



UNIVERSITY OF  
**WEST LONDON**

**Advanced Software Engineering**  
**Software Development of Online Theatre Ticket System**

Hamza Tariq	21462189
Borys Kubisty	21383564
Ahmed Yunus	21499384
Christoforos Arampidis	21510792
Preethi Merline Anthony	21478214

# Contents

Table of figures .....	3
1. Introduction .....	5
2. Work Division .....	6
3. Structure of the plan driven development .....	7
4. Requirements Engineering.....	10
5. System Modelling.....	12
5.1 Data Flow Diagrams .....	12
5.2 Use Case Diagram .....	14
5.3 Activity Diagrams .....	14
5.4 Sequence Diagrams.....	18
5.5 Class diagrams.....	23
5.6 Wireframing .....	25
5.7 Algorithm flowchart.....	25
6. Implementation .....	27
7. Testing Procedures.....	41
7.1 Behaviour Driven Development.....	41
7.2 Test Driven Development .....	43
8. Critical discussion – Conclusion .....	48
References .....	49
Appendix .....	50

## Table of figures

Figure 1 - Plan Driven Approach .....	8
Figure 2 - Gantt Chart.....	9
Figure 3 - Level 0 DFD.....	12
Figure 4 - Level-1 DFD .....	13
Figure 5 - Use Case Diagram .....	14
Figure 6 - Guest/Visitor Activity Diagram .....	15
Figure 7 - Staff Activity Diagram .....	16
Figure 8 - Admin Activity Diagram .....	17
Figure 9 - Staff Sequence Diagram.....	19
Figure 10 - Member Sequence Diagram .....	20
Figure 11 - Visitor Sequence Diagram.....	21
Figure 12 - Admin Sequence Diagram.....	22
Figure 13 - Business Concept Model (Class Diagram).....	23
Figure 14 - Business Type Model (Class Diagram) .....	24
Figure 15 - Algorithm Flowchart .....	26
Figure 16 - Database Table- Response Metrics.....	28
Figure 17 - Database Table-2 .....	28
Figure 18 - Backend.....	29
Figure 19 - Frontend.....	30
Figure 20 - Frontend.....	31
Figure 21 - Customer Homepage .....	32
Figure 22 - Customer Registration .....	32
Figure 23 - Customer Login .....	33
Figure 24 - View Events for Customers .....	33
Figure 25 - Customer Booking an Event.....	34
Figure 26 - Customer Checkout Screen.....	34
Figure 27 - Customer Successful Booking .....	35
Figure 28 - Customer Account Screen with Cancel Option.....	35
Figure 29 - Staff and Admin Login.....	36
Figure 30 - Staff Dashboard .....	36
Figure 31 - Staff Event List .....	37
Figure 32 - Staff Booking for Customer.....	37
Figure 33 - Staff Booking Confirmation.....	38

Figure 34 - Staff Reference Search and Cancel Ticket Screen.....	38
Figure 35 - Admin Dashboard .....	39
Figure 36 - Admin Create Event Screen .....	39
Figure 37 - Admin Modify or Delete Events.....	40
Figure 38 - BDD and TDD.....	41
Figure 39 - TDD 1.....	44
Figure 40 - TDD 2.....	44
Figure 41 - TDD 3.....	44
Figure 42 - Manual Testing 1.....	46
Figure 43 - Manual Testing 2.....	46
Figure 44 - Manual Testing 3.....	47
Figure 45 - Account Page .....	50
Figure 46 - Book Event .....	51
Figure 47 - Checkout .....	52
Figure 48 - Event List.....	53
Figure 49 - Homepage.....	54
Figure 50 - Register .....	55
Figure 51 - Sign In.....	56

## 1. Introduction

The aim of this project is to create a software application for Online Theatre Ticket system for a new movie theatre in Ealing that can be accessible via a mobile app or a website. Customers will be able to buy tickets online for the movie and theatre of their choice, as well as have some other facilities. For the development of this application we were asked to use plan driven development approach in this assignment. Plan driven development is an sequential process which includes all the phases of software development life cycle. The goal of this assessment is to develop application using customer given requirements, analysing, and implementing them. The scope of this assessment is to produce an algorithm which helps in decreasing single seat scattering in the theatre. This report shows the implementation of the application and the usage of plan driven development.

This report gives an overview of development of a web application for the topic Online Theatre Ticket System by following plan driven approach. This report comprises of the phases involved in plan driven development and has following sections; section 2 provides information about the work division among all the team members for successful completion of the application, section 3 explains the structure of plan driven development used, section 4 is an overview of requirements engineering where we investigate the requirements and expectations of our application, section 5 shows the system modelling where we designed different UML models for understanding aim of our application in detail, section 6 provides the implementation of the application, section 6 comprises of testing procedures used to test our application, section 7 and 8 includes conclusion describing the analysis of whole development process and the references.

## 2. Work Division

Planning of general structure which included UML drafting and selection technology stack of our project was discussed as a group during meeting throughout the lifecycle of the project.

- Hamza – Frontend engineer which performed wireframing that is matching the requirements and feature UI and UX aspects that compliment behaviour-driven development. Sourced and setup styling used within the websites, and coded visual elements.
- Chris – Frontend engineer responsible for handling website state management and flow of actions with the backend. Setup model-view-controller structure within the React environments alongside routing for different pages.
- Borys – Frontend engineer overseen progress and handled issues occurring between frontend and backend
- Yunus – Head engineer for backend functionality. Orchestrated cloud architecture and automated deployment to the cloud from our GitHub repository.
- Preethi – Backend engineering and responsible for ensuring tests are covering the whole functionality. Reviewed if test-driven development tests were in-line with behaviour-driven development specification.

### 3. Structure of the plan driven development

Development of different software products within given time and budget providing best quality is complex and many software companies follow an approach for developing software. One such approach is plan driven development. Plan driven development can be defined as a software development method which comprises of planning and developing all the requests which user needs. In this method, all the planning is executed in different levels. Plan driven approach is suitable to many software products because of its task specific levels, planning, gathering requirements, implementing those requirements, testing, and maintaining the software product. The most popular plan driven approaches are waterfall model, spiral model and DSDM. For our application, we used waterfall model to manage the development process. (Satyabrata, 2021)

Waterfall model can be defined as software development approach which comprises of various stages to emphasise starting to end of the project. It is a sequential approach which has the following phases:

- Planning
- Requirement Analysis
- Software Design
- Implementation
- Testing
- Maintenance

In this model, each phase should be completed to start the next phase. Waterfall model is advantageous as the customer requirements are gathered in the beginning and each member of team know what is needed to be done. It is also beneficial to calculate the time and duration to complete the software development. These waterfall model steps of our project are explained in below sections in detail. (The Manager's Guide to Mixing Agile and Waterfall, n.d.)

The below figure shows the waterfall model approach used for our project:



Figure 1 - Plan Driven Approach

To further explain, the usage of waterfall model and structure of our project, we have made a project schedule with Gantt chart. Gantt chart can be defined as a bar chart which is used to display project schedule, it includes the tasks and duration. This Gantt chart helped in planning our work, deadlines, and resources. The below figure shows the structure of our project with deadlines and the waterfall model steps,



Figure 2 - Gantt Chart

## 4. Requirements Engineering

Requirement engineering can be explained as a procedure of investigating the needs, documenting it, and managing our software system requirements which can be used in design process and in implementation (JavaTPoint, 2021). Requirements engineering is beneficial to understand the function of the application, how the end product should be and how it performs. While analysing system requirements all the team members discussed and have listed the requirements and understood what needs to be done. We defined requirements in two different categories:

### Functional Requirements

Functional requirements can be referred as a requirement to define the system and its components. The main aim of functional requirements is to describe how our software should work (Chitrasingla, 2021). Few functional requirements for our application are given below:

1. Guest shall be able to login – a guest of website must have the option to login to the system if he is already registered
2. Visitor shall be able to register – a visitor of website must be able to register to system
3. Visitor shall be able to view movie list – a movie list should be visible for visitor without any criteria fulfilled
4. Guest shall be able to view the seating plan per theatre – a seating plan can be viewed only by the guest not the visitor
5. Anyone can view all performances per theatre with dates and prices – a list of all performances, in all locations, should be available to be seen by anyone
6. Only registered members shall be able to buy tickets – a guest should be logged in and be registered member in order to make the payment and secure his booking
7. Members shall be able to choose seat as per his choice – a guest can choose any seat to his likeness according to the amount of the party he is booking and following app's algorithm to prevent scattering
8. Members shall be able to pay online – a booking payment should be able to complete with an online payment
9. Customer shall be able to cancel tickets – customers must be able to amend or cancel their booking through their account.

10. Admin shall be able to add/delete/modify performances – the administrator should be able to add delete and modify performances according to new events, popularity, and sales
11. Staff should be able to book tickets for customers – customers with no access to online bookings, should have the option to call or visit and staff should be able to book a ticket for them.

### Non-functional Requirements

It can be referred as a requirement that mainly focus on the quality of the application. The aim of non-functional requirements is to give information on how system performs (Chitrasingla, 2021). Few non-functional requirements for our application are given below:

1. Client should receive a booking reference after successful booking – after the completion of a booking, a customer should receive a confirmation of booking, providing a reference number.
2. A ticket should not be able to be cancelled in less than 24 hours – a customer should be able to manage his booking and cancel it but if he tries to do it in less than 24 hours, system should not allow it as it's too close to the show time.
3. Web application should be loaded within 10 seconds when the application is running.

## 5. System Modelling

System modelling helps in designing a conceptual model to describe a different perspective or view of the system (Tutorial Space, 2021) . System modelling is represented in pictorial or graphical way using unified modelling language (UML). In this phase, we created different system models by taking the above functional requirements in consideration. Below few UML diagrams are provided for online theatre ticket system.

### 5.1 Data Flow Diagrams

Data flow diagrams are used to understand the flow of information of the system. For our application, we used dataflow diagrams to show the interaction between system and external entities (Lucidchart, 2021). There are multi-level DFD's however, in this section level 0 is given which is then levelled up to level 1 diagram. These diagrams have different shapes to describe different functionalities.

Level-0 DFD is also referred as context diagram, it gives overview of whole process and relationship between external entities, below diagram shows context diagram of online theatre ticket system,

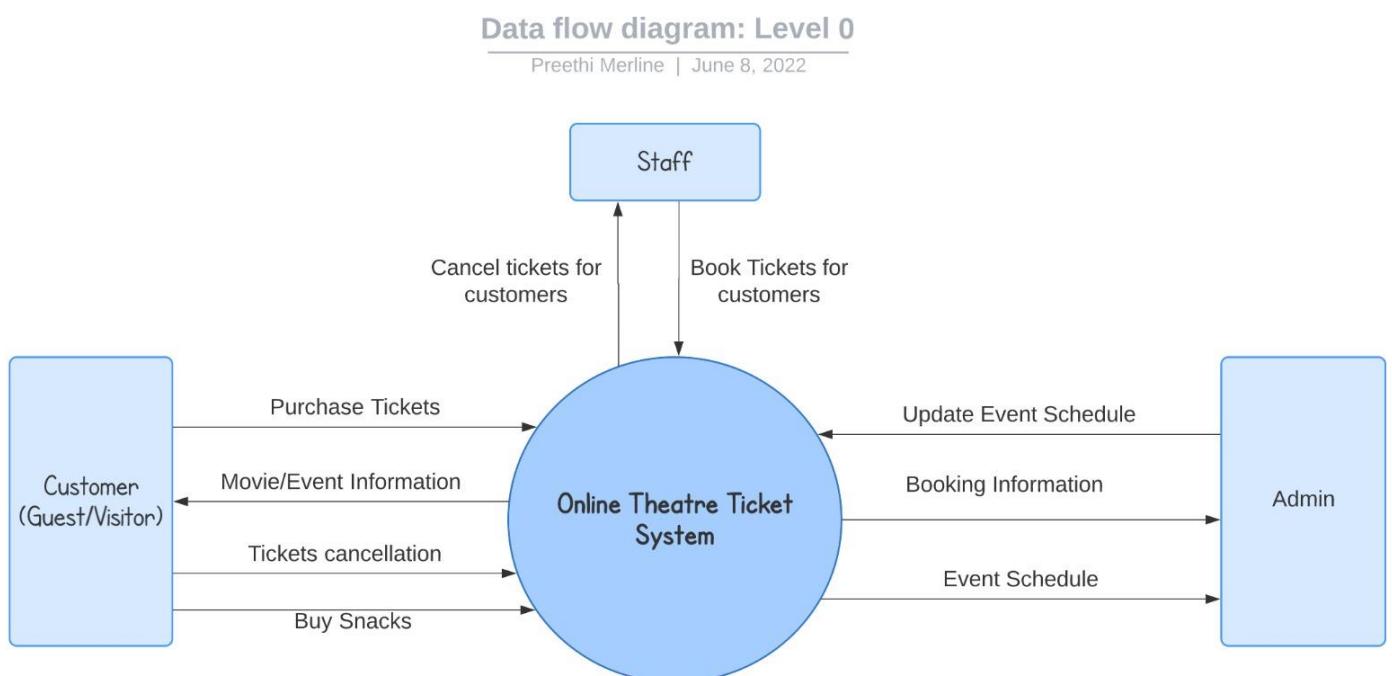


Figure 3 - Level 0 DFD

The above level-0 DFD is further split into level-1 DFD which has different process to provide detailed information of functions present in the system. Below diagram shows the levelled up 1 DFD.

## Data flow diagram: Level 1

Preethi Merline | June 8, 2022

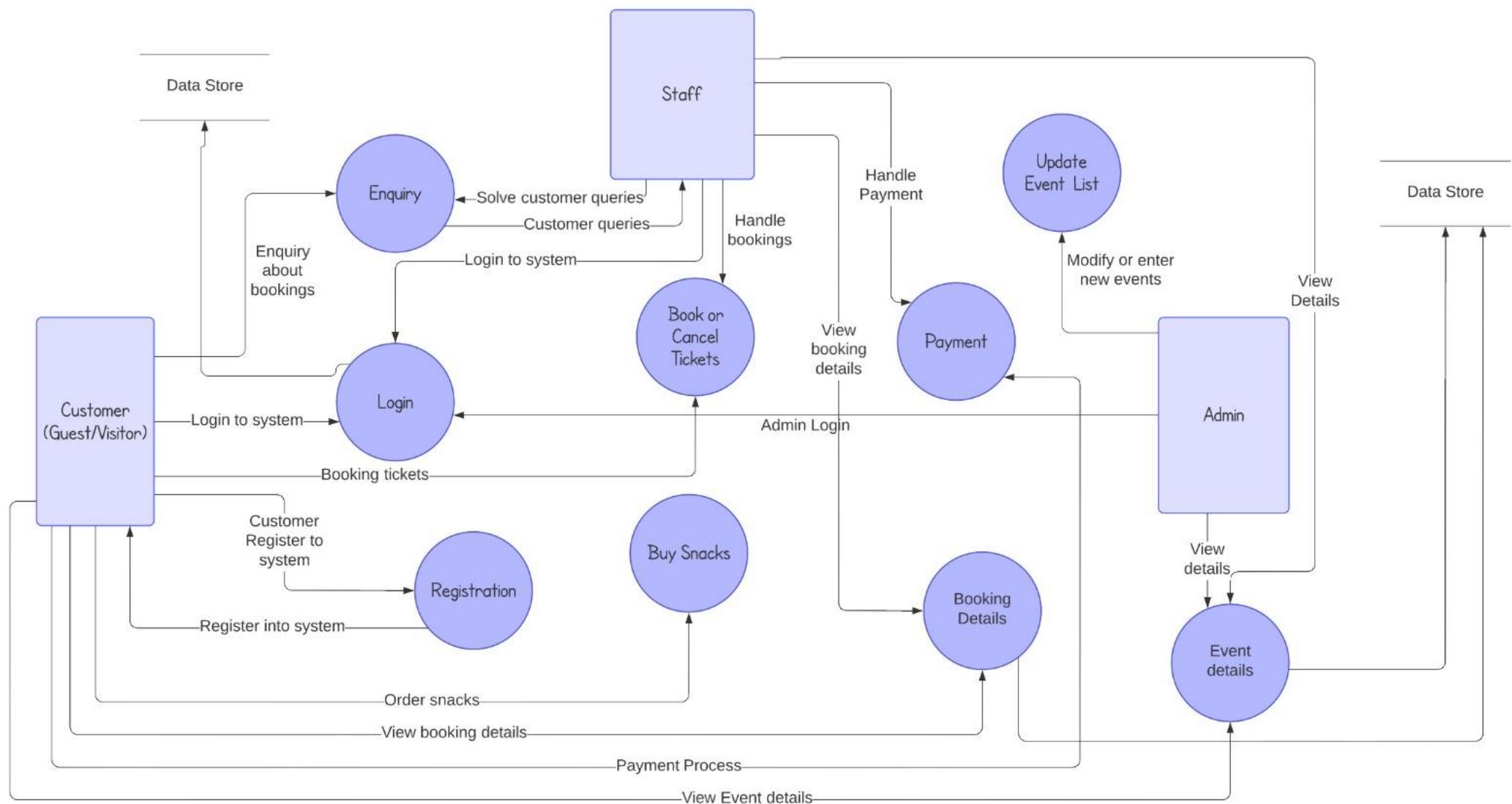


Figure 4 - Level-1 DFD

## 5.2 Use Case Diagram

Use Case diagram uses the above-mentioned functional requirements which system needs to fulfil and analyse it. Here all high-level requirements are considered as use cases and it shows relationship between its requirements with different actors present in system and association between them (Tallyfly, 2021).

