FINAL SEMESTER ASSESSMENT (FSA) B.TECH. (CSE)
VI SEMESTER

UE18CS355 – OBJECT ORIENTED ANALYSIS AND DESIGN WITH SOFTWARE ENGINEERING LABORATORY
PROJECT REPORT
ON

MyMovie (Book My Show)

SUBMITTED BY

	NAME	SRN
1)	Vishwa Pravin	PES1201800143
2)	Anup Nagareddy	PES1201800283
3)	K. Sreesh Reddy	PES1201801580

JANUARY – MAY 2021

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

RR CAMPUS, BENGALURU – 560085, KARNATAKA, INDIA

TABLE OF CONTENTS

Abstract

Online movie booking system is a web portal where you can book movie tickets prior to going to the show. The objective of our project is to provide the facility of booking movie tickets online.

There are two types of users who can log in to the website. One is the customer who wants to book the tickets. And the others are the admins who have special privileges such as adding and removing a particular show. The movies are set up in rooms/auditoriums based on their availability at that instance of time. A search bar has also been implemented for the ease of navigation for the user. The user can search for a term and the movies containing that term will popup.

For seat selection, the customer is displayed a seat matrix wherein they can visually observe where their seats would be in the theater. The cost of the seats vary proportional to their position from the screen and the confirmation of the booking is shown to the customer later.

1. Software Requirement Specification

1. Introduction

1.1 Purpose

The purpose of an online movie ticket booking platform like bookmyshow is to provide an alternative method to purchasing movie tickets besides the traditionally available ways. This particular document talks only about the ‘movie’ booking part but the website offers options to book various other entertainment tickets like for concerts, comedy shows, theatre, etc. This is an Internet based application and requires the user to be connected to a network .

1.2 Intended Audience

The purpose of this document is to describe the functionality and requirements of the project for developers and project managers.

1.3 Product Scope

The objective of the product is to let users book movie tickets remotely without actually having to go to the theatre. This application will automate the reservation of tickets and ensure availability of tickets.

Scope:

- The system should allow the customer and admin to login successfully through the application’s first page

- The user should be able to view the list of movies which are running near to his location(based on GPS).
- The system shall enable the user to enter the search text on the screen.
- The system shall enable the user to select multiple options on the screen to search.
- The system shall enable the user to navigate between the search results.
- The system shall notify the user when no matching product is found on the search.
- The system shall allow the user to create a profile and set his credential.
- The system shall authenticate user credentials to view the profile.

Business Strategy: On its initial launch, it was a one of a kind application that let users book their movie tickets without having to stand in queues or be worried that the tickets won't be available. It lets them view the availability of their favourite movie, makes the payment process easy and hassle free and increases the user convenience exponentially.

It eventually gave the option to book tickets not only for movies but for various entertainment purposes like IPL, theatre, comedy shows, etc.

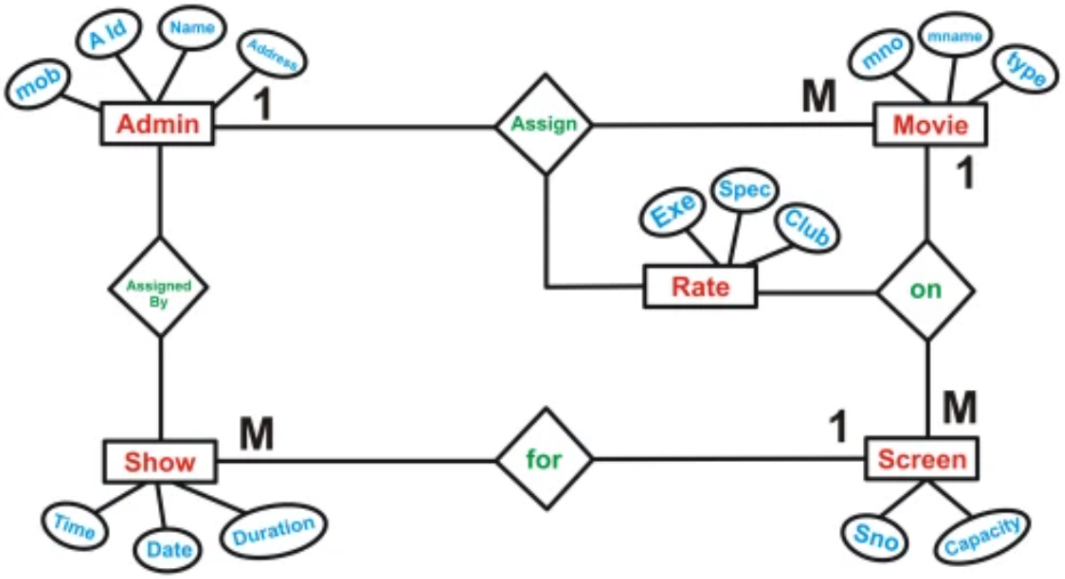
1.4 References

Marketing and Business strategy : https://in.bookmyshow.com/static/press-release/2021/jan
Product Perspective: here

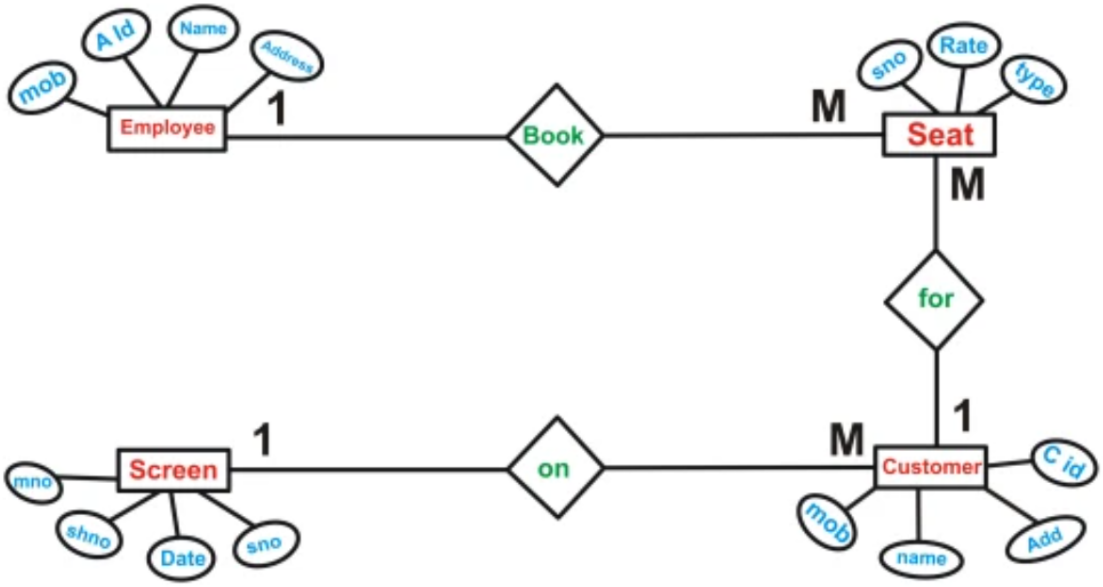
2. Overall Description

2.1 Product Perspective

The application initially when launched was a one of a kind platform to provide a hassle free experience while booking movie tickets by eliminating the process of having to wait in a queue. It also shows the availability of tickets. It's a replacement for the traditional method. Current Competitors to bookmyshow: Although BMS has no major competitors for movie ticket booking in India, http://pvrcinemas.com/ comes close. However, it has other competitors with respect to event tickets booking, which is out of scope for this project.



ER Diagram for assign movies



ER Diagram for book ticket

2.2 Product Functions

The following is a list of major functions that the application will perform:

- Registration
- Login or Sign up
- Search Movie
- Seat Viewing
- Ticket Cancellation

- Payment Logout
- Generate ticket
- Add Movies
- Search Movie

2.3 User Classes and Characteristics

The user types that would use the application are as follows:
Administrator: Administrator is responsible for updating and the maintenance of the web site content such as adding/removing information on the site, changing the logo.
Customer: Customers are people who shall use the application. To use this service people should have the basic computer ability. Users can view the general information, FAQ can use search. They will have a user account to be able to book tickets.
External Users: External users are people who have not got any user account for the web site. They can view the general information, FAQ.
Support staff: Consist of specialists who have good analytical and problem solving skills, up-to-date technical knowledge, good interpersonal and customer care skills.

2.4 Operating Environment

The designed system is thought to be a website and will be available via any web-browser application. It will not be dependent on the technical capabilities or operating system of the user's device. It can be used on PC or mobile through a browser.

2.5 Design and Implementation Constraints

It is assumed that the user has internet access and can do online payments. The performance depends on the quality and speed of the internet connection.

2.6 Assumptions and Dependencies

The user must have the ability to use the internet.
The user must have connected to the internet to use the system.
The user's computer must be Windows 95 or later version platforms and Microsoft Internet Explorer version 5.5 or later
TCP/IP protocol must be installed to communicate through HTTP messages.
The accuracy of the information of users is the responsibility of all users.

