



Dharmsinh Desai University, Nadiad

Faculty of Technology

Department of Computer Engineering

B.Tech. CE

Semester – V

Subject : (CE – 520) Advanced Technologies

Project Title: Movie Ticket Booking Application

Created By:(Group 5)

1) Bhagatji Maharshi- CE006 – 21CEUOG067

2) Chauhan Shyam- CE016 – 21CEUOS005

Guided by: Prof. Siddharth P. Shah

Dharamsinh Desai University,Nadiad
Faculty of Technology
Department of Computer Engineering

CERTIFICATE

This is to certify that the practical / term work carried out in the subject of **Advanced Technology** and Recorded in this Journal is the bonafide work of

Maharshi Bhagatji(CE006) (21CEUOG067)
Chauhan Shyam(CE016) (21CEUOS005)

Of B.Tech semester V in the branch of computer Engineering during the academic year 2023-2024

Prof. Siddharth P. Shah
Dept. of Computer Engineering
Faculty of Technology

Dr. C.K. Bhensdadia
Head of Dept. of Computer engg.
Faculty of Technology

Table of Contents

Abstract	6
Introduction	7
Technology/platform/Tools used	7
Software Requirements Specification(SRS)	9
Introduction	9
Overall Description	11
External Interface Requirements	13
Overview of Functional Requirements	14
Description of Functional Requirements	14
Non Functional Requirements	17
Database Design	18
Implementation Details	20
Modules and their description	20
Testing	25
Screen-shots	27
Conclusion	30
Limitation and Future Extension	31
Bibliography	32

Abstract

The abstract of this project is to create a website that allows users to book a movie of their interest. This project aims to provide a seamless and efficient solution for both moviegoers and cinema operators alike.

Key features of the system user registration and authentication enabling secure account creation and login and user friendly interface for browsing movies. Movie information including posters and descriptions about movies is available for user convenience.

User profile enables customers to manage their information and view their booking history, with options for canceling bookings. On the cinema operator's side, the admin dashboard allows management of movies.

In conclusion, The Movie Ticket Booking System aims to provide the best way to enhance the experience for moviegoers by providing a user friendly, efficient and convenient platform for booking tickets.

Introduction

Technology/Platform/Tools used

- **Database: MongoDB**

- An NOSQL based database aimed to provide scalability, simplicity, less code and easy maintenance.

- **Backend: Node.js**

- A javascript framework aimed to provide fast delivery, scalability, cross platform support, community support And ease of adoption.
- Used with express.js to provide simplicity, flexibility and minimalism.

- **Fronted: React.js**

- A javascript framework aimed to provide ease of creating dynamic web applications, reusable components and performance enhancement.
- Along with that **Material UI** is used to design different pages in front-end like
 - Icons
 - Buttons
 - Text-fields
 - Cards

- **Development Tools**

- Used Visual Studio Code for editing code of projects.
- Used github for better collaboration of the team.
- Used env variables for securing and using crucial information of the website.
- Used **postman** API client for testing backend apis.

- **Miscellaneous**

- **Axios** – Axios is a JavaScript library used for making HTTP requests from the frontend or backend of a web application. It simplifies the process of sending asynchronous HTTP requests to interact with APIs, retrieve data from servers, and send data to servers.
- **Redux** – React Redux is a library that provides a state management solution for React applications. It is often used in conjunction with React to manage the state of a complex application more efficiently.
- Used **Mongoose** package for providing models for database access.
- Used **bcrypt** for encryption of password of users.

Software Requirement Specifications (SRS)

1. Introduction

1.1 Purpose

- The purpose of this document is to present a detailed description of the Movie Ticket Booking System . It will explain the purpose and features of the system, the interfaces of the system, what the system will do, the constraints under which it must operate and how the system will react to external stimuli. This document is intended for both the stakeholders and the developers of the system.

1.2 Intended Audience and Reading Suggestions

- Intended audience covers individuals who are interested in watching different types of movies. This could include people working in different Industries, students studying in schools and colleges.

1.3 Product Scope

- The scope of this project is to develop a movie ticket booking web application that allows users to search different movies , manage their booking . It will have a user-friendly interface that allows users to easily book the movie ticket. It will also have a registration and login system to provide a secure and personalized experience.

1.4 References

- IEEE. IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications. IEEE Computer Society, 1998.

2. OverAll Description

2.1 Product Perspective

- There are many similar software like this but the way this is different from those is that this provides a user friendly experience with the simple UI, quick and easy product search and easily book or cancel the ticket of the movie.