Below the use case diagram of online theatre ticket system is given.

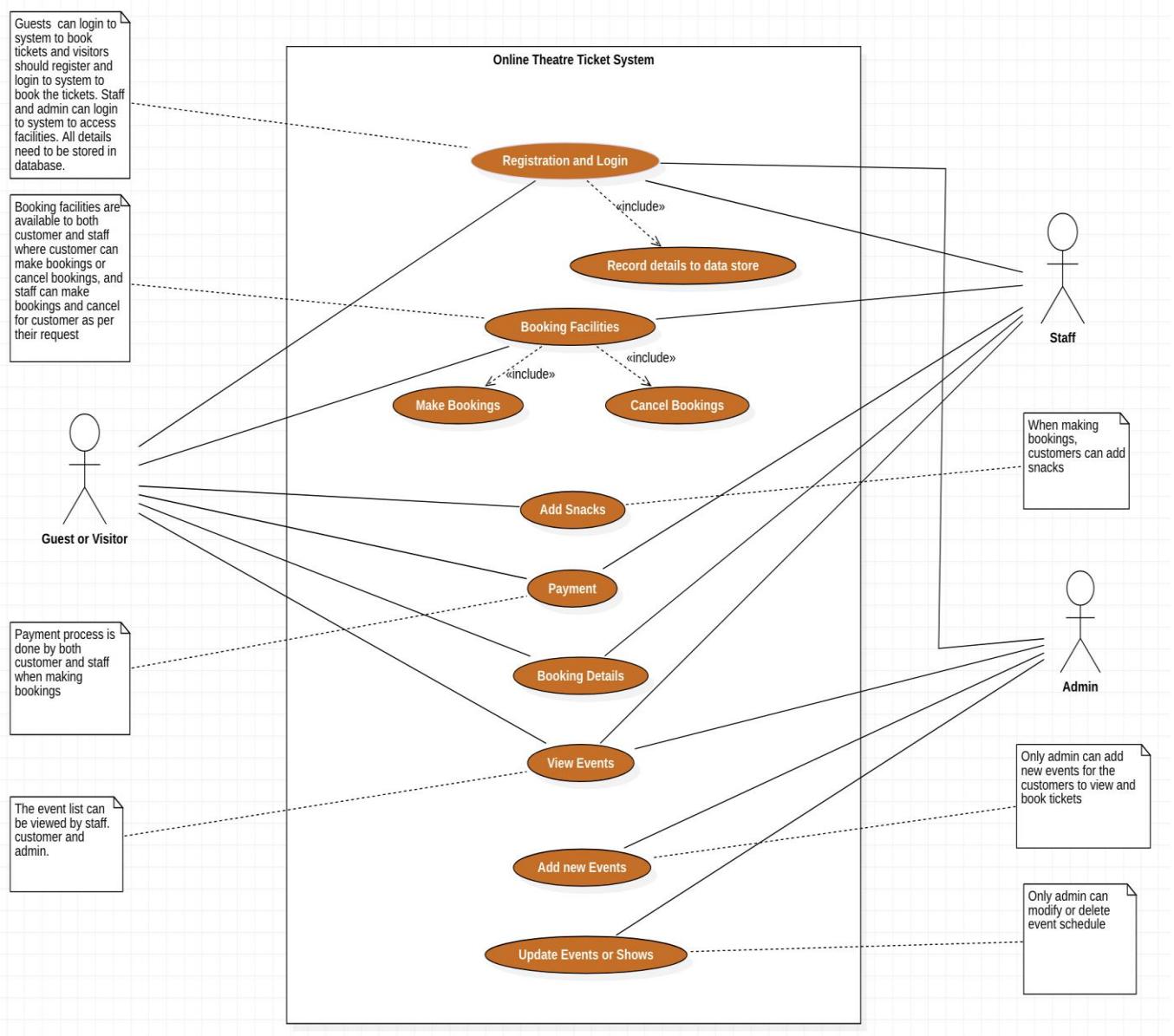


Figure 5 - Use Case Diagram

## 5.3 Activity Diagrams

Activity diagram is widely used to describe the business process model by giving information about flow of activities which can be parallel or sequential, it even describes relationships, conditions of the system (Tallyfly, 2021). Below 3 activity diagrams are given which gives details about flow of activities involved for customer, staff, and the admin. The customer here is guest/visitor, and this activity diagram shows activities of the customer to book events, cancel, register, login, adding snacks, and payment.

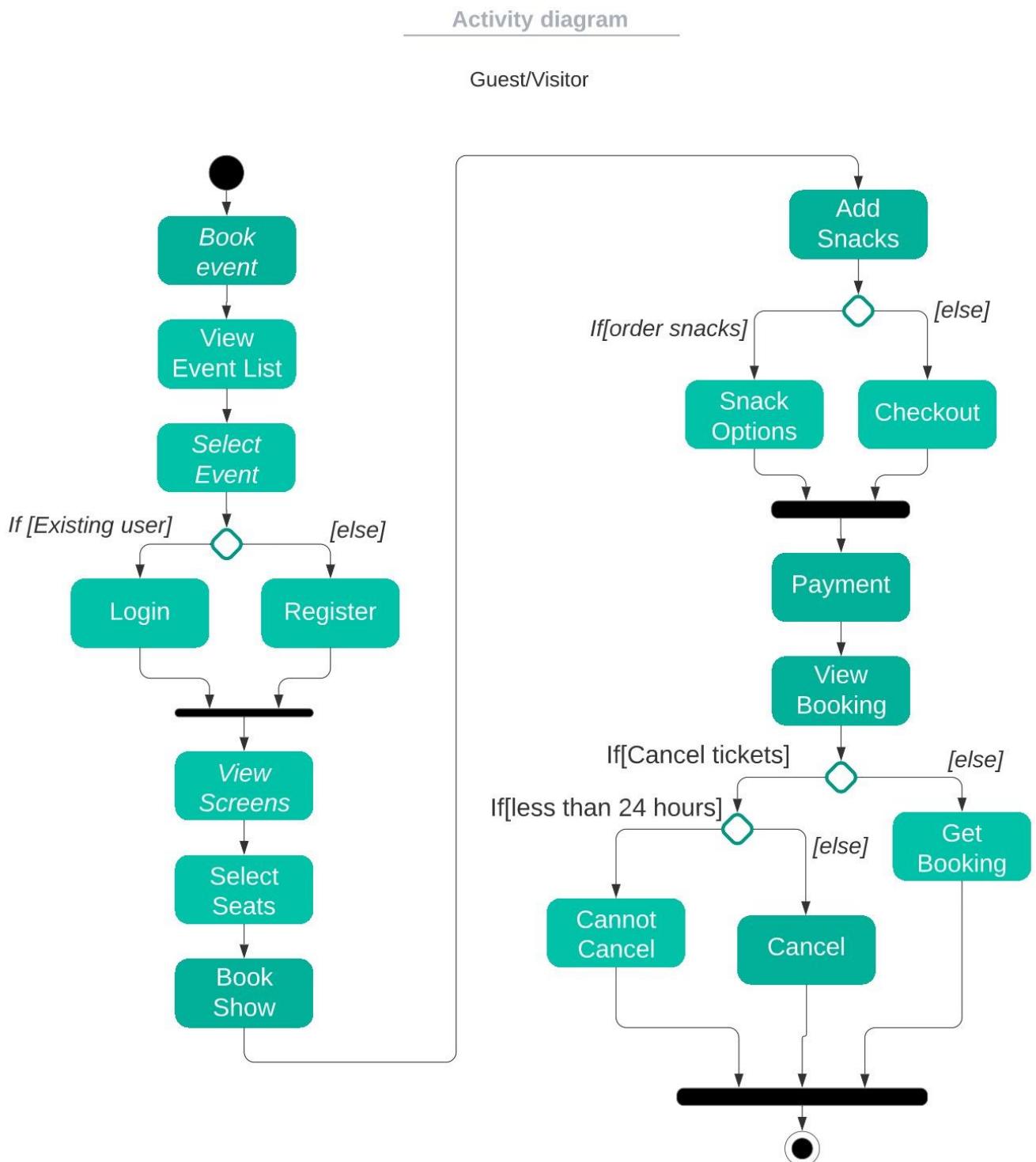


Figure 6 - Guest/Visitor Activity Diagram

The next diagram is the staff activity diagram which explains the actions of staff which are logging in, booking, or cancelling tickets as per customers request and other conditions as shown below.

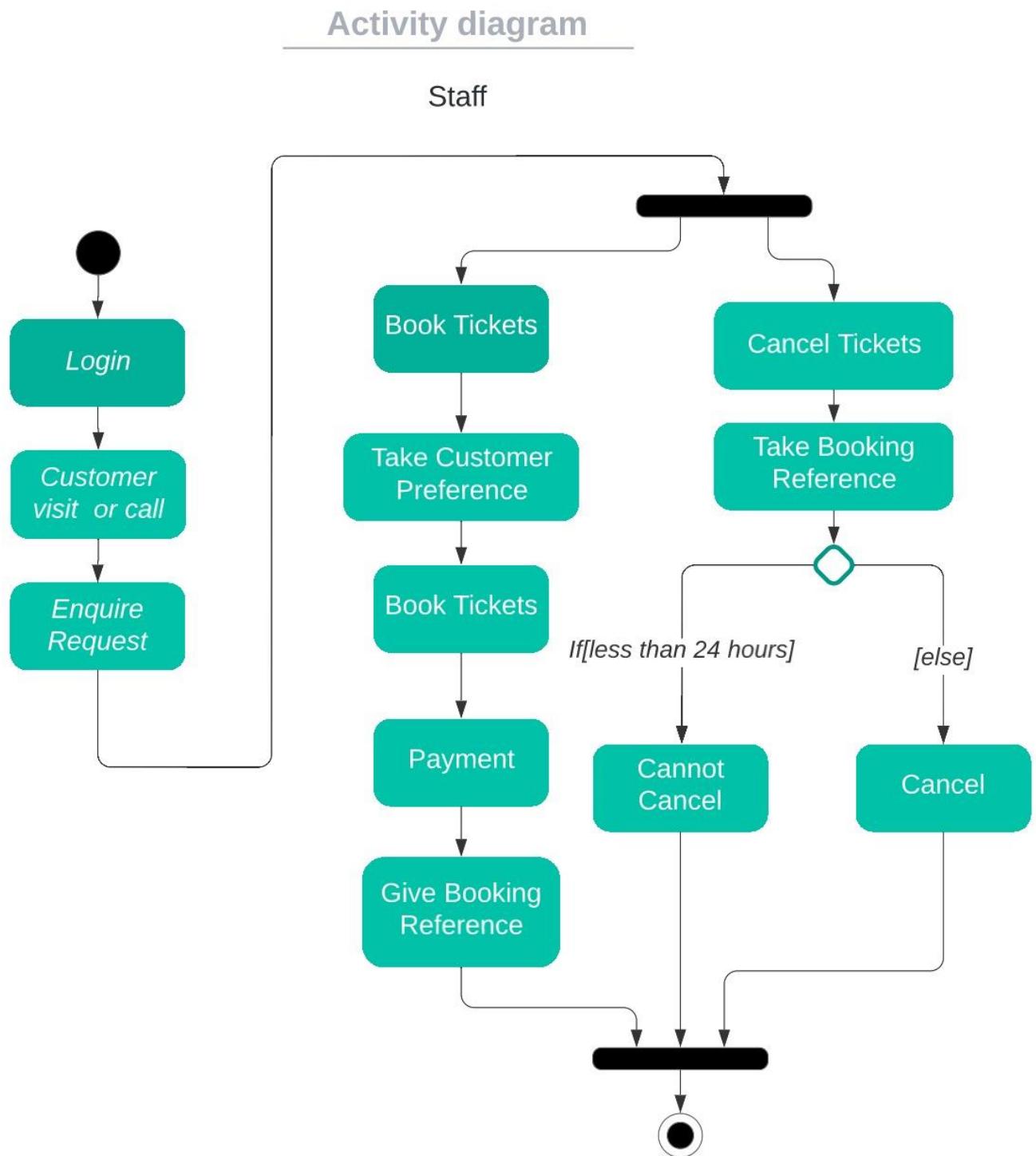


Figure 7 - Staff Activity Diagram

The below figure shows the admin activities and its flow in the system and demonstrates the business process.