3. External Interface Requirements

3.1 User Interfaces

The application will have the following interfaces,

- The main page
- Sign up or Sign in page
- Movie Listing

- Seating Arrangement
- Payment
- Ticket Viewing

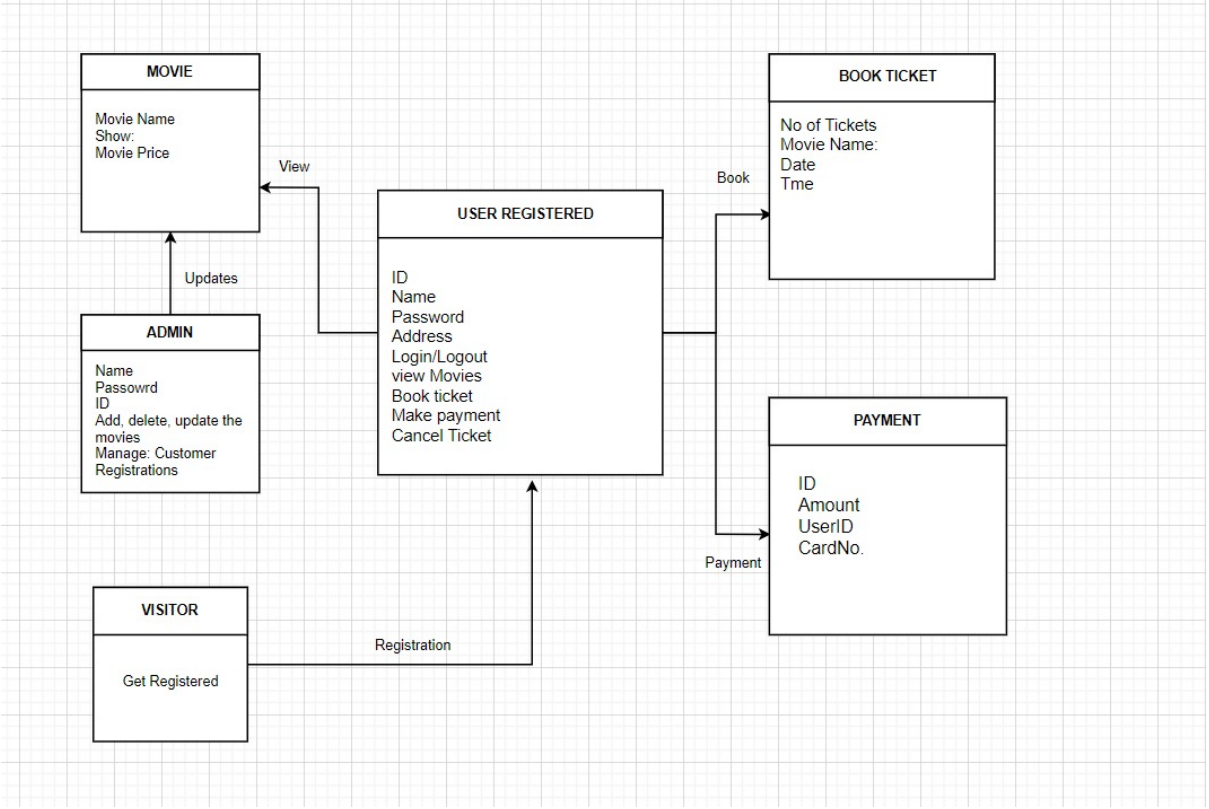
3.2 Software Interfaces

1. Any Web Browser Application
2. Apache HTTP Server
3. PHP
4. MySQL

3.3 Communications Interfaces

The default communication protocol for data transmission is Transmission Control Protocol/ Internet Protocol (TCP/IP). At the upper level HyperText Transfer Protocol (HTTP).

4. Analysis Models



5. System Features

5.1 Registration

If a customer wants to book the ticket then he/she must be registered, an unregistered user can’t book the ticket.

5.2 Login

Customer logins to the system by entering valid user id and password for booking the ticket.

5.3 Search Movie

The system shall have a search function. Customers or visitors can search movies based on movie name, date, time and venue.

5.4 Seat Viewing

The customer shall be shown a 2D image of the seats from which the desired seats are selected.

5.5 Ticket Canceling

The customer shall be given an option to cancel the ticket one hour before the movie with some fine.

5.6 Payment

For the customer there are many types of secure billing options provided, like debit or credit card.

5.7 Logout

After the payment or browse the movie, the customer will log out.

5.8 Generate ticket

After booking, the system can generate the portable document file and then send one copy to the customer’s Email-address and another one as an SMS to the customer's phone.

5.9 Add movies

The system shall have a feature for admin to add movies and their details.

5.10 Search Movie

The system shall have a feature for admin to remove movies.

6. Other Nonfunctional Requirements

6.1 Performance Requirements

Whenever users select some operation to perform then that operation should be able to execute as fast as possible. Hence its response time should be as little as possible.

6.2 Safety Requirements

If there is extensive damage to a wide portion of the database due to catastrophic failure, such as a disk crash, the recovery method restores a past copy of the database that was backed up to archival storage and reconstructs a more current state by reapplying or redoing the operations of committed transactions from the backed up log, up to the time of failure.

6.3 Security Requirements

The application should be secure enough such that no unauthorized or illegal user can access customer's data.

6.4 Software Quality Attributes

- **Availability:** The movie shows should be available on the specified date and time as many customers are doing advance reservations.
- **Correctness:** The shows should be correct according to the user's choice.
- **Maintainability:** The administrators and movie in chargers should maintain correct schedules of movies.
- **Usability:** The movie schedules should satisfy a maximum number of customers' needs

Appendix C: Requirement Traceability Matrix

Sl. No	Requirement ID	Brief Description of Requirement	Architecture Reference	Design Reference	Code File Reference	Test Case ID	System Test Case ID
1	MM001	List All Movies	1,2,4	1	/templates/movies.html /model/movie.py	UT_01	ST_05
2	MM002	Search for a term	1,2	1	/templates/movies.html /model/movie.py	UT_02	ST_06
3	MM003	Select a Movie	1,1	1	/templates/movies.html	UT_04	ST_09

					/model/movie.py		
4	MM004	View movie details	1,2	1	/templates/movie.html /model/movie.py	UT_04	ST_09
5	MM005	View all shows available	1,2,3	2	/templates/show.html /models/show.py	ST_14	ST_14
6	MM006	Select a show	1,3,	2	/templates/show.html /models/show.py	ST_15	ST_15
7	MM007	View seats available	1,4	2	/templates/show.html /models/show.py	UT_10	ST_16
8	MM008	See price for selection	1,2,3	2	/templates/show.html /models/show.py	UT_10	ST_16
9	MM009	Book ticket	1,2,3	3	/templates/show.html /services/show.py	UT_10	ST_17
10	MM010	View the booked ticket	1,2,3	4	/templates/myttickets.html /services/seat.py	UT_13	ST_17
11	MM011	Get details about movie using imdb id	1,2,3	1	/templates/addmovie.html /models/movie..py	UT_23	ST_31
12	MM012	Add movie	1,2,3	1	/templates/addmovie.html /models/movie..py	UT_23	ST_31
13	MM013	Delete Movie	1,2,3	1	/templates/mivie.html /models/movie.py	ST_32	ST_32
14	MM014	Add a show	1,2,	2	/templates/show..html /models/movie.py	UT_21	ST_29
15	MM015	Remove a show	1,2,	2	/templates/mivie.html /models/movie.py	ST_30	ST_30

2. Project Plan

1: Lifecycle to be followed for the execution of your project and justify why you have chosen the model.

The Agile software development life cycle would be best suited for the product being created.

The main tool of Agile development is iteration. Iteration is a process wherein a set of actions is repeated in a sequence until a condition is met, the focus being end-user satisfaction.
We think this is the best suited for our product because requirements, plans, and results are evaluated continuously so teams have a natural mechanism for
responding to change quickly. This will help us in adapting to all phases of the development life cycle

The following advantages are why we decided to go ahead with this model:

- People and interactions are emphasized rather than process and tools. Customers, developers and testers constantly interact with each other.
- Continuous attention to technical excellence and good design.
- Regular adaptation to changing circumstances.
- Even late changes in requirements are welcomed.

We have a fair idea of the tools and the plan we want to go ahead with. By using the agile software development lifecycle, we will stress more on the product and it's iterative development.

2: Tools used at different phases of SDLC

Planning Tools:

- Trello
- LucidChart

Design Tools:

- Adobe XD
- Figma

Development Tools:

Front end -

- HTML

- CSS
- Javascript
- REST API

Back-end-

- MySQL/Firebase
- Apache Server

IDE

- VS Code

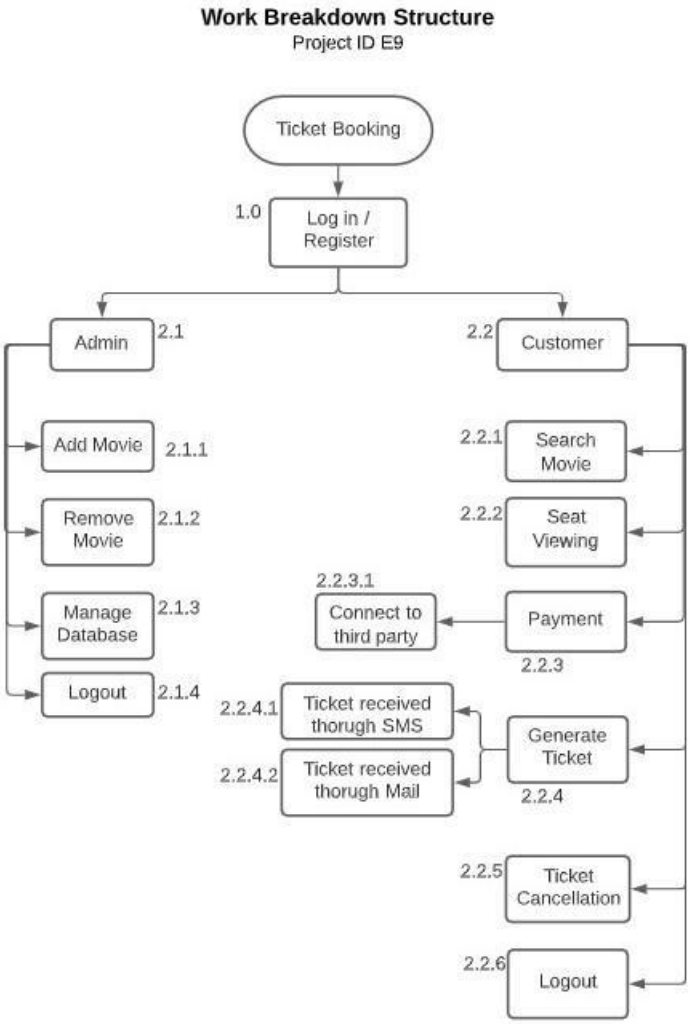
Bug Tracking and Testing Tool:

- Bugzilla
- Github
- Postman

Version Control:

- Git

3: Work Breakdown Structure for the entire functionalities



4: Determine all the deliverables and categorise them as reuse/build components

- **1.0** Login / Register - Build
  A new user has to register to be able book tickets. An user with an existing account can login and use the facilities. The additional functionalities will be available when an Admin logs in.

- **2.1** Admin - Build
  Admin has access to view the database of booking and add or remove movies from the listing.

- **2.1.1** Add Movie - Build
  The admin can add movies and it’s details.

- **2.1.2** Remove Movie - Build
  Admin can remove movies from the listing.

- **2.1.3** Manage Database - Build
  Admin can view the booking details of each movie.

- **2.1.4** Logout - Build
  Admin can logout when done using the functionalities.

- **2.2** Customer - Build
  Customers can view the movie listing ,book tickets and cancel.

- **2.2.1** Search Movie - Build
  Customers can search movies based on movie name, date, time and venue.

- **2.2.2** Seat Viewing - Build
  Customers shall be shown a 2D image of the seats available/ booked and be able to choose from the available options.

- **2.2.3** Payment - Build
  There are many types of secure billing options provided.

- **2.2.3.1** Connect to third party - Reuse
  Each secure payment will be redirected to the respective payment option service provider.

- **2.2.4** Generate Ticket - Build
  The system will generate a portable softcopy of the ticket.

- **2.2.4.1** Ticket received through SMS - Reuse
  The ticket can be sent through SMS via the customer's phone number.

- **2.2.4.2** Ticket received through Mail - Reuse
  The ticket can be sent through mail via the customer's email-id.

- **2.2.5** Ticket Cancellation - Build
  Customers can cancel the ticket one hour before the movie starts with a percent of ticket pricing as a fine.

- **2.2.6** Logout - Build
  Customers can logout after being done using the functionalities.