2.2 Product Functions:

Functionality of the system:

Admin:

- Login & Logout
- Add a new movie

Users:

- Signup
- Login & Logout
- View the movies
- Search the desired movie
- Book the ticket
- Cancel the ticket

2.3 User Classes and Characteristics

- **Normal user** : normal user is expected to invoke the

functionalities such as login, search,book the ticket and cancel the ticket.

- **Super user** : Super user is expected to invoke the functionalities such as admin login,add the movie.

2.4 Operating Environment

- The system can easily operate on all popular web browsers like Chrome, Firefox, Safari. Computer with better internet connection is required for the use of the system.

2.5 Assumption and Dependencies

Assumption

- The system will require an active internet connection for booking or canceling the ticket.
- The software will assume that the user's device is running a compatible operating system, as mentioned in the operating Environment section.

Dependencies

- The system will rely on server and database to store user's data and for Hosting the software.

3. External Interface Requirements

3.1 Hardware Interface:

- The system is compatible with a wide range of hardware, including desktop computers, laptops, and mobile devices. The system will be able to run on popular operating systems such as Windows, MacOS, and Linux.

3.2 Software Interface:

- The system uses Ract for frontend development, MongoDB for storing the data and Node-JS Express is used for backend.

3.3 Communication Interfaces

- This uses standard network protocols, such as HTTPS. The system will comply with industry standards for data encryption and secure communication to ensure that customer information and orders are protected from unauthorized access or attack.

4.System Feature

Overview of functional requirements

R.1 admin login

R.2 movie management(adding new movie)

R.3 User registration and login

R.3.1 User Login

R.3.2 User Registration

R.4 Movie booking

R.5 Cancel a booking

Description of Functional Requirements

R.1 : admin login

Input : enter email and password

process: check if the credentials are correct or not.

Output : redirected to home page.

R.2: Movie management

Input: title of new movie,description,release date, photoUrl

Process: validation and processing of input data

Output: added movie will be displayed on home page.

R.3 User Registration and Login :

Description : Users should be able to create an account and login to the application.

R.3.1 For Login of Users :

Input : Email-id and passwords created by users.

Process : The system will validate the entered input and give appropriate feedback.

Output : If Email-id and password are correct, then displays "Home page" otherwise go to R.1.1 again.

R.3.2 For signup (New User) :

Input : Name, Email-id, password.

Process : The system will validate the input and give appropriate feedback.

Output : If the input details are correct, then new account will be created and redirected to the login page otherwise go to R.1.2

R.4 Movie Booking

Description: Users should be able to book a movie by selecting the desired movie and date.

Input: User will provide movie name, seat number and date

Process: System will book a ticket according to the input.

Output: Confirmation message will be displayed and redirected to the user profile page.

R.5 Cancel the ticket

Description: By clicking on the delete button, the user should be able to cancel the ticket.

Input: User will click on the delete button.

Process: The system will update the database

Output: Updated user profile will be displayed.

5. Non Functional Requirement :

5.1 Performance

- Performance can be increased by limiting the excess information like one phone number and only necessary information will be taken.

5.2 Easy to use

- User interface will be made from react and will be user friendly and easy to use.
- Users can access websites on all different devices with similar design. So, for them it is easy to use the website.

5.3 Maintenance

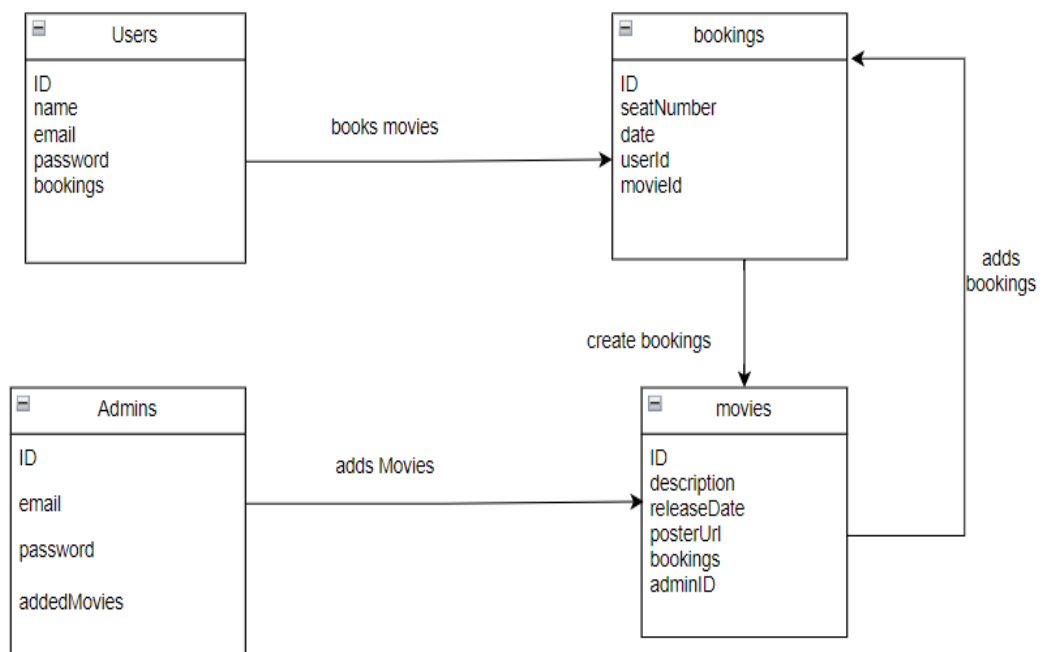
- Website information will be updated on a regular basis and new movies will be displayed on the top of our website.
- When user traffic increases day by day then try to maximize user traffic without any lagging.

