CineMates

Movie Ticket Booking

Project By-

1.Vinit Sunil Chavan

en20169879@git-india.edu.in

2.Yash Ravindra Chawan

en20140555@git-india.edu.in

3.Omkar Vijay Ghorpade

en20183138@git-india.edu.in

ABSTRACT

A Java Spring Boot-based CineMates is a movie ticket booking streamlines the process of reserving seats for movies, offering a convenient and user-friendly experience for cinema enthusiasts. This application encompasses a robust three-layer architecture, comprising controllers, services, and data access objects (DAOs), ensuring efficient data flow.

Users can search for movies based on parameters like movie name, city, and date, which provides a list of available shows. Further, they can access detailed show information and the list of available seats for a particular show. To facilitate booking, the system offers real-time seat reservation, allowing users to select their preferred seats. The booking process takes into account factors like seat availability and overall booking status, ensuring a smooth and secure experience.

By employing Java Spring Boot, this application leverages the benefits of rapid development, robust security, and scalability, making it an ideal solution for the dynamic world of movie ticket bookings.

ACKNOWLEDGEMENT

We are very glad to present this project named
”Cinemates: Movie Ticket Booking ”.
We take this opportunity to express our gratitude to many people ,whose good wishes and positive approach have inspired us to take this project as success. However, it is imperative for us to mention some of them who played significant role in my project.

DECLARATION

We here by declaring that the project entitled,
” Cinemates: Movie Ticket Booking “has not been any case duplicated to submit to any other university for the award of any degree. To the best of my knowledge other than me, no one has submitted to any other university.

TABLE OF CONTENTS

INTRODUCTION

In the digital age, Java Spring Boot has revolutionized the way we book movie tickets. Queueing at cinema counters or using traditional booking methods is a thing of the past, thanks to this user-friendly movie ticket booking application.

This application seamlessly combines the power of Java with the simplicity of Spring Boot. It features a well-organized architecture with controllers, services, and data access objects (DAOs), ensuring smooth and robust data flow.

A standout feature is the search functionality. Users can easily find movies by specifying parameters like movie name, city, and date, instantly getting a list of matching shows. Show details, including timings and availability, empower users to make informed choices.

Additionally, the application streamlines seat selection and reservation. Users can effortlessly browse available seats, pick their preferred ones, and complete bookings in just a few clicks. Real-time updates prevent double bookings and ensure a hassle-free experience.

Built on Java Spring Boot, this application boasts rapid development, robust security, and scalability. It's the modern solution for moviegoers, meeting their evolving needs in today's tech world.

FEATURES AND FUNCTIONALITIES

1. Dashboard: An admin dashboard displaying key information, including the total number of movies, theaters, and bookings.

2. Movie Management:
2.1 Add Movies: Ability to add new movies to the H2 database. Includes movie title, description, duration, language, release date, country, and genre.
2.2 Edit Movies: Modify movie details, in case of updates or corrections.
2.3 Delete Movies: Remove movies that are no longer being screened.

3. Theater/Screen Management:
3.1 Add Theaters/Screens: Administrators can add new theaters or screens to the database, specifying their names and capacities.
3.2 Edit Theaters/Screens: Modify theater or screen details, such as name or capacity.
3.3 Delete Theaters/Screens: Remove theaters or screens that are no longer in use.

4. Show Management:
4.1 Create Shows: Define showtimes, assign movies to theaters/screens, and set seat availability for each show.
4.2 Edit Shows: Modify show details, such as showtime or seat availability.
4.3 Cancel Shows: Administrators can cancel or reschedule shows as needed.

5. Seat Booking Management:
5.1 View Bookings: Admins can view all seat bookings, including show details, user information, and booked seats.
5.2 Cancel Bookings: Ability to cancel bookings if necessary.

6. Seat Availability: Ensures that once a seat is booked, it becomes unavailable for further booking during that showtime.

7. Data Persistence: The website stores all movie, theater, screen, show, and booking data in the H2 database.

8. Data Validation: Input data is validated to ensure accuracy and consistency.

9. Error Handling: The system provides error messages and logs for effective issue resolution.

10. Reports: Generate reports on movie bookings, theater occupancy, or revenue for data analysis.

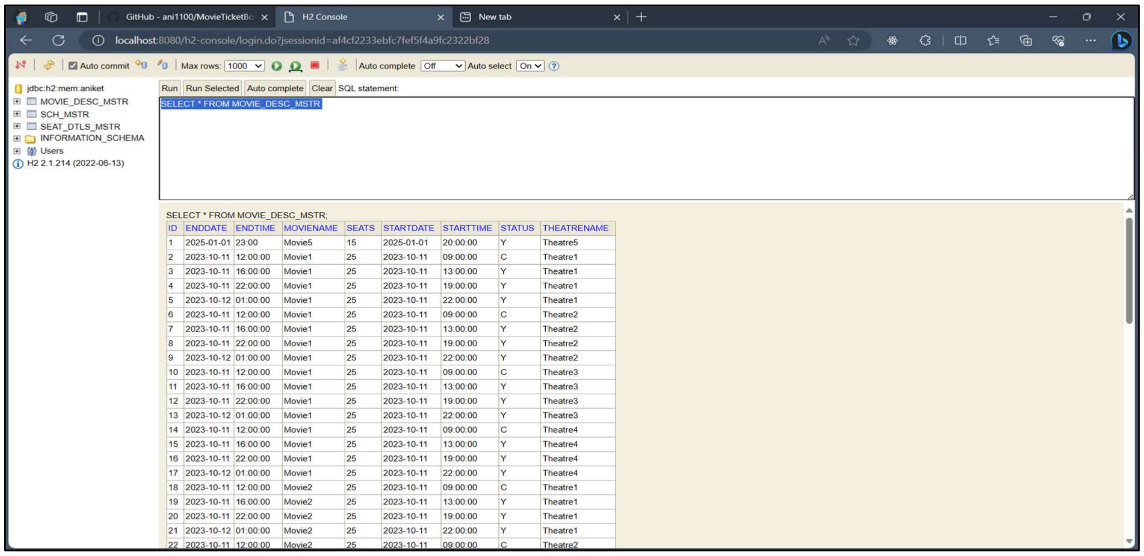
11. User Management: Add, edit, or remove administrators who can access the admin portal.

12. Data Backup and Restore: Option to create backups of the H2 database and restore data if needed.

This movie ticket booking website simplifies movie and show management, ensuring efficient administration and seat booking without the risk of double bookings. The H2 database keeps data organized and accessible, offering a complete solution for movie theater administrators.

