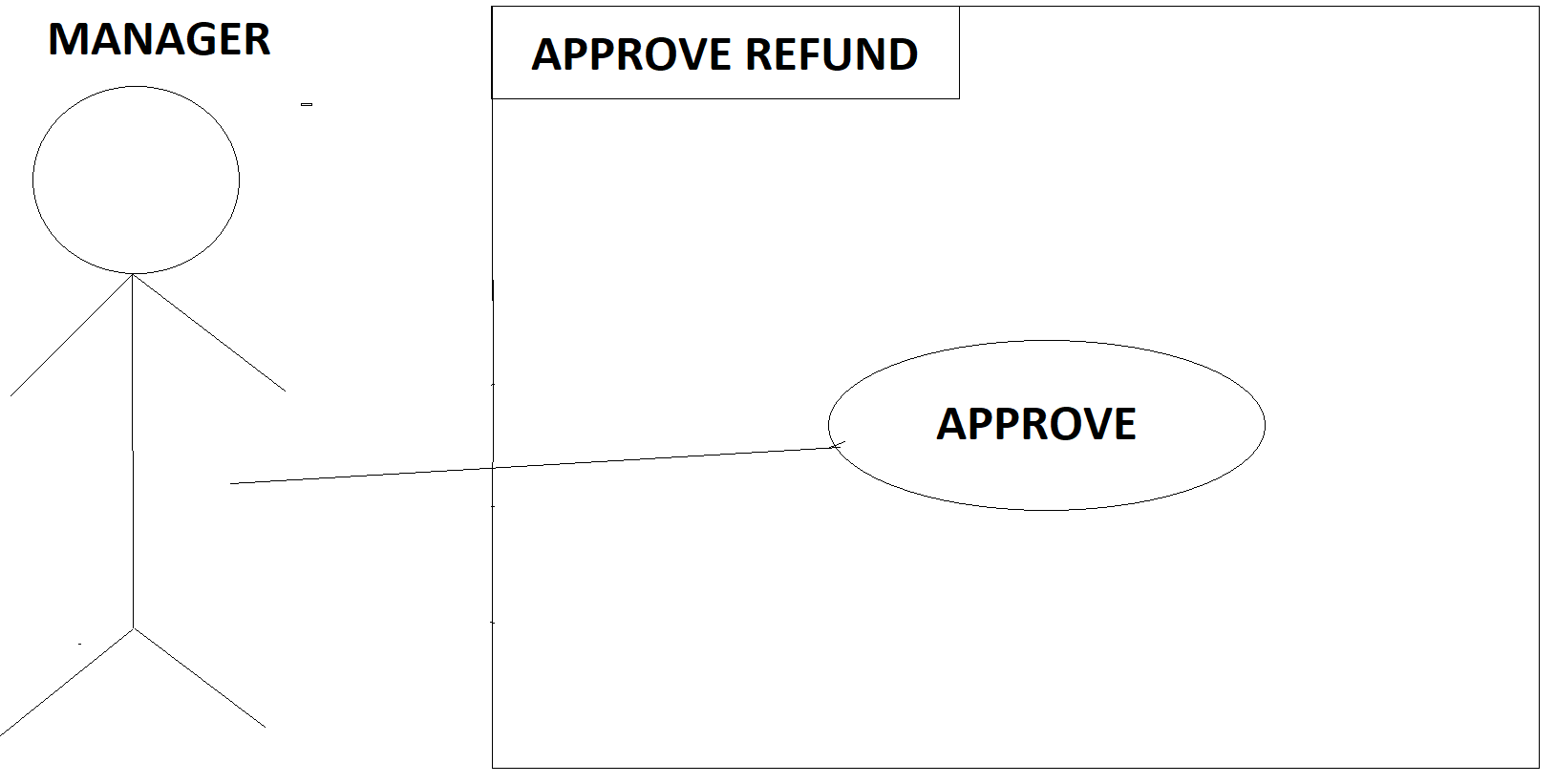












**Use Case Descriptions**

**Use Case 01: Login**

|  |  |
| --- | --- |
| **ACTOR ACTION** | **SYSTEM RESPONSE** |
| 1. Actor choses Login | 1. Displays Login Screen |
| 1. Actor Enters Login Details | 1. Displays Main Menu |

**Non-Functional dependency:** Security

The username and password should be encrypted and stored safely

**Alternate route:** If user enters in wrong details the system shows an error to the user and asks to enter the details again.