5:Rough estimate of effort required to accomplish each task in terms of personal months.

Basic UI
Includes Basic HTML and CSS: 1-2 weeks

Database Implementation
Using MYSQL and connecting it to frontend: 1 week

Interactive tools
Creating accounts, browsing through the site for movies: 2 weeks

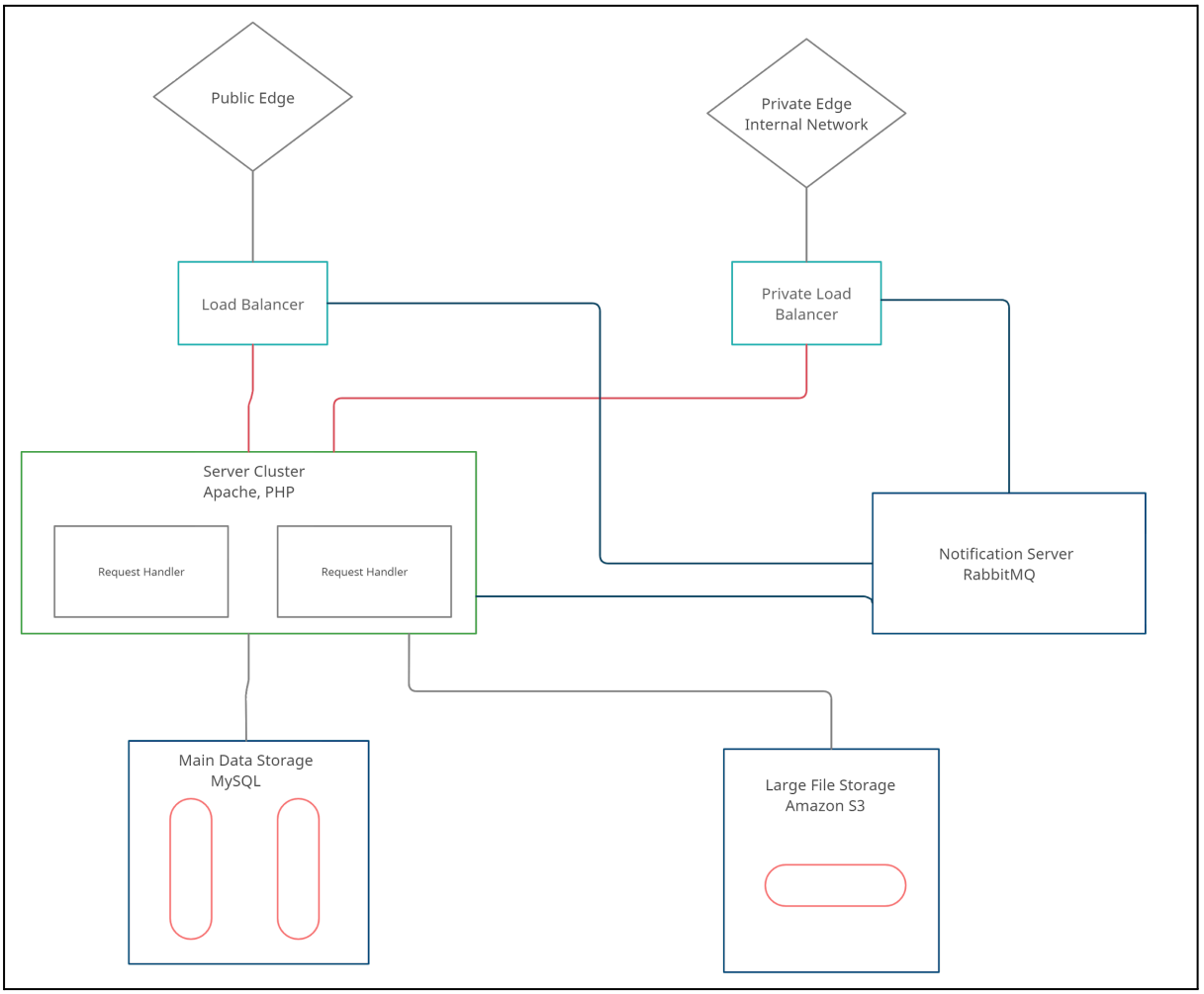
Booking tickets
Seat viewing, payment methods sending tickets to email and via SMS: 2 weeks

Debugging
Additional corrections and debugging: 1 week

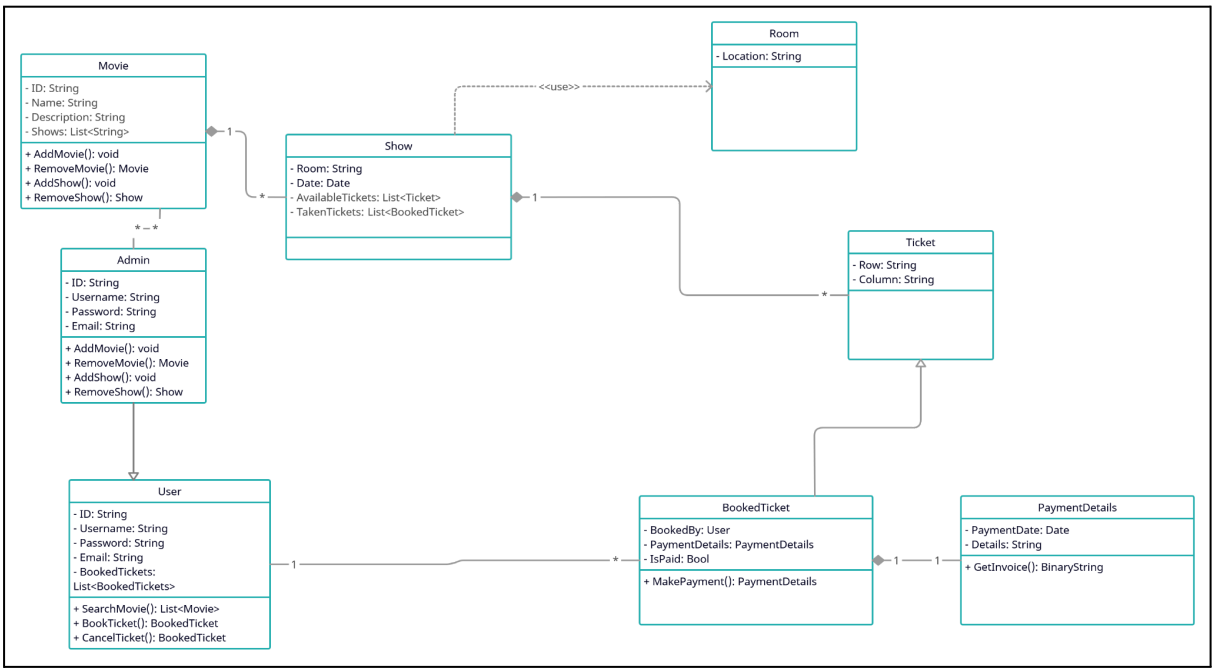
Total development
The complete product will be a movie ticket booking website: 8-10 weeks

3. Design Diagrams

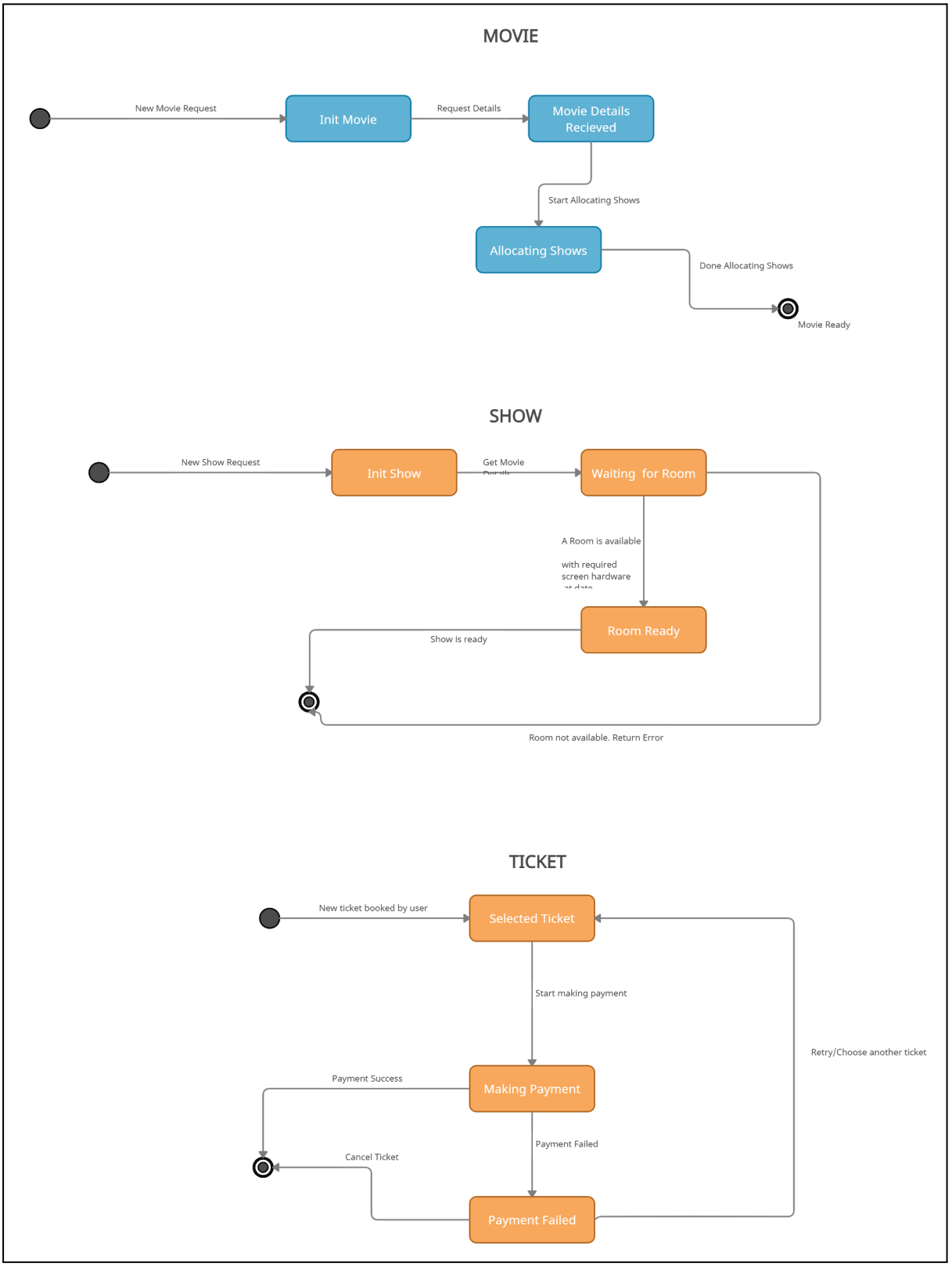
Architecture Diagram:



Class Diagram:

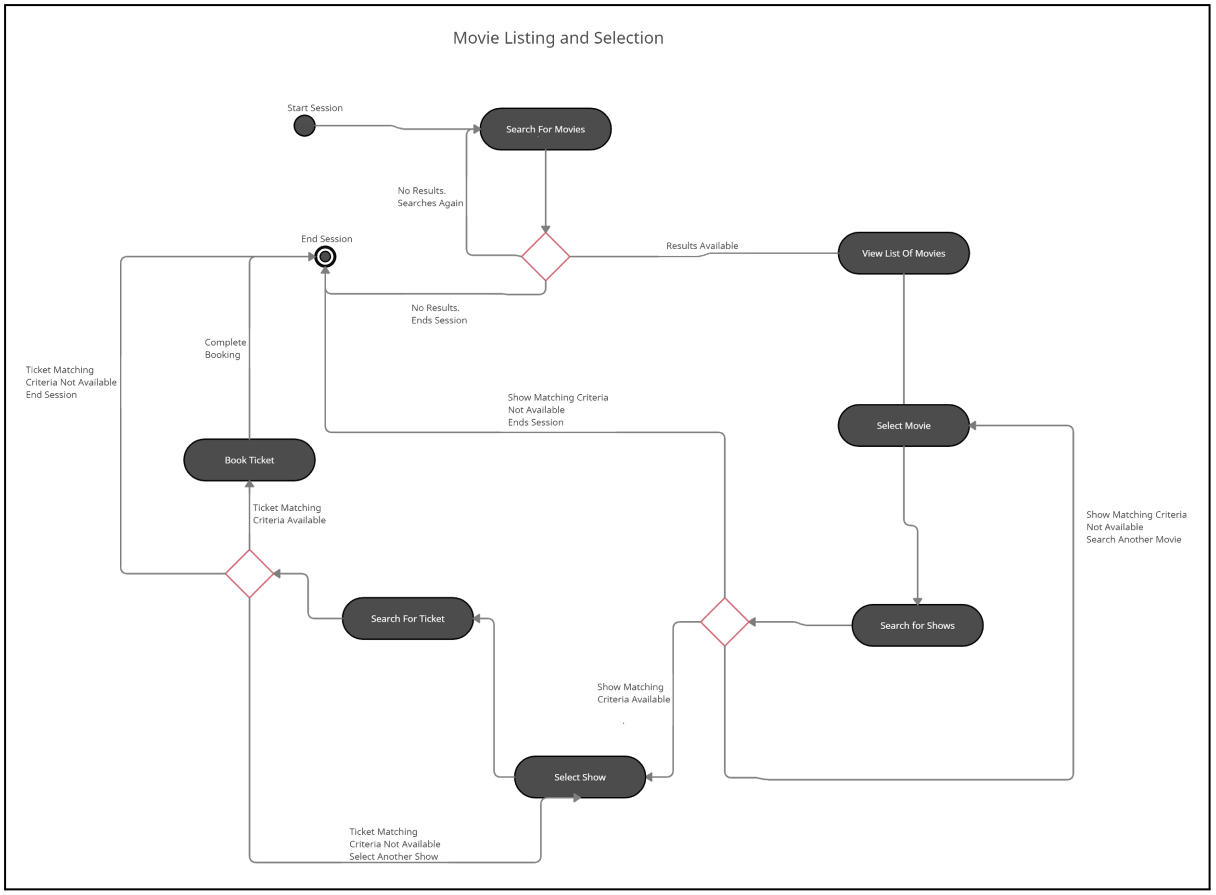


State Diagram:



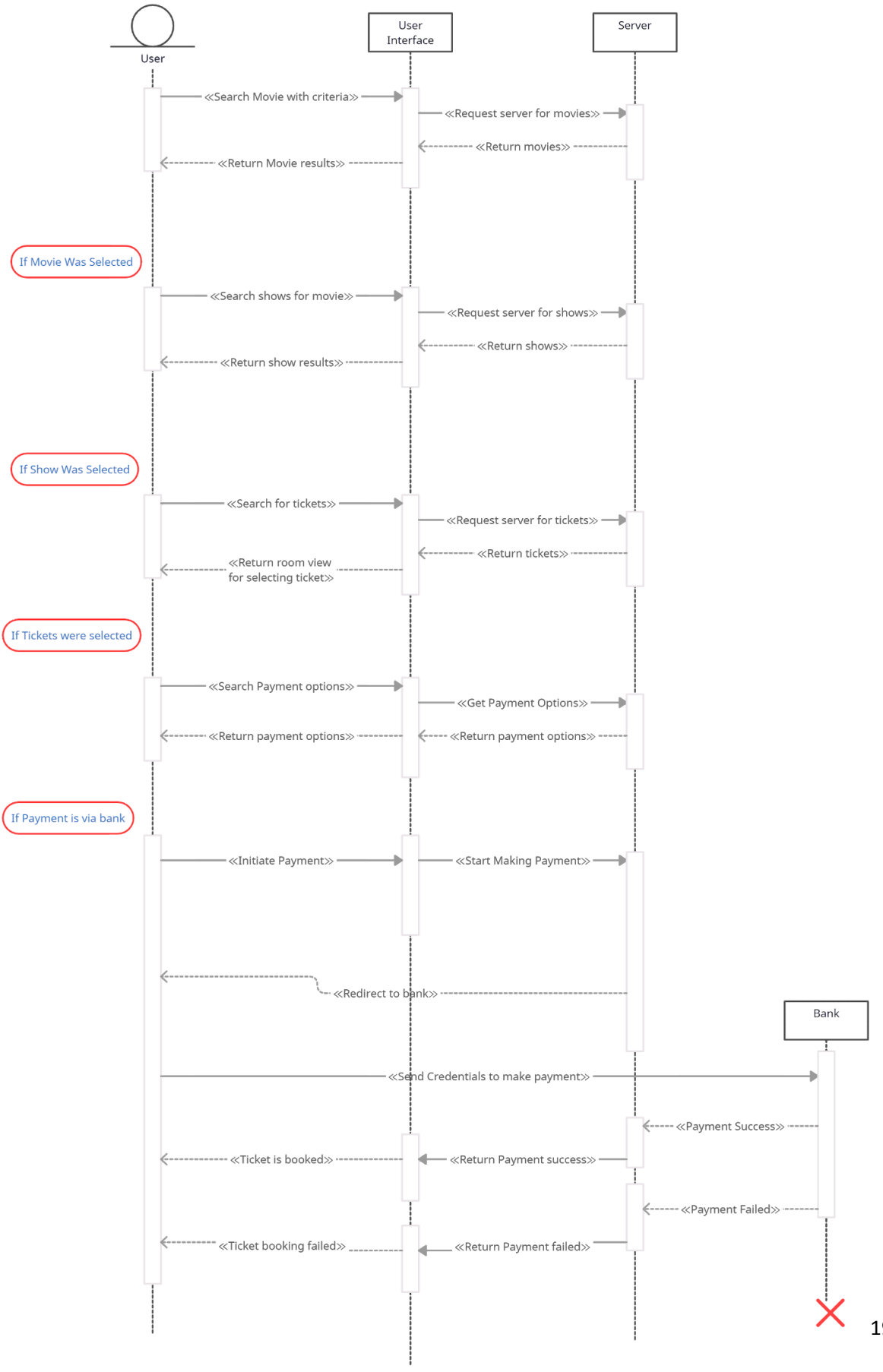
Use Case - Movie Listing and Selection

Activity Diagram:



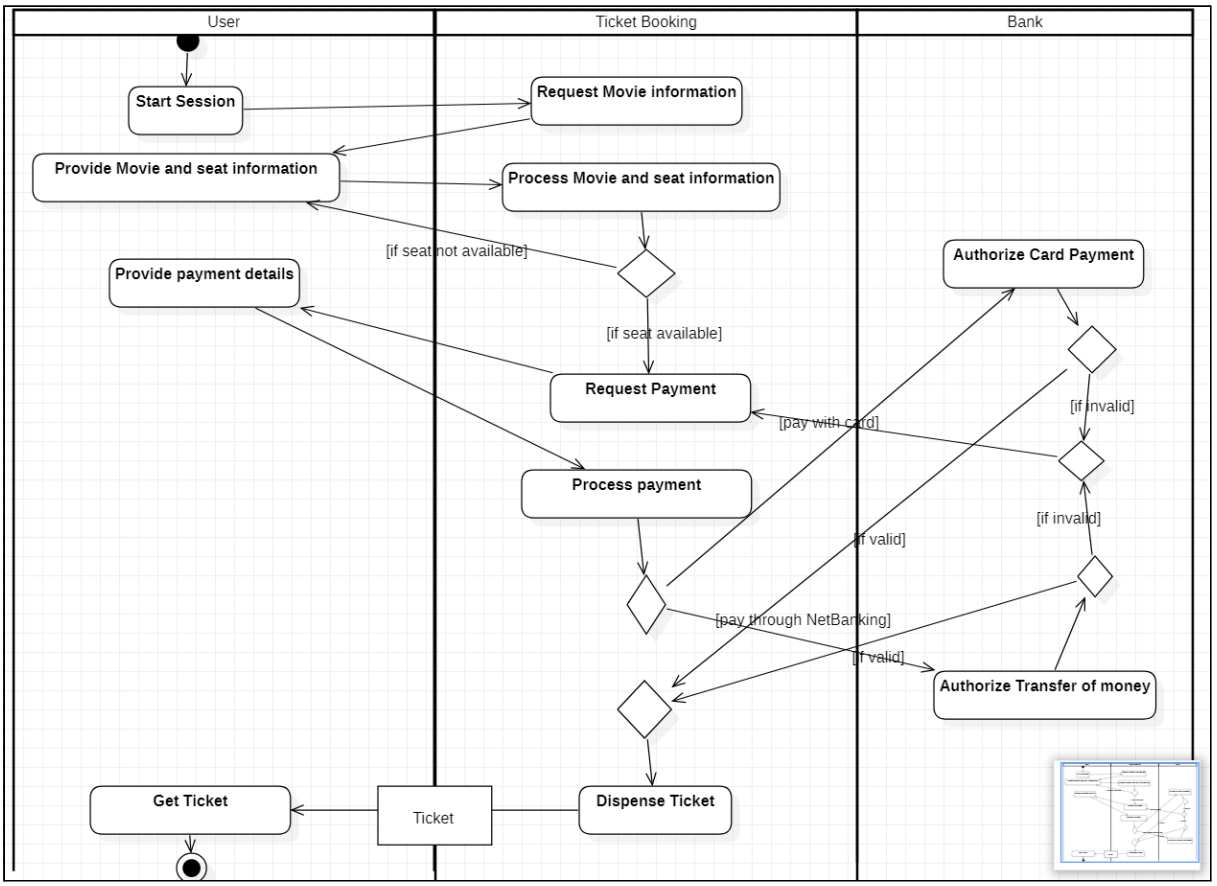
Sequence Diagram:

Movie Listing and Selection

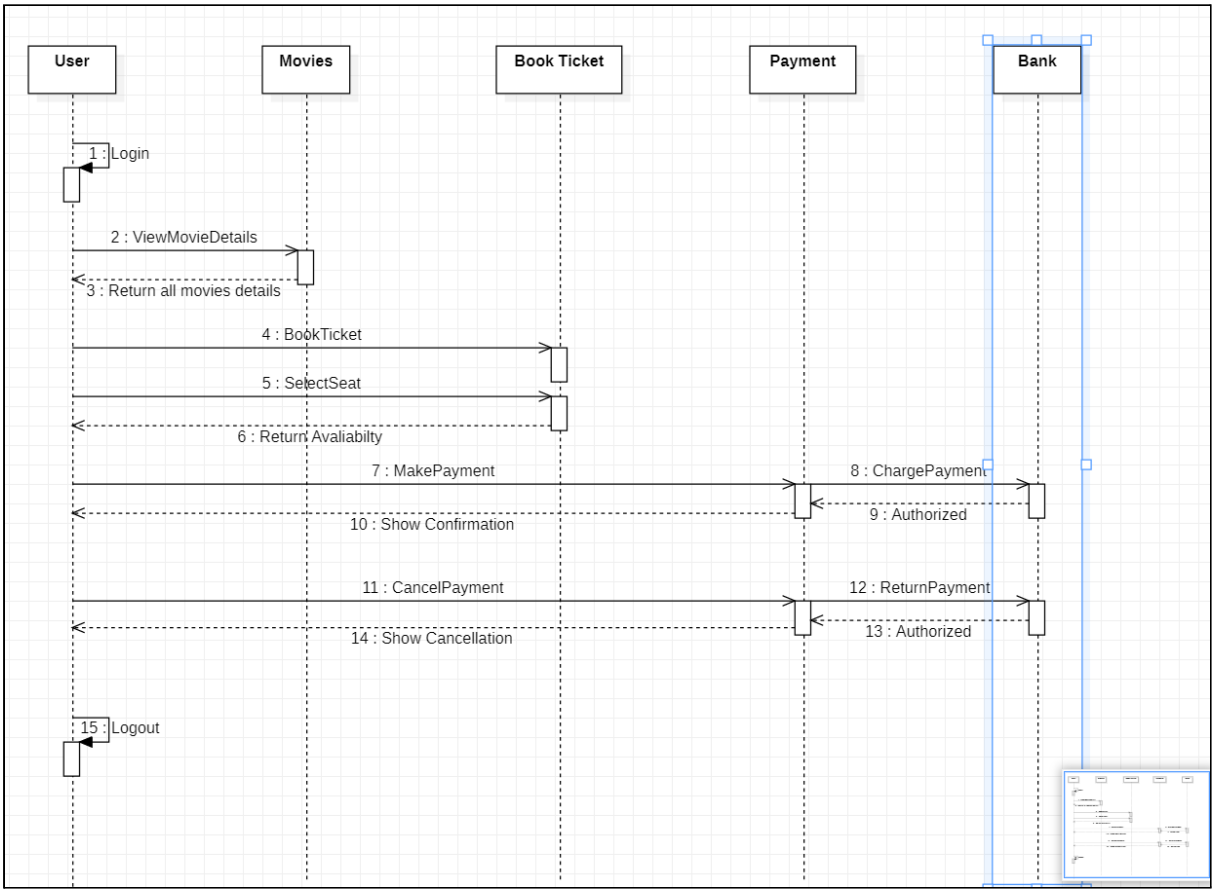


Use case - Seat Selection and Booking

Activity Diagram:

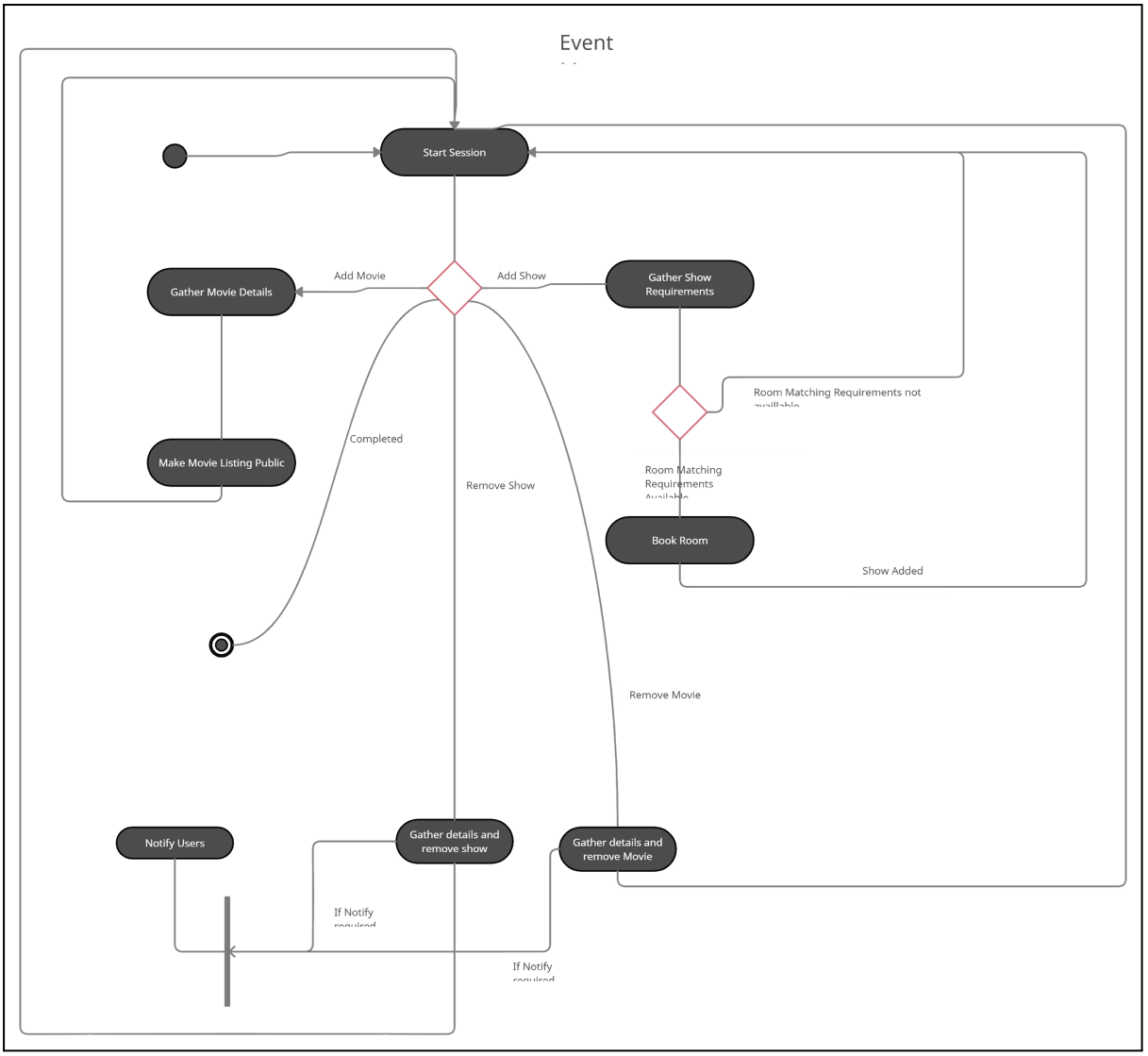


Sequence Diagram:

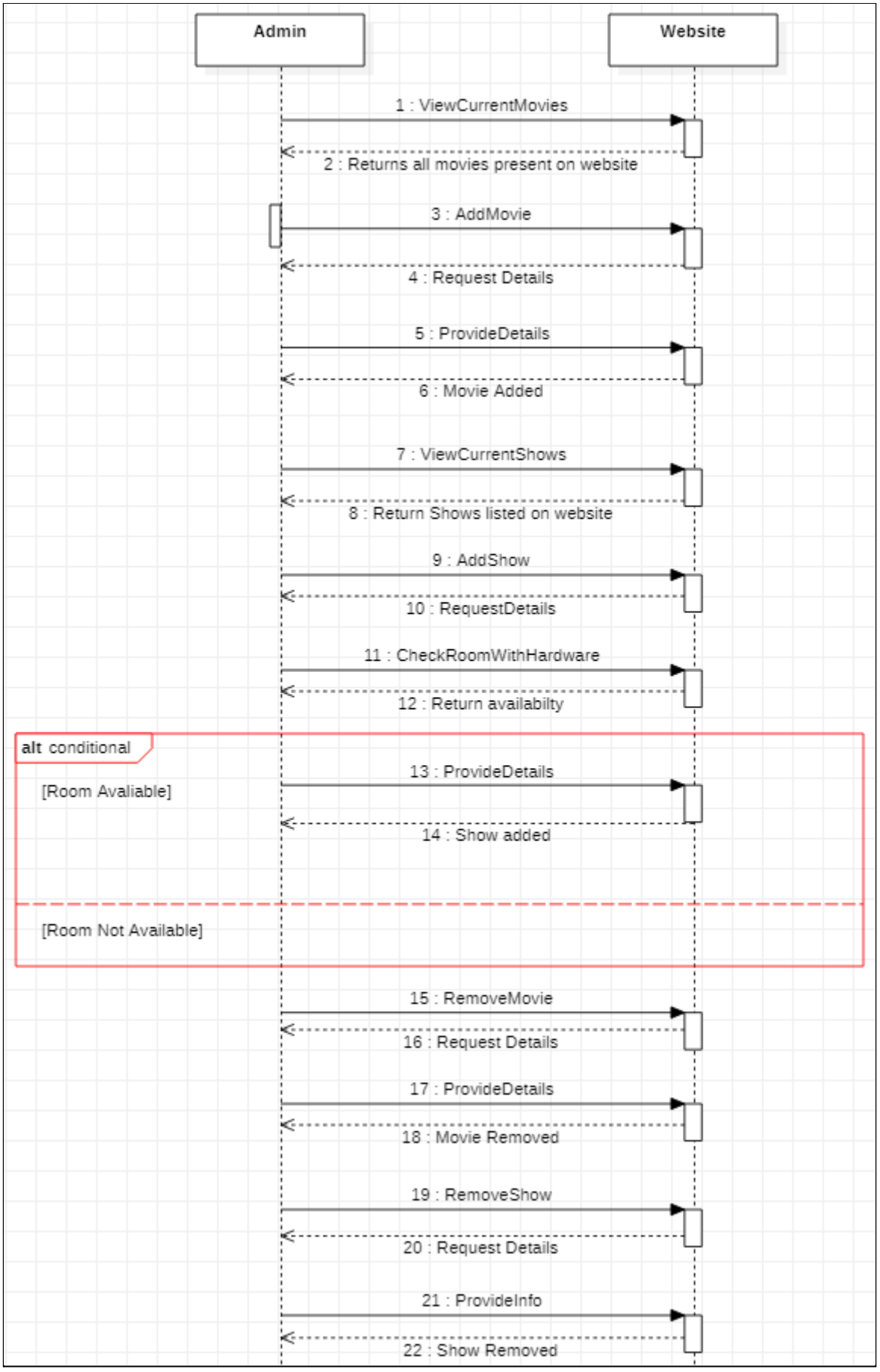


Use Case - Movie Management

Activity Diagram:



Sequence Diagram:


Admin
Website
1 : ViewCurrentMovies
2 : Returns all movies present on website
3 : AddMovie
4 : Request Details
5 : ProvideDetails
6 : Movie Added
7 : ViewCurrentShows
8 : Return Shows listed on website
9 : AddShow
10 : RequestDetails
11 : CheckRoomWithHardware
12 : Return availabilty
alt conditional
[Room Avaliable]
13 : ProvideDetails
14 : Show added
[Room Not Available]
15 : RemoveMovie
16 : Request Details
17 : ProvideDetails
18 : Movie Removed
19 : RemoveShow
20 : Request Details
21 : ProvideInfo
22 : Show Removed

4. Module Description

Movie Listing and Selection

As soon as the user logs into his account, this is the first page visible to the user. It lists out all the movies running across different theatres at different times in the city. Based on the preference of the type of movie the user wants to watch, the user can click on that particular movie/show. After clicking on the movie, he will be directed to the show selection page. This page contains a summary of the generic plot of the movie along with the duration of the movie. It contains information about the age groups who are allowed to watch the movie.

If the user is interested in the movie after reading the summary, one can click on the show selection box and the user can select the show and timing accordingly. After selecting this he'll be directed to the next use case which is the seat selection and booking.

Seat Selection and booking

This use case is based upon the selection of the seats for the show after the user has selected the movie they want to watch. They will be displayed a seat matrix consisting of rows and columns to identify their seat. The user can select the seat they want by clicking on the seat and it turns green indicating that it is under reservation. In the same way they can unselect the seat by clicking again on the same seat they reserved.

After selecting the desired seats the cost will be displayed on the customer's screen. This cost of the seats is dependent on the position of the seats from the screen. For eg: The seats in the last row are worth twice the seats in the first row. After the user is comfortable with the seats, he can go ahead and book them. Their tickets will now be listed under "Your Tickets" section.

Movie Management

There are two types of users who can log in to the website. One is the customer who wants to book the tickets. And the others are the admins who have special privileges such as adding and removing a particular show. This use case helps the latter in managing the movies/shows for a particular day by selecting the time slot and the room the movie/show is hosted. The movie/show admin can login as an admin since only admins have the access to manipulate the movies/shows. After logging in, the admin types in the IMDB id of the movie he wants to remove or add a show for.

After typing in the id, the details of the movie such as name, actors, rating and brief description, etc. are visible to the admin. The admin now selects the time and the room where the movie has to be hosted on that particular day. Similar to adding a movie, the procedure for deleting a show is also the same.

Stack Used

Python Flask - Backend Server
MongoDB - Database

Jinja - HTML Rendering Engine

Server Logic

Middleware

- auth.py
- common.py

Models

- movie.py
- user.py

Services

- movie.py
- seats.py
- shows.py

Static

- … static HTML/CSS/JS

Templates

- addmovie.html
- addshow.html
- layout.html
- login.html
- movie_page.html
- movies.html
- mytickets.html
- navbar.html
- show_page.html
- simple_error.html

Utils

- conf.py
- helpers.py
- responses.py

app.py

Scripts

- mongotest.py
- movies.json
- reset.py
- update.py
- view.py

Tests

- runner.py

5. Test Cases

Abbreviation Used

ST - System Test

UT - Unit Test

Movie listing and selection - MLS

Seat Selection and Booking - SSB

Event Management - EM

Selenium Test - Sel

BackendTest - Bac

Movie Listing and Selection

Test Case ID	Name of Module	Test case description	Pre-conditions	Test Steps	Test data	Expected Results	Actual Result	Test Result
UT_01	MLS _Bac	Send empty term to route for getting search results	User must have a session cookie which gives them access to the route	Make a request to POST /search	{ “Term”:” ” }	[… List of movies]	[… List of movies]	PASS
UT_02	MLS _Bac	Send “avengers” term to route for getting search results	User must have a session cookie which gives them access to the route	Make a request to POST /search	{ “Term”:” avengers” }	[… List of three movies]	[… List of three movies]	PASS
UT_03	MLS _Bac	Check if “avengers” and “AVENGERS” return the same result	User must have a session cookie which gives them	Make a request to POST /search	{ “Term”:” avengers” } {	[… List of three movies]	[… List of three movies]	PASS

			access to the route	Repeat for upper case term	“Term”:” AVENGERS” }			
UT_04	MLS_Bac	Check a valid movie route and an invalid, verify results	User must have a session cookie which gives them access to the route	Make a request to GET request to each route	Movie ids	Movie page is shown	Movie page is shown	PASS
ST_05	MLS_Sel	List all movies in the home page and check if all are shown	User must be logged in	Open home page		List of 21 movies must be shown	List of 21 movies is shown	PASS
ST_06	MLS_Sel	Search for a term and check results	User must be logged in	Open home page and type term in search bar	Avengers	Same List of 3 movies are shown in the two searches	List of 3 movies is shown	PASS
ST_07	MLS_Sel	Make sure searches are case sensitive	User must be logged in	Open home page and type a term in lower case and upper case	Avengers AVENGERS	Same List of 3 movies are shown in the two searches	Same List of 3 movies are shown in the two searches	PASS
ST_08	MLS_Sel	Make sure no results are shown for a term for which no movie is found	User must be logged in	Open home page, type a random term	12122121	No movies are shown	No movies are shown	PASS

ST_09	MLS _Sel	Select a movie from the list of movies	User must be logged in	On home page select a movie and click on it		The movie page is shown	The movie page is shown	PASS

Seat Selection and Booking

Test Case ID	Name of Module	Test case description	Pre-conditions	Test Steps	Test data	Expected Results	Actual Result	Test Result
UT_10	SSB_ Bac	Request to book tickets which are available and valid	User must have a session cookie which gives them access to the route	Send a request POST /movie/<mid>/0/book	{ 'tickets': tickets }	{ … Booking data }	{ … Booking data }	PASS
UT_11	SSB_ Bac	Request to book tickets whose body is invalid	User must have a session cookie which gives them access to the route	Send a request POST /movie/<mid>/0/book	{ 'tickets': tickets }	A 503 error response	A 503 error response	PASS
UT_12	SSB_ Bac	Request to book unavailable tickets	User must have a session cookie which gives them access to the route	Send a request POST /movie/<mid>/0/book	{ 'tickets': tickets }	A 403 error response	A 403 error response	PASS
UT_13	SSB_ Bac	Verify that the tickets are booked	User must have a session cookie	Send a post reques		{ … Show details	{ … Show details	PASS

			which gives them access to the route	t to GET /api/movie/<mid>/0		}	}	
ST_14	SSB_Sel	Check if shows are displayed for a particular movie	User must be logged in	Open the home page and click on the desired Movie		Show found on April 22	Show found on April 22	PASS
ST_15	SSB_Sel	Select a show and verify if the seat matrix is displayed	User must be logged in	Click on the movie on the home page and select from the available shows		Show is selected and seat matrix is viewed	Show is selected and seat matrix is viewed	PASS
ST_16	SSB_Sel	Select the seat and verify if the correct price is shown	Show must be selected	Click on the seats not blackened in the seat matrix	Seats : 30, 31, 32	Correct price is displayed	Correct price is displayed	PASS
ST_17	SSB_Sel	Book the selected seats and verify the booking	Seats must be selected	Click on book tickets in the seat matrix page		Ticket is booked	Ticket is booked	PASS
ST_18	SSB_sSel	Verify that the previously booked	Booking must have been done	Click on the same show		Seats are already booked	Seats are already	PASS

		seats are blackened when visiting again		for the same movie for the seat matrix			booked	

Movie Management

Test Case ID	Name of Module	Test case description	Pre-conditions	Test Steps	Test data	Expected Results	Actual Result	Test Result
UT_19	EM_Bac	Verify that the same date can't be chosen for a show again	User must have a session cookie which gives them access to the route	Send a request POST movie/{mid}/will_showdate_collide	{ 'date': '2021-04-22' }	Same date can't be chosen for a show	Same date can't be chosen for a show	PASS
UT_20	EM_Bac	Verify that all free rooms are returned for a particular date and slot	User must have a session cookie which gives them access to the route	Send a request POST movie/{mid}/free_rooms	{ 'date': '2021-04-22', 'slot': '9:AM' }	All free rooms are returned for a particular date and slot	All the free rooms for the particular date and slot are shown	PASS
UT_21	EM_Bac	Verify that a valid show is added	User must have a session cookie which gives them access to the route	Send a request POST movie/{mid}/addshow	{ 'date': '2021-04-22', 'slot': '9:AM',	Only a valid show should be added	Only a valid show can be added	PASS

					'room': 0, 'price': 500 }			

UT_22	EM_Bac	Verify that an existing show can't be added again	User must have a session cookie which gives them access to the route	Send a request POST movie/{mid}/addshow	{
					
					'date': '2021-04-22',
					
					'slot': '9:AM',
					
					'room': 0,
					
					'price': 500
					}
						A		PASS

UT_23	EM_Bac	Verify that a new movie can be added with imdb id	Users must have a session cookie which gives them access to the route.					PASS
UT_24	EM_Bac	Verify that an invalid movie is not added which does not exist in imdb	User must have a session cookie which gives them access to the route					PASS
ST_25	EM_Sel	Login as different users, check that only admin can add movies	Nil	Login as different users and then admin	Username and password	Access is correct	Access is correct	PASS
ST_26	EM_Sel	Select an invalid date	User is logged in	Login and	{	Invalid date is	Invalid date is	

		and check if it isn't allowed	and is in the date selection page for adding a show	select a show that has already been added on that day.	'date': '2021-04-19' }	not accepted	not accepted	PASS
ST_27	EM_Sel	Select an valid date and check if does not allow it	User is logged in and is in the date selection page for adding a show	Select a show to be added and the date for it.	{ 'date': '2021-04-22' }	Valid date is accepted	Valid date is accepted	PASS
ST_28	EM_Sel	Select show date and slot, verify if all rooms are returned	User should be logged in and a date and slot should be selected	The date and slot should be selected on the respective page.	Date 2021-04-22 Slot 9:AM	All rooms are returned	All rooms are returned	PASS
ST_29	EM_Sel	Add show with required details, verify that it is added	User should be logged in.	Click in the show adding page and select the show to be added	Date 2021-04-22 Slot 9:AM Rooms 0	The show was added	The show was added	PASS
ST_30	EM_Sel	Remove a show, verify that is removed	Users should be logged in and a show should have been added previously.	Click on the edit show page and click on the show to be	Show id	The show was removed	The show was removed	PASS

				remove d				
ST_3 1	EM_Sel	Add a movie, verify that is added	User must be logged in and a show should be selected	Type the IMDB id and click add movie and the click on the link shown to go to the movie page	Imdb id tt4432 021	The movie was added	The movie was added	PASS
ST_3 2	EM_Sel	Remove a movie, verify that is removed	Users should be logged in and a movie should have been added previously.	Type the IMDB id and remove movie and the click on the link shown to go to the home page	Movie id	The movie was remove d	The movie was remov ed	PASS

