Software Requirements Elicitation and Modelling of Movie Ticket Booking System

Four Week Industrial Training Report on Software Development (September 19, 2022- October 14, 2022)

Submitted by

Sandeep Singh, MCA-II Year
Department of Computer Science and Engineering
Mahatma Jyotiba Phule Rohilkhand University Bareilly
A State University, Government of Uttar Pradesh, India

Division of Communications and Information Technology
Indraprastha Institute of Information Sciences Delhi
(A Unit of Indraprastha Institute of Information Sciences Private Limited, New Delhi, India)

October 2022

Table of Contents

List of Figures

Chapter 1:

Introduction

This project is aimed at developing a ticket reservation system for Cinema Halls. The Ticket Reservation System is an Internet based application that can be accesses throughout the Net and can be accessed by anyone who has a net connection. This application will automate the reservation of tickets and Enquiries about availability of the tickets. This application includes email confirmation for the tickets.

In this chapter we will discuss about four main key points which helps to easy to understand this project or this system.

1.1 Software Requirements Elicitation:-

Requirements Elicitation is the process to find out the requirements for an intended software system by communicating with client, end users, system users and others who have a stake in the software system development.

1.2 Software Requirements Modelling:-

Requirements modelling in software engineering is essentially the planning stage of a software application or system. Generally, the process will begin when a business or an entity (for example, an educational institution) approaches a software development team to create an application or system from scratch or update an existing one. Requirements modelling comprises several stages, or '**patterns**': scenario-based modelling, data modelling, flow-oriented modelling, class-based modelling and behavioural modelling. Each of these stages/patterns examines the same problem from a different perspective.

1.3 Motivation:-

According to the KAOS Report maximum software project was failed (approximate 92%) because they do not focus to gathering the requirements in a proper way. So it is very essential to gathering every single information very carefully. If we find the all possible requirements with the help of various stack holders like customer, market demand, user friendly environment and so on then it will definitely increase the chance to develop a beautiful software.

1.4 Contribution:-

The main purpose of this article to cover all the parties that are involves in the process of developing a software project . In this we identify all the possible members and resources that take parts directly or indirectly in a project.

1.5 Report Outline:-

The software development report deals with various situations like selecting best requirements that are good for project and able to full fill the need of customer. Our aim is to select best ever requirements.

Chapter 2:

Elicitation and Modelling of Movie Ticket Booking System

2.1 Introduction:-

In this chapter we discuss how to gathering the requirements and how to model them in different ways. Elicitation of requirements is one of the different work that is performed by well trained developers. In this process the developer communicate with customer to identify his need to the software and then developer discuss to his team members at last developer communicate with random persons to clear what is new requirements to be added in the project. In the movie ticket system main aim of the developer to reveal all the obstacle to a person face while booking a ticket on the particular software. This is observing now days that user friendly software capture the market instead of others.

2.2 Elicitation of Movie Ticket Booking System Using Traditional Method:-

Gathering the requirements is the first process to built a good software. With the help of traditional method we identified some basic requirement that are obtained by communicating with various stakeholders like customer, team members and individual persons of the circle. Maximum requirements are given by a customer and to deal with other requirements communication is compulsory. In our module (Movie ticket booking system) there are some requirements that we collect are given below-

(i) Functional requirements
(ii) Non Functional requirements
Now discuss each requirement in details. Functional Requirements can be classified as-

(i) Functional requirements:-

a) **Registration –**
If a customer wants to book the ticket then he/she must be registered, an unregistered user can't book the ticket.

b) **Login-**
Customer logins to the system by entering valid user id and password for booking the ticket.

c) **Search Movie-**
The system shall have a search function. Customer or visitor can search movies based on movie name, date, time and venue

d) **Seat Viewing-**
The customer shall be shown a 2D image of the seats from which the desired seats are selected.

e) **Ticket Canceling-**
The customer shall be given an option to cancel ticket one hour before the movie with some fine.

f) **Payment-**
For the customer there are many types of secure billing will be prepaid by debit or credit card. The security will provide by the third party like Pay-Pal etc.

g) **Logout-**
After the payment or browse the movie, the customer will log out.

h) **Generate Ticket-**
After booking, the system can generate the portable document file (.pdf) and then sent one copy to the customer's Email-address and another one as an SMS to customer's phone.

i) **Add movies-**
The system shall have a feature for admin to add movies and their details.

j) **Remove Movies-**
The system shall have a feature for admin to remove movies.