**Use Case 02: Create User**

|  |  |
| --- | --- |
| **ACTOR ACTION** | **SYSTEM RESPONSE** |
| 1. Actor choses create user on the login screen | 1. Displays create user screen |
| 1. Actor enters in their details | 1. System checks if the username is available if it is then it registers the user |

**Actors:** Client

**Non-Functional dependency:** Security and performance

The user data should be encrypted and stored safely.

The system should check existing username as fast as possible.

**Alternate Route:** If the user enters an already existing username the system prompts the user to enter a new username.

**Use Case 03: New Booking**

|  |  |
| --- | --- |
| **ACTOR ACTION** | **SYSTEM RESPONSE** |
| 1. Actor clicks on new booking from main menu | 1. Screen with list of movies is shown |
| 1. Actor selects the desired movie and show time | 1. System Displays the seats for that movie. |

**Actors:** Client

**Non-Functional Dependency:** Performance and Accuracy

System should load the data for movies quickly and the switch between display movie screen and display seats should be fast.

**Alternate Route:**

**1.A** If the user clicks back button main menu is displayed

If the user doesn’t select all the fields system prompts the user.

**Use Case 04: Confirm Seats**

|  |  |
| --- | --- |
| **Description** | This use case describes the process and interaction between client and system when client decides to confirm his/her seats. |
| **Preconditions** | At least one seat is selected to confirm. |
| **Postconditions** | Screen requesting payment details is displayed. |
| **Success End Condition** | The seats for the movie have been reserved and the payment has been completed. |
| **Failed End Condition** | The System was unable to reserve the database/ All seats were already reserved or the payment was unsuccessful. |
| **Actors** | Client. |
| **Trigger** | Actor has selected the desired movie he/she wants to make the bookings for. |

|  |  |  |
| --- | --- | --- |
| **Description** | **Step Number** | **Action** |
|  | 1. | Client selects the movie he/she wants to make the booking for. |
|  | 2. | Client selects the desired seats. |
|  | 3. | Client clicks on confirm seats. |
| **Extension** | **Step Number** | **Branching Condition** |
|  | 2.a | User clicks go back to View Movie screen. |
|  | 2.b | User isn’t logged in. So, login screen is displayed to the Client before the booking can be confirmed. |
| **Variation** | **Step Number** | **Branching Condition** |
|  | 2.a | All the seats have already been booked. |
|  | 2.b | Unable to load the seats. |

**Use Case 05: Add Movie**

|  |  |
| --- | --- |
| **Description** | This use case describes the process and interaction between manager and system when manager wants to add new movie. |
| **Preconditions** | Actor has successfully logged into the computer and Actor is manager. |
| **Postconditions** | Movie is added into the database with the details provided by the manager. |
| **Success End Condition** | Movie is added into the database and the main menu is displayed for the manager. |
| **Failed End Condition** | System is unable to update the database because movie already exists. Error is displayed to the user. |
| **Actors** | Manager. |
| **Trigger** | Actor selects add new movie from the main menu. |

|  |  |  |
| --- | --- | --- |
| **Description** | **Step Number** | **Action** |
|  | 1. | Manager choses add movie from the main menu. |
|  | 2. | User enters the movie details. |
|  | 3. | User enters the movie showtimes. |
|  | 4. | User clicks confirm button after entering all the details. |
| **Extension** | **Step Number** | **Branching Condition** |
|  | 4.a | User clicks view all show times from the add movie menu. |
|  | 4.b | User clicks on edit movie from the add movie menu and is bought to the edit movie screen. |
| **Variation** | **Step Number** | **Branching Condition** |
|  | 2.a | User enters the name for an already existing movie. |
|  | 2.b | User enters an already taken show time. |

**Use Case 06: Edit Movie/Delete Movie**

|  |  |
| --- | --- |
| **ACTOR ACTION** | **SYSTEM RESPONSE** |
| Actor clicks edit movie on the main menu or from the add movie screen. | Edit movie menu is displayed. |
| Actor edits the movie details. | The details for the movie are updated in the database. |

**Actors:** Manager

**Non-Functional dependency:** Performance and Accuracy

System must be fast in checking if the showtimes are already taken.