DATABASE DESIGN

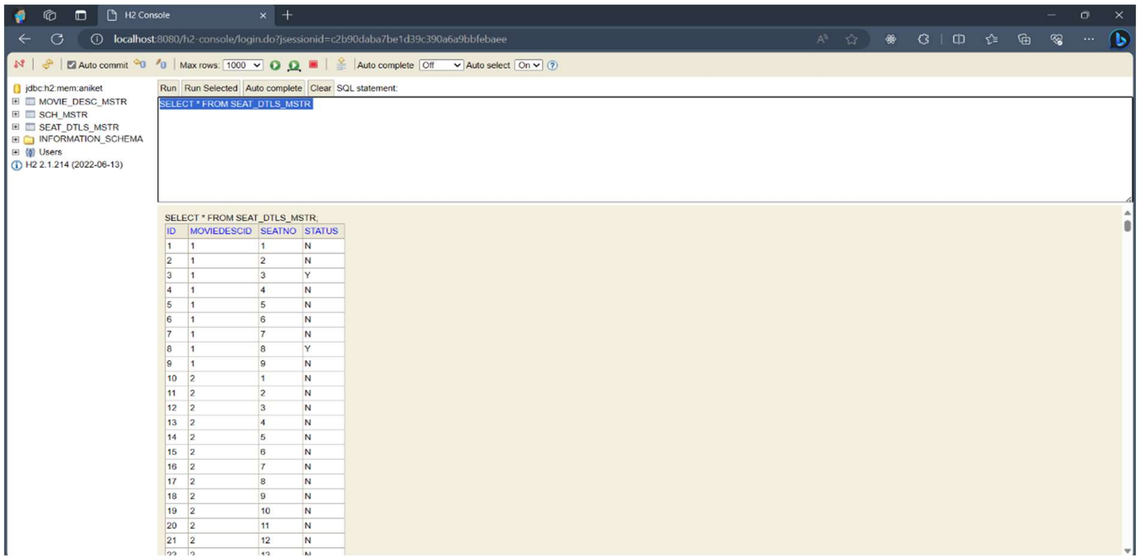
H2 is a lightweight, open-source, and in-memory relational database that is commonly used with Spring Boot applications, especially during development and testing phases. It provides a convenient way to work with a database without requiring a separate database server installation. Here's some key information about using H2 in Spring Boot.



```
SELECT * FROM MOVIE_DESC_MSTR
```

SELECT * FROM MOVIE_DESC_MSTR;

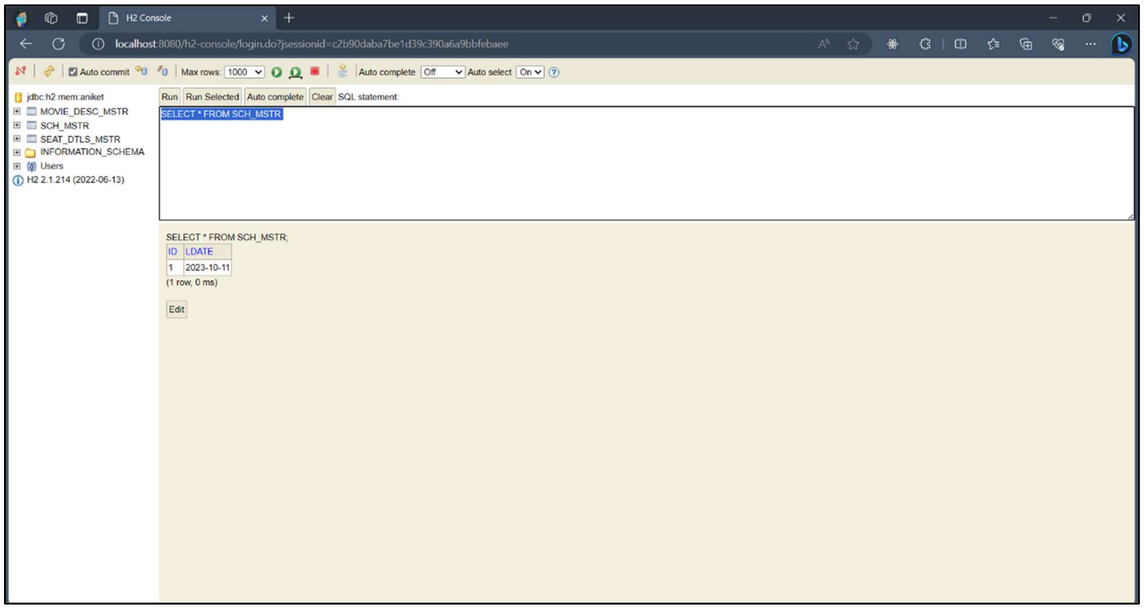
ID	ENDDATE	ENDTIME	MOVIENAME	SEATS	STARTDATE	STARTTIME	STATUS	THEATRENAME
1	2025-01-01	23:00	Movie5	15	2025-01-01	20:00:00	Y	Theatre5
2	2023-10-11	12:00:00	Movie1	25	2023-10-11	09:00:00	C	Theatre1
3	2023-10-11	16:00:00	Movie1	25	2023-10-11	13:00:00	Y	Theatre1
4	2023-10-11	22:00:00	Movie1	25	2023-10-11	19:00:00	Y	Theatre1
5	2023-10-12	01:00:00	Movie1	25	2023-10-11	22:00:00	Y	Theatre1
6	2023-10-11	12:00:00	Movie1	25	2023-10-11	09:00:00	C	Theatre2
7	2023-10-11	16:00:00	Movie1	25	2023-10-11	13:00:00	Y	Theatre2
8	2023-10-11	22:00:00	Movie1	25	2023-10-11	19:00:00	Y	Theatre2
9	2023-10-12	01:00:00	Movie1	25	2023-10-11	22:00:00	Y	Theatre2
10	2023-10-11	12:00:00	Movie1	25	2023-10-11	09:00:00	C	Theatre3
11	2023-10-11	16:00:00	Movie1	25	2023-10-11	13:00:00	Y	Theatre3
12	2023-10-11	22:00:00	Movie1	25	2023-10-11	19:00:00	Y	Theatre3
13	2023-10-12	01:00:00	Movie1	25	2023-10-11	22:00:00	Y	Theatre3
14	2023-10-11	12:00:00	Movie1	25	2023-10-11	09:00:00	C	Theatre4
15	2023-10-11	16:00:00	Movie1	25	2023-10-11	13:00:00	Y	Theatre4
16	2023-10-11	22:00:00	Movie1	25	2023-10-11	19:00:00	Y	Theatre4
17	2023-10-12	01:00:00	Movie1	25	2023-10-11	22:00:00	Y	Theatre4
18	2023-10-11	12:00:00	Movie2	25	2023-10-11	09:00:00	C	Theatre1
19	2023-10-11	16:00:00	Movie2	25	2023-10-11	13:00:00	Y	Theatre1
20	2023-10-11	22:00:00	Movie2	25	2023-10-11	19:00:00	Y	Theatre1
21	2023-10-12	01:00:00	Movie2	25	2023-10-11	22:00:00	Y	Theatre1
22	2023-10-11	12:00:00	Movie2	25	2023-10-11	09:00:00	C	Theatre2



```
SELECT * FROM SEAT_DTLS_MSTR
```

SELECT * FROM SEAT_DTLS_MSTR;

ID	MOVIEDESCID	SEATNO	STATUS
1	1	1	N
2	1	2	N
3	1	3	Y
4	1	4	N
5	1	5	N
6	1	6	N
7	1	7	N
8	1	8	Y
9	1	9	N
10	2	1	N
11	2	2	N
12	2	3	N
13	2	4	N
14	2	5	N
15	2	6	N
16	2	7	N
17	2	8	N
18	2	9	N
19	2	10	N
20	2	11	N
21	2	12	N
22	2	13	N



```
SELECT * FROM SCH_MSTR
```

