**Book-A-Movie**

Gabriel Lemos Rodrigues - glemosro 251101800

Abbas Saeed Hasan - ahasan62 251094486

Stefan Pisic - spisic 251110215

Mihir Nipeshkumar Kadiya - mkadiya 251129342

**Problem Definition**

Our client owns and operates out of a small movie theater and has requested a system capable of:

* Separate access to system for registered employees and general users (defined as “customers”)
* Allowing employees to add, remove, and keep track of current movie showings
* Allowing customers to book tickets for movie showings and change/cancel their tickets upon request
* Displaying the movies being shown at the theater as well as their showtimes to customers and employees

**Project Objective**

The tools present in today’s world that are used for the online booking of movies in movie theaters present a lot of friction on many aspects of its user journey. Mainly, the amount of new pages visited by the user makes it so the experience is not as fluent as it could be. The objective of this project is to innovate this experience in such a way to minimize the friction seen today, by reducing the amount of steps while not overwhelming the users with information, presenting them with a fluent and enjoyable experience.

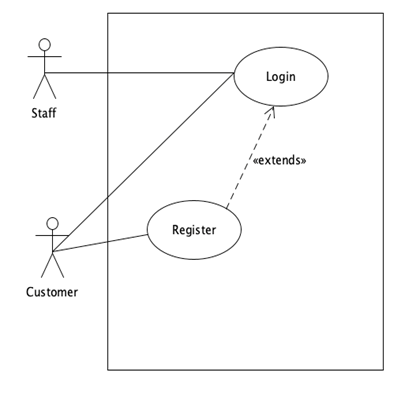
**Stakeholders List**

1. Theater chains
2. Entertainment companies

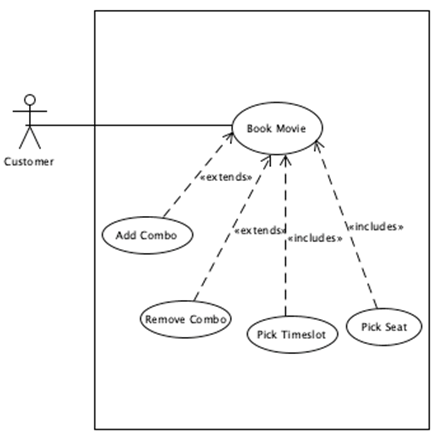
**Success/ Acceptance Criteria for each Stakeholder**

1. Ability for theatre chain staff members to add the latest movies.
2. Entertainment companies want seamless movie booking for customers to provide an increase in revenue for the company.

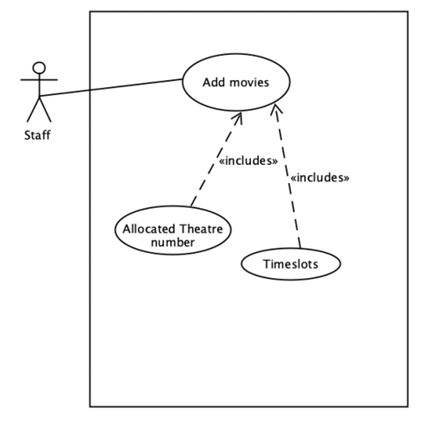
**Uses Cases**

****

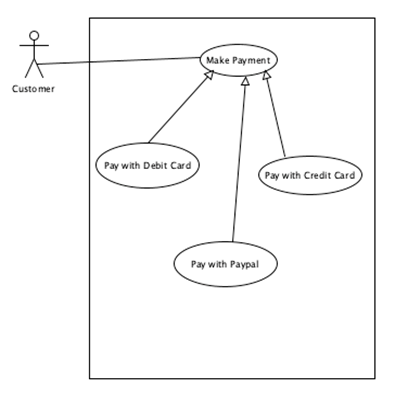
**(Figure 1):** User enters their credentials and clicks login, or clicks the register button to create an account.



**(Figure 2):** If the user presses the movie page, they are greeted with a page to add a combo, remove a combo, pick a timeslot and pick a seat.