**Alternate Route:**

A: If the user clicks back button main menu is displayed.

B: If the user Deletes a movie from the database automatic refunds should be made to the clients for that movie.

**Use Case 07: Display Movies**

|  |  |
| --- | --- |
| **ACTOR ACTION** | **SYSTEM RESPONSE** |
| 1. Actor Choses Display movies on start Screen | 1. System Displays Movie screen |

**Trigger:** Start Screen, Case 03, Case 06, Case 07

**Actors:** Client and Manager

**Non-Functional dependency:** Performance

**Alternate Route:** User selects back button,

**1A.** If the user is logged in,then main menu should be displayed.

**1B.** otherwise start screen is displayed.

**Use Case 08: Confirm Booking**

|  |  |
| --- | --- |
| **ACTOR ACTION** | **SYSTEM RESPONSE** |
| 1. Actor clicks on confirm seats. | 1. New Screen is displayed requesting the payment details. |
| 1. Actor enters the payment details. | 1. System checks if the payment has been received and then updates the database with the new booking. |

**Trigger:** After use case 05 is successful

**Non-Functional dependency:** Security and performance

The user payment details should be encrypted, and the system should reserve the seats for the user fast and accurately.

**Alternate Route:**

1A If the user is not logged in user is displayed the login screen and message is displayed “please login to confirm booking.”

4A.If the payment details are not current the user is prompted an error stating payment was unsuccessful, try again.

**Use Case 09: Request Refund**

|  |  |
| --- | --- |
| **ACTOR ACTION** | **SYSTEM RESPONSE** |
| Actor clicks on request refund from booking/Request refund screen | Screen with user’s old bookings is displayed |
| Actor selects the desired movie which he/she wants to request refund for | System updates the database with the refund request |

**Trigger:** User has already made a booking.

The option is not displayed if the user has no existing bookings

**Actors:** Client

**Alternate Route:** Refunds can only be requested until 24hours before showtime. Thus, if the user tries to request refund for the same. The user is displayed a message stating that the refund can’t be requested.

**Use Case 10: Approve refunds**

|  |  |
| --- | --- |
| **Description** | This use case describes the process and interaction between manager and system when manager decides to approve requested refunds |
| **Preconditions** | Actor has successfully logged into the computer and Actor is manager.  And there exists refund requests. |
| **Postconditions** | The refund has been successful. |
| **Success End Condition** | The refund has been successful, and it updates system by removing the reservation and emptying the seats. |
| **Failed End Condition** | The refund wasn’t successful, or system was unable to update the database. |
| **Actors** | Manager. |
| **Trigger** | Actor selects approve refunds from the main menu. |

|  |  |  |
| --- | --- | --- |
| **Description** | **Step Number** | **Action** |
|  | 1. | Manager choses approve refunds from the main menu. |
|  | 2. | Manager selects the refund he wants to approve. |
|  | 3. | Manager clicks on approve the selected refund. |
| **Extension** | **Step Number** | **Branching Condition** |
|  | 2.a | User clicks go back to main menu |
|  | 2.b | User clicks on automatic refund for selected movie. A pop-up menu is displayed for all the movies and the user can select the movie and make refund for all customers. |
| **Variation** | **Step Number** | **Branching Condition** |
|  | 2.a | Refund was requested too late to be approved. |
|  | 2.b | Refund was unsuccessful. |

**Use Case 10: Edit Prices**

|  |  |
| --- | --- |
| **ACTOR ACTION** | **SYSTEM RESPONSE** |
| Actor clicks on edit pricing | Screen with user’s old bookings is displayed |
| Actor selects the desired movie which he/she wants to request refund for | System updates the database with the refund request |