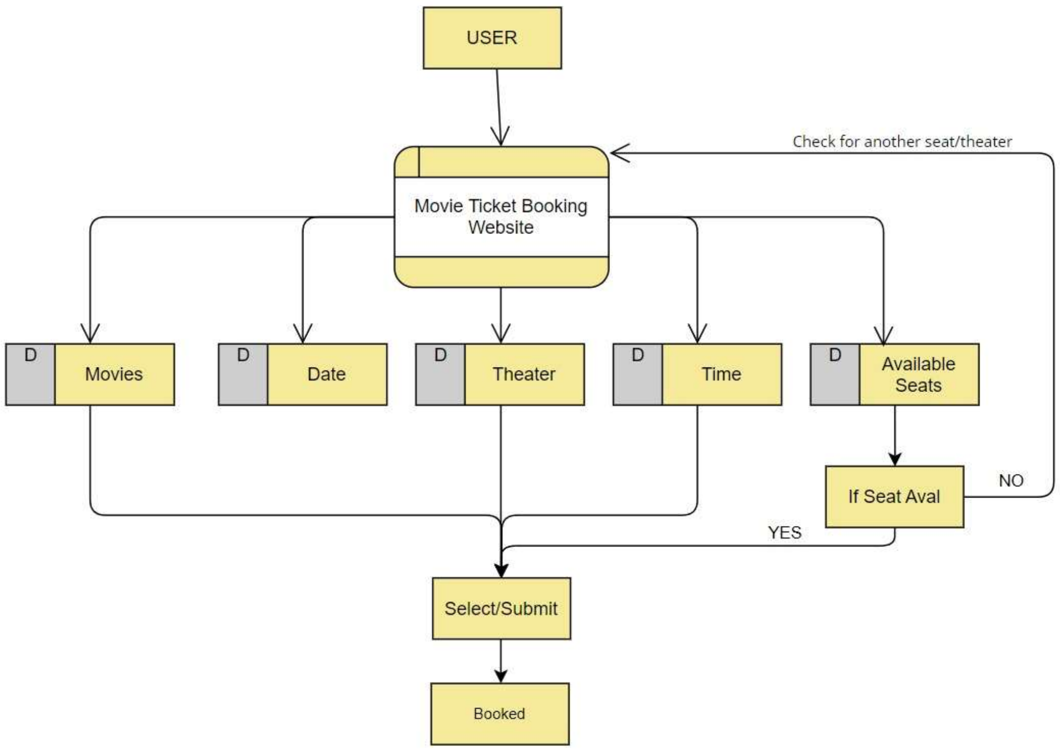
```
SELECT * FROM SCH_MSTR;
```

ID	LDATE
1	2023-10-11

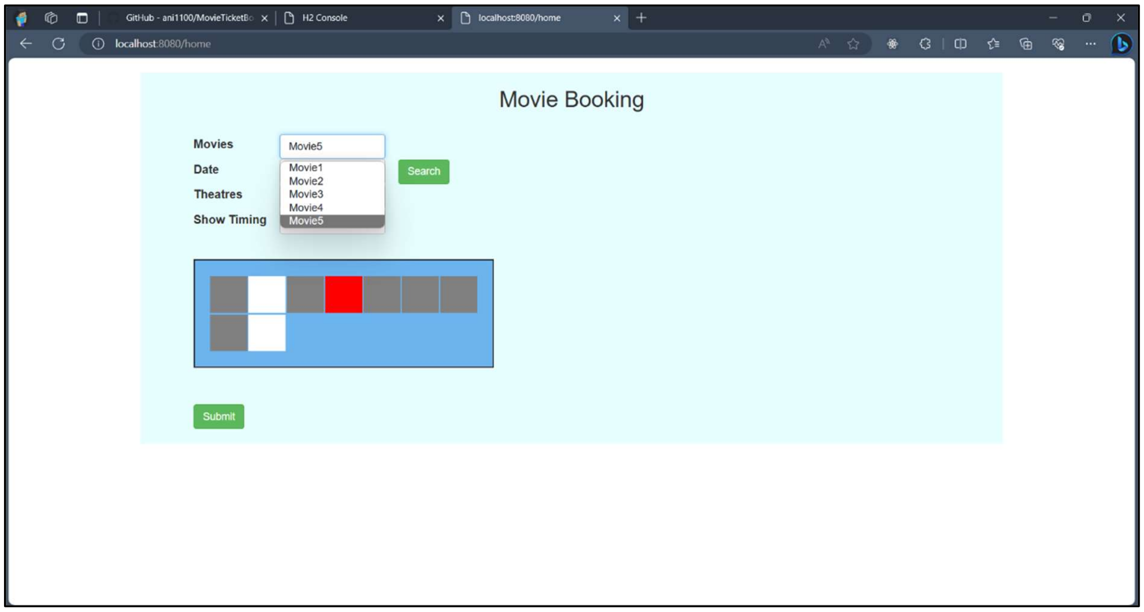
(1 row, 0 ms)

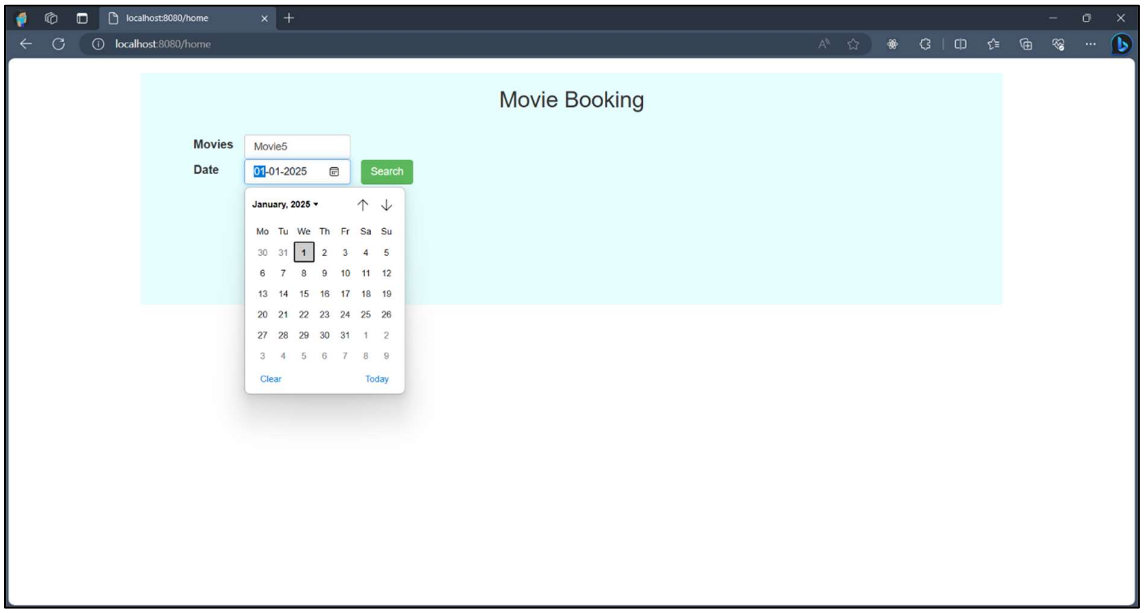
1. Create Spring Boot Application (A): You create a Spring Boot application.
2. Define Entity Classes (B): You define entity classes annotated with @Entity that represent the structure of the data you want to store.
3. Define Repositories (C): You define repositories by extending JpaRepository interfaces, allowing you to perform CRUD (Create, Read, Update, Delete) operations on entities.
4.Configure H2 Database (D): You configure the H2 database in your Spring Boot application, specifying database connection details and enabling the H2 console if needed.
5. Initialize H2 Database (E): The application initializes the H2 database based on your entity classes and configuration.
6. CRUD Operations (F): The application can perform CRUD operations on the database through the repositories.
7. Query Data (G): The application can query data from the H2 database to retrieve and display information