5.4 Security Requirements

- The system should use encryption to protect sensitive data, such as user passwords.
- The system should have regular vulnerability assessments and penetration tests to identify and remediate any security weaknesses.

Database Design

Models Overview



- **Users**

- **name:** the name of the user
- **email:** the email of the user
- **password:** password of the user
- **bookings:** to store the booking details of the user

- **Admins**

- **email:** the email of the user
- **password:** the password of the admin
- **addedMovies:** to store the movies information which is added by the admin

- **Movies**

- **title:** name of the movie
- **description:** brief about the movie
- **Release-date:** release date of the movie
- **posterUrl:** image address of movie poster

- **Bookings**

- **date:** booking date
- **seatNumber:** seat number selected by user
- **user:** user_id of the user
- **movie:** movie_id of the movie

Implementation Details

User Management Module:

- **User Registration:**
 - Implement a registration form that collects user information, including username, email, and password.
 - The user's password is securely hashed using 'bcrypt' for storage in database.
 - If credentials are valid then redirect to the login page.
- **User Login:**
 - Implement a login form that accepts the user's email and password.
 - Verify the entered credentials against the stored, hashed password in the database.
 - Login details are stored in localStorage.
- **User Logout:**
 - By clicking on the logout button, user's information which is stored in localStorage is destroyed.

Admin Management Module:

- **Admin login:**

- Implement a login form that accepts the admin's email and password.
- Verify the entered credentials against the stored, hashed password in the database.
- If the credentials are correct, then JWT (JSON Web Token) is generated and stored in the localStorage.

- **Admin logout:**

- By clicking on the logout button, JWT is removed from the localStorage.

- **Add a movie:**

- First of all, verification of JWT token is done.
- Then, a new Movie object is created using the provided data from the request body.
- A database session is created using Mongoose, which allows for better control over database operations.
- A transaction is started within the session. It is used to ensure that both movie creation and admin information are either both successfully committed or rolled back in case of an error.

- The newly created movie is added to the 'addedMovies' array of the admin which keeps track of the information about which movie is added by which admin.
- At the last session's transaction is committed, which means all changes are saved to the database.

Booking Management

- **Book a ticket:**

- Before the booking process starts, it is checked if the user logged in or not. If the user is not logged in, then redirects to the login page.
- Otherwise, a new 'Bookings' object is created with the provided data, including movie, booking date, seat number.
- A database session is created using Mongoose, which allows for better control over database operations.
- Then a transaction is started and then the new booking is associated with the user and movie by adding it to their respective 'bookings' arrays.
- The changes made to the user, movie and new booking are saved within the context of the session, ensuring that the

changes are not immediately persisted to the database until the transaction is explicitly committed.

- Finally, the transaction is committed, which permanently saves all the changes to the database.

- **Delete a booking**

- By clicking on the delete button, it attempts to find and remove the booking with the specified 'booking id' from the database. The 'populate' method is used to retrieve additional data associated with the booking, like 'user' and 'movie.'
- A database session is created using Mongoose, which allows for better control over database operations.
- It removes the booking from the 'bookings' array of both the associated 'user' and 'movie'. This ensures that the booking is disassociated from both the user who made the booking and the movie for which the booking was made.
- After making the changes, the sessions' transaction is committed, saving the changes to the database.