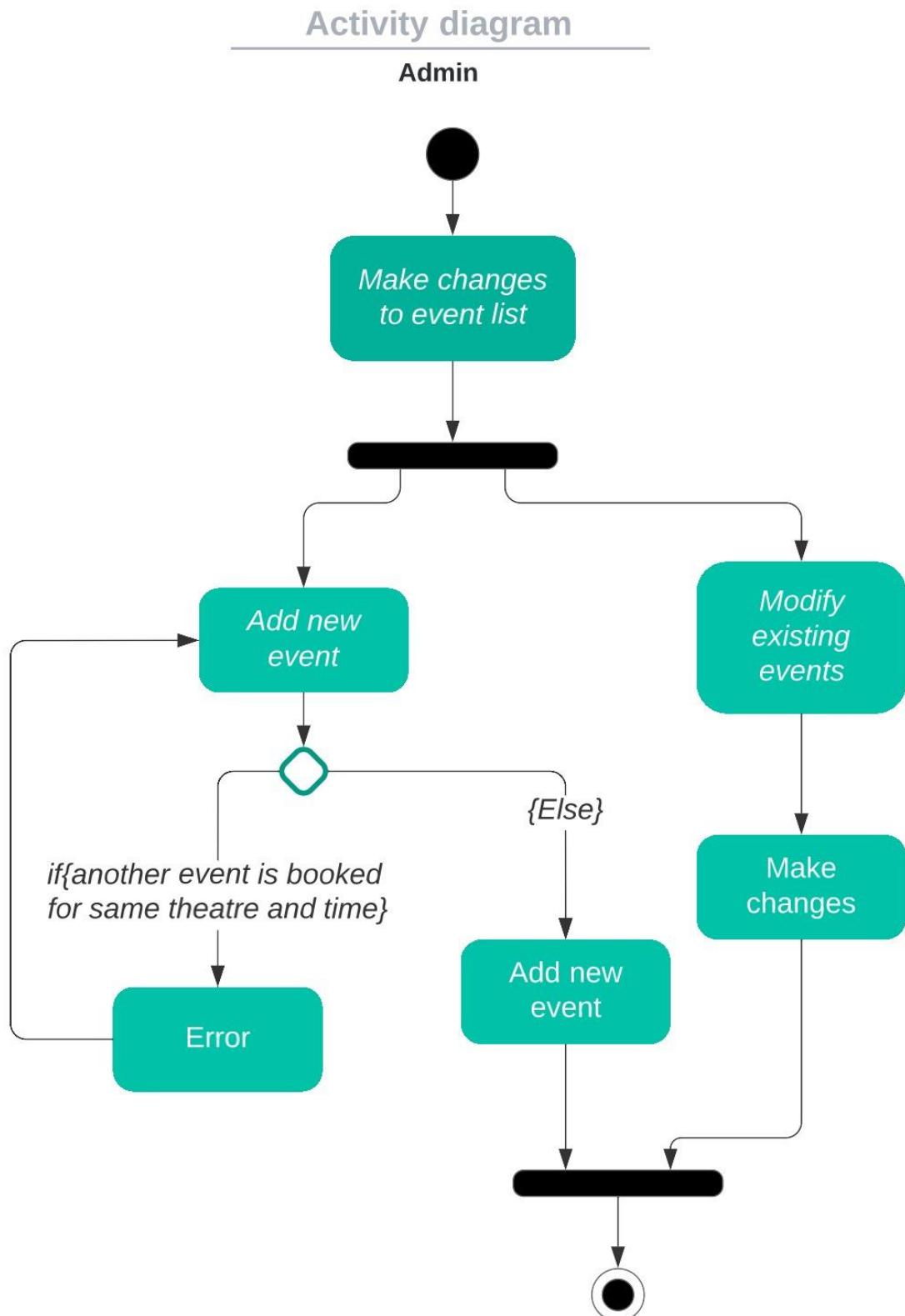


Figure 8 - Admin Activity Diagram

## 5.4 Sequence Diagrams

Sequence diagram is another widely used UML model which describes the behaviour of the system. This diagram shows the interaction between the system objects. These interactions have an order and are sequential between system components and the actors. There can be multiple objects or actors in a sequence diagram to support the system interaction (Tallyfly, 2021). This part consists of 4 sequence diagrams, each describing the behaviour of visitor (who is using the system for first time), member (an existing user), staff and the admin.

The below diagram is the staff diagram which shows the interaction of staff with the system and the customer showing its functionalities,

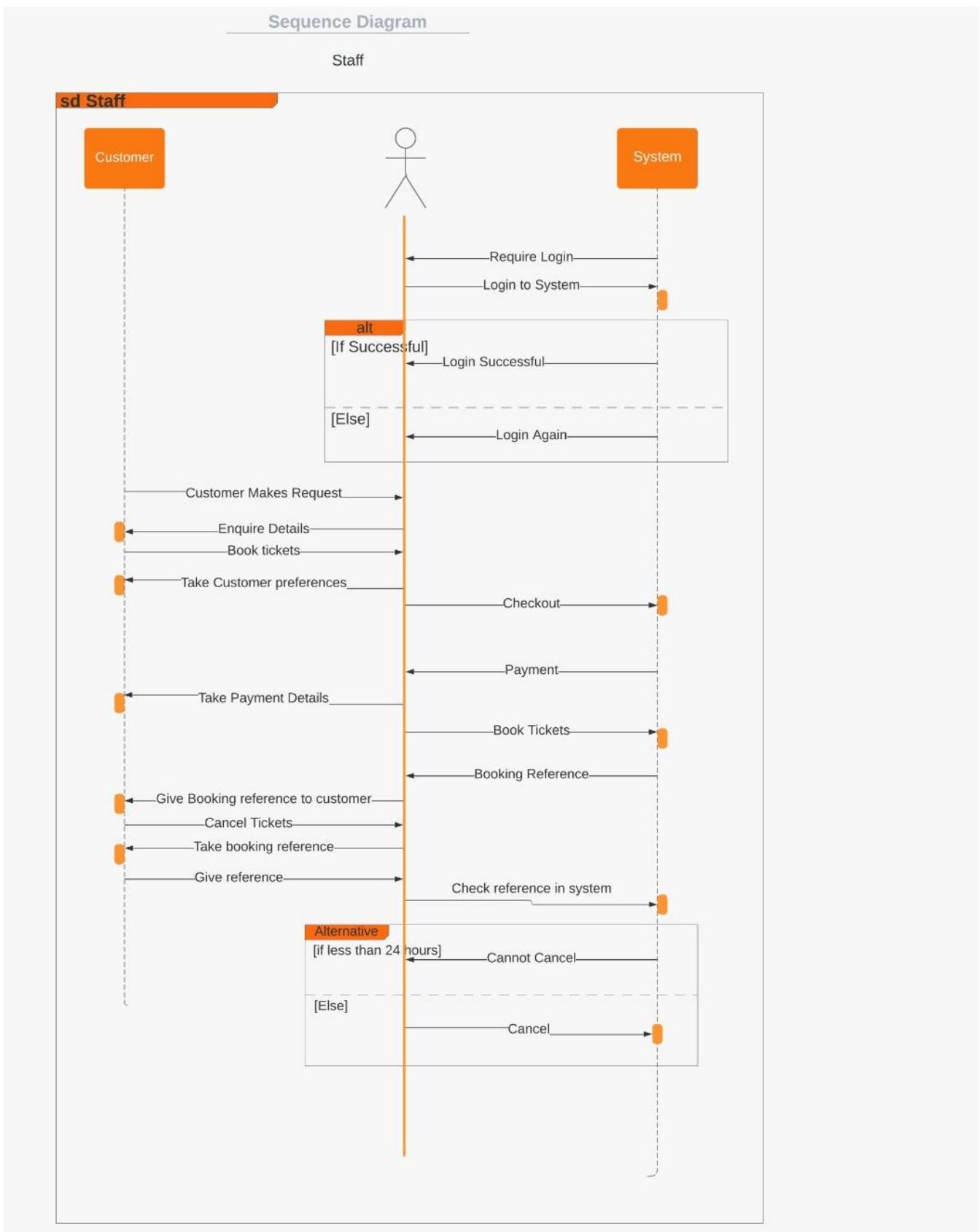


Figure 9 - Staff Sequence Diagram

The below diagrams are the visitor and member sequence diagrams which explains the customer behaviour of the system and shows the interaction and flow between customer with the system,

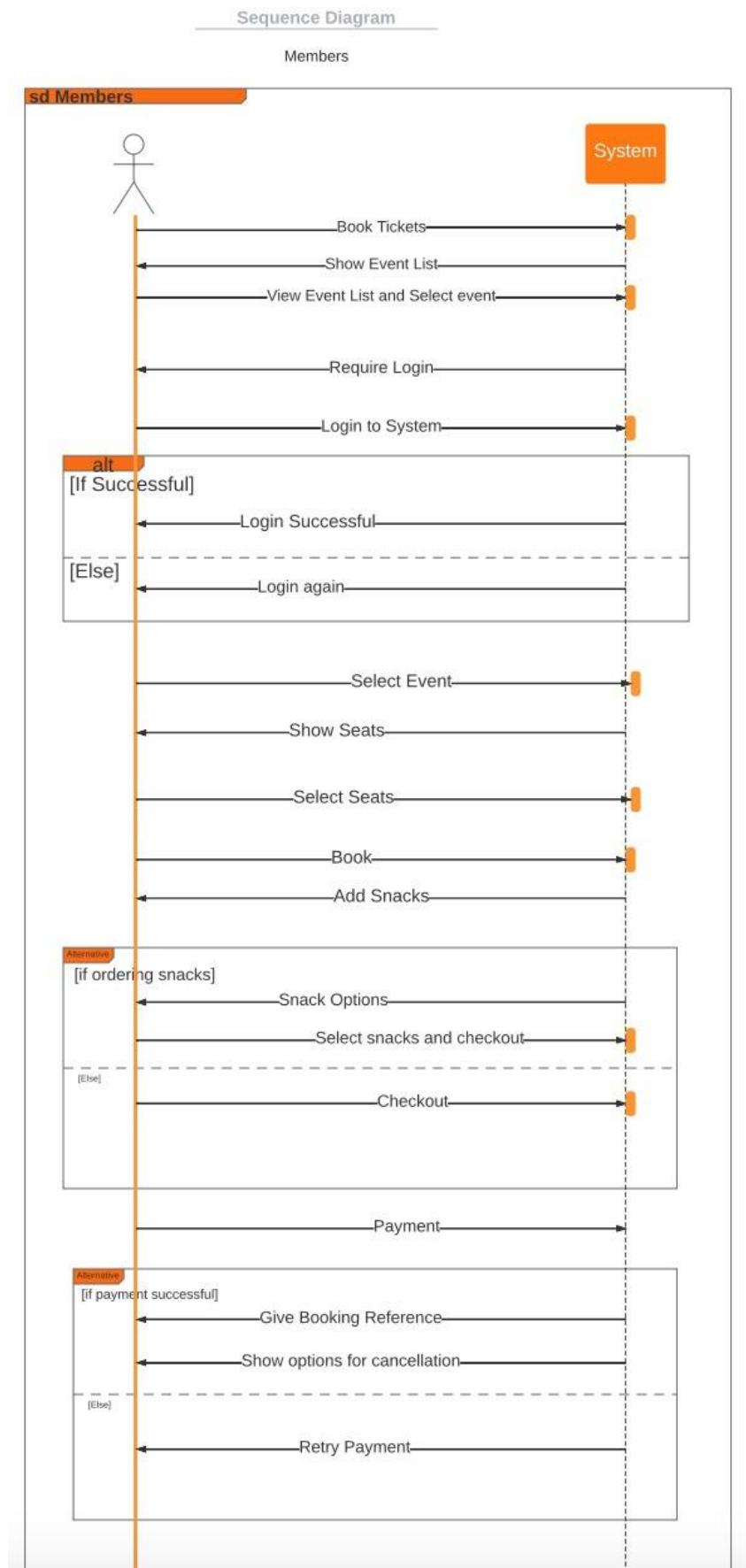


Figure 10 - Member Sequence Diagram

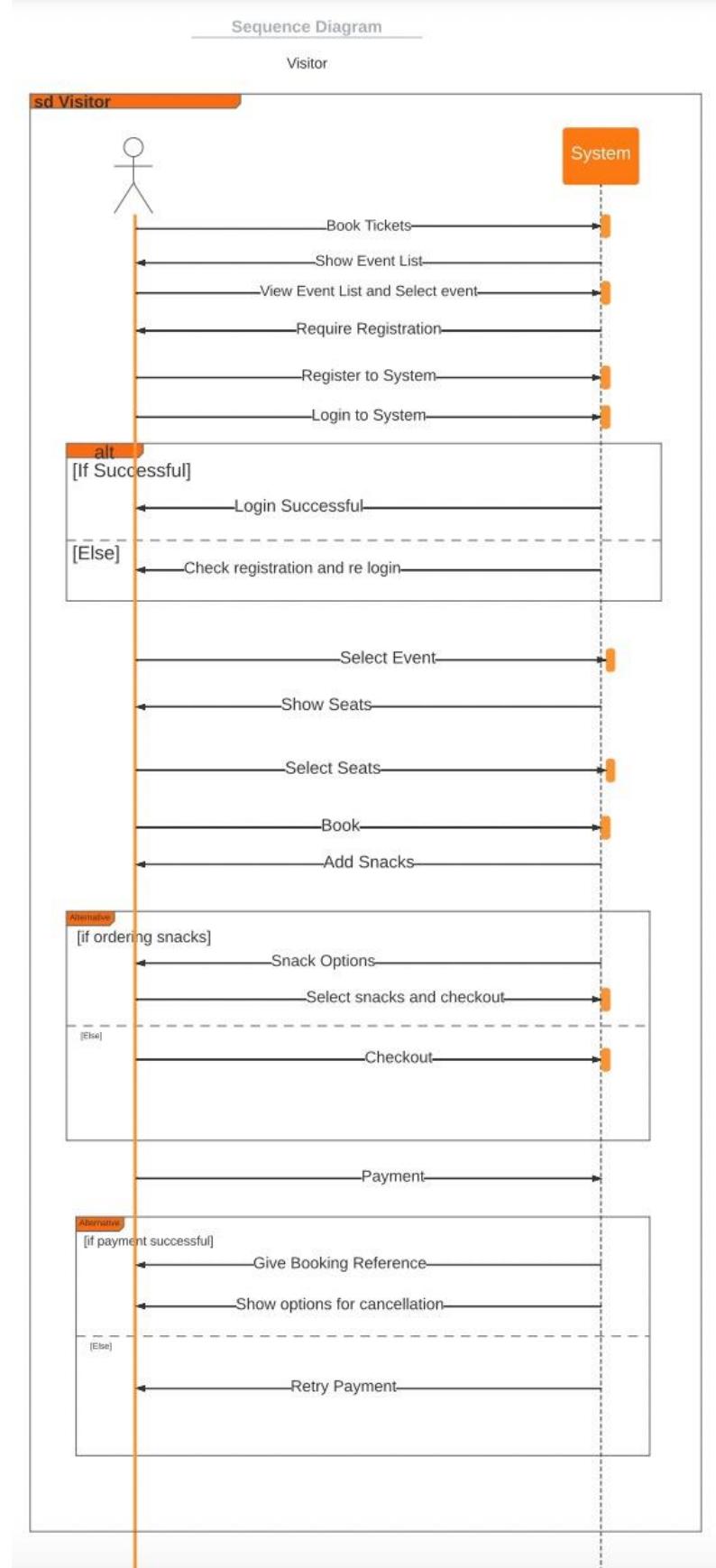


Figure 11 - Visitor Sequence Diagram

The next sequence diagram gives an overview of admin interaction with system and shows behaviour of admin and how he manages the application by adding events and other activities as shown below,