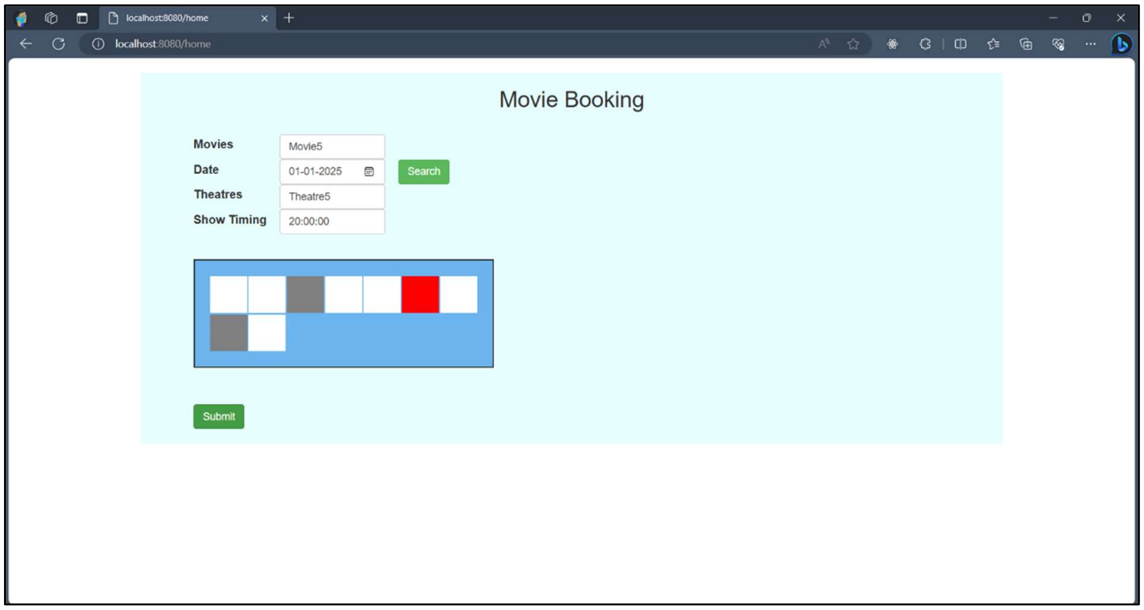
FLOW CHART

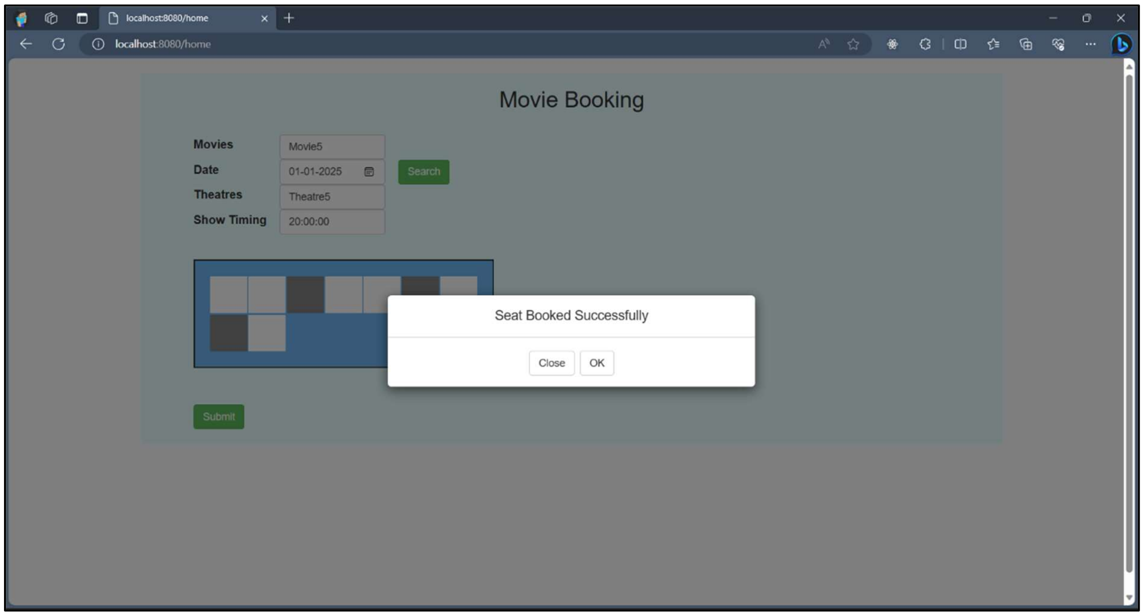

USER
Check for another seat/theater
Movie Ticket Booking Website
D
Movies
D
Date
D
Theater
D
Time
D
Available Seats
If Seat Aval
NO
YES
Select/Submit
Booked