(ii) Non-functional requirement:-

a) **Security –** The system uses SSL (secured socket layer) in all transactions that include any confidential customer information.
The system must automatically log out all customers after a period of inactivity. The system should not leave any cookies on the customer's computer containing the user's password. The system's back-end servers shall only be accessible to authenticated administrators. Sensitive data will be encrypted before being sent over insecure connections like the internet.

b) **Reliability –** The system provides storage of all databases on redundant computers with automatic switchover.

The reliability of the overall program depends on the reliability of the separate components. The main pillar of the reliability of the system is the backup of the database which is continuously maintained and updated to reflect the most recent changes.

Thus the overall stability of the system depends on the stability of container and its underlying operating system.

c) **Availability –** The system should be available at all times, meaning the user can access it using a web browser, only restricted by the downtime of the server on which the system runs. In case of an of a hardware failure or database corruption, a replacement page will be shown. Also in case of a hardware failure or database corruption, backups of the database should be retrieved from the server and saved by the administrator. Then the service will be restarted. It means 24 X 7 availability.

d) **Maintainability –** A commercial database is used for maintaining the database and the application server takes care of the site. In case of a failure, a re-initialization of the program will be done. Also, the software design is being done with modularity in mind so that maintainability can be done efficiently.

e) **Portability –** The application is HTML and scripting language based. So The end-user part is fully portable and any system using any web browser should be able to use the features of the system, including any hardware platform that is available or will be available in the future.

   An end-user is using this system on any OS; either it is Windows or Linux.
   The system shall run on PC, Laptops, and PDA etc.

f) **Accessibility –** The system will be a web-based application it is going to be accessible on the web browser.

g) **Back up –** We will take a backup in our system database. In order to enable the administrator and the user to access the data from our system!

h) **Performance –** The product shall be based on web and has to be run from a web server.

   The product shall take initial load time depending on internet connection strength which also depends on the media from which the product is run.

   The performance shall depend upon hardware components of the client/customer

i) **Accessibility –** The system shall provide handicap access.

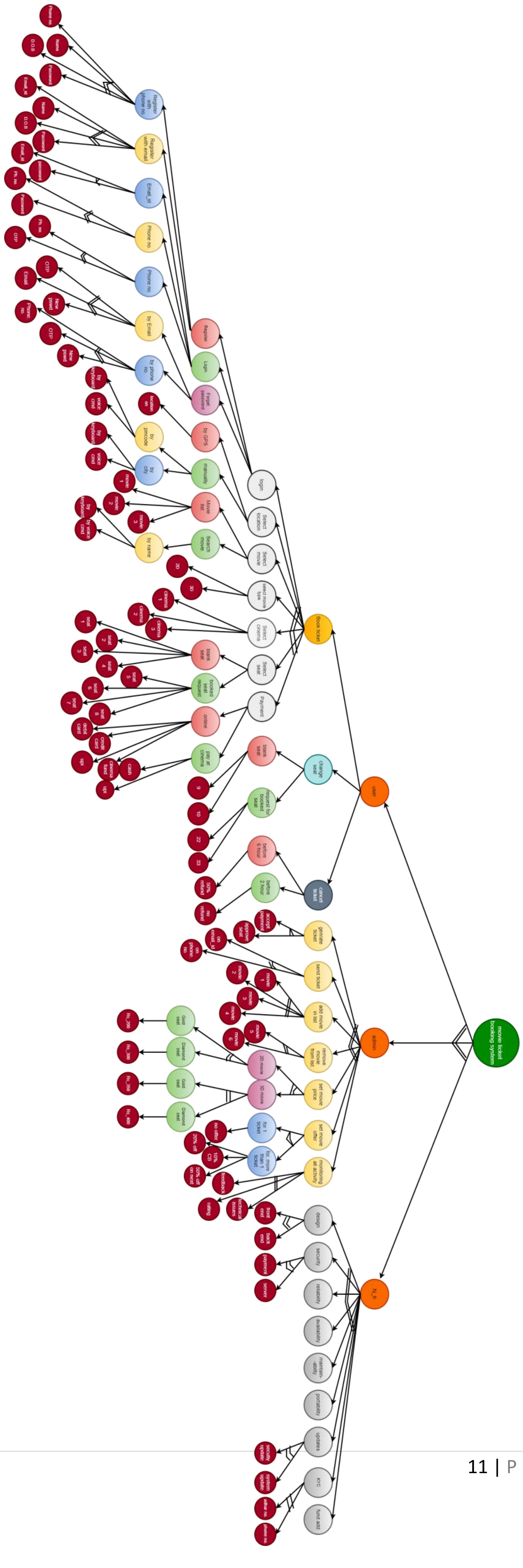
The system shall provide multi-language support.