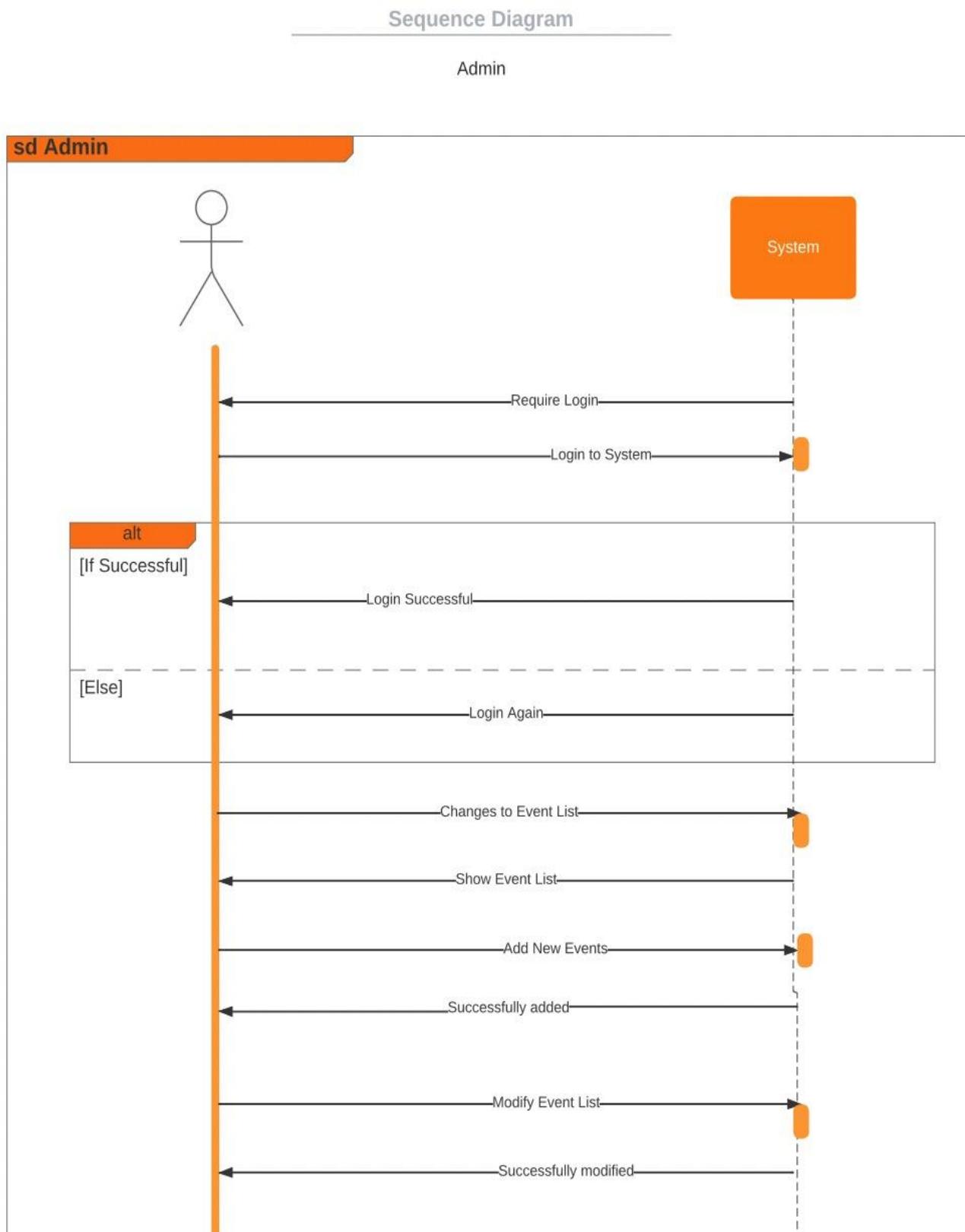


Figure 12 - Admin Sequence Diagram

## 5.5 Class diagrams

Class diagrams are most important structural UML model which helps in describing and developing the structure of the application. It helps and gives a path for the implementation phase. Class diagram follows the object-oriented concepts and consists of relationships and classes. In this section, we did 2 class diagrams, one showing the business concept model and other business type model.

The business concept model gives an overview of describing the business processes present and the organisation structure. It is an optimised class diagram (Tallyfly, 2021). The below figure shows the business concepts.

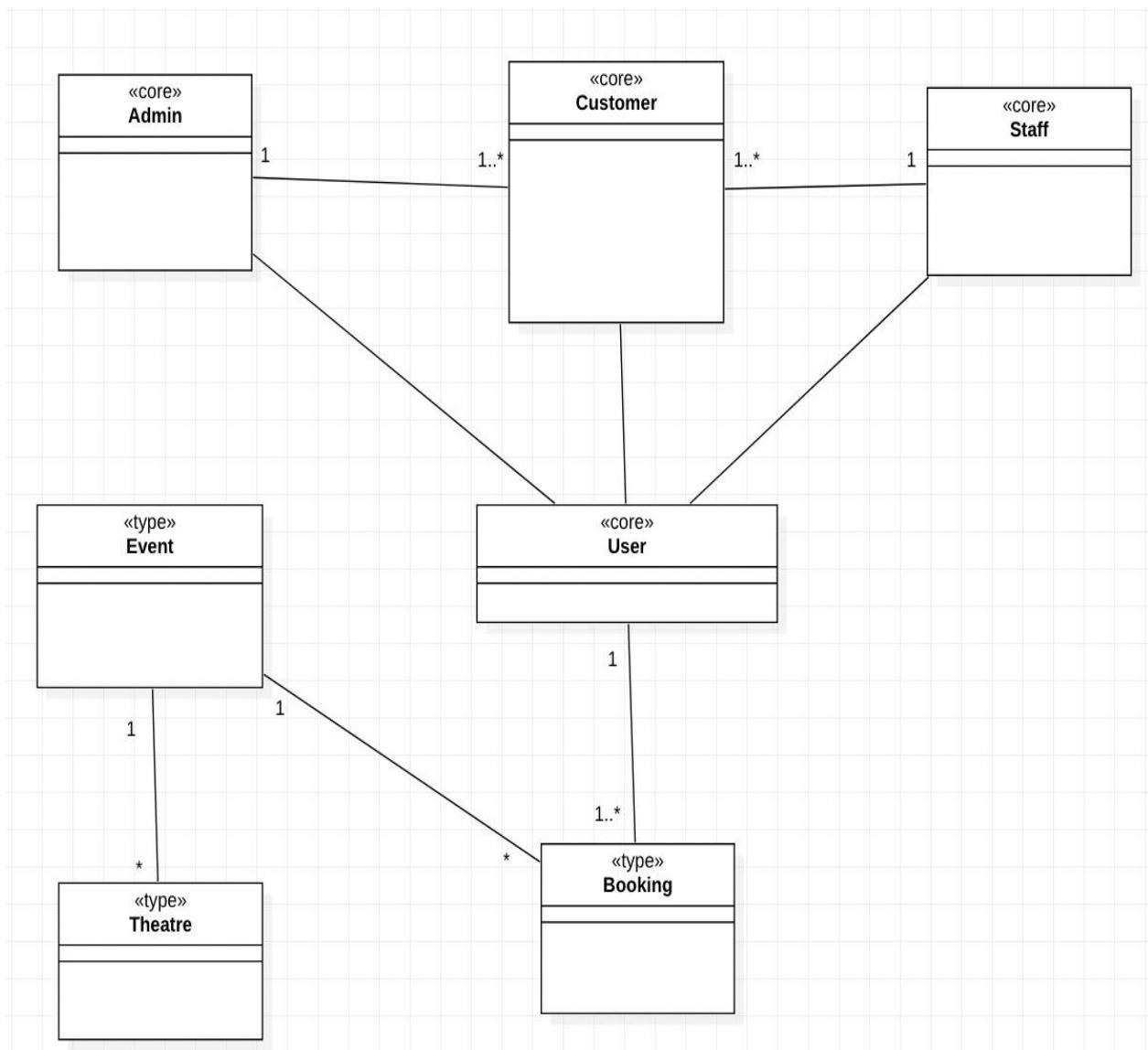


Figure 13 - Business Concept Model (Class Diagram)

The business type model can be defined as a static model which provides a static perspective or view of a system. This diagram shows classes, its attributes, operations, and relationship between the classes in a

detailed way (Tallyfly, 2021). Business type model is like an extension of business concept model. It creates the structure of the system. The below figure shows the business type model.

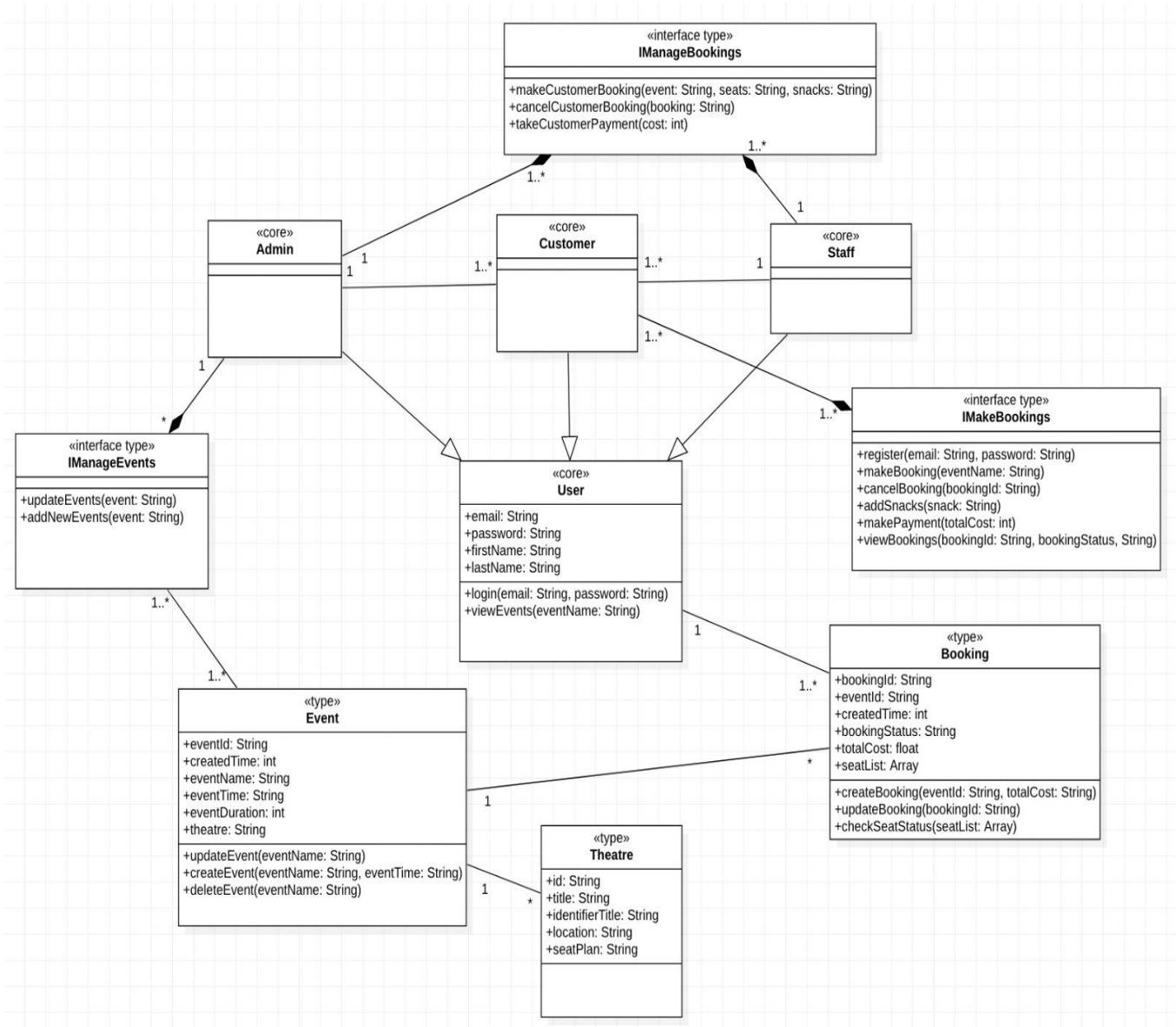


Figure 14 - Business Type Model (Class Diagram)

## 5.6 Wireframing

Within the frontend team, wireframing was used to pass information to other team members of the group of how the whole structure is supposed to look like. Please see appendix for the charts.

## 5.7 Algorithm flowchart

The main objective of the algorithm implemented is to avoid scattering of single seats in the theatre as it helps in not splitting up the small to large group of customers. The algorithm we developed works on the concept of not letting the customer book a single seat if neighbour's neighbour seat is empty on both sides of the seat. Further it can be explained by the example where if the range of seats in a row are 5 and the single customer tries to book a seat in the middle, then system will not allow the customer to go ahead with booking as neighbour's neighbour seats are empty on both sides. This algorithm also gives a concept, where if there are only 10 last seats remaining in the entire hall then it provides the customers to book seats freely without any restrictions.

Below an algorithm flowchart is given, which describes the working of an algorithm with using pseudo code.

### NOTE:

A sparse seat is an individual seat in a row which is identified as an available seat to be booked but its' left neighbour seat and right neighbour seat is already booked.

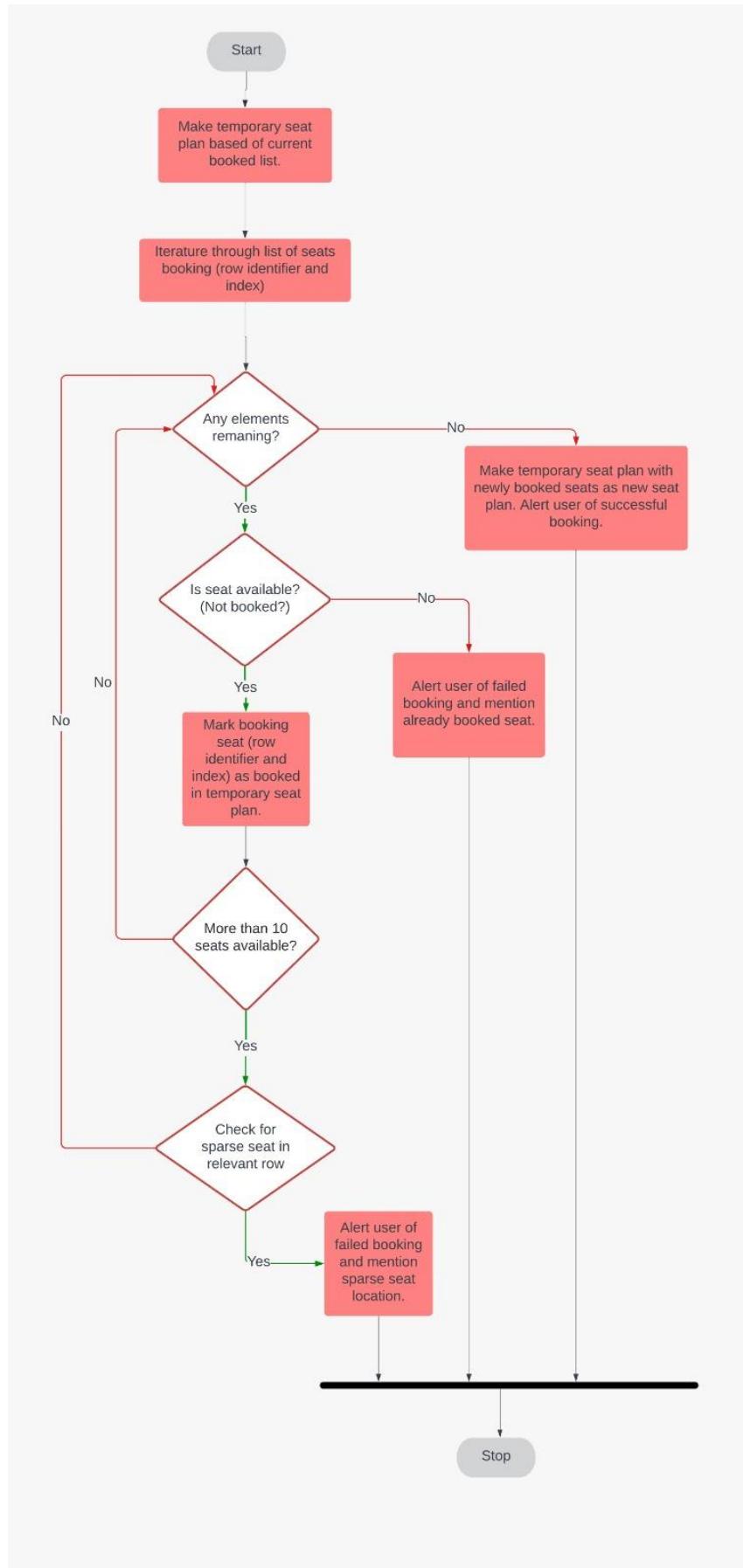


Figure 15 - Algorithm Flowchart

## 6. Implementation

After completing the design phase, we started the implementation phase. In this phase all team members worked on programming the code by following the given requirements and implement the product under plan driven approach. The entire application has been developed in this phase. After team meetings and discussions, we agreed on creating a website and using the following tools and technologies,