User Profile:

- **Get all booking details**

- User clicks on the profile button, then all the bookings done by the user are shown.
- It extracts the 'id' (user id) from the request's URL parameters.
- Then, it attempts to find bookings in the database where the 'user' field matches the provided 'id'.
- In the query, `populate("movie")` is used to expand reference to the associated 'movie' document. This means that when bookings are fetched, they will include details about the associated movies.
- There is also a delete button, which will delete the booking. How it's done is described in the 'Delete a booking' section.

9) Testing

For testing our application, a mixed approach integration and regression testing is used.

- **Integration testing**

- Integration testing is performed by systematically combining individual components or modules of a software system and testing their interaction.
- It helps to identify and resolve some issues that may arise when these components are integrated.

- **Regression testing**

- Regression testing is a quality assurance process that involves retesting a software application with the same test cases after code changes or updates have been made.
- Main purpose is to confirm that recent modifications have not adversely affected the existing functionality.

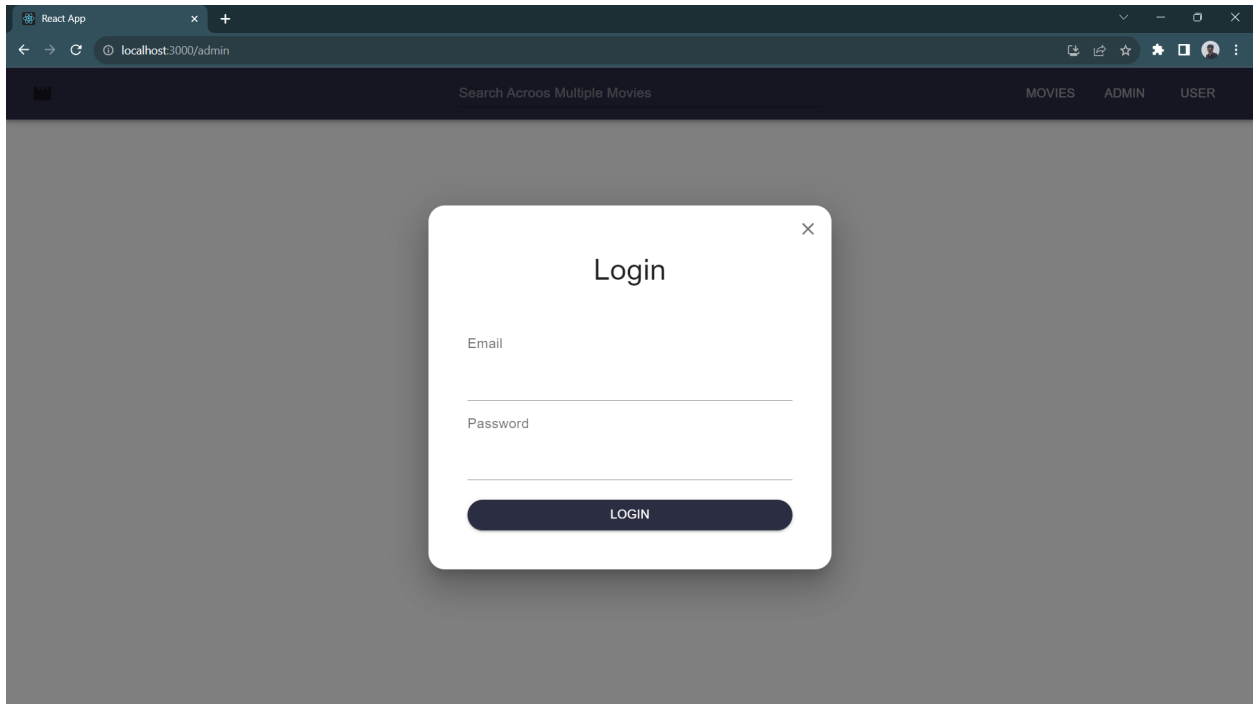
- **Manual testing**

- In manual testing, an interaction with the software application is done to evaluate its performance, functionality and user interface and also to find bug in the system.