****

**(Figure 3):** Staff adds a movie, and allocates a theatre number and a timeslot for it.



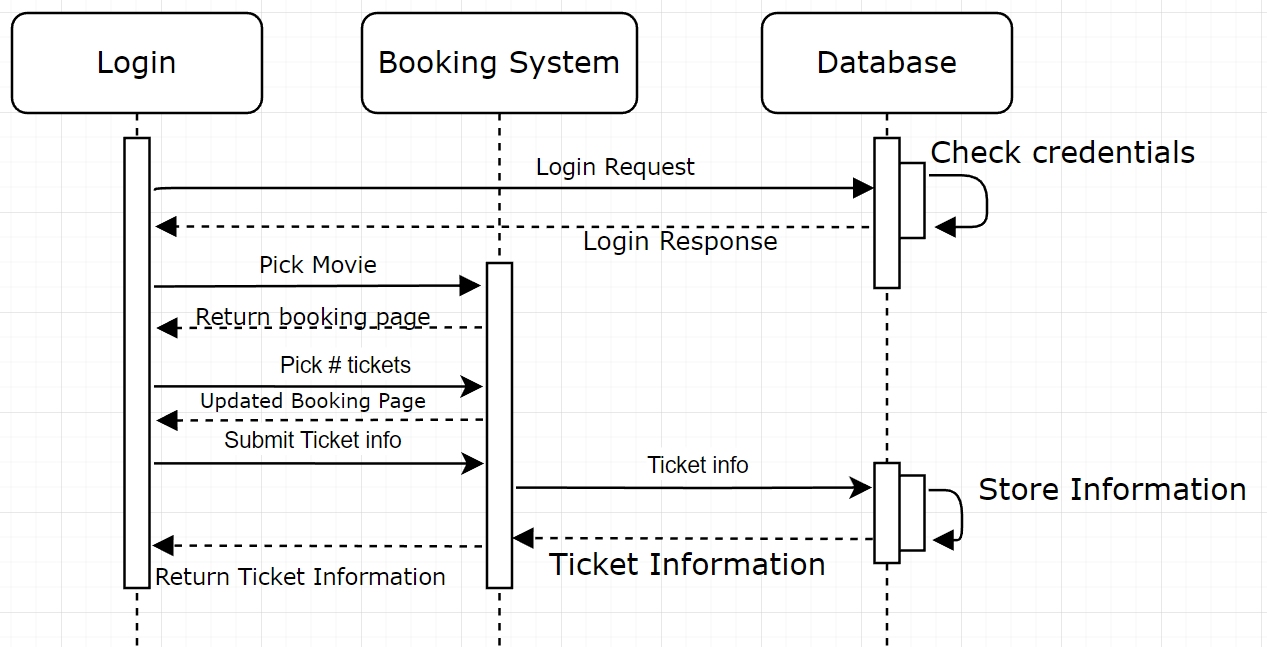
**(Figure 4):** User enters payment credentials on payment page.

**Selected Use Case Descriptions**

|  |  |  |
| --- | --- | --- |
| Use Case Name: | Login | |
| Scenario | Staff or User logs in. | |
| Triggering Event: | User enters their credentials and clicks login. | |
| Brief Description : | When a user enters his credentials and presses login, the system validates the credentials. If they are wrong, user will be prompted to enter credentials again | |
| Actors: | Staff, Customer | |
| Related Use Cases: |  | |
| Stakeholders: | 1. Theatre chains 2. Entertainment companies | |
| Preconditions: | User or Staff member must exist.  Login page must be functional. | |
| Postconditions: | Login credentials are validated - if correct, and the user is greeted with the main page. If incorrect, the user is prompted to enter it again. | |
| Flow of Activities: | Actor | System |
| 1. User enters credentials 2. If correct, user gets the main page or staff page if staff member 3. If incorrect, user prompted to re-enter credentials | 1. Posts entered credentials to backend 2. Validates login |
| Exception Conditions: | User credentials does not exist. Re-prompt for credentials | |