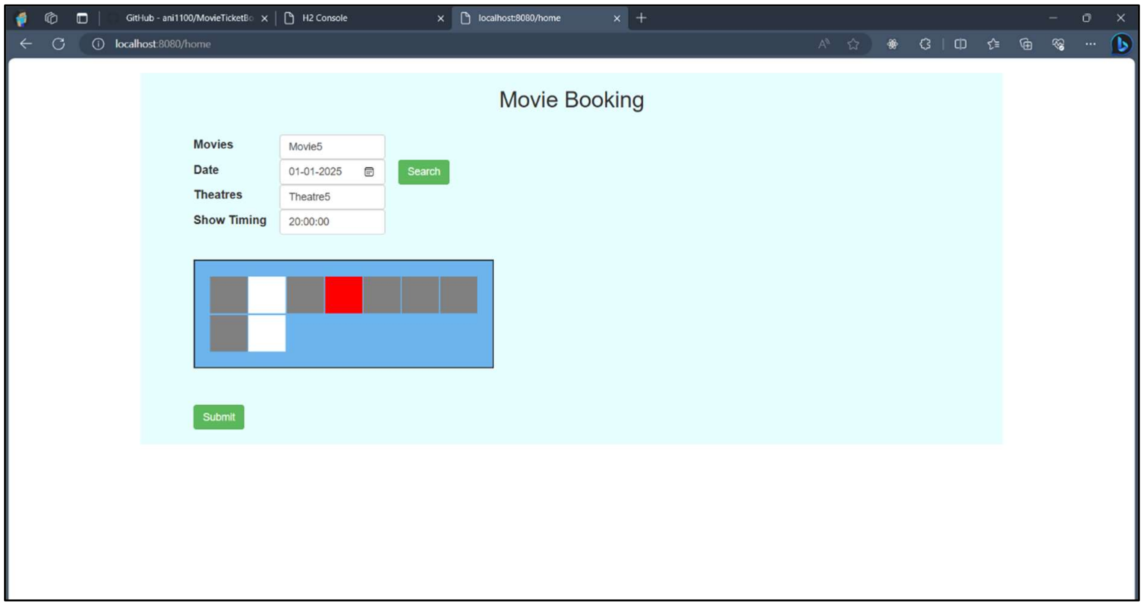
IMPLEMENTATION





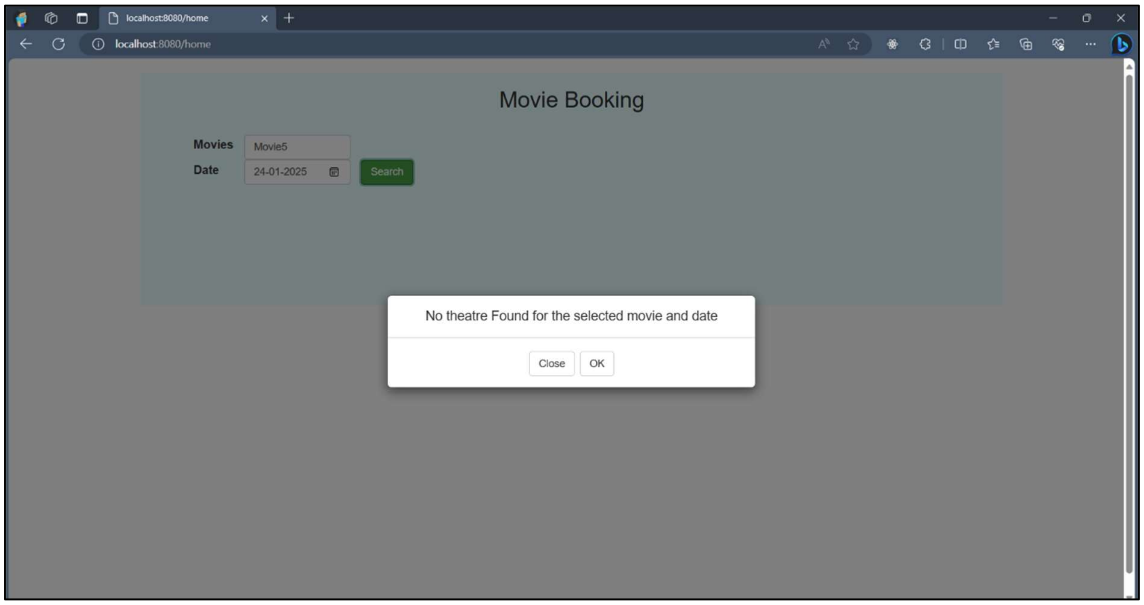

localhost:8080/home
Movie Booking
Movies
Movie5
Date
01-01-2025
Search
Theatres
Theatre5
Show Timing
20:00:00
Submit


localhost:8080/home
Movie Booking
Movies
Movie5
Date
01-01-2025
Search
Theatres
Theatre5
Show Timing
20:00:00
Seat Booked Successfully
Close
OK
Submit



Movie Booking

Movies	Movie5
Date	01-01-2025
Theatres	Theatre5
Show Timing	20:00:00



Movie Booking

Movies	Movie5
Date	24-01-2025

No theatre Found for the selected movie and date

SOURCE CODE

--HomeController--

```
package com.portal.movieticketbooking.controller;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
@Controller
public class HomeController {
	@RequestMapping("home")
	public String home()
	{
		return "templates/index.jsp";
	}
}
```

--TicketBookingController

```
package com.portal.movieticketbooking.controller;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;
import com.portal.movieticketbooking.model.MovieDescModel;
import com.portal.movieticketbooking.model.SeatDtlsModel;
import com.portal.movieticketbooking.pojo.MovieInfoPojo;
import com.portal.movieticketbooking.pojo.Savepojo;
import com.portal.movieticketbooking.pojo.Theatreinfopojo;
import com.portal.movieticketbooking.service.TicketBookingSrvc;
@RestController
public class TicketBookingController {
	@Autowired
	TicketBookingSrvc srvc;
	@GetMapping("/getactivemovies")
	public List<String> getactivemovies()
	{
		return(srvc.fetchactivemovie());
	}
	@PostMapping("/fetchtheatresbymovieanddate")
	public List<MovieDescModel> fetchtheatresbymovieanddate(@RequestBody 
Theatreinfopojo pojo)
	{
	return(srvc.fetchtheatresbyname(pojo.getThname(),pojo.reqpojo.getMoviename(),pojo.reqp
ojo.getDate(),pojo.reqpojo.getTime()));
	}
	@PostMapping("/fetchseatdtls")
	public List<SeatDtlsModel> fetchseatdtls(@RequestBody MovieDescModel moviedtls)
	{
		return(srvc.fetchseatdtls(moviedtls));
	}

	@PostMapping("/savedetails")
	public List<String> savedetails(@RequestBody Savepojo pojo)
	{
		return(srvc.savedetails(pojo));
	}

	@PostMapping("/gettheatres")
	public List<String> fetchtheatres(@RequestBody MovieInfoPojo reqpojo)
	{

	return(srvc.fetchtheatres(reqpojo.getMoviename(),reqpojo.getDate(),reqpojo.getTime()));
	}

}
```

--MovieDescModel--

```
package com.portal.movieticketbooking.model;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Table;
@Entity
@Table(name = "movie_desc_mstr")
public class MovieDescModel {
	@Id
	@Column(name = "Id", updatable = false, nullable = false)
	private Long id;
	@Column(name = "Moviename", nullable = false)
	private String moviename;
	@Column(name = "Theatrename", nullable = false)
	private String theatrename;
	@Column(name = "Startdate", nullable = false)
	private String startdate;
	@Column(name = "Starttime", nullable = false)
	private String starttime;
	@Column(name = "Enddate",nullable = false)
	private String enddate;
	@Column(name = "Endtime", nullable = false)
	private String endtime;
```

```
@Column(name = "Seats", nullable = false)
private int seats;
@Column(name = "Status", nullable = false)
private String status;
public Long getId() {
        return id;
}
public void setId(Long id) {
        this.id = id;
}
public String getMoviename() {
        return moviename;
}
public void setMoviename(String moviename) {
        this.moviename = moviename;
}
public String getTheatrename() {
        return theatrename;
}
public void setTheatrename(String theatrename) {
        this.theatrename = theatrename;
}
public String getStartdate() {
        return startdate;
}
public void setStartdate(String startdate) {
        this.startdate = startdate;
}
public String getStarttime() {
        return starttime;
}
public void setStarttime(String starttime) {
        this.starttime = starttime;
}
public String getEnddate() {
        return enddate;
}
public void setEnddate(String enddate) {
        this.enddate = enddate;
}
public String getEndtime() {
        return endtime;
}
public void setEndtime(String endtime) {
        this.endtime = endtime;
}
```

```
	public int getSeats() {
		return seats;
	}
	public void setSeats(int seats) {
		this.seats = seats;
	}
	public String getStatus() {
		return status;
	}
	public void setStatus(String status) {
		this.status = status;
	}
}
```

--SchModel--

```
package com.portal.movieticketbooking.model;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Table;
@Entity
@Table(name = "sch_mstr")
public class SchModel {
	@Id
	@Column(name = "id")
	private Long id;
	@Column(name = "ldate")
	private String ldate;
	public Long getId() {
		return id;
	}
	public void setId(Long id) {
		this.id = id;
	}
	public String getLdate() {
		return ldate;
	}
	public void setLdate(String ldate) {
		this.ldate = ldate;
	}

}
```