j) **Supportability –** The source code developed for this system shall be maintained in configuration management tool.

2.3 Elicitation of Movie Ticket Booking System Using Goal Oriented Method:-

In this goal oriented method we will divide the main goals into the sub-goals that help to collect all the requirements easily.

Using goal oriented method we connect all the requirements finding in the article 2.2 by AND OR graph which is shown below-



movie ticket booking system
user
admin
N_fr
Book ticket
change seat
cancel ticket
login
Select location
Select movie
select movie type
Select cinema
Select seat
Payment
Register
Login
Forgot password
Register with phone no
Register with email
Email_id
Phone no
by Email
by phone no
by GPS
manually
by pincode
by city
Movie list
Search movie
by name
2D
3D
blank seat
booked seat request
online
pay at cinema
debit card
credit card
upi
cash
before 6 hour
before 2 hour
50% refund
no refund
generate ticket
send ticket
add movie in list
remove movie from list
set movie price
set movie offer
monitoring all activity
2D movie
3D movie
Gold seat
Diamond seat
for 1 ticket
for more than 1 ticket
feedback
rating
design
security
reliability
availability
maintainability
portability
updates
KYC
fund add
front end
back end
payment
server
security update
system update

Fig 2.1

2.4 Modelling of Movie Ticket Booking System:- Using the class diagram we model these requirements in such a way-

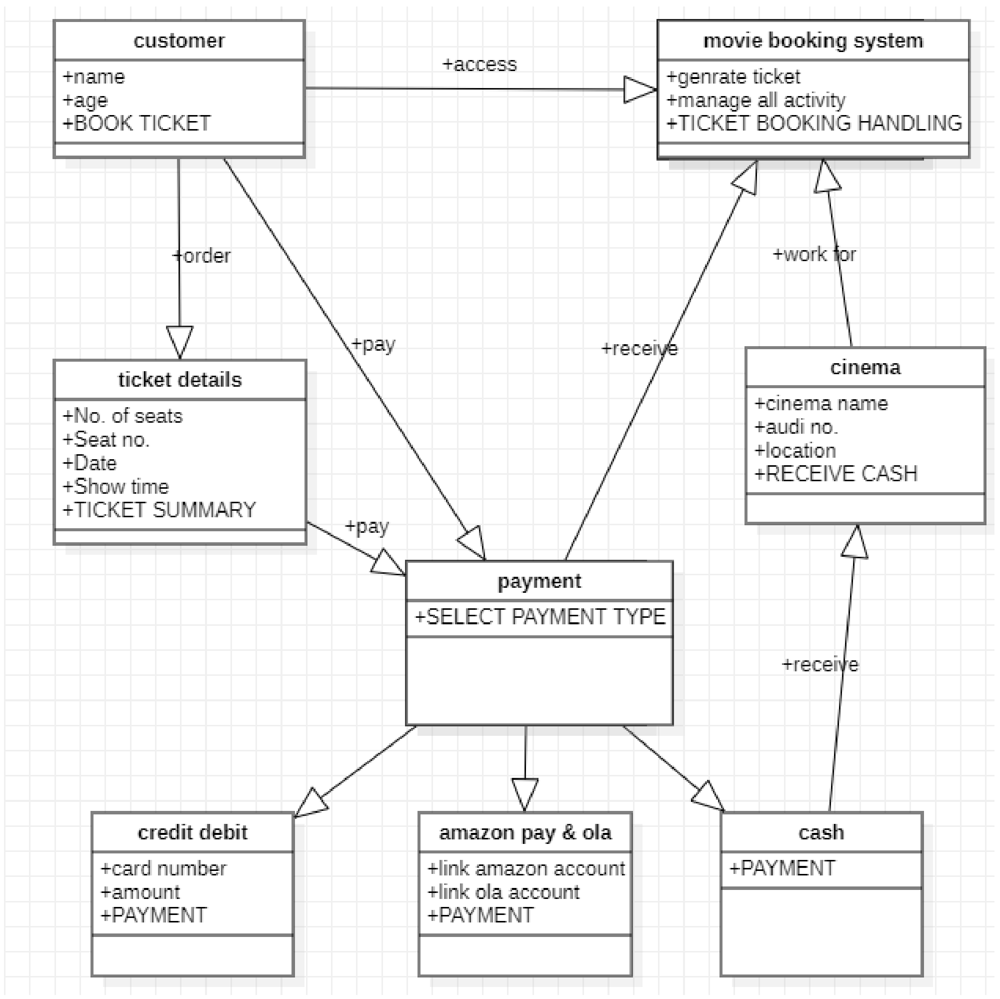


Fig 2.2

2.5 Conclusion:- We have discuss all the possible aspects of during the development of Movie ticket booking system.

Chapter 3:

Implementation

3.1 Introduction:- In this chapter we discuss how to apply the code to implement any requirement of the project. If you know the basics of HTML and CSS concept then it will take only one hour to understand it. HTML provides the basic structure of a web page and it also consider as a skeleton of the body and CSS provides the colours of page like skin of the skeleton.

3.2 Implementation of Login module:-

---Index.html---

Let's create *index.html* and add the following code to it:

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <meta http-equiv="X-UA-Compatible" content="ie=edge" />