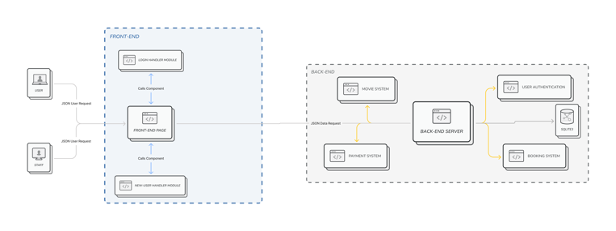
|  |  |  |
| --- | --- | --- |
| Use Case Name: | Book Movie | |
| Scenario | User books a movie | |
| Triggering Event: | User clicks a certain movie on the main page. | |
| Brief Description : | If the user presses the movie page, they are greeted with a page to add a combo, remove a combo, pick a timeslot and pick a seat. | |
| Actors: | Customer | |
| Related Use Cases: |  | |
| Stakeholders: | 1. Theatre companies 2. Entertainment companies | |
| Preconditions: | User must exist.  Movie booking process must be functional. | |
| Postconditions: | Tickets, combos, timeslots, seats and user credentials stored in the database and receipt returned to display to user. | |
| Flow of Activities: | Actor | System |
| 1. User selects a movie 2. User selects combo, timeslot and seat. | 1. System validates if timeslot and seat selection available, if not user prompted. 2. System posts to database. |
| Exception Conditions: | Movie booking already booked, user asked to reprompt another timeslot. | |

**Sequence Diagram**

****

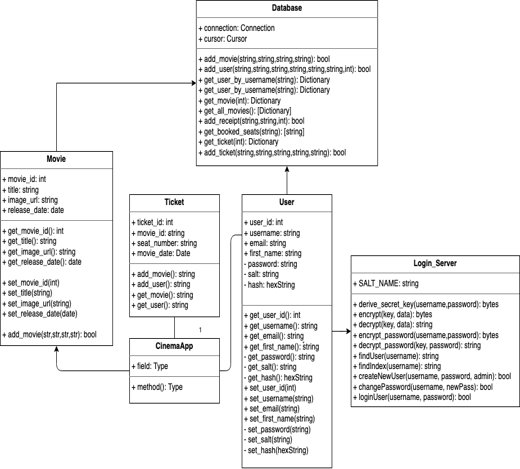
**(Figure 5):** Sequence Diagram describing the user journey throughout the application.

**Software Architecture**

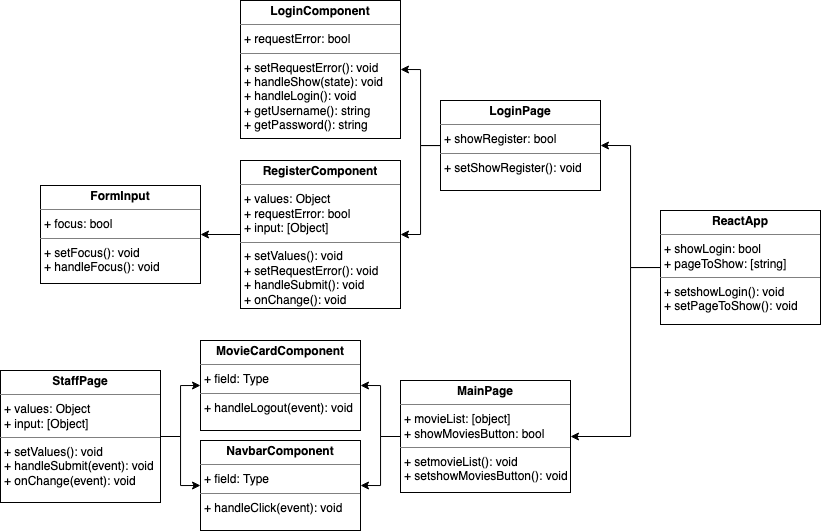
****

**(Figure 6):** System Architecture that shows the Service Oriented Architecture that our group has used for this project.