Frontend – React hosted on AWS amplify

Backend and API – NodeJS using AWS Lambda and AWS HTTP Gateway

Database - AWS DynamoDB

Our team has split into two to work on front and backend respectively. Front end and backend work was done concurrently by team coordination. In this stage, the algorithm has been produced to support the requirement of minimising scattering of seats in theatre. The code for the algorithm is given below:

```
// Seatplan is an object containing the row and the index of which seat is booked
for (let j = 0; j < seatPlan[row].length; j++) {
    const occurrences = occurrencesOf(3, seatPlan[row].slice(0, j))

    // 3 is identified in seatplan as a white blank spot in the frontend
    // 1 is identified in seatplan as an available seat
    // 0 is identified in seatplan as a booked seat
    if (seatPlan[row][j] === 1 && seatPlan[row][j - 1] === 3 && seatPlan[row][j + 1] === 3) {
        continue
    }
    if (seatPlan[row][j] === 1 && seatPlan[row][j - 1] === 3 && seatPlan[row][j + 1] === 0) {
        return sendSeatSparseErrorMessage({ rowKey: row, index: (j - occurrences) })
    }
    if (seatPlan[row][j] === 1 && seatPlan[row][j - 1] === 0 && seatPlan[row][j + 1] === 3) {
        return sendSeatSparseErrorMessage({ rowKey: row, index: (j - occurrences) })
    }
    if (j === 0 && seatPlan[row][j] === 1 && seatPlan[row][j + 1] === 0) {
        return sendSeatSparseErrorMessage({ rowKey: row, index: (j - occurrences) })
    }
    else if (j === (seatPlan[row].length - 1) && seatPlan[row][j] === 1 && seatPlan[row][j - 1] === 0) {
        return sendSeatSparseErrorMessage({ rowKey: row, index: (j - occurrences) })
    }
    else if (
        seatPlan[row][j] === 1 &&
        (seatPlan[row][j - 1] === 0) &&
        (seatPlan[row][j + 1] === 0)
    ) {
        return sendSeatSparseErrorMessage({ rowKey: row, index: (j - occurrences) })
    }
}
```

The database we used for our application is DynamoDB and below figures shows a brief information of our database and table:



Figure 16 - Database Table- Response Metrics

Items returned (23)													
	Id	createdTime	booking...	customer...	customer...	dataType	email	eventDu...	eventId				
□	USER_0000f03_55ab_4cde_3d2_40e2e0549fd0	1654591962619				User	123@gmail.com						
□	BOOKING_a4372084_c5f4_407c_8591_c5dd7217ed0a	1654592096024	CONFIRMED	USER_808d...	123 123	TicketBooking		60	EVENT_a15c2bf5_977b_4928_abbd_72b91b3afc37				
□	EVENT_a15c2bf5_977b_4928_abbd_72b91b3afc37	1654592096024				EventBooking		60					
□	2	61009				MovieTheater							
□	BOOKING_a4c2b3a1_ec81_f70_0937_a1da841b6ef1	1654592096024	CONFIRMED	USER_a7e4...	asd asd	TicketBooking		60	EVENT_a15c2bf5_977b_4928_abbd_72b91b3afc37				
□	USER_a7e4a546_b139_4738_a8d3_4624d93784c9	1654592120189				User	asd@asd.com						
□	1	58429				MovieTheater							
□	1	74091				User	adminS@movie.com						
□	BOOKING_d0d9d649_23fd_45d7_a538_e0865fb9b7	1654592096024	CONFIRMED	USER_a7e4...	asd asd	TicketBooking		60	EVENT_a15c2bf5_977b_4928_abbd_72b91b3afc37				
□	BOOKING_cac351b1_b66a_4f9e_9971_219288ebf628	1654539810204	CANCELLED	0	AdminM Ad...	TicketBooking		110	EVENT_fa676101_50a8_4239_8372_894099492ddf				
□	BOOKING_4cb48f8e_c0e8_44c1_bbbc_b4ea04eaddfe	1654592096024	CONFIRMED	USER_808d...	123 123	TicketBooking		60	EVENT_a15c2bf5_977b_4928_abbd_72b91b3afc37				
□	BOOKING_12c8d1c5_7925_4675_a958_ca28c208e10d	1654593052394	CONFIRMED	USER_a7e4...	asd asd	TicketBooking		60	EVENT_B243065a_bcd6_4e5c_a7eb_10853aea3dc6				
□	EVENT_db001363_77c9_4297_9bf1_61ad0cf485ae	1654539811222				EventBooking		110					
□	USER_50be0413_8cb8_4e05_8533_c208508e1f70	1654540989288				User	yunus@gmail.com						
□	0	86798				MovieTheater							
□	0	88310				User	adminM@movie.com						
□	EVENT_B243065a_bcd6_4e5c_a7eb_10853aea3dc6	1654593052394				EventBooking		60					
□	BOOKING_42f3cdab_ba0f_4f9e_b47c_af60f77219ac	1654593052394	CONFIRMED	USER_a7e4...	asd asd	TicketBooking		60	EVENT_B243065a_bcd6_4e5c_a7eb_10853aea3dc6				
□	BOOKING_cdd0dc3_40ba_4c08_a018_0ef19ea1fea2	1654592096024	CONFIRMED	USER_808d...	123 123	TicketBooking		60	EVENT_a15c2bf5_977b_4928_abbd_72b91b3afc37				
□	BOOKING_c0d76431_a9b7_42d9_082e_4f7160f7ab5	1654592096024	CONFIRMED	USER_808d...	123 123	TicketBooking		60	EVENT_a15c2bf5_977b_4928_abbd_72b91b3afc37				
□	USER_8aa009a5_5976_a498_93b2_3213c3db6df	1654541067776				User	newuser@gmail.com						
□	USER_c856673f_97e2_4d2d_ab6b_034ca7c2a88e	1654541102390				User	chris@gmail.com						
□	BOOKING_a498d330_a087_4a13_bb6d_3469845781a0	1654592096024	CONFIRMED	USER_a7e4...	asd asd	TicketBooking		60	EVENT_a15c2bf5_977b_4928_abbd_72b91b3afc37				

Figure 17 - Database Table-2

After implementing all the requirements, the application has been developed and ready for the testing phase.

Our application is available on GitHub platform and the link for the repository is:

<https://github.com/YunusOor1/CP70032E-A3-Adv-backend-main>

<https://github.com/YunusOor1/CP70032E-A3-Adv-frontend-client>

<https://github.com/YunusOor1/CP70032E-A3-Adv-frontend-admin>

Preethi-Merline committed 23 hours ago	Fixed and Push Event List Bug	Preethi-Merline committed 23 hours ago	4d6ccc4	<>
Working on Get Event Bug	Preethi-Merline committed 23 hours ago		c95301f	<>
Updated	YunusOor1 authored and Preethi-Merline committed 23 hours ago		9df310e	<>
Fixed Seat Check Algo	YunusOor1 authored and Preethi-Merline committed 23 hours ago		2ba1b6e	<>
Added EventPRice	YunusOor1 authored and Preethi-Merline committed 23 hours ago		6ff7397	<>
B0oking SYstyem Updated	YunusOor1 authored and Preethi-Merline committed 23 hours ago		8998107	<>
Post Man Collection Updated	YunusOor1 authored and Preethi-Merline committed 23 hours ago		e1c61f3	<>
Get Booking List and Booking APIS are done	YunusOor1 authored and Preethi-Merline committed 23 hours ago		17b4e4b	<>
Confirm And Cancel Booking Working fine	YunusOor1 authored and Preethi-Merline committed 23 hours ago		896264b	<>
Updated API Call file structure	YunusOor1 authored and Preethi-Merline committed 23 hours ago		c24c9cd	<>
Get THeater List is done	YunusOor1 authored and Preethi-Merline committed 23 hours ago		d841621	<>
Fixed Set Event Bug	YunusOor1 authored and Preethi-Merline committed 23 hours ago		54a1e70	<>
Updated	YunusOor1 authored and Preethi-Merline committed 23 hours ago		7416336	<>
SeatPlan system is working fine	YunusOor1 authored and Preethi-Merline committed 23 hours ago		fde50d0	<>
Working on Seat Planning System	Preethi-Merline committed 23 hours ago		6ee87d6	<>
> Commits on May 27, 2022				
Collection Added	YunusOor1 committed 12 days ago		8f854b8	<>
Updated	YunusOor1 committed 12 days ago		3c2b014	<>
Get Event List Done	YunusOor1 committed 12 days ago		09c7fb8	<>

Figure 18 - Backend

admin cancel disappear booking fix	<a href="#">diff</a>	3527b27	<a href="#">diff</a>
dropcmd authored and Hymzaa committed 31 seconds ago			
checking if <24h until event	<a href="#">diff</a>	10474a2	<a href="#">diff</a>
Hymzaa committed 35 seconds ago			
impl. of confirm and cancel booking	<a href="#">diff</a>	ad4ab63	<a href="#">diff</a>
Hymzaa committed 43 seconds ago			
user level auth	<a href="#">diff</a>	0b380b0	<a href="#">diff</a>
dropcmd authored and Hymzaa committed 1 minute ago			
events list filter	<a href="#">diff</a>	3b25c07	<a href="#">diff</a>
Hymzaa committed 1 minute ago			
basic dashboard	<a href="#">diff</a>	d562214	<a href="#">diff</a>
dropcmd authored and Hymzaa committed 1 minute ago			
booking search done*	<a href="#">diff</a>	a8d72f5	<a href="#">diff</a>
dropcmd authored and Hymzaa committed 1 minute ago			
event create, edit, delete Test	<a href="#">diff</a>	03249ef	<a href="#">diff</a>
Hymzaa committed 1 minute ago			
auth/login* ...	<a href="#">diff</a>	ddbd1b8	<a href="#">diff</a>
dropcmd authored and Hymzaa committed 1 minute ago			
version showcased during meeting	<a href="#">diff</a>	e0ff3b4	<a href="#">diff</a>
dropcmd authored and Hymzaa committed 1 minute ago			
header/wrapper rework	<a href="#">diff</a>	7ee20f3	<a href="#">diff</a>
Hymzaa committed 2 minutes ago			
create event process	<a href="#">diff</a>	cf2692a	<a href="#">diff</a>
Hymzaa committed 2 minutes ago			
-o- Commits on May 21, 2022			
create event venue details	<a href="#">diff</a>	8581605	<a href="#">diff</a>
dropcmd committed 18 days ago			
header, ... ...	<a href="#">diff</a>	5aec096	<a href="#">diff</a>
dropcmd committed 18 days ago			
login	<a href="#">diff</a>	aa23d6	<a href="#">diff</a>
dropcmd committed 18 days ago			
register form	<a href="#">diff</a>	0443b2e	<a href="#">diff</a>
dropcmd committed 18 days ago			
-o- Commits on May 20, 2022			
basic auth flow	<a href="#">diff</a>	2d6dc23	<a href="#">diff</a>
dropcmd committed 19 days ago			

Figure 19 - Frontend

checkout and prices	<a href="#">7b04fd9</a>	< >
list -> book nav and prop pass ...	<a href="#">449e0f6</a>	< >
theatre 2 color fix	<a href="#">87616fa</a>	< >
theatre 3	<a href="#">aa0d518</a>	< >
curve seat row ability + theatre 2	<a href="#">3ed7fea</a>	< >
seperation of concerns	<a href="#">71640da</a>	< >
basic seat layout	<a href="#">edebbb5</a>	< >
custom logo added	<a href="#">0ef539a</a>	< >
-o- Commits on May 22, 2022		
quick fix 🎟	<a href="#">653254f</a>	< >
home page impl.	<a href="#">dc5c36d</a>	< >
footer	<a href="#">d9539bf</a>	< >
header adjustments	<a href="#">5a5c1f1</a>	< >
login register impl.	<a href="#">57e514e</a>	< >
base	<a href="#">5bb4458</a>	< >

Figure 20 - Frontend

There are two portals in our application, customer portal, staff, and admin portal. The web application is available on links,

Admin/Staff: <https://master.d3r9ted7jxv5qy.amplifyapp.com>

Customer: <https://master.d2g1ov2tfcuju4.amplifyapp.com/>

The login details for staff are [adminS@movie.com](mailto:adminS@movie.com), 123123

The login details for admin are [adminM@movie.com](mailto:adminM@movie.com), 123123

The output screens of our application are given below:

## UPGRADE YOUR SUNDAYS

Enjoy wide range of shows with venues all over London.