  <link rel="stylesheet" href="style.css" />
  <title>Movie Seat Booking</title>
 </head>
 <body>
  <div class="movie-container">
   <label>Pick a movie:</label>
   <select id="movie">
    <option value="100">Avengers: Endgame (100)</option>
    <option value="120">Joker (Rs.120)</option>
    <option value="80">Harry Potter (Rs.80)</option>
    <option value="90">The Lion King (Rs.90)</option>
   </select>
  </div>

  <ul class="showcase">
   <li>
    <div class="seat"></div>
    <small>N/A</small>
   </li>
   <li>
    <div class="seat selected"></div>
    <small>Selected</small>
   </li>
   <li>
    <div class="seat occupied"></div>
    <small>Occupied</small>
  </li>
</ul>

<div class="container">
  <div class="screen"></div>

  <div class="row">
    <div class="seat"></div>
    <div class="seat"></div>
    <div class="seat"></div>
    <div class="seat"></div>
    <div class="seat"></div>
    <div class="seat"></div>
    <div class="seat"></div>
    <div class="seat"></div>
  </div>
  <div class="row">
    <div class="seat"></div>
    <div class="seat"></div>
    <div class="seat"></div>
    <div class="seat occupied"></div>
    <div class="seat occupied"></div>
    <div class="seat"></div>
    <div class="seat"></div>
    <div class="seat"></div>
  </div>
  <div class="row">
    <div class="seat"></div>
    <div class="seat"></div>
    <div class="seat"></div>
    <div class="seat"></div>
    <div class="seat"></div>
    <div class="seat"></div>
    <div class="seat occupied"></div>
    <div class="seat occupied"></div>
  </div>
  <div class="row">
    <div class="seat"></div>
    <div class="seat"></div>
    <div class="seat"></div>
    <div class="seat"></div>
    <div class="seat"></div>
    <div class="seat"></div>
    <div class="seat"></div>
    <div class="seat"></div>
  </div>
  <div class="row">
    <div class="seat"></div>
    <div class="seat"></div>
    <div class="seat"></div>
    <div class="seat occupied"></div>
    <div class="seat occupied"></div>
    <div class="seat"></div>
    <div class="seat"></div>
    <div class="seat"></div>
  </div>
  <div class="row">
    <div class="seat"></div>
    <div class="seat"></div>
    <div class="seat"></div>
    <div class="seat"></div>
    <div class="seat occupied"></div>
```