--SeatDtlsModel--

```
package com.portal.movieticketbooking.model;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Table;
@Entity
@Table(name = "seat_dtls_mstr")
public class SeatDtlsModel {
	@Id
	@Column(name = "Id", updatable = false, nullable = false)
	private Long id
	@Column(name = "Moviedescid", nullable = false)
	private Long moviedescid;
	@Column(name = "Seatno", nullable = false)
	private int seatno;
	@Column(name = "Status", nullable = false)
	private String status;

	public Long getId() {
		return id;
	}
	public void setId(Long id) {
		this.id = id;
	}
	public Long getMoviedescid() {
		return moviedescid;
	}
	public void setMoviedescid(Long moviedescid) {
		this.moviedescid = moviedescid;
	}
	public int getSeatno() {
		return seatno;
	}
	public void setSeatno(int seatno) {
		this.seatno = seatno;
	}
	public String getStatus() {
		return status;
	}
	public void setStatus(String status) {
		this.status = status;
	}
}
```

--SchedulerSrvc-

```
package com.portal.movieticketbooking.service;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Arrays;
import java.util.Calendar;
import java.util.Date;
import java.util.List;
import java.util.Optional;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Configuration;
import org.springframework.scheduling.annotation.EnableScheduling;
import org.springframework.scheduling.annotation.Scheduled;
import com.portal.movieticketbooking.model.MovieDescModel;
import com.portal.movieticketbooking.model.SchModel;
import com.portal.movieticketbooking.model.SeatDtlsModel;
import com.portal.movieticketbooking.repository.LastDateSaveRepo;
import com.portal.movieticketbooking.repository.MovieDescRepository;
import com.portal.movieticketbooking.repository.SeatDtlsRepository;
@Configuration
@EnableScheduling
public class SchedulerSrvc {
	@Autowired
	MovieDescRepository movierepo;
	@Autowired
	SeatDtlsRepository seatrepo;
	@Autowired
	LastDateSaveRepo schrepo;
	@Scheduled(fixedDelay = 60000)
   public void initialization() throws InterruptedException, ParseException {
		List<String> movies= Arrays.asList("Movie1", "Movie2", "Movie3","Movie4");
		List<String> theatres= Arrays.asList("Theatre1", "Theatre2", "Theatre3","Theatre4");
		List<String> stime= Arrays.asList("09:00:00", "13:00:00","19:00:00","22:00:00");
		SimpleDateFormat formatter = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
      Date date = new Date();
      String date1=formatter.format(date).split(" ")[0];
      SchModel zobj=schrepo.findAll().get(0);
      if(!(zobj.getLdate().equals(date1)))
      {
		for(String mov:movies)
		{
		     for(String th:theatres)
		     {
			for(String st:stime)
			{
				String temp=date1.concat(" ").concat(st).concat(":00");
				Date datear=formatter.parse(temp);
				Calendar caldep = Calendar.getInstance();
```

```
                                        caldep.setTime(datear);
                                        caldep.add(Calendar.HOUR_OF_DAY, 3);
                                        Date date3 = caldep.getTime();
                                        SimpleDateFormat format1 = new
SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
                                        String date4 = format1.format(date3).split(" ")[0];
                                        String time4 = format1.format(date3).split(" ")[1];
                                MovieDescModel obj=new MovieDescModel();
                                Long id1=movierepo.fetchmaxid();
                                obj.setEnddate(date4);
                                obj.setEndtime(time4);
                                obj.setMoviename(mov);
                                obj.setId(id1);
                                obj.setSeats(25);
                                obj.setStartdate(date1);
                                obj.setStarttime(st);
                                obj.setStatus("Y");
                                obj.setTheatrename(th);
                                movierepo.save(obj);
                        }
                }
            }
    }
    zobj.setLdate(date1);
    schrepo.save(zobj);
                List<MovieDescModel> lisobj=movierepo.fetchallacticemovies();
                for(MovieDescModel obj:lisobj)
                {
                        List<SeatDtlsModel> lis1=seatrepo.fetchseatdtls(obj.getId());
                        if(lis1.size()==0)
                        {
                                for(int i=1;i<=obj.getSeats();i++)
                                {
                                        Long id1=seatrepo.fetchmaxid();
                                        SeatDtlsModel seatobj=new SeatDtlsModel();
                                        seatobj.setId(id1);
                                        seatobj.setMoviedescid(obj.getId());
                                        seatobj.setSeatno(i);
                                        seatobj.setStatus("N");
                                        seatrepo.save(seatobj);
                                }
                        }}
    date = new Date();
    String dt1=formatter.format(date).split(" ")[0];
    String dt2=formatter.format(date).split(" ")[1];
    movierepo.autorunning(dt1, dt2);
    movierepo.autocompletion(dt1, dt2);
    }}
```

--LastDateSaveRepo--

```
package com.portal.movieticketbooking.repository;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
import com.portal.movieticketbooking.model.MovieDescModel;
import com.portal.movieticketbooking.model.SchModel;

@Repository
public interface LastDateSaveRepo extends JpaRepository<SchModel,Long> {

}
```

--MovieDescRepository--

```
package com.portal.movieticketbooking.repository;
import java.util.List
import javax.transaction.Transactional;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Modifying;
import org.springframework.data.jpa.repository.Query;
import org.springframework.stereotype.Repository;
import com.portal.movieticketbooking.model.MovieDescModel;



@Repository
public interface MovieDescRepository extends JpaRepository<MovieDescModel,Long>
{
	@Query(value = "select distinct Moviename from movie_desc_mstr where Status='Y'",
nativeQuery = true)
	List<String> fetchmovies();

	@Query(value = "select distinct Theatrename from movie_desc_mstr where Status='Y' and
Moviename=?1 and Startdate=?2 and Starttime>=?3", nativeQuery = true)
	List<String> fetchtheatres(String moviename,String date,String time);

	@Transactional
	@Modifying
	@Query(value = "update movie_desc_mstr set Status='C' where Enddate<=?1 and Status='R'
and Endtime<=?2", nativeQuery = true)
	void autocompletion(String date,String time);

	@Transactional
	@Modifying
	@Query(value = "update movie_desc_mstr set Status='R' where Startdate<=?1 and
Status='Y' and Starttime<=?2", nativeQuery = true)
	void autorunning(String date,String time);

	@Query(value = "select * from movie_desc_mstr where Status='Y' and Moviename=?1 and
Startdate=?2 and Starttime>=?3 and Theatrename=?4", nativeQuery = true)
	List<MovieDescModel> fetchtheatresbyname(String moviename,String date,String
time,String thname);

	@Query(value = "select * from movie_desc_mstr where Status='Y'", nativeQuery = true)
	List<MovieDescModel> fetchallacticemovies();

	@Query(value = "select max(Id)+1 from movie_desc_mstr", nativeQuery = true)
	Long fetchmaxid();

}
```

