Project Title

Movie Mate: Where Every Seat Tells A Store

Introduction:-

Project Title:-

Online-Movie Tickets (Movie Mate)

Team members members and their roles:

- Injamuri Nathaniyelu (218x1a0528)
- Kamjula Vikas (218x1a0528563)
- Bathula Eswar (218x1a0504)
- Duvada Teja (218x1a0548)

Project Overview :-

Purpose or objectives and goals:

Purpose :-

The purpose of an Online Movie Booking System is to provide a convenient and efficient platform for users to browse movies, check showtimes, book tickets, and make payments seamlessly from anywhere. This system aims to eliminate the need for physical visits to cinema halls, reduce wait times, and offer a hassle-free ticket booking experience.

Objectives :-

Ease of Access: Provide a user-friendly interface that allows customers to search for movies, view showtimes, and book tickets online effortlessly.

Real-Time Availability: Ensure real-time access to movie schedules, ticket availability, and seat selection across multiple theaters.

User Registration and Profiles: Allow users to register and maintain personal profiles, view booking history, and receive notifications for upcoming movies or special offers.

Admin Management: Provide administrative access to cinema managers to manage movie listings, showtimes, pricing, and seat allocations.

Feedback and Reviews: Enable users to provide feedback or reviews on movies and cinema services.

Goals :-

Enhance User Convenience: Offer a smooth and convenient way to book movie tickets without physically visiting a theater.

Improve Efficiency: Streamline the booking process by reducing time spent on purchasing tickets and eliminating the need for manual handling of transactions.

Increase Revenue: Encourage more bookings by providing promotional discounts, loyalty programs, and exclusive online deals, thereby increasing overall cinema revenue.

Features:-

User Registration and Login:

Sign-Up/Sign-In: Allows users to create accounts with email or social media credentials.

User Profiles: Personalized profiles where users can store their preferences, booking history, and payment details.

Password Recovery: Option to recover passwords through email or SMS.

Movie Browsing:

Search Movies: Users can search for movies based on genre, language, title, and rating.

Upcoming Movies Section: View trailers, details, and release dates of upcoming movies.

Filter Options: Filters by showtime, cinema location, and movie ratings.

Detailed Movie Info: Movie descriptions, cast and crew details, trailers, runtime, and IMDb ratings.

Showtime and Theater Selection:

Theater Listings: Users can view available theaters nearby and select from multiple showtimes.

Interactive Seat Map: A real-time seat map showing available, reserved, and booked seats for selection.

Multiple Theater Chains: Ability to book tickets across different cinema chains.

Ticket Booking and Seat Reservation:

Flexible Seat Selection: Interactive seating arrangement to pick preferred seats.

Multiple Ticket Types: Selection of regular, premium, or VIP seat categories.

Group Booking: Allows users to book multiple seats together for groups or families.

Online Payment Integration:

Multiple Payment Methods: Secure payment gateway integration supporting credit/debit cards, digital wallets, net banking, and UPI.

Coupon and Promo Codes: Option to apply discount codes or loyalty points for reduced ticket prices.

Payment Confirmation: Real-time booking confirmation sent via email or SMS.

Booking History and E-Ticketing:

Booking History: Users can view and manage their past bookings, including cancellations or changes.

E-Tickets: Digital ticket generation with a QR code or barcode for easy scanning at theaters.