**Detailed Class Diagram**

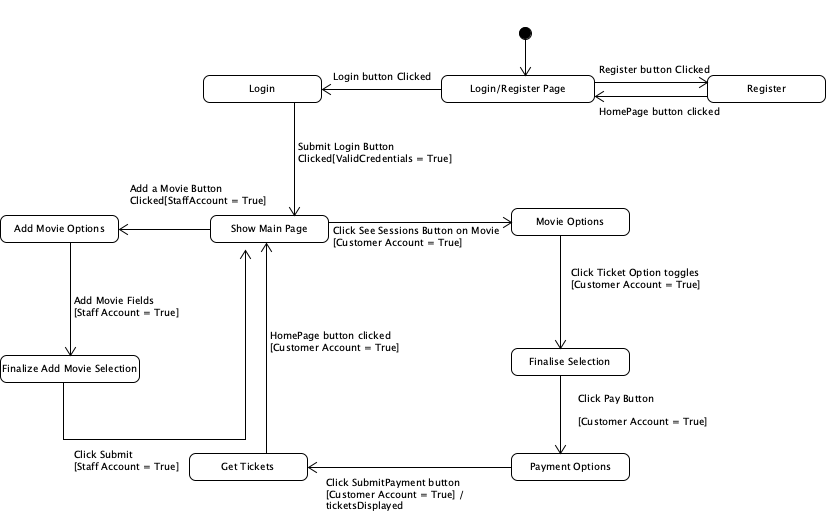
****

**(Figure 7):** Backend class diagram for serving user and staff requests. The backend is connected to the frontend to display the UI and information from the databases.

****

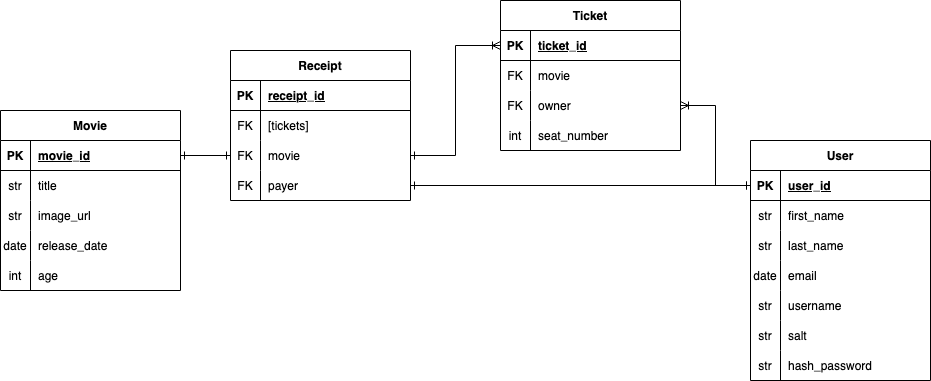
**(Figure 8):** The frontend class diagram mapping out the path the user can take depending on if they are a regular user or staff.

**State Diagram**

****

**(Figure 9):** The state diagram for the entire program. User or staff log in and depending on if the account is user or staff different UI is presented with different options.

**E-R Diagram**

****

**(Figure 10):** sqlite3 database that holds all the data for the web application which connects to the backend and is used for data presented in the frontend.