--TicketBookingSrvcImpl--

```
package com.portal.movieticketbooking.service;
import java.util.ArrayList;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import com.portal.movieticketbooking.model.MovieDescModel;
import com.portal.movieticketbooking.model.SeatDtlsModel;
import com.portal.movieticketbooking.pojo.Savepojo;
import com.portal.movieticketbooking.repository.MovieDescRepository;
import com.portal.movieticketbooking.repository.SeatDtlsRepository;

@Service
public class TicketBookingSrvcImpl implements TicketBookingSrvc{

	@Autowired
	MovieDescRepository movierepo;

	@Autowired
	SeatDtlsRepository seatrepo;

	@Override
	public List<String> fetchactivemovie() {
		List<String> resp=new ArrayList<String>();
		try
		{
			List<String> lismovies=movierepo.fetchmovies();
			if(lismovies.size()==0)
			{
				resp.add("Failed");
				resp.add("No Movies Available");
			}
			else
			{
				resp.add("Success");
				resp.addAll(lismovies);
			}
			return resp;
		}
		catch(Exception e)
		{
			resp.clear();
			resp.add("Failed");
			resp.add("Error in processing data");
			return resp;
		}
	}

	@Override
	public List<String> fetchtheatres(String moviename, String date, String time) {
		try
		{
			List<String> lisobj=movierepo.fetchtheatres(moviename, date, time);
			if(lisobj.size()==0)
			{
				return null;
			}
			else
			{
				return lisobj;
			}
		}
		catch(Exception e)
		{
			return null;
		}
	}

	@Override
	public List<SeatDtlsModel> fetchseatdtls(MovieDescModel moviedtls) {
		try
		{
			List<SeatDtlsModel> seatlis= seatrepo.fetchseatdtls(moviedtls.getId());
			return seatlis;
		}
		catch(Exception e)
		{
			return null;
		}

	}

	@Override
	public List<String> savedetails(Savepojo pojo) {
		List<String> resp=new ArrayList<String>();
		try
		{
			if(movierepo.getById(pojo.getMovieid()).getStatus().equals("Y"))
			{
				List<SeatDtlsModel> liseats=new ArrayList<SeatDtlsModel>();
				for(Integer st:pojo.getSeats())
				{
					SeatDtlsModel
seatobj=seatrepo.fetchseat(pojo.getMovieid(),st);
					if(seatobj.getStatus().equals("Y"))
					{
						resp.add("Failed");
						resp.add("Seat Already Booked");
						return resp;
					}
					else
					{
						seatobj.setStatus("Y");
						liseats.add(seatobj);
					}
				}
				seatrepo.saveAllAndFlush(liseats);
				resp.add("Success");
				resp.add("Seat Booked Successfully");
				return resp;
			}
			else
			{
				resp.add("Failed");
				resp.add("Booking Closed for this show");
				return resp;
			}

		}
		catch(Exception e)
		{
			resp.clear();
			resp.add("Failed");
```

```
				resp.add("Error in processing data");
				return resp;
			}
	}

	@Override
	public List<MovieDescModel> fetchtheatresbyname(String thname, String moviename,
String date, String time) {
			// TODO Auto-generated method stub
			try
			{
				List<MovieDescModel> lisobj=movierepo.fetchtheatresbyname(moviename,
date, time,thname);
				if(lisobj.size()==0)
				{
					return null;
				}
				else
				{
					return lisobj;
				}
			}
			catch(Exception e)
			{
				return null;
			}
	}

}
```

--index.jsp--

```
<html ng-app="myApp">
<head>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0/dist/css/bootstrap.min.css"
rel="stylesheet"
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0/dist/js/bootstrap.bundle.min.js"
integrity="sha384-A3rJD856KowSb7dwlZdYEkO39Gagi7vIsF0jrRAoQmDKKtQBHUuLZ9AsSv4jD4Xa"
crossorigin="anonymous"></script>
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-
awesome.min.css">
```

```
<script src="js/Booking.js"></script>
<link rel="stylesheet" href="css/stylesheet.css">
</head>

<body>
 <br>
 <div data-ng-controller="Booking" data-ng-init= "getlist()" class="container" style="background-
color:#e7feff;">
 <h2 class="d-flex justify-content-center">Movie Booking</h2>
 <br>
 <form>
		<table style="margin-left:5%">
			<tbody>
				<tr>
					<td><label>Movies</label></td>
					<td style="width:5%"></td>
					<td><select ng-model="movie" data-ng-options="x for x in
lismovies" class="form-control" ng-change="moviesel()"></select></td>
				</tr>
				<tr ng-show="movie!=null">
					<td><label>Date</label></td>
					<td style="width:5%"></td>
					<td><input type="date"  id="depdate" data-ng-model="date"
class="form-control"></td>
					<td style="width:5%"></td>
					<td><button type="button" data-ng-click="dateselect()"
class="btn btn-success">Search</button></td>
				</tr>
				<tr ng-show="listheatre.length>0">
					<td><label>Theatres</label></td>
					<td style="width:5%"></td>
					<td><select ng-model="theatre"  data-ng-options="x for x in
listheatre" class="form-control" ng-change="gettimings()"></select></td>
				</tr>
				<tr ng-show="listiming.length>0">
					<td><label>Show Timing</label></td>
					<td style="width:5%"></td>
					<td><select ng-model="timing"  data-ng-options="x.starttime for
x in listiming" class="form-control" data-ng-change="getseats()"></select></td>
				</tr>
			</tbody>
		</table>
 <br>
 </form>
 <div class="custbook1" ng-show="rows.length>0">
		<table>
		<tr data-ng-repeat="m2 in rows">
```

```
            <td data-ng-repeat="m3 in m2"><div class="custbook2" style="background-
color:{{m3.color}};" data-ng-click="selectseat(m3)"></div></td>
            </tr>
            </table>
</div>
<br>
<h1 class="clear1">&nbsp;</h1>
  <button type="button" data-ng-click="save()" class="btn btn-success" style="margin-left:5%" ng-
show="rows.length>0">Submit</button>

  <br>
  <br>
  <button type="button" data-toggle="modal" data-target="#myModal" id="id1" ng-
show="false">Alert</button>
 <div class="modal fade" id="myModal" role="dialog" data-backdrop="static" data-keyboard="true">
    <div class="modal-dialog modal-dialog-centered">
      <div class="modal-content">
        <div class="modal-header">
          <h4 class="modal-title d-flex justify-content-center">{{alertmessage}}</h4>
        </div>
        <div class="modal-footer d-flex justify-content-center">
          <button type="button" class="btn btn-default" data-dismiss="modal">Close</button>
          <button type="button" class="btn btn-default" data-dismiss="modal">OK</button>
        </div>
      </div>
    </div>
  </div>
</div>
</body>
</html>
```