Successful Run log

Loading Driver
E:\Dev\MyMovie\tests\runner.py:27: DeprecationWarning: use options instead of chrome_options
  w = webdriver.Chrome(

DevTools listening on ws://127.0.0.1:56669/devtools/browser/b392b1d0-1afc-48a3-8dee-bc557eeef409
Loaded driver
Press enter to continue

Resetting DB
Reset DB
Movie Listing and Selection

Backend Tests

1: Search route: EMPTY TERM
-> Search for an empty and verify results
Result: PASSED
Movie items found: 21

2: Search route: Avengers
-> Search for a term and verify results
Result: PASSED
Movie items found: 3

3: Search route: Case insensitive
-> Make sure searches are case insensitive
Result: PASSED
Movie items found: 3

4: Movie routes
-> Check a valid movie route and an invalid, verify results
Result: PASSED
Movie routes are handled correctly

Selenium Tests

5: List All Movies
-> List all movies in the home page and check if all are shown
Result: PASSED
Movie items found: 21

6: Search for a term
-> Search for term "avengers" and verify results
Result: PASSED
Movie items found: 3

7: Case Insensitive search
-> Make sure searches can be case insensitive
Result: PASSED
Movie items found: 3

8: A term not found
-> Make sure no results are returned when term doesn't match all movies
Result: PASSED
Movie items found: 0

9: Select a movie
-> Select a movie and check if it shows more details about it when clicked
Result: PASSED
Found Iron Man 2

Seat Selection and Booking

Backend Tests

Reset DB
10: Book ticket route
-> Request to book tickets which are available and valid, verify response
Result: PASSED
Received correct response {"result": "Ticket booked", "data": {"mid": "60797618920c419d02d26e9d", "sid": 0, "tickets": [[3, 0], [3, 1]], "cost": 4000.0, "showdate":

{"$date": 1621641600000}, "name": "C
aptain America: The First Avenger"}}

11: Book ticket route: Invalid tickets
-> Request to book tickets with invalid body, verify response
Result: PASSED
Received correct error response {"err": "invalid literal for int() with base 10: '-'"}

12: Book ticket route: Unavailable tickets
-> Request to book unavailable tickets, verify response
Result: PASSED
Received correct error response {"err": "Some of the tickets are not available"}

13: Show route
-> Verify that booked tickets are really booked
Result: PASSED
Ticket confirmed to be booked

Selenium Tests

Reset DB
14: Check Shows
-> Check if shows are displayed for a movie
Result: PASSED
Show found May 22
9:AM

15: Select Show
-> Select a show and verify seat matrix is displayed
Result: PASSED
Show is selected

16: Select seats
-> Select seats and make sure the price displayed is correct
Result: PASSED
Price is correct

17: Book seats
-> Book selected seats and make sure the booking is made
Result: PASSED
Ticket booked

18: Check seats are booked
-> Verify that the seats booked are black when visiting it again
Result: PASSED
Seats are booked

Event Management

Backend Tests

Reset DB
19: will_showdate_collide
-> Verify that the same date can't be chosen for a show again
Result: PASSED
Returned correct result

20: get_free_rooms
-> Verify that all free rooms are returned for a particular date and slot
Result: PASSED
Returned correct result

21: add_show_valid
-> Verify that a valid show is added
Result: PASSED
Show was added

22: add_show_existing
-> Verify that an existing show can't be added again
Result: PASSED
Existing show was identified

23: add_movie_valid
-> Verify that a new movie can be added with imdb id
Result: PASSED
Movie was added successfully

24: add_movie_invalid
-> Verify that a invalid movie is not added which doesn't exist in imdb
Result: PASSED
Invalid Movie was identified

Selenium Tests

Reset DB
25: Check Admin access
-> Login as different users, check that only admin can add movies
Result: PASSED
Access is correct

26: Show date invalid

-> Select an invalid date and check if it isnt allowed
Result: PASSED
Invalid date is not accepted

27: Show date valid
-> Select a valid date and check if doesn't allow it
Result: PASSED
Valid date is accepted

28: Check if rooms are available
-> Select show date and slot, verify if all rooms are returned
Result: PASSED
All rooms are returned

29: Add a show
-> Add show with required details, verify that it is added
Result: PASSED
The show was added

30: Remove a show
-> Remove a show, verify that is removed
Result: PASSED
The show was removed

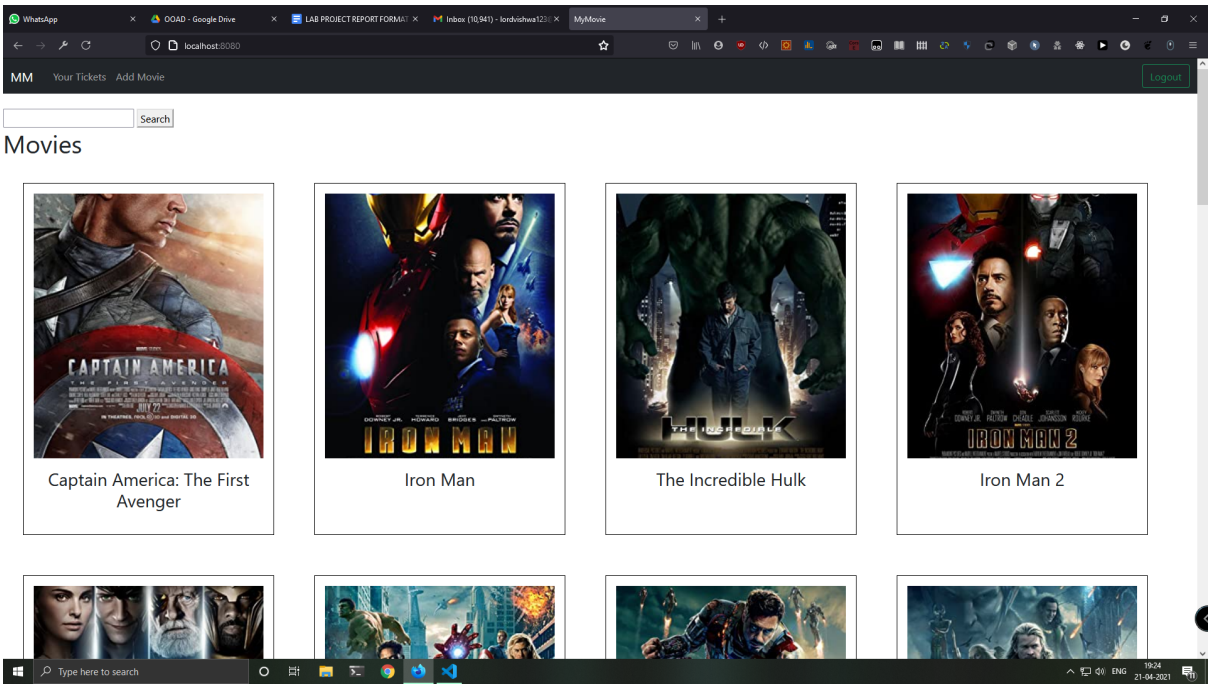
31: Add a movie using imdb id
-> Add a movie, verify that is added
Result: PASSED
The movie was added

32: Remove a movie
-> Remove a movie, verify that is removed
Result: PASSED
The movie was removed