Non-Functional Requirements

* Security
* Performance
* Extensibility

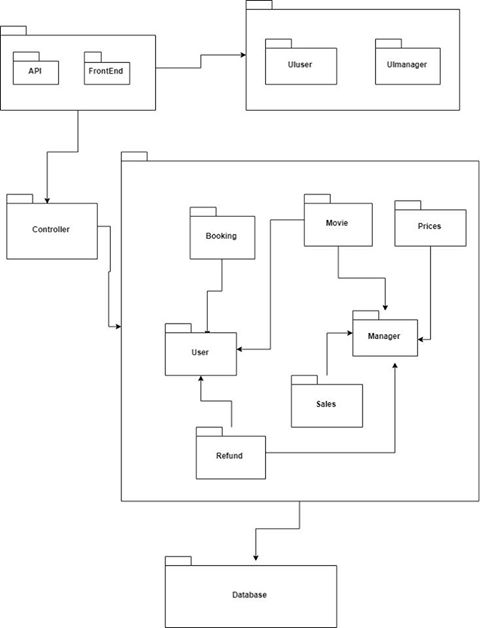
**Tactics to consider to support quality attributes**

**Security**: There are many open source frameworks which facilitate security features. Such as Acegi Security or ki. Ki is flexible, open source framework which handles authorization, enterprise session management and cryptography.

**Performance**: A tactic one might consider improving performance is to use a framework such as Sof4j. This contains many utilities and components to fix performance issues.

**Extensibility**: The use of design patterns such as the state design pattern as well as the factory design pattern will facilitate the extensibility of this program.

System Architecture



We split our Architecture into three tiers. The UI tier, the business tier and the database tier. We have controller classes which act as middle men to allow all three tiers to communicate with each other.

List of Candidate objects:

User

Booking

Movie

Refund

ShowingTime

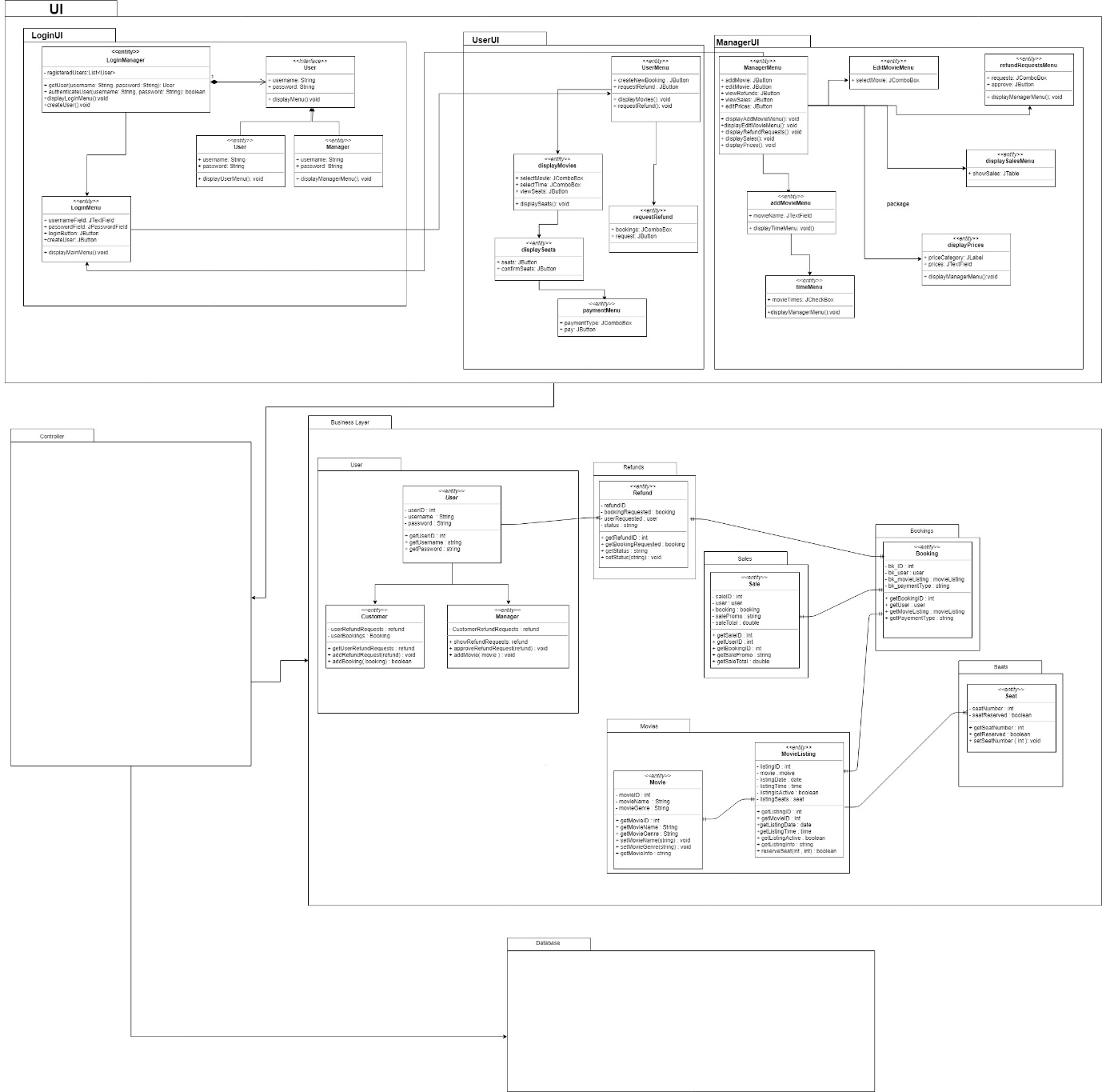
Manager

Seat

UI

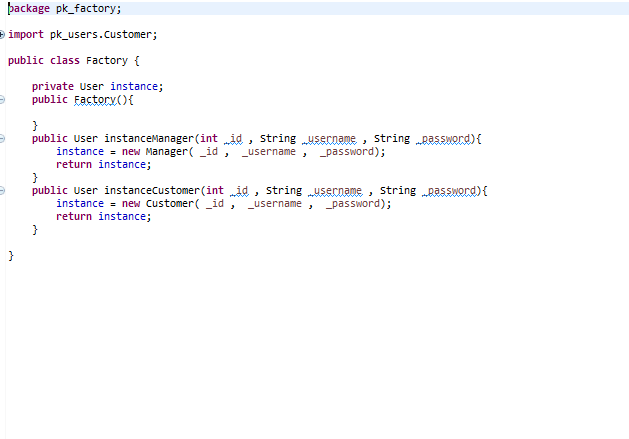
DataManager

Analysis Time Class Diagram



Coding Fragments

Factory Design Pattern

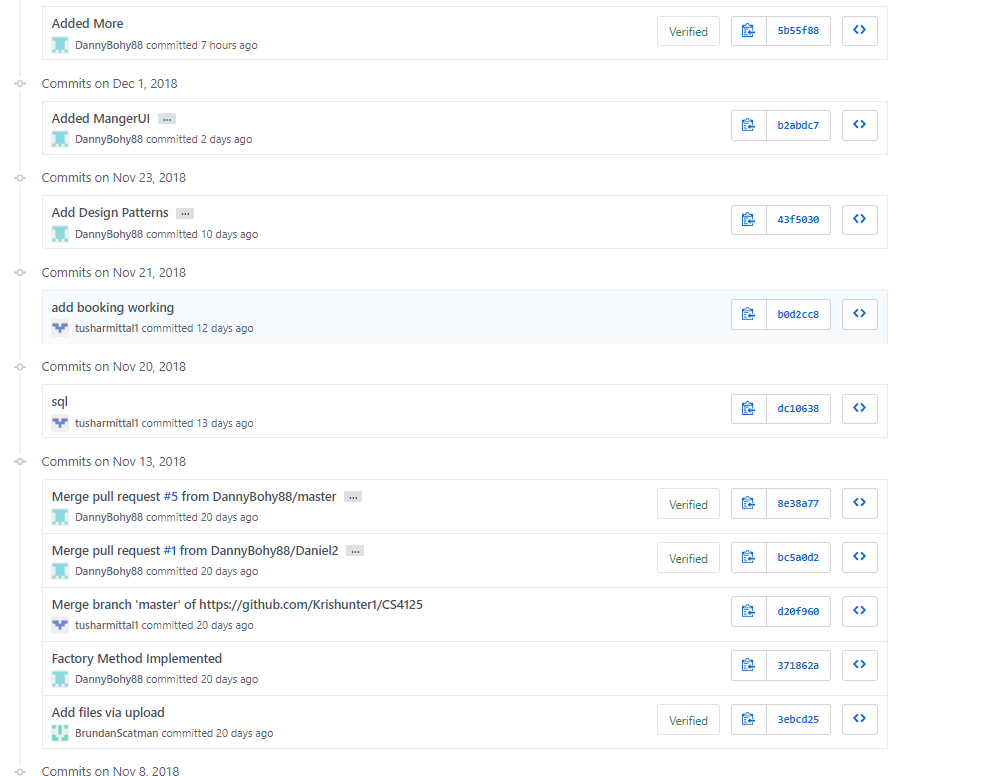


Workload

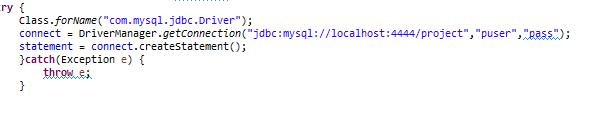
|  |  |  |  |
| --- | --- | --- | --- |
| **PACKAGE** | **CLASSES** | **LOC** | **AUTHOR** |
| pk\_business |  |  |  |
|  | Approved | 21 | 14167077 |
|  | Booking | 75 | 16177797 |
|  | CalculatePrices | 63 | 14167077 |
|  | Context | 17 | 15169081 |
|  | Login | 70 | 15169081 |
|  | ManagerBL | 32 | 15182304 |
|  | Menu | 115 | 16123921 |
|  | NotRequested | 22 | 14167077 |
|  | Requested | 21 | 14167077 |
|  | State | 7 | 14167077 |
| pk\_controller |  |  |  |
|  | UIController | 114 | 16177797 |
| Pk\_database |  |  |  |
|  | DatabaseController | 268 | 16123921 |
| Pk\_decorator |  |  |  |
|  | AdultTicket | 13 | 15169081 |
|  | ChildTicket | 13 | 15182304 |
|  | Price | 12 | 15182304 |
|  | PriceCreationTest | 30 | 14167077 |
|  | PriceDecorator | 8 | 14167077 |
|  | StudentTicket | 13 | 14167077 |
|  | Weekday | 18 | 14167077 |
|  | Weekend | 20 | 14167077 |