**Github Link**

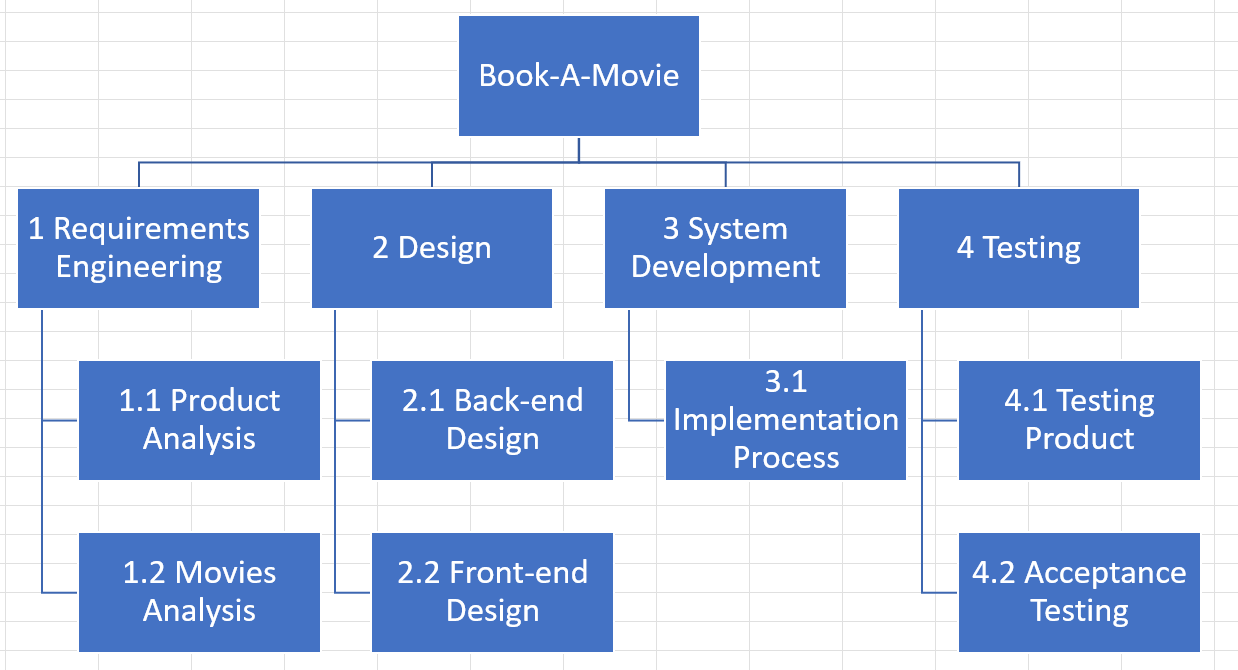
[**https://github.com/Lemos00/CS4471-Project-2022**](https://github.com/Lemos00/CS4471-Project-2022)

**Conclusion**

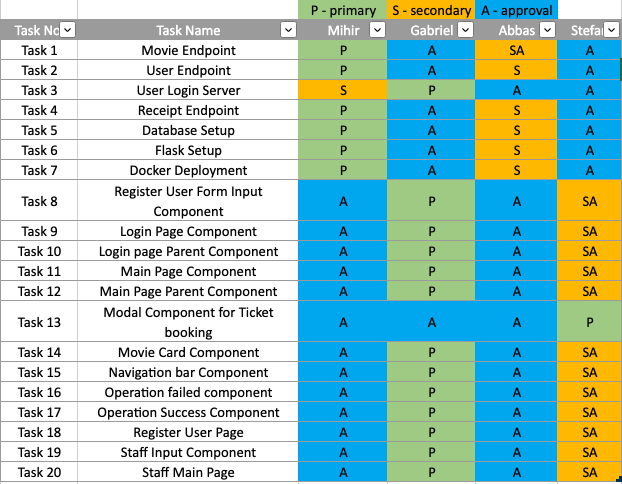
We learnt an optimal way to plan a software project and the benefits of the AGILE method. We also learnt the importance of planning our software project with the system diagrams to provide a solid basis to abstract from. With these methodologies, we were able to create a seamless movie booking process much smoother than the market, as it excluded a lot of unnecessary steps in the user journey.