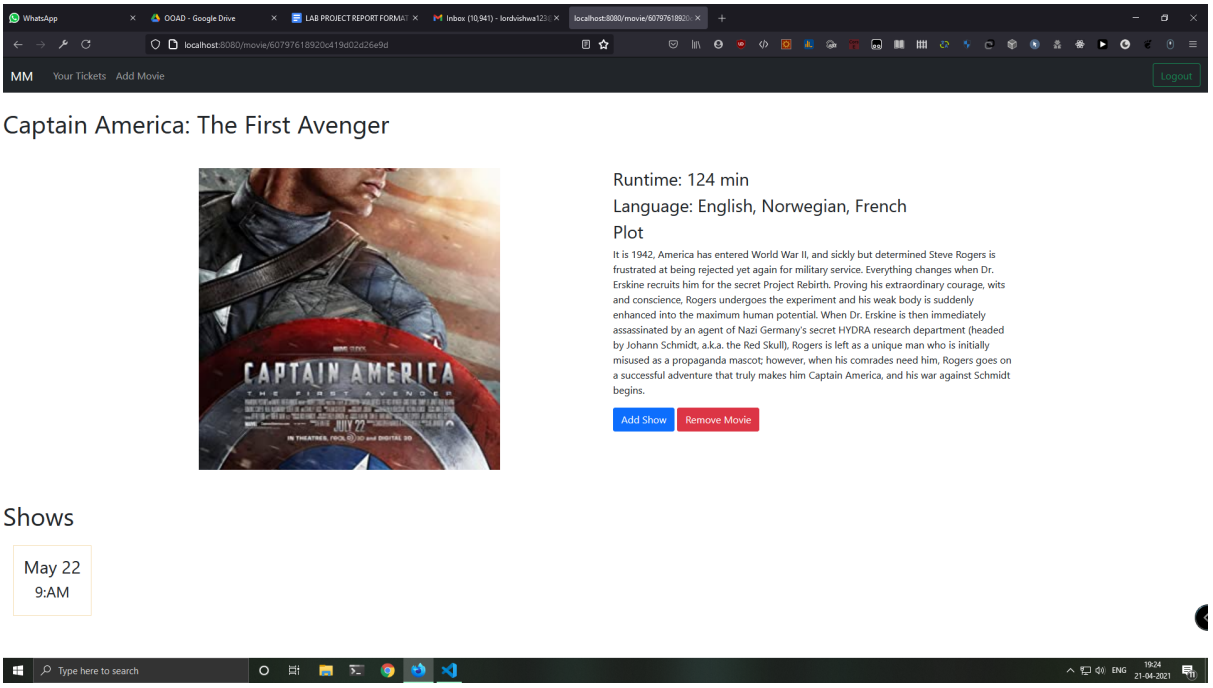
PASSED 32 / 32

6. Screenshots of Output

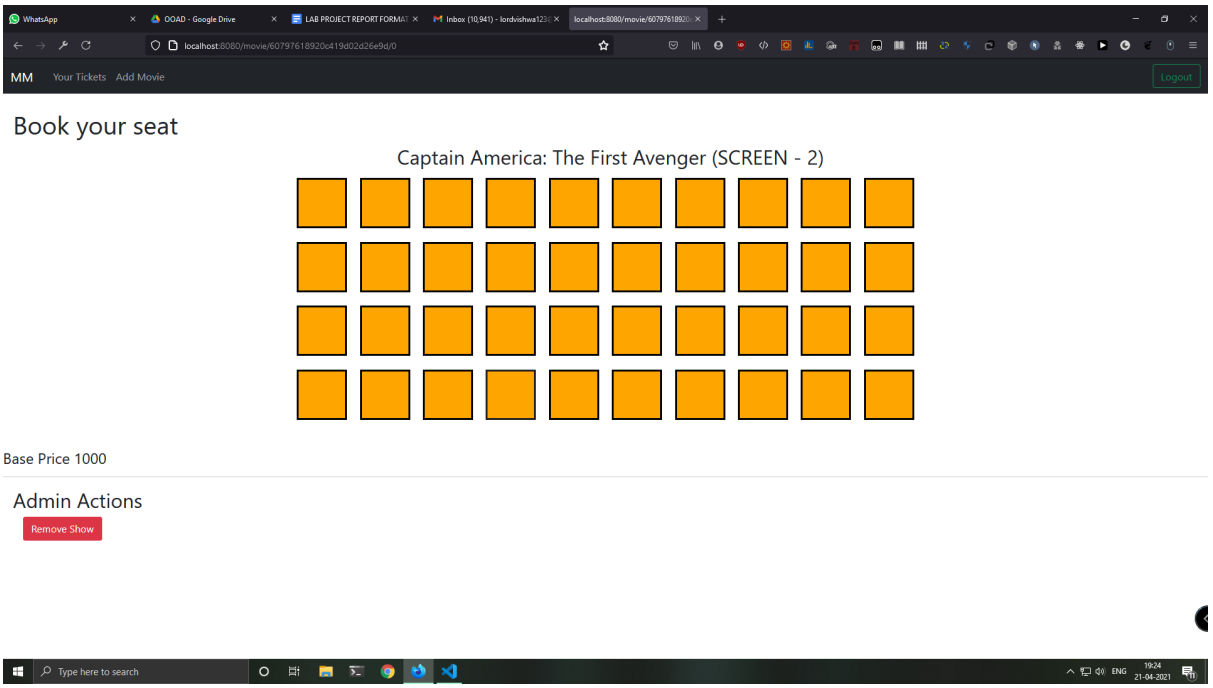
Main Page



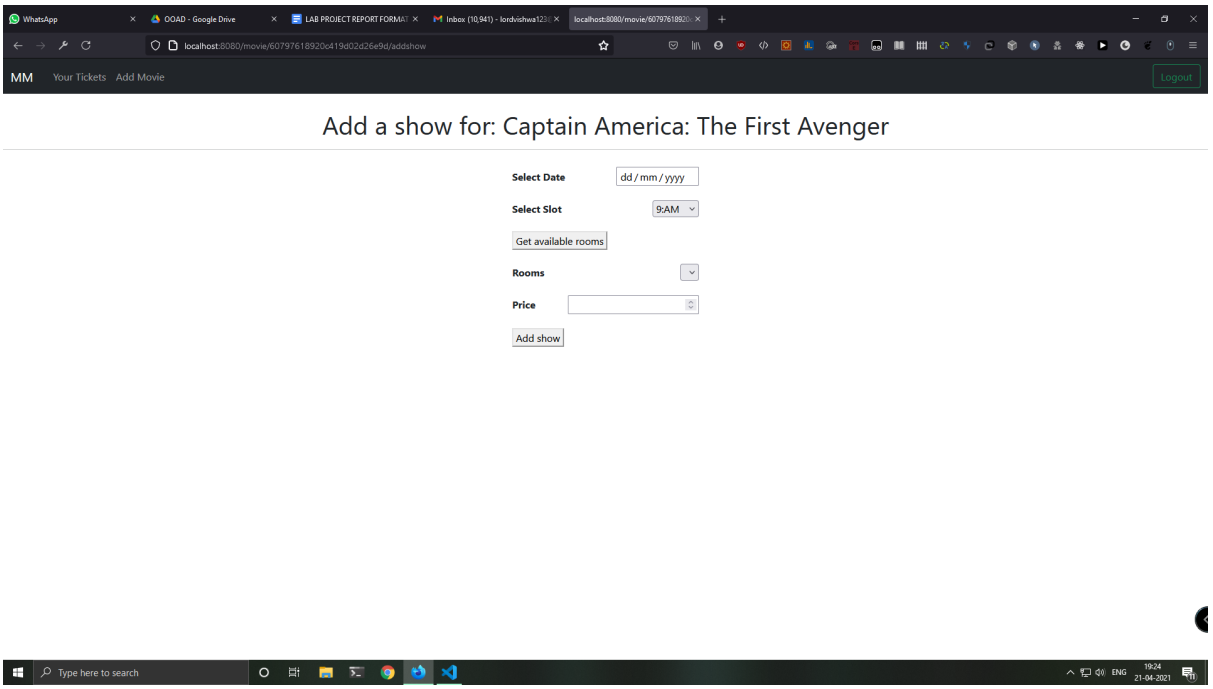
Movie Page



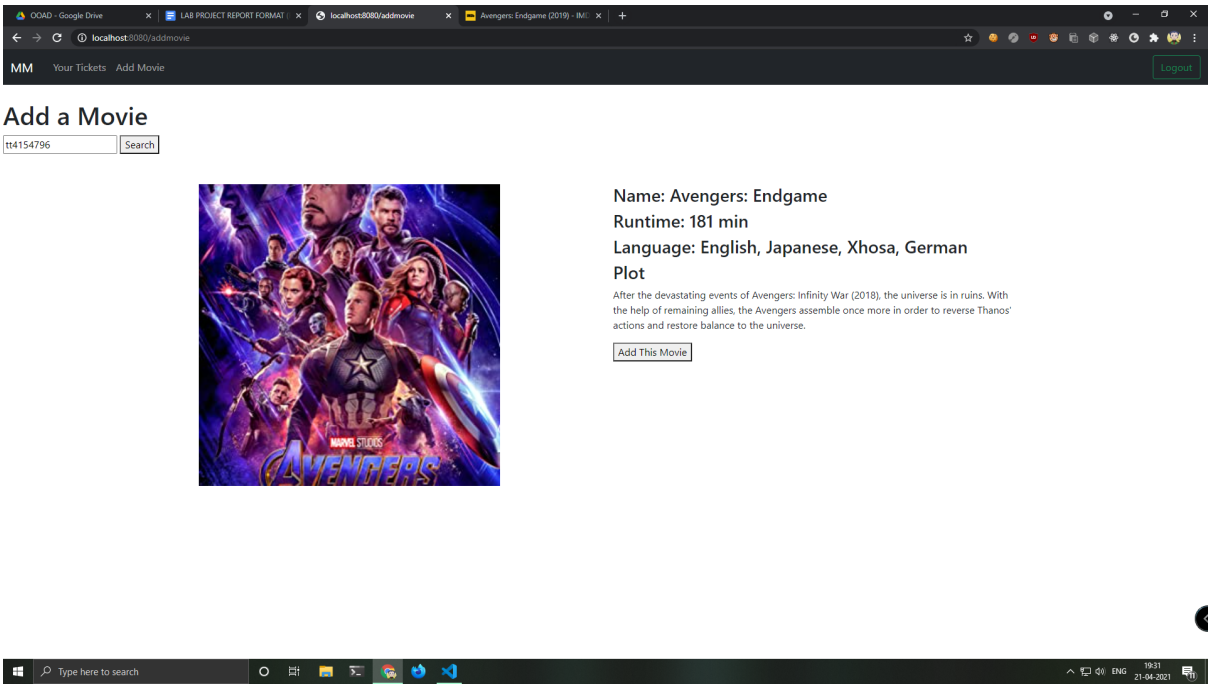
Seat Selection Page



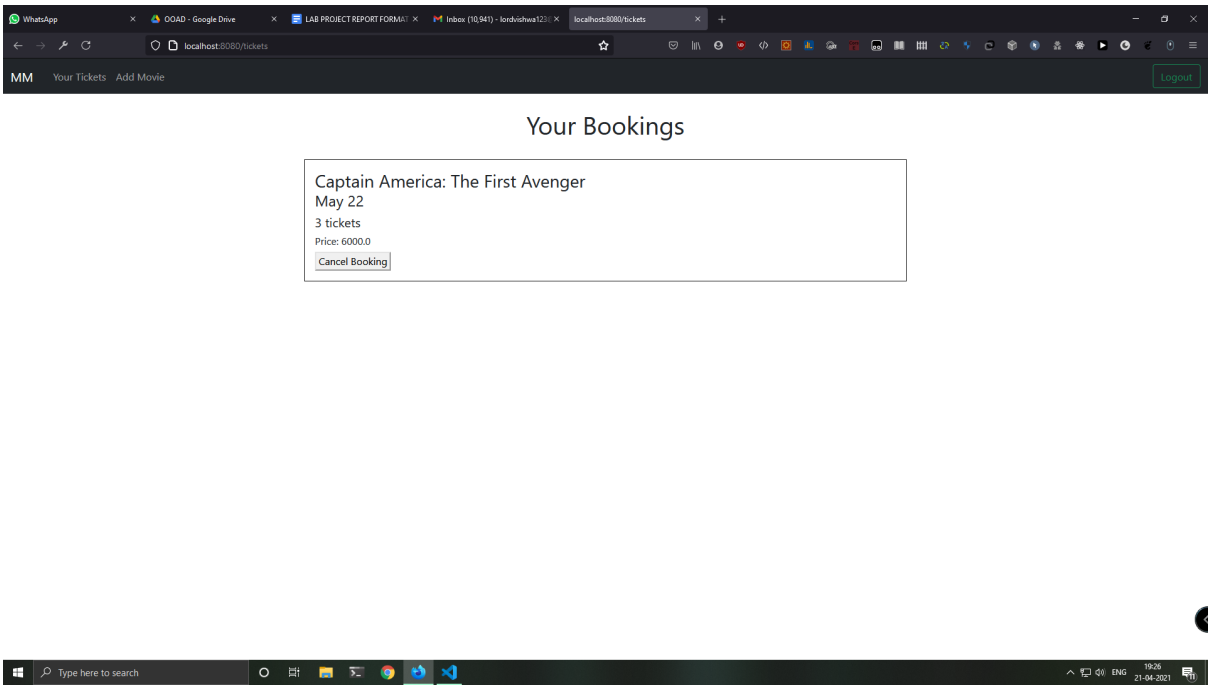
Add Show Page



Add Movie Page



Your Bookings



Test Case Run