```
      <div class="seat occupied"></div>
      <div class="seat occupied"></div>
      <div class="seat"></div>
    </div>
  </div>

  <p class="text">
    You have selected <span id="count">0</span> seats for a price of Rs.<span
      id="total"
      >0</span
    >
  </p>

  <script src="script.js"></script>
 </body>
            </html>
```

---Script.js---

Let's create a JavaScript file named *script.js* and add the following JavaScript code to it:

```
const container = document.querySelector('.container');
const seats = document.querySelectorAll('.row .seat:not(.occupied)');
const count = document.getElementById('count');
const total = document.getElementById('total');
const movieSelect = document.getElementById('movie');

populateUI();

let ticketPrice = +movieSelect.value;

// Save selected movie index and price
function setMovieData(movieIndex, moviePrice) {
  localStorage.setItem('selectedMovieIndex', movieIndex);
  localStorage.setItem('selectedMoviePrice', moviePrice);
}

// Update total and count
function updateSelectedCount() {
  const selectedSeats = document.querySelectorAll('.row .seat.selected');

  const seatsIndex = [...selectedSeats].map(seat => [...seats].indexOf(seat));

  localStorage.setItem('selectedSeats', JSON.stringify(seatsIndex));

  const selectedSeatsCount = selectedSeats.length;

  count.innerText = selectedSeatsCount;
  total.innerText = selectedSeatsCount * ticketPrice;

  setMovieData(movieSelect.selectedIndex, movieSelect.value);
}

// Get data from localstorage and populate UI
function populateUI() {
  const selectedSeats = JSON.parse(localStorage.getItem('selectedSeats'));

  if (selectedSeats !== null && selectedSeats.length > 0) {
    seats.forEach((seat, index) => {
      if (selectedSeats.indexOf(index) > -1) {
        seat.classList.add('selected');
      }
    });
  }

  const selectedMovieIndex = localStorage.getItem('selectedMovieIndex');

  if (selectedMovieIndex !== null) {
    movieSelect.selectedIndex = selectedMovieIndex;
  }
}

// Movie select event
movieSelect.addEventListener('change', e => {
  ticketPrice = +e.target.value;
  setMovieData(e.target.selectedIndex, e.target.value);
  updateSelectedCount();
});

// Seat click event
container.addEventListener('click', e => {
  if (
    e.target.classList.contains('seat') &&
    !e.target.classList.contains('occupied')
  ) {
    e.target.classList.toggle('selected');

    updateSelectedCount();
  }
});

// Initial count and total set
updateSelectedCount();
```

---Style.css---

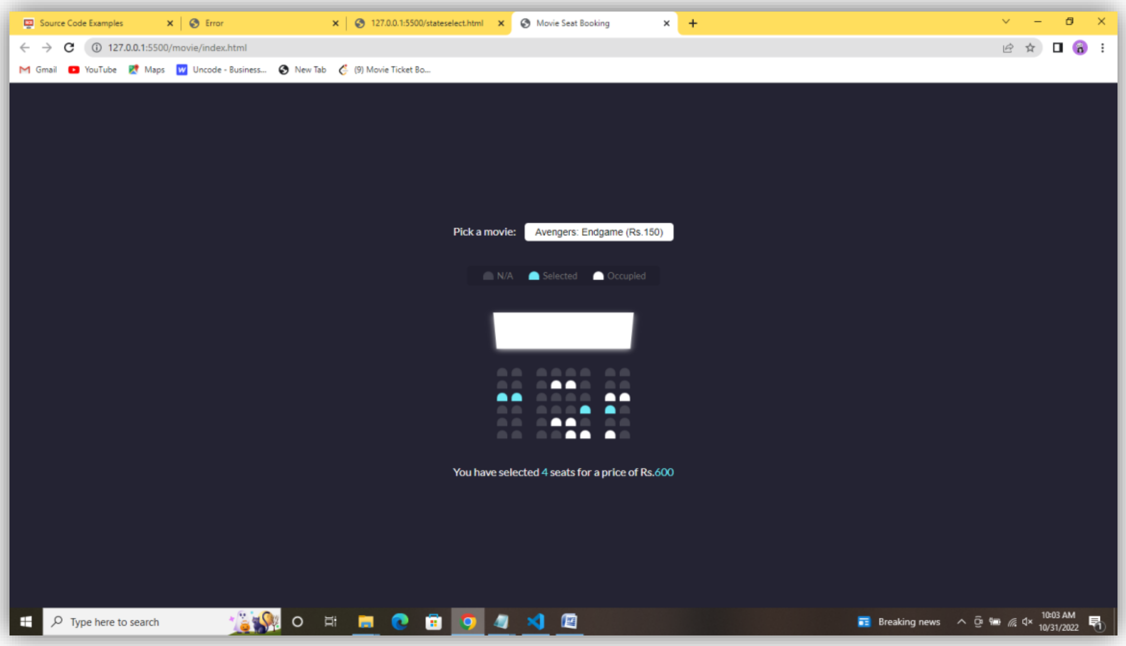
Let's create a CSS file named *style.css* and add the following CSS code to it:

```
@import url('https://fonts.googleapis.com/css?family=Lato&display=swap');

* {
  box-sizing: border-box;
}

body {
  background-color: #242333;
  color: #fff;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  height: 100vh;
  font-family: 'Lato', sans-serif;
  margin: 0;
}

.movie-container {
  margin: 20px 0;
```

```
}

.movie-container select {
  background-color: #fff;
  border: 0;
  border-radius: 5px;
  font-size: 14px;
  margin-left: 10px;
  padding: 5px 15px 5px 15px;
  -moz-appearance: none;
  -webkit-appearance: none;
  appearance: none;
}

.container {
  perspective: 1000px;
  margin-bottom: 30px;
}

.seat {
  background-color: #444451;
  height: 12px;
  width: 15px;
  margin: 3px;
  border-top-left-radius: 10px;
  border-top-right-radius: 10px;
}

.seat.selected {
  background-color: #6feaf6;
}

.seat.occupied {
  background-color: #fff;
}

.seat:nth-of-type(2) {
  margin-right: 18px;
}

.seat:nth-last-of-type(2) {
  margin-left: 18px;
}

.seat:not(.occupied):hover {
  cursor: pointer;
  transform: scale(1.2);
}

.showcase .seat:not(.occupied):hover {
  cursor: default;
  transform: scale(1);
}

.showcase {
  background: rgba(0, 0, 0, 0.1);
  padding: 5px 10px;
  border-radius: 5px;
  color: #777;
  list-style-type: none;
  display: flex;
  justify-content: space-between;
```

```
}

.showcase li {
  display: flex;
  align-items: center;
  justify-content: center;
  margin: 0 10px;
}

.showcase li small {
  margin-left: 2px;
}

.row {
  display: flex;
}

.screen {
  background-color: #fff;
  height: 70px;
  width: 100%;
  margin: 15px 0;
  transform: rotateX(-45deg);
  box-shadow: 0 3px 10px rgba(255, 255, 255, 0.7);
}

p.text {
  margin: 5px 0;
}

p.text span {
  color: #6feaf6;
}
```

3.3 **Conclusion:-** front page of movie ticket booking system is successfully created with using HTML, CSS, and javascript.



Pick a movie: Avengers: Endgame (Rs.150)

N/A Selected Occupied

You have selected 4 seats for a price of Rs.600

-------output------

Chapter 4:

Contributions and Future Work

4.1 Introduction:- To implement a login page we study various page carefully it will help us to built a good page. By this way running websites are very useful to provides us a clear idea about the page. Our main work to maintain the software because it will work for a long time period without any errors.

4.2 Contribution:- In this software project the party that participates are market software, friends, juniors and customers demand. although if a developer has any requirement that around to the project he will also consider it as a primary requirement.

4.3 Future Work:- Our future work is to maintain software ,testing in a certain period of time. And one of the most important thing is update it because update provides a software security and more efficiency to perform better it also reduce chance of any error.

★ **References:-** **While start any software project once do case study of top using same software available in the market. It will helps you to built a very good software product**.