Sr No.	Test Case	Expected Result	Actual Result	Status
1	Login	The user or admin should be able to login	The user or admin is able to login	Success
2	Register	The user should be able to create an account	The user is able to make an account	Success
3	Book a ticket	The user should be able to book a ticket	Ticket booking is done successfully	Success
4	Delete a ticket	Ticket should be deleted	Movie Ticket deleted successfully	Success
5	Add a new movie	Admin should be able to add new movie	Admin is able to add a new movie	Success

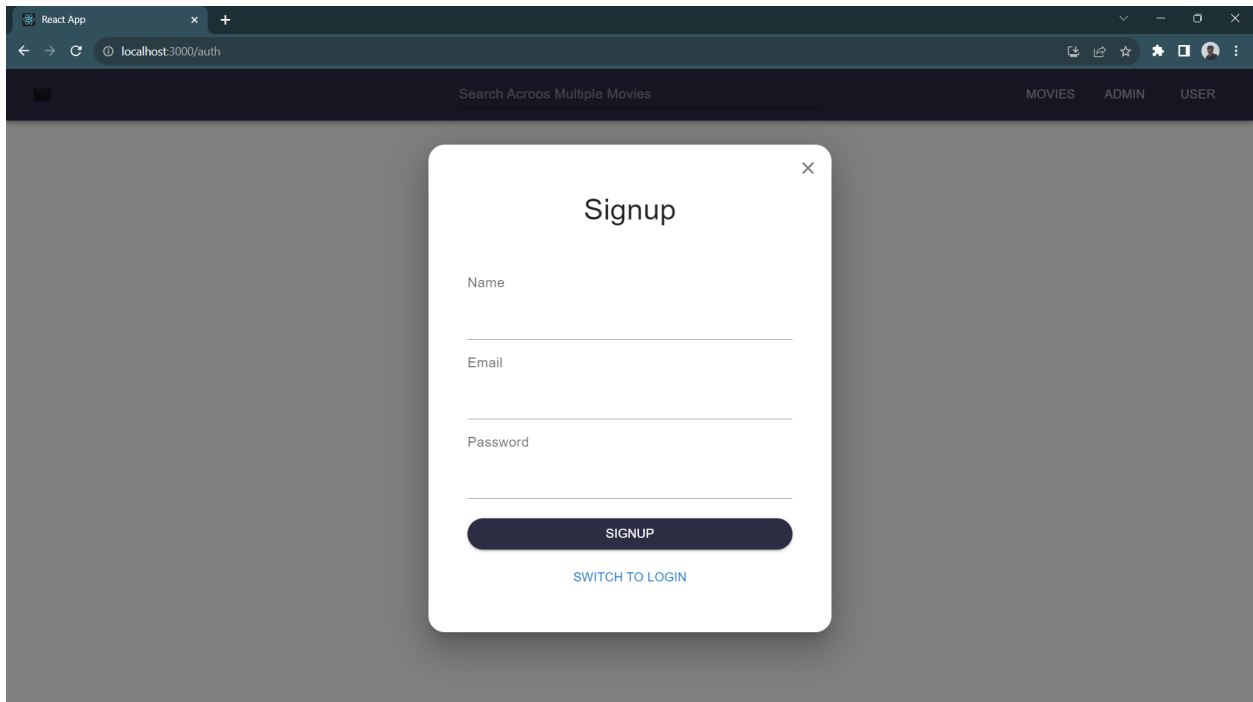
Screen-shots

Login Page



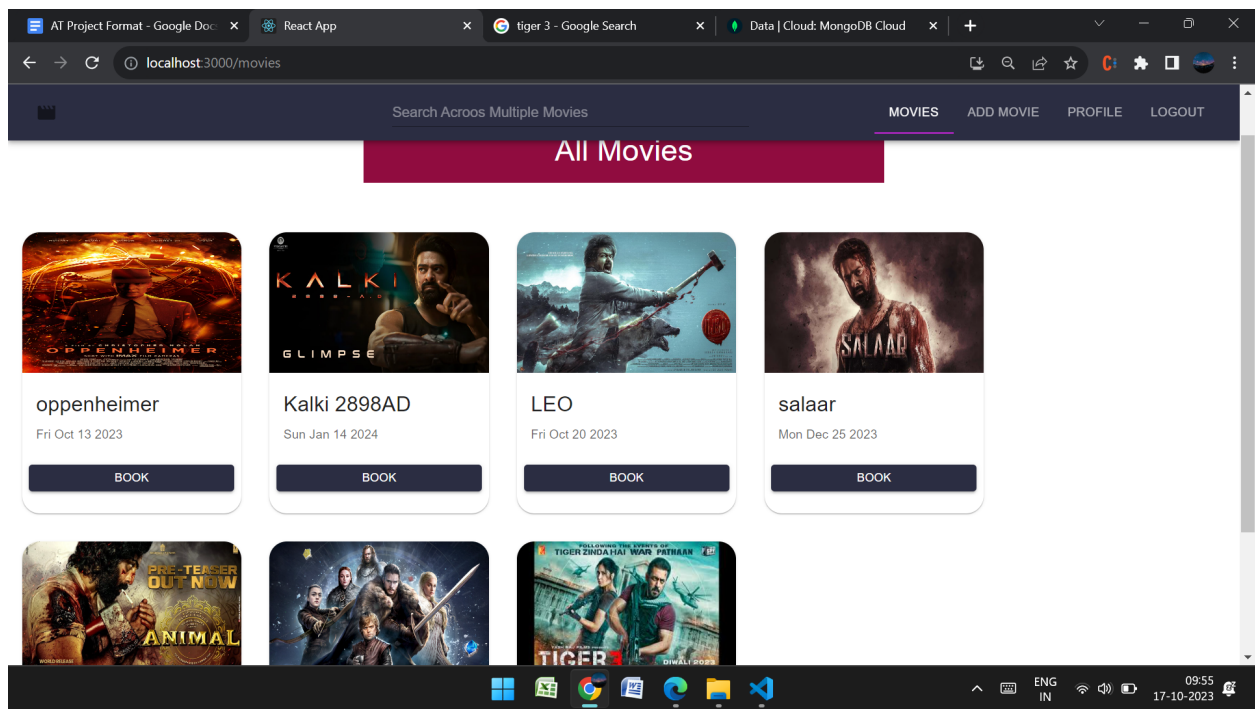
The screenshot shows a web browser window with the address bar displaying 'localhost:3000/admin'. The page has a