**Appendix**

**WBS**

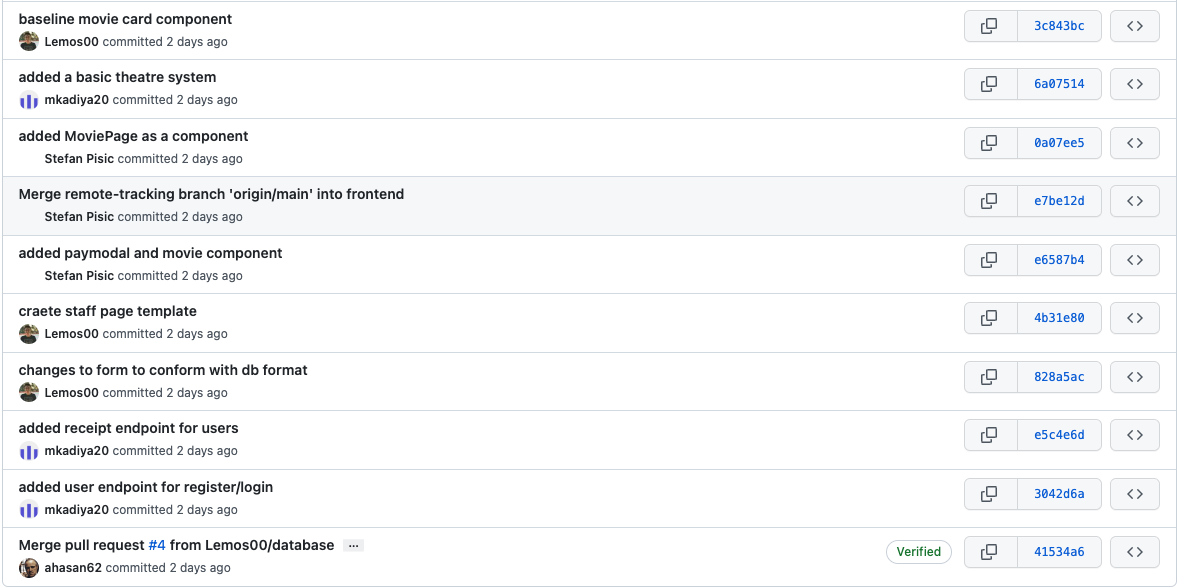
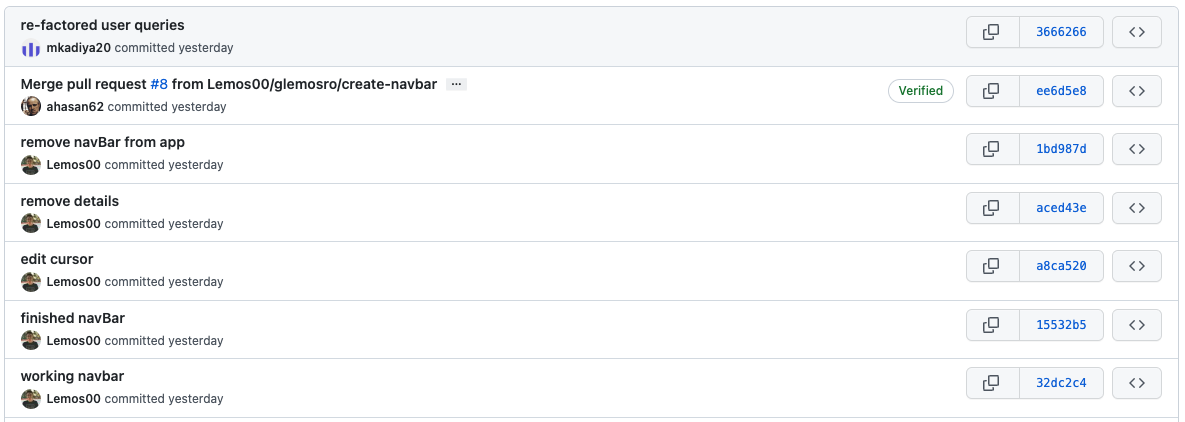
****

**(Figure 11):** Work breakdown structure with tasks done by the group.

**Task Assignment Matrix**

**(Figure 12):** Task assignment matrix that maps each task done by each member. We organized the tasks by the lets P,S, and A which represent primary, secondary and approval respectively.

**Sample of Commits**

****