| Pk\_factory |  |  |  |
|  | factory | 22 | 15182304 |
| Pk\_movies |  |  |  |
|  | movie | 27 | 16177797 |
|  | movieListing | 34 | 15182304 |
|  | seat | 27 | 15182304 |
| Pk\_payments |  |  |  |
|  | PaymentController | 9 | 14167077 |
| Pk\_Program |  |  |  |
|  | CinemaProgram | 18 | 15182304 |
|  | ManagerTest | 16 | 14167077 |
| Pk\_refund |  |  |  |
|  | Refund | 17 | 14167077 |
| Pk\_users |  |  |  |
|  | Customer | 28 | 15182304 |
|  | Manager | 25 | 16177797 |
|  | User | 19 | 15182304 |
| Pk\_userUI |  |  |  |
|  | displayMovies | 81 | 15169081 |
|  | displaySeats | 83 | 16123921 |
|  | Frame | 12 | 14167077 |
|  | LoginMenu | 49 | 14167077 |
|  | ManagerMenu | 88 | 14167077 |
|  | PaymentMethod | 92 | 15182304 |
|  | RegisterMenu | 56 | 15169081 |
|  | UserMenu | 67 | 16177797 |
|  | refundReq | 103 | 15169081 |
| Total |  | 1825 |  |
| 14167077 | 414 |  |  |
| 16123921 | 466 |  |  |
| 16177797 | 308 |  |  |
| 15169081 | 284 |  |  |
| 15182304 | 297 |  |  |

Added Value

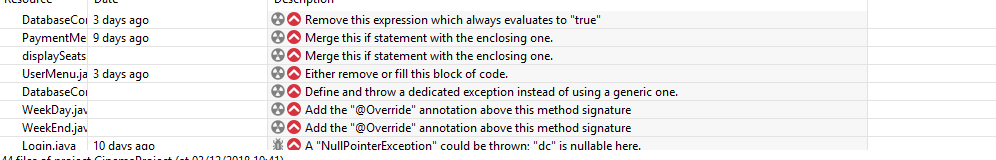
1. Github : We used Github to help with the implementation of the project



2: Database: We used xampp to create a Mysql database for the project.