dark header with the text 'Search Across Multiple Movies' and navigation links 'MOVIES', 'ADMIN', and 'USER'. A white modal box titled 'Login' is centered on the page. It contains two input fields labeled 'Email' and 'Password', and a dark blue button labeled 'LOGIN'.

Signup Page

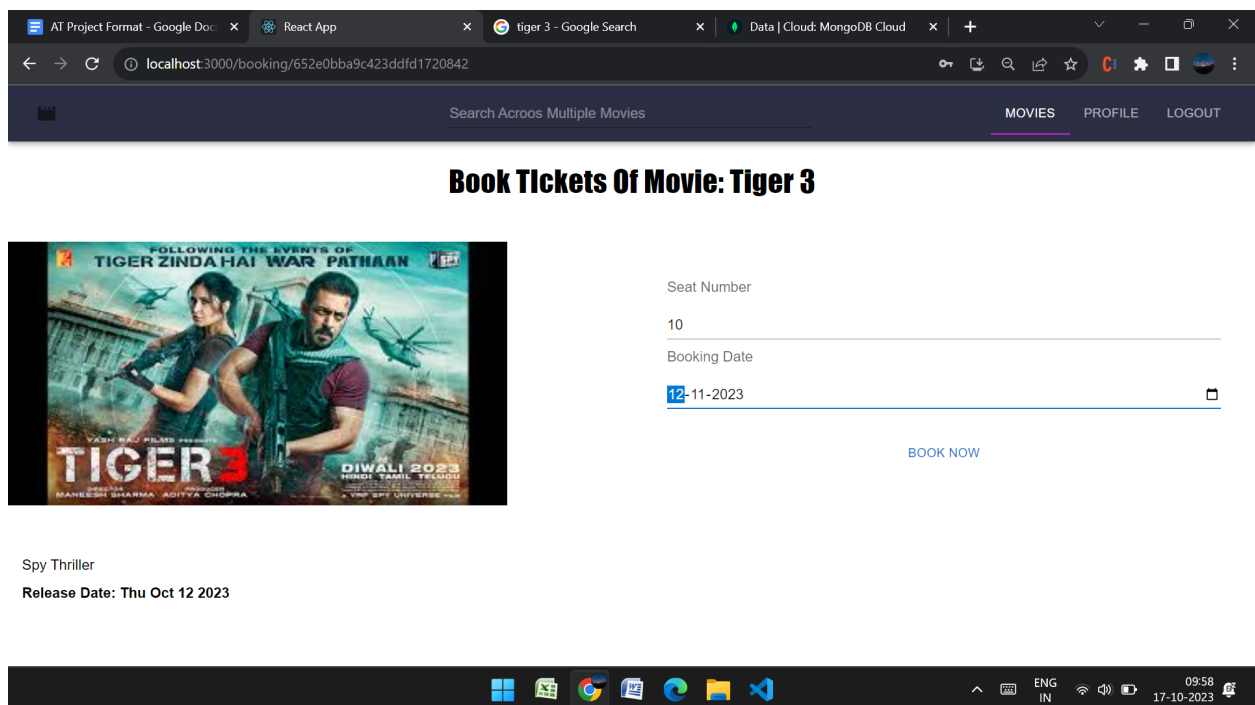


The screenshot shows a web browser window with the address bar displaying 'localhost:3000/auth'. The page has a dark header with the text 'Search Across Multiple Movies' and navigation links 'MOVIES', 'ADMIN', and 'USER'. A white modal box titled 'Signup' is centered on the page. It contains three input fields labeled 'Name', 'Email', and 'Password', a dark blue button labeled 'SIGNUP', and a link labeled 'SWITCH TO LOGIN'.