--Booking.js--

```
"use strict"
var myApp = angular.module('myApp',[]);
myApp.controller('Booking',Booking);
function Booking($scope,$http)
{
            $scope.lismovies=[];
            $scope.getlist=function()
            {
                        $http({
               method : "GET",
                 url : "getactivemovies",
              }).then(function mySuccess(response) {
```

```
 console.log(response.data);
 if(response.data[0]=="Failed")
 {
         $scope.alert(response.data[1]);
 }
 else
 {
         $scope.lismovies=response.data.slice(1);
 }
 }, function myError(response) {
 $scope.alert("Something went wrong.Please try Again Later");
});
}

$scope.moviesel=function()
{
        $scope.rows=[];
        $scope.timing=null;
        $scope.listiming=[];
        $scope.theatre=null;
        $scope.listheatre=[];
        $scope.date=null;
}
$scope.dateselect=function()
{
        $scope.rows=[];
        $scope.timing=null;
        $scope.listiming=[];
        $scope.theatre=null;
        $scope.listheatre=[];
        if($scope.date==null)
        {
                $scope.alert("Please select a date");
                return;
        }
        var d1 = new Date();
        var year1 = d1.getFullYear().toString();
        var month1 = (d1.getMonth()+1).toString();
if(month1.length==1)
{
                month1="0".concat(month1);
        }
var day1 = d1.getDate().toString();
if(day1.length==1)
{
         day1="0".concat(day1);
}
$scope.currentdate=year1.concat("-").concat(month1).concat("-").concat(day1);
```

```
            var d=$scope.date;
            year1 = d.getFullYear().toString();
            month1 = (d.getMonth()+1).toString();
if(month1.length==1)
{
                        month1="0".concat(month1);
            }
day1 = d.getDate().toString();
if(day1.length==1)
{
             day1="0".concat(day1);
}
$scope.date1=year1.concat("-").concat(month1).concat("-").concat(day1);
if($scope.date1<$scope.currentdate)
{
      $scope.alert("Past Date cannot be selected");
      $scope.date=null;
      return;
            }
            else
            {
                        if($scope.date1==$scope.currentdate)
                        {
                                    var h1=d1.getHours().toString();
                              if(h1.length==1)
                              {
                                                      h1="0".concat(h1);
                                    }
                              var m1=d1.getMinutes().toString();
                              if(m1.length==1)
                              {
                                                      m1="0".concat(m1);
                                    }
                                    var s1=d1.getSeconds().toString();
                                    if(s1.length==1)
                                    {
                                                      s1="0".concat(s1);
                                    }
                                    $scope.time1=h1.concat(":").concat(m1).concat(":").concat(s1);
                        }
                        else
                        {
                                    $scope.time1="00:00:00";
                        }
            }
$scope.gettheatres();
}
```

```
$scope.gettheatres=function()
{
	$scope.reqpojo={};
	$scope.reqpojo.moviename=$scope.movie;
	$scope.reqpojo.date=$scope.date1;
	$scope.reqpojo.time=$scope.time1;
	$http({
    method : "POST",
      url : "gettheatres",
      data: $scope.reqpojo
  }).then(function mySuccess(response) {
   if(response.data=="")
   {
	$scope.alert("No theatre Found for the selected movie and date");
   }
   else
   {
	 $scope.listheatre=response.data;
	}
  }, function myError(response) {
  $scope.alert("Something went wrong.Please try Again Later");
  });
}

$scope.gettimings=function()
{
	$scope.rows=[];
	$scope.timing=null;
	$scope.listiming=[];
	$scope.timingreq={};
	$scope.timingreq.reqpojo=$scope.reqpojo;
	$scope.timingreq.thname=$scope.theatre;
	$http({
    method : "POST",
      url : "fetchtheatresbymovieanddate",
      data: $scope.timingreq
  }).then(function mySuccess(response) {
   if(response.data=="")
   {
	$scope.alert("No Show Available");
   }
   else
   {
	 $scope.listiming=response.data;
	}
  }, function myError(response) {
  $scope.alert("Something went wrong.Please try Again Later");
  });
```

```
}

$scope.alert=function(val)
{
$scope.alertmessage=val;
document.getElementById("id1").click();
}

$scope.getseats=function()
{
		$scope.rows=[];
		$http({
  method : "POST",
    url : "fetchseatdtls",
    data: $scope.timing
 }).then(function mySuccess(response) {
 $scope.rows=[];
 var ct=0;
 $scope.items=[];
 for(var i=0;i<response.data.length;i++)
 {
		var obj={};
		obj.seat=response.data[i].seatno;
		if(response.data[i].status=="Y")
		{
			obj.color="grey";
			obj.status="A";
		}
		else
		{
			obj.color="white";
			obj.status="U";
		}
		$scope.items.push(obj);
		ct+=1;
		if(ct==7)
		{
			ct=0;
			$scope.rows.push($scope.items);
			$scope.items=[];
		}
 }
 if($scope.items.length>0)
 {
		$scope.rows.push($scope.items);
 }
 }, function myError(response) {
 $scope.alert("Something went wrong.Please try Again Later");
```

```
  });
}

$scope.selectseat=function(obj)
{
	if(obj.color=="grey")
	{
		return;
	}
	else
	{
		if(obj.color=="white")
		{
			obj.color="red";
			obj.status="O";
		}
		else
		{
			obj.color="white";
			obj.status="U";
		}
	}
}

$scope.save=function()
{
	$scope.seatslis=[];
	for(var i=0;i<$scope.rows.length;i++)
	{
		for(var j=0;j<$scope.rows[i].length;j++)
		{
			if($scope.rows[i][j].status=="O")
			{
				$scope.seatslis.push($scope.rows[i][j].seat);
			}
		}
	}
	if($scope.seatslis.length==0)
	{
		$scope.alert("No Seat Selected");
		return;
	}
	$scope.pojo={};
	$scope.pojo.seats=$scope.seatslis;
	$scope.pojo.movieid=$scope.timing.id;
	$http({
   method : "POST",
     url : "savedetails",
                data: $scope.pojo
            }).then(function mySuccess(response) {
                    if(response.data[0]=="Failed")
                    {
                            if(response.data[1]=="Booking Closed for this show")
                            {
                                    $scope.alert(response.data[1]);
                                    $scope.rows=[];
                                    $scope.timing=null;
                                    $scope.listiming=[];
                                    $scope.theatre=null;
                            }
                            else
                            {
                                    $scope.alert(response.data[1]);
                                    $scope.getseats();
                            }
                    }
                    else
                    {
                            $scope.alert(response.data[1]);
                            $scope.getseats();
                    }

            }, function myError(response) {
            $scope.alert("Something went wrong.Please try Again Later");
            });
           }

}
```

--stylesheet.css--

```
.custbook1
{
  background-color:#6CB4EE;
  float:left;
  padding:20px;
  margin-left:5%;
  margin-right:5%;
  text-align:center;
  border:2px ridge black;
}
.custbook2
{
  width:50px;
  height:50px;
  float:left;
  margin:1px;
}
```

CONCLUSION

In conclusion, the Java Spring Boot-based movie ticket booking is a powerful and user-centric solution that simplifies the seat reservation process for cinema enthusiasts. Its well-structured three-layer architecture ensures efficient data flow, comprising controllers, services, and data access objects (DAOs).

Users can easily search for movies based on various parameters, such as movie name, city, and date, and access comprehensive show information along with available seat listings. The application excels in providing real-time seat reservations, enabling users to select their preferred seats conveniently. The booking process is designed with a focus on seat availability and overall booking status, ensuring a smooth and secure experience for users.

Leveraging Java Spring Boot, the application benefits from rapid development, robust security measures, and scalability, making it an ideal choice for the ever-evolving world of movie ticket bookings. Overall, it offers a seamless and user-friendly platform for movie enthusiasts to reserve their seats and enjoy a hassle-free cinema experience.

FUTURE WORK

The future scope of the Java Spring Boot-based CIneMates :movie ticket booking includes:

1. Improved Frontend: Enhancing the user interface with modern web technologies and responsive design.
2. Mobile App Development: Expanding to Android and iOS platforms for wider accessibility.
3. User Support and Feedback: Implementing real-time support and feedback systems for better customer service.
4. Performance Optimization: Focusing on database and algorithm improvements for efficiency.
5. Personalization: Offering user profiles, recommendations, and loyalty programs.
6. Enhanced Security: Continuously updating security measures.
7. Social Media Integration: Allowing users to share bookings and reviews for viral marketing.
8. Event Booking: Extending the platform to book live events and other entertainment.
9. Accessibility Features: Ensuring accessibility for users with disabilities
10. Community and Social Features: Creating a user community for interactions.

Adapting to these changes is crucial to remain competitive in the online movie ticket booking industry.