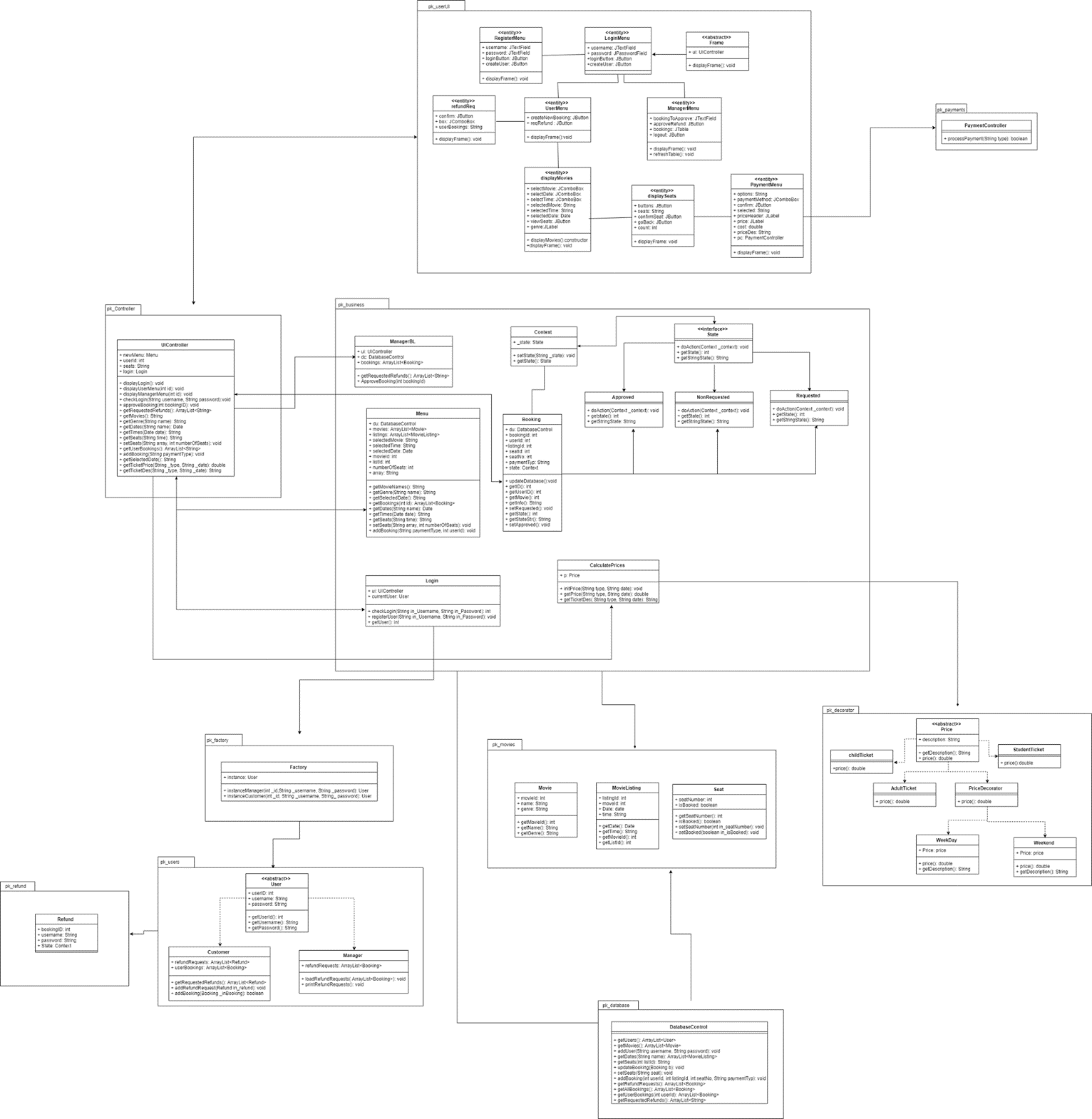


3: We used an eclipse plugin called SonarLit to refactor our code. SonarLit analysis code and shows bad code smells.



Recovered Architecture blueprints

Recovered Analysis Time Class Diagram



Critique of analysis time class diagram

Our original analysis time diagram was missing a lot of what finally ended up in the implementation. We made a significant amount of design changes during implementation which meant our original diagram changed a lot.