[VIEW EVENTS](#)

Discover the experience



Figure 21 - Customer Homepage

## REGISTER

[Already have an account?](#)

First name \*

Last name \*

Email \*

Password \*

[REGISTER](#)

Figure 22 - Customer Registration

## SIGN IN

[Not a member yet? Register here](#)

Email \*

Required

Password \*

**SIGN IN**

Figure 23 - Customer Login

## EVENTS LIST

Select Venue

- All
- The Supermax, Ealing Broadway
- Regent Feature Cinema, Regent Street
- Ultimate 3D, Shepherd's Bush

<b>Test-Event-1</b>  <b>Wed Jun 08 2022, 09:53:00</b> 215 of 231 seats available. The Supermax, Ealing Broadway	<b>Homecoming</b>  <b>Thu Jun 09 2022, 15:30:00</b> 59 of 59 seats available. Ultimate 3D, Shepherd's Bush
<b>Test-2-Event</b>  <b>Wed Jun 15 2022, 10:09:00</b> 219 of 231 seats available. The Supermax, Ealing Broadway	

Figure 24 - View Events for Customers

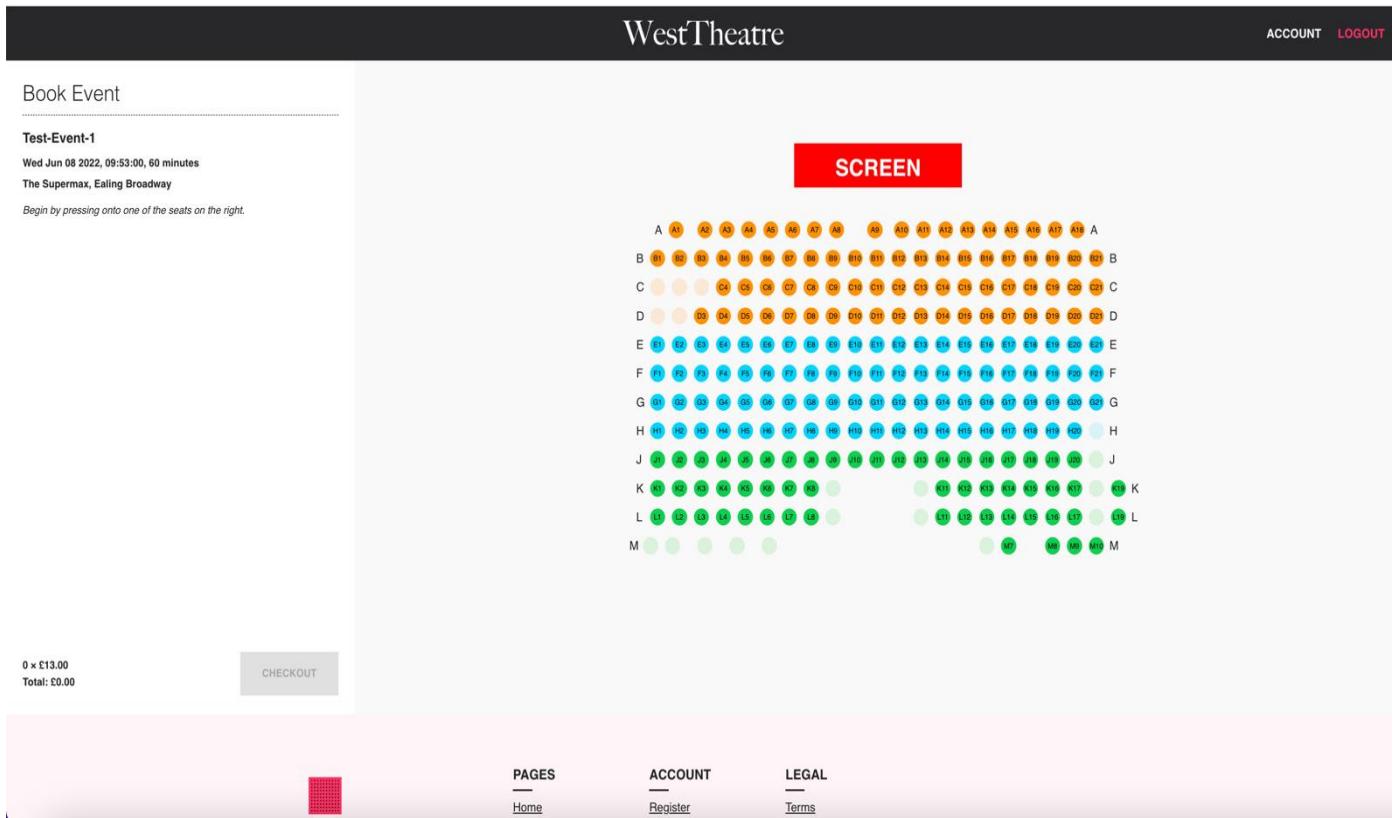


Figure 25 - Customer Booking an Event

The screenshot shows the checkout screen for the same event. It displays two selected seats, M1 and M2, with "AMEND" buttons. The "Seat Subtotal: £26.00" is shown. On the right, there are sections for "Snacks" (Soft Drink, Popcorn, Chocolate) and "Payment" (Card Number, Valid until, CVV, Card Holder). The "Snack Subtotal: £0.00" and a "PURCHASE - £26.00" button are also present. At the bottom, there are links for PAGES (Home, Events List), ACCOUNT (Register, Login), and LEGAL (Terms, Privacy).

Figure 26 - Customer Checkout Screen

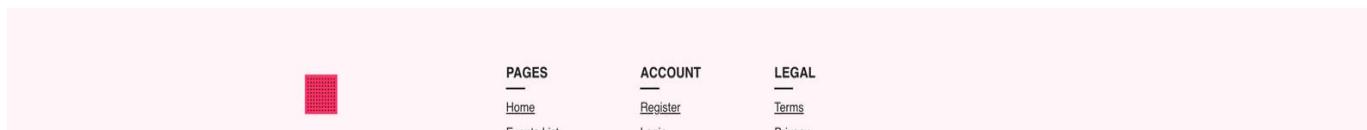


Figure 27 - Customer Successful Booking



Figure 28 - Customer Account Screen with Cancel Option



## Login

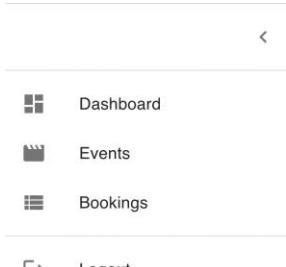
Email Address \*

Password \*

LOGIN

WestTheatre Copyright © 2022

Figure 29 - Staff and Admin Login



WestTheatre Dashboard

Welcome, **David Paul**

EVENT LIST    BOOKING LIST

WestTheatre Copyright © 2022

Figure 30 - Staff Dashboard

The screenshot shows a staff interface for managing events. On the left is a sidebar with icons for Dashboard, Events, Bookings, and Logout. The main area is titled 'Events' and displays 'Upcoming events, most recent first.' A search bar labeled 'Filter Venue' has a 'CLEAR' button. Three events are listed:

- Test-Event-1**: 60 minutes, The Supermax, Ealing Broadway, Wed Jun 08 2022 09:53:00 GMT+0100 (British Summer Time). Status: 215 of 231 Available, 16 Booked. Buttons: BOOK (purple), EDIT, DELETE.
- Spider-Man: Homecoming**: 110 minutes, Ultimate 3D, Shepherd's Bush, Thu Jun 09 2022 15:30:00 GMT+0100 (British Summer Time). Status: 59 of 59 Available, 0 Booked. Buttons: BOOK (purple), EDIT, DELETE.
- Test-2-Event**: 60 minutes, The Supermax, Ealing Broadway, Wed Jun 15 2022 10:09:00 GMT+0100 (British Summer Time). Status: 219 of 231 Available, 12 Booked. Buttons: BOOK (purple), EDIT, DELETE.

Figure 31 - Staff Event List

In above figure, view events for staff where the create button is greyed out to restrict them to add any events as it is admins responsibility.

The screenshot shows a staff interface for booking a client. On the left is a sidebar with icons for Dashboard, Events, Bookings, and Logout. The main area is titled 'Book Client' and shows a 'Event Confirmation' step for 'Test-Event-1'. The details are as follows:

- Event: Test-Event-1
- Date: Wed Jun 08 2022
- Time: 9:53 am
- Duration: For 60 minutes
- Location: The Supermax, Ealing Broadway
- Ticket Price: £13.00

A 'NEXT' button is at the bottom right. At the very bottom, a footer note says 'WestTheatre Copyright © 2022'.

Figure 32 - Staff Booking for Customer

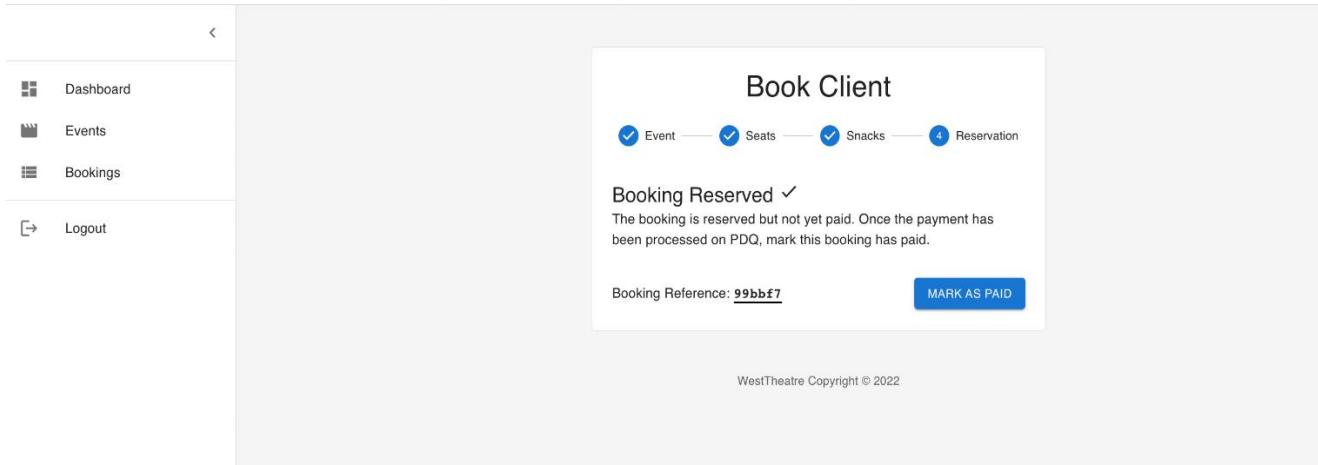


Figure 33 - Staff Booking Confirmation

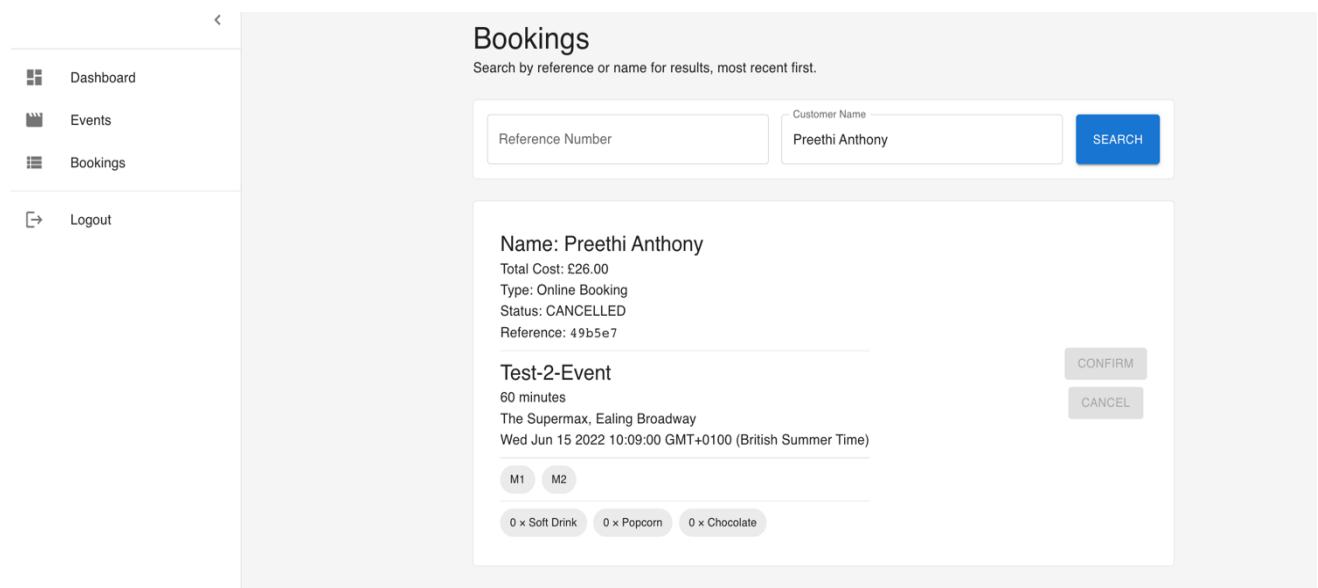


Figure 34 - Staff Reference Search and Cancel Ticket Screen

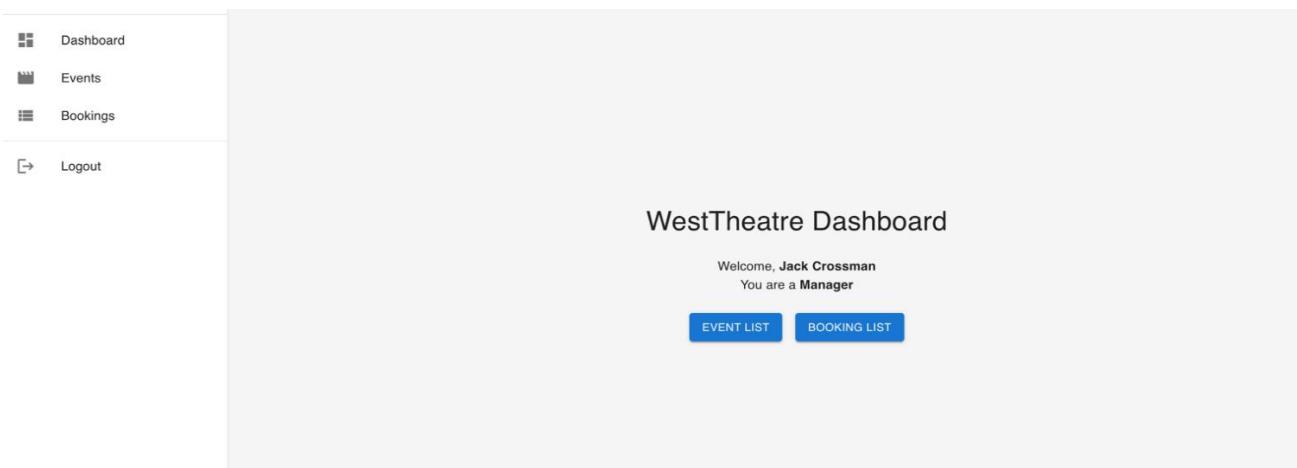


Figure 35 - Admin Dashboard

The screenshot shows the "Create Event" screen. The sidebar on the left is identical to Figure 35. The main form has three steps: 1. Event, 2. Venues, and 3. Confirm. Step 1 is active. It contains fields for "Event Name" (with placeholder "Event Name \*"), "Date" (set to June 7 2022), "Time" (set to 12:14 PM), "£ 13" (Ticket Price), "60 minutes (runtime)", and a "NEXT" button.

Figure 36 - Admin Create Event Screen

**Events**  
Upcoming events, most recent first.

Filter Venue  [CLEAR](#)

**Test-Event-1**  
60 minutes  
The Supermax, Ealing Broadway  
Wed Jun 08 2022 09:53:00 GMT+0100 (British Summer Time)  
214 of 231 Available, 17 Booked

[BOOK](#) [EDIT](#) [DELETE](#)

**Spider-Man: Homecoming**  
110 minutes  
Ultimate 3D, Shepherd's Bush  
Thu Jun 09 2022 15:30:00 GMT+0100 (British Summer Time)  
59 of 59 Available, 0 Booked

[BOOK](#) [EDIT](#) [DELETE](#)

**Test-2-Event**  
60 minutes  
The Supermax, Ealing Broadway  
Wed Jun 15 2022 10:09:00 GMT+0100 (British Summer Time)  
219 of 231 Available, 12 Booked

[BOOK](#) [EDIT](#) [DELETE](#)

Figure 37 - Admin Modify or Delete Events

## 7. Testing Procedures

Testing is a most important phase in software development life cycle which helps in ensuring that the software is working as per the requirements, ensure great user experience and with no errors. Test cases or scenarios as known as user stories are created in this phase from taking requirements and modelling details from previous stages. Then, these test cases are implemented to test the system. For this application, we used various testing methods which are behaviour driven development, test driven development and manual testing of algorithm through created website. Below, these testing methods are discussed.

### 7.1 Behaviour Driven Development

Behaviour Driven Development or BDD has same concept of test-driven development (TDD) which is test first development. However, in BDD the test is more focused on behaviour of the system and is an extension of TDD. In BDD, we created major features and different scenarios to test the behaviour of the system. In, BDD we write the failing feature test, and it is written in simple language. The below figure shows how BDD and TDD work together.