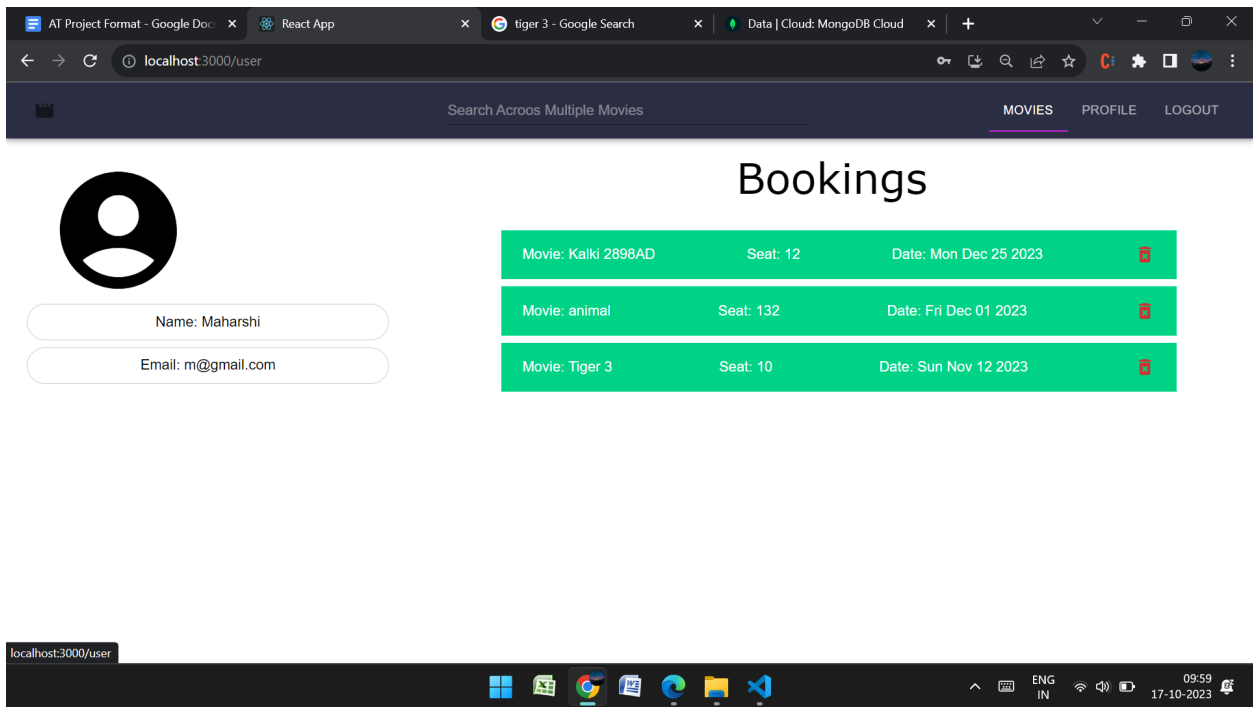
Movies Page



Movie Booking Page



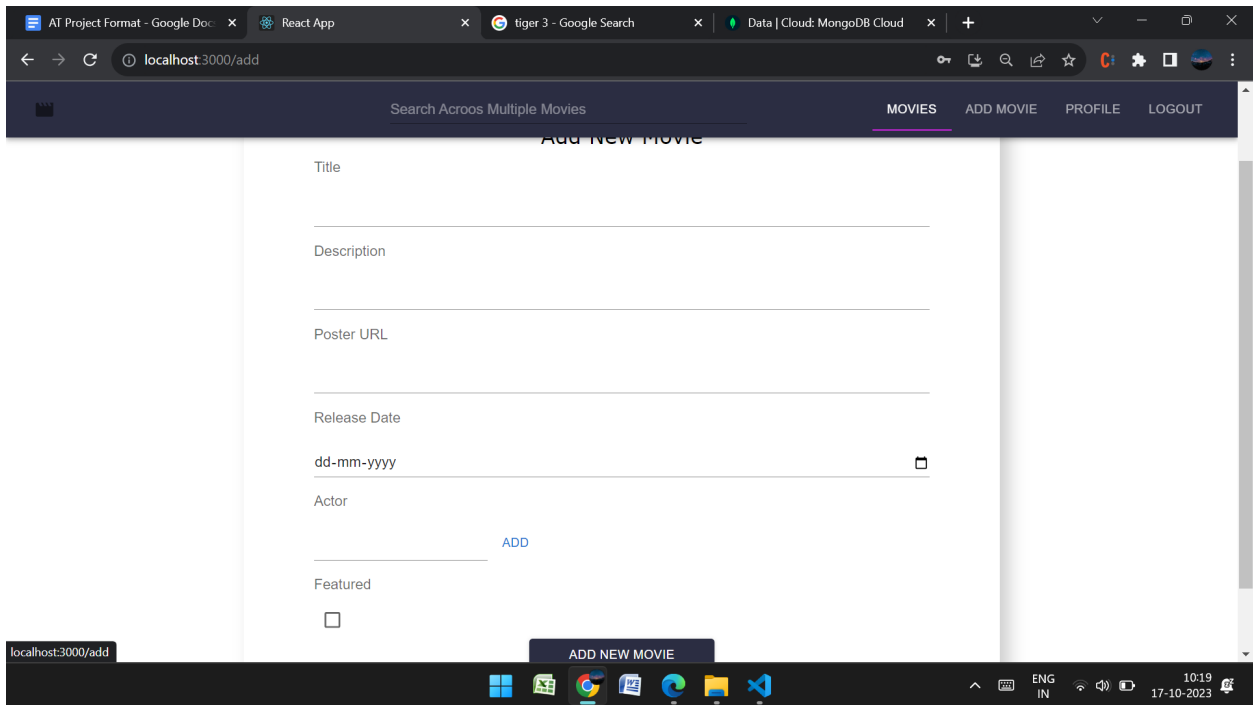
User Profile Page



The screenshot shows a web browser at localhost:3000/user. The page has a dark header with a search bar and navigation links: MOVIES, PROFILE, and LOGOUT. The PROFILE link is active. On the left, there is a user profile section with a circular placeholder for a profile picture, a name field containing 'Name: Maharshi', and an email field containing 'Email: m@gmail.com'. On the right, there is a section titled 'Bookings' which contains a table of movie bookings.

Movie	Seat	Date	Action
Kalki 2898AD	Seat: 12	Date: Mon Dec 25 2023	
animal	Seat: 132	Date: Fri Dec 01 2023	
Tiger 3	Seat: 10	Date: Sun Nov 12 2023	

Add a movie page(Admin)



The screenshot shows a web browser at localhost:3000/add. The page has a dark header with a search bar and navigation links: MOVIES, ADD MOVIE, PROFILE, and LOGOUT. The ADD MOVIE link is active. The main content area is titled 'Add New Movie' and contains a form with the following fields: Title, Description, Poster URL, Release Date (with a date picker set to dd-mm-yyyy), Actor, and a Featured checkbox. An 'ADD' button is located at the bottom right of the form.

11) Conclusion

Hereby, we conclude that we have successfully implemented the User Management Module, Admin Management Module, Booking Management Module and User Profile Module.

New User can create his/ her account. The users who already have an account can log in. Users can browse through different movies. Authenticated users can book a ticket of their desired movie. Users can also browse through their booking history. Users can also cancel their booking.

Authenticated admin will be able to add a new movie in the system. Admin profile is also maintained to keep track of which are the movies added by the particular admin.

12) Limitation and Future Extension

- **Limitation**

- Currently users are not able to visualize the seating arrangement because there is not a functionality of interactive seat selection with real time availability updates.
- Users are only able to book a single ticket, multiple ticket booking is not allowed.
- The users are not able to give feedback about the movie or feedback about the facilities provided by Cinema Authority.

- **Future Extension**

- There should be an interactive seat selection interface with real time availability updates for better visualization of seat selection.
- Users should be allowed to book multiple tickets.
- Users can rate and review the movie and can give feedback about the facilities provided by Cinema Authority and Cinema Authority can respond to that feedback.
- There should be a good error handling mechanism so that if users are not getting desired output then they can identify what is error.

13) Bibliography

<https://react.dev/>

<https://www.mongodb.com/>

<https://expressjs.com/>

<https://mui.com/material-ui/>

<https://www.npmjs.com/package/axios>