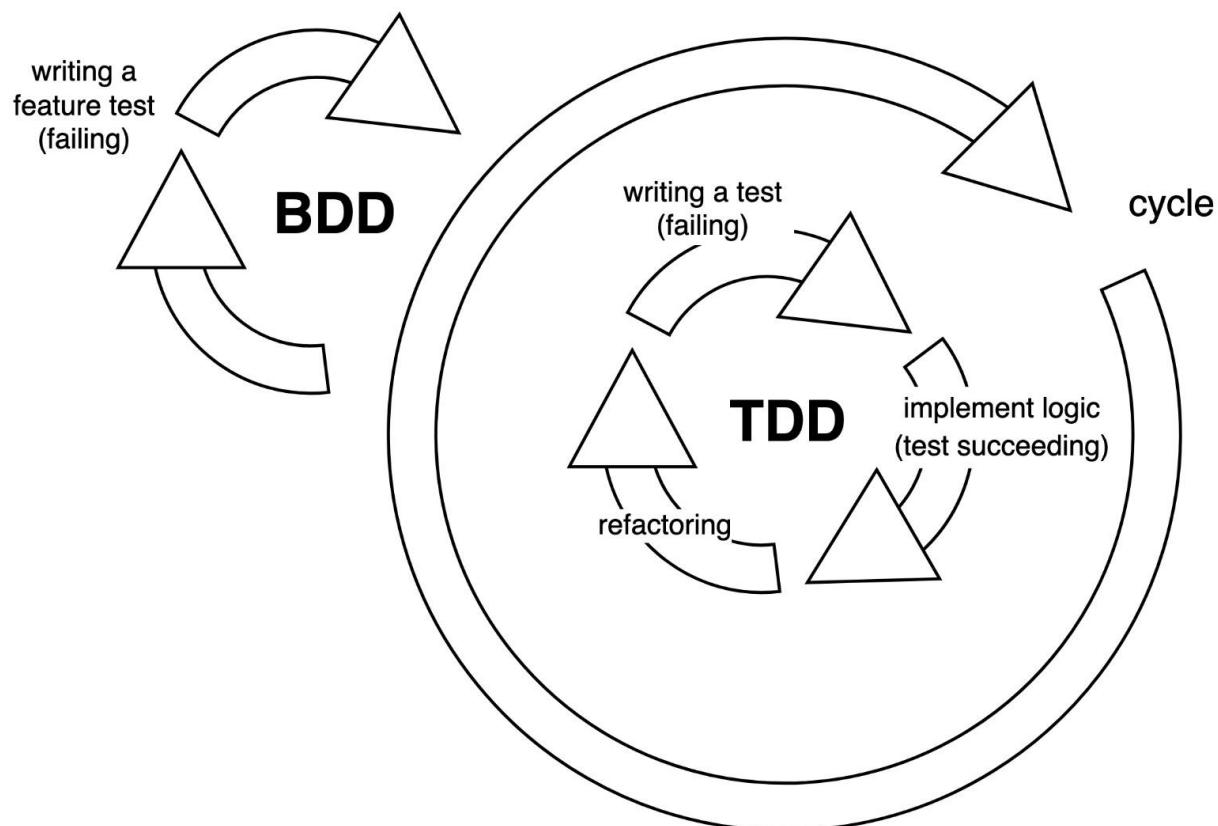


Figure 38 - BDD and TDD

Below few BDD scenarios of our application are written:

#### Feature: User Authentication

- **Scenario:** Check the registration form
  - **Given** I am not an existing user
  - **When** I open registration page
  - **And** give correct details
  - **Then** I am registered into system
- 
- **Scenario:** Check the login form
  - **Given** I am existing user of the system
  - **When** I open login page
  - **And** give correct username and password
  - **Then** I am logged into system

See figure 39 for TDD testing based on User Authentication Scenarios.

#### Feature: Booking

- **Scenario:** Booking a ticket of an event
  - **Given** I am a potential buyer
  - **When** I view events listed
  - **And** I am existing user
  - **And** select preference, make payment
  - **Then** I get booking reference
- 
- **Scenario:** Cancelling ticket of an event
  - **Given** I am existing user
  - **When** I have booking reference
  - **And** I request cancelling in system
  - **And** it is more than 24 hours for the event
  - **Then** my ticket is cancelled

See figure 40 for TDD testing based on Booking Scenarios.

## Feature: Events

- **Scenario:** Adding new events
  - **Given** I am the admin
  - **When** I login to system
  - **And** I add new event to the list
  - **Then** event is added in the system
  
- **Scenario:** Modifying events
  - **Given** I am the admin
  - **When** I login to system
  - **And** I make changes to existing event list
  - **Then** event list is modified

See figure 41 for TDD testing based on Events Scenarios.

## Feature: Staff

- **Scenario:** Booking a ticket for the customer
  - **Given** I am the staff
  - **When** I get booking request from customer
  - **And** I take customer preferences, payment
  - **Then** customers ticket is booked
  
- **Scenario:** Cancelling a ticket for the customer
  - **Given** I am the staff
  - **When** I get cancelling request from customer
  - **And** I take booking reference
  - **And** it is more than 24 hours for the event
  - **Then** customers ticket is cancelled

## 7.2 Test Driven Development

Test driven development or TDD mainly focuses on creating unit test cases and test the functionality of the system before implementation. TDD is beneficial as it helps in faster development and clean code. In this application, we have used TDD for the user stories which are taken from requirements stage and BDD scenarios and implemented the test code. The below figures show the proof of TDD usage in our application,

```
> jest "user_auth"

PASS ./user_auth.test.js
  ✓ LOGIN USER DATA (404 ms)
  ✓ REGISTER USER DATA (266 ms)

Test Suites: 1 passed, 1 total
Tests:       2 passed, 2 total
Snapshots:   0 total
Time:        1.345 s, estimated 2 s
Ran all test suites matching /user_auth/i.
```

Figure 39 - TDD 1

```
> jest "booking"

PASS ./booking.test.js
  ✓ CHECK BOOKING SEAT (393 ms)
  ✓ SET TICKET BOOKING (826 ms)
  ✓ UPDATE TICKET BOOKING (809 ms)
  ✓ GET BOOKING LIST (487 ms)

Test Suites: 1 passed, 1 total
Tests:       4 passed, 4 total
Snapshots:   0 total
Time:        3.182 s, estimated 4 s
Ran all test suites matching /booking/i.
```

Figure 40 - TDD 2

```
> jest "event"

PASS ./event.test.js
  ✓ SET EVENT (425 ms)
  ✓ UPDATE EVENT (275 ms)
  ✓ GET EVENT LIST (358 ms)
  ✓ DELETE EVENT (271 ms)
  ✓ GET THEATER LIST (272 ms)

Test Suites: 1 passed, 1 total
Tests:       5 passed, 5 total
Snapshots:   0 total
Time:        2.295 s, estimated 3 s
Ran all test suites matching /event/i.
```

Figure 41 - TDD 3

We have done unit testing on the algorithm to check if the requirement of algorithm is met. The algorithm test code is:

```
let bookedSeat = [
  {
    "rowKey": "G",
    "index": 1
  },
  {
    "rowKey": "H",
    "index": 2
  },
  {
    "rowKey": "H",
    "index": 3
  },
  {
    "rowKey": "G",
    "index": 3
  },
  {
    "rowKey": "G",
    "index": 2
  },
  {
    "rowKey": "H",
    "index": 8
  },
  {
    "rowKey": "H",
    "index": 10
  },
]

const requestBody = {
  "eventId": "EVENT_cd01fca2_2b2b_474f_a0fd_64c4ef9b9810",
  "createdTime": 1654434376201,
  "bookedSeat": bookedSeat
}

const checkSeatBookingResponse = await checkSeatBooking(requestBody);
expect(checkSeatBookingResponse).toBeTruthy();
```

The manual testing results of the algorithm through website are given below.

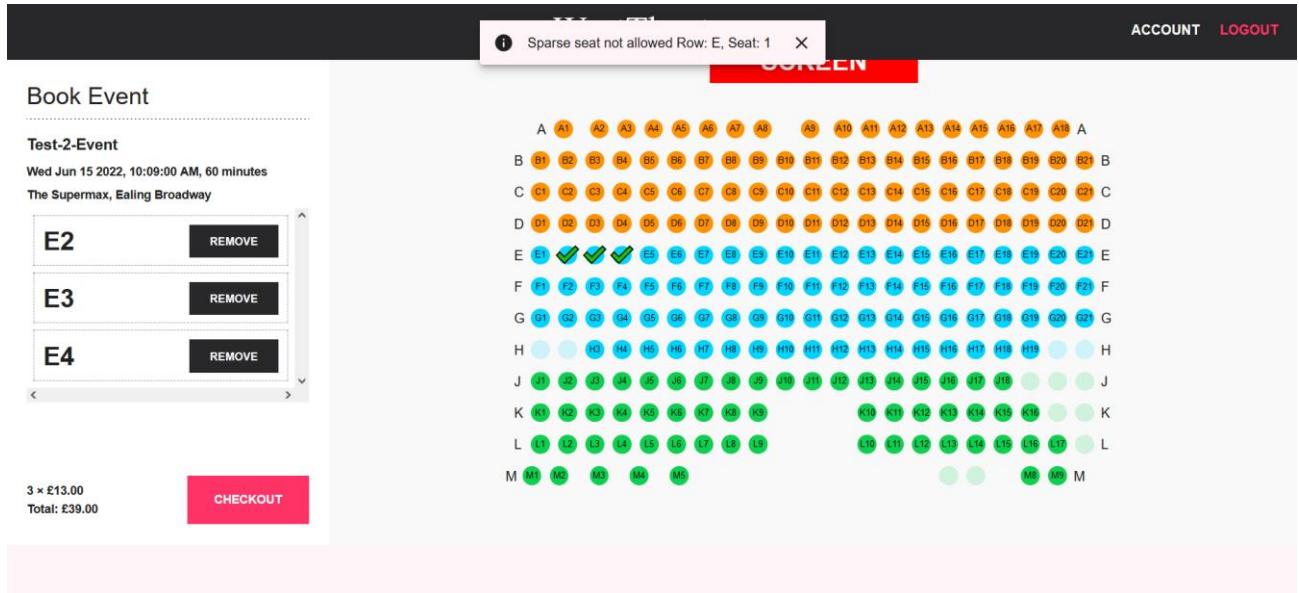


Figure 42 - Manual Testing 1

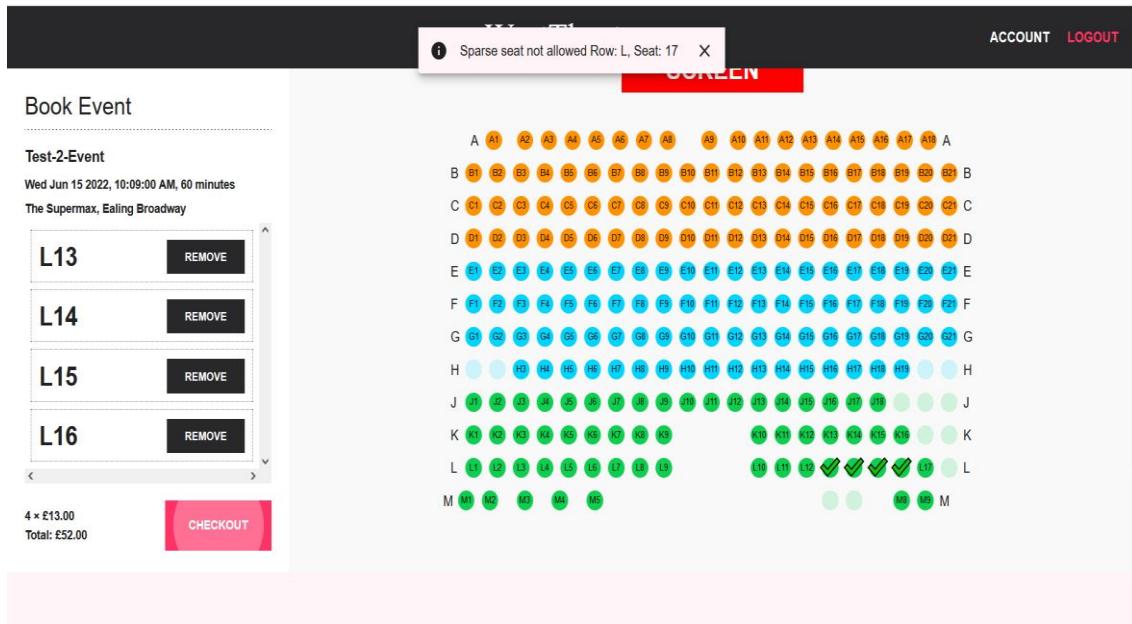


Figure 43 - Manual Testing 2

ACCOUNT [LOGOUT](#)

**SCREEN**

### Book Event

**Test-2-Event**  
Wed Jun 15 2022, 10:09:00 AM, 60 minutes  
The Supermax, Ealing Broadway

B2	<a href="#">REMOVE</a>
B3	<a href="#">REMOVE</a>
B4	<a href="#">REMOVE</a>
B5	<a href="#">REMOVE</a>

4 x £13.00  
Total: £52.00

[CHECKOUT](#)

Figure 44 - Manual Testing 3

## 8. Critical discussion – Conclusion

Since day one of the part 3 assignment we set up a group meeting with all members to establish our plan and work distribution to make sure we deliver the best possible outcome. Upon meeting, we had to decide who is going to be in charge of which division and set a plan towards that.

We decided that two people would be in charge of the backend development and three in charge of the front end development, based on previous projects, skills and experience.

Based on the plan, deadlines were set and meetings were decided every week to go through with our work and share it amongst us to make sure that everyone is at the same point and no one is left behind. Meetings ideally were supposed to be in person but due to various factors, could be done online as well, as that was the only way to secure that everything is done according to our plan.

Working on our project, came with a variety of issues that were not anticipated from the beginning but with all team's effort, we reduced the issues to a bare minimum. The biggest problem that surfaced was that everyone had a different schedule or availability due to full time/part time jobs outside of university, which is something that might have been not be the case if we had all been employed and assigned specifically this task to implement as a team. On top of this, as everyone in the group has a different background in coding and programming, there was a knowledge gap that we had to cover between us, making the task harder to complete as we had to analyse and learn everything from the beginning in order to implement it. Which although it was more time consuming, it was an opportunity to learn at the same time and it made the whole project a source of knowledge and learning. Apart from these two, we faced couple of issues with our github repositories as well resulting us to rebase.

Taking issues aside, all group delivered what was expected from them and the effort to complete the project was up to standards. All ideas that were initially put on table were implemented and made sure that they are functioning the way they were supposed to. And although, we had one month to deliver a complete project, all deadlines with some small delays were met and our work was delivered. However there is always room for improvement.

In our initial discussion the group decided to implement two different version of the website, one facing the front of the business which handles customer bookings, and another that was used by administrations (staff and managers). The focus was on separation of concerns but in hindsight, we had to reuse some code elements like venue stage where the user has a visible seat layout to select seats from, which made making updates required in two different places. Apart from making one website which handled both customer and administrator actions, an alternative would include creating a monorepo. The goal of monorepos is to have an external set of visual elements and business logic which could be used in multiple areas, hosted on version control platforms like GitHub (Fernandez, 2021). Within our project, the application of such a concept could be applied to reusing the visual venue stage for seat selection as both implementations use the same code.

Aside from what we could have done differently, we were exposed to the benefits of working in a group project. The main one being that we were all exposed to a variety of perspectives and with a combination of different personalities gave us the opportunity to learn how to listen, interact and leverage the talent of our peers in order to achieve the best overall result for our project.

## References

- Chitrasingla, 2021. *geeksforgeeks.org*. [Online]  
Available at: <https://www.geeksforgeeks.org/functional-vs-non-functional-requirements/#:~:text=A%20functional%20requirement%20defines%20a,system%20fulfill%20the%20functional%20requirements%3F%E2%80%9D>  
[Accessed June 2022].
- Fernandez, T., 2021. *semaphoreci.com*. [Online]  
Available at: <https://semaphoreci.com/blog/what-is-monorepo>  
[Accessed June 2022].
- JavaTPoint, 2021. *javatpoint.com*. [Online]  
Available at: <https://www.javatpoint.com/software-engineering-requirement-engineering>  
[Accessed June 2022].
- Lucidchart, 2021. *Lucidchart.com*. [Online]  
Available at: <https://www.lucidchart.com/pages/data-flow-diagram>  
[Accessed June 2022].
- Satyabrata, J., 2021. *Geeks for Geeks*. [Online]  
Available at: <https://www.geeksforgeeks.org/overview-of-plan-driven-development-pdd/>  
[Accessed June 2022].
- Tallyfly, 2021. *Tallyfly*. [Online]  
Available at: <https://tallyfy.com/uml-diagram/>  
[Accessed June 2022].
- The Manager's Guide to Mixing Agile and Waterfall, n.d. *Workfront*. [Online]  
Available at: <https://www.workfront.com/en-gb/project-management/methodologies/waterfall#:~:text=The%20Waterfall%20methodology%E2%80%94also%20known,before%20the%20next%20phase%20begins>  
[Accessed June 2022].
- Tutorial Space, 2021. *tutorialsspace.com*. [Online]  
Available at: <http://www.tutorialsspace.com/Software-Engineering/SE-2nd-Unit/05-System-Modelling.aspx>  
[Accessed June 2022].

## Appendix

The screenshot shows a web browser window for the WestTheatre website. The header includes a 'Page 1' button, a search bar with the URL 'https://www.Assignment3.cain', and a dark navigation bar with 'WestTheatre', 'Account', and 'Logout' buttons.

The main content area is titled 'ACCOUNT' and contains a 'BOOKINGS' section. A large rectangular dialog box is centered, containing the following information:

- Test-1**
- Date, Time**
- Location**
- A series of small, illegible icons or text entries.
- A 'Cancel' button in the top right corner.
- A note below the buttons: "Cannot cancel less than 24 hours before viewing."

At the bottom left of the page, there is a summary: "0 x £13.00" and "Total: £0.00", followed by a "Check Out" button.

At the bottom right, there are three small rectangular boxes labeled "Button 4" and "Button 3".

At the very bottom left, there is a small square icon with an 'X' inside it. At the bottom center, there is a copyright notice: "© WestTheatre 2022".

Figure 45 - Account Page

Page 1

https://www.Assignment3.cain

WestTheatre

Account Logout

## Book Event

**Test-1**

**Date, Time**

**Location**

...

SCREEN

Door                                  Door

0 x £13.00  
Total: £0.00

Check Out

© WestTheatre 2022

Button 4  
Button 3

Button 4  
Button 3

Button 4  
Button 3

Figure 46 - Book Event

Page 1

https://www.Assignment3.cain

WestTheatre

Account Logout

## CHECKOUT

Test-1

Date, Time

Location

Snacks

Snack Subtotal

Card Number

Valid Until

CVV

Card Holder

Purchase

0 x £13.00  
Total: £0.00

Check Out

Button 4  
Button 3

Button 4  
Button 3

Button 4  
Button 3

© WestTheatre 2022

Figure 47 - Checkout

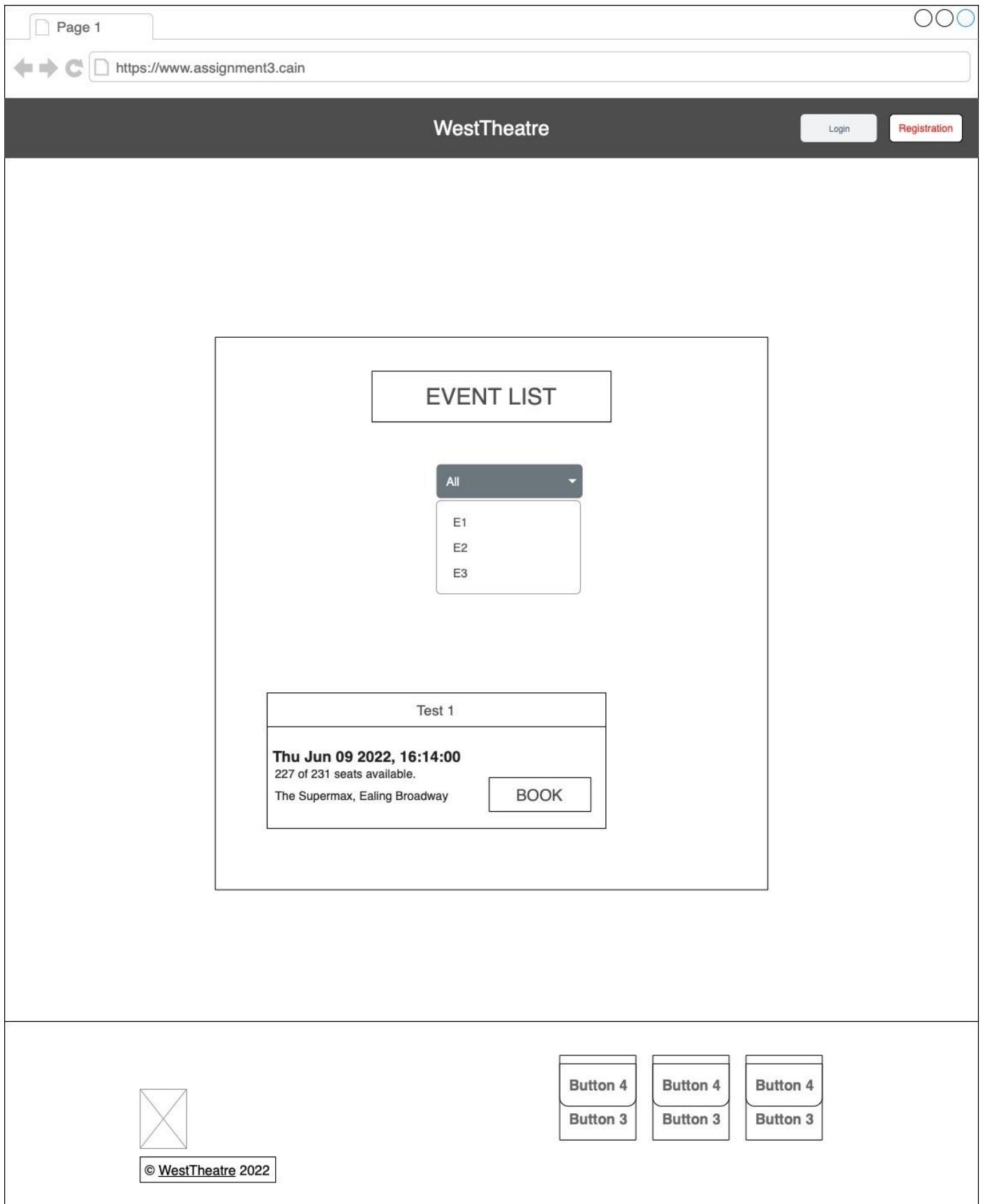


Figure 48 - Event List

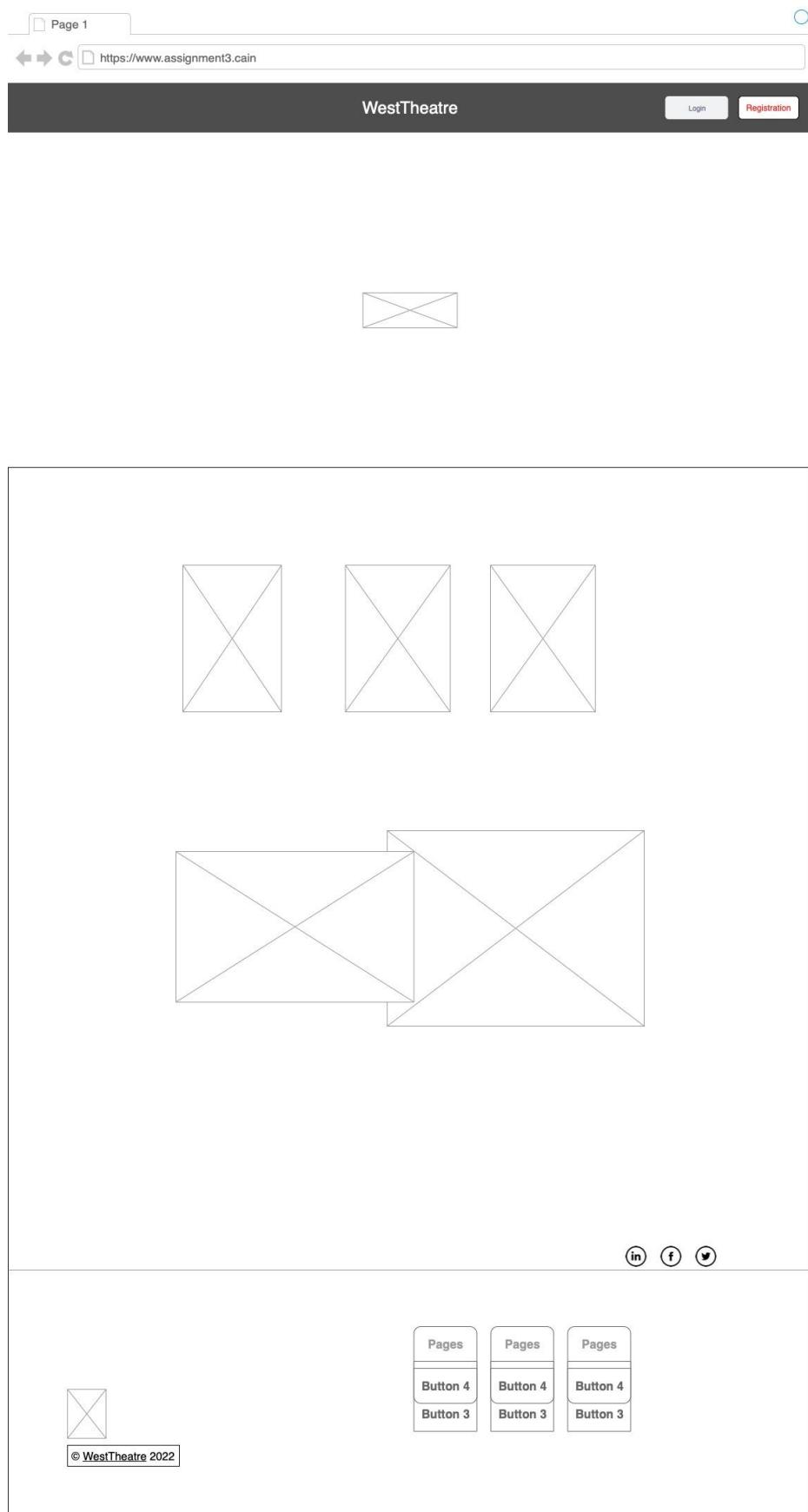


Figure 49 - Homepage

The diagram illustrates a wireframe representation of a web browser interface. At the top, there is a header bar with a back/forward button, a refresh/circular arrow button, and a search/address bar containing the URL "https://www.assignment3.cain". Below the header is a dark navigation bar featuring the "WestTheatre" logo in white text. To the right of the logo are two buttons: "Login" and "Registration", with "Registration" highlighted in red. The main content area is a large rectangular box labeled "REGISTER" in bold capital letters at the top center. Below this, there is a small link "Already a member? Sign in". The registration form consists of several input fields: "First Name" and "Last Name" (both in grey placeholder text), "Email address" (in grey placeholder text), and "Password" (in grey placeholder text). At the bottom of the form is a "Sign in" button. In the bottom left corner of the main content area, there is a small square icon with an 'X' inside. The bottom navigation bar contains three identical rectangular buttons, each labeled "Button 4" on top and "Button 3" on the bottom.

Figure 50 - Register

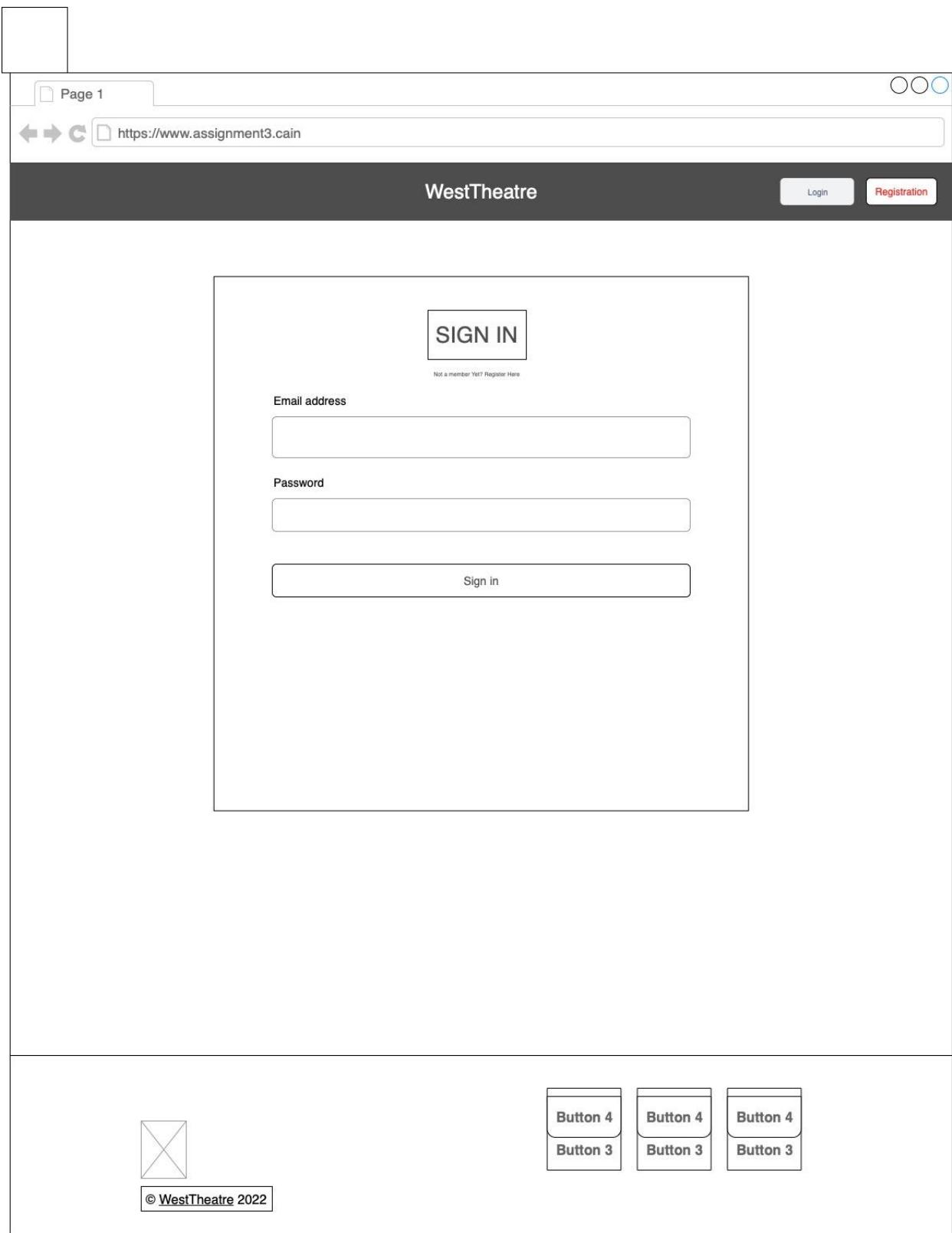


Figure 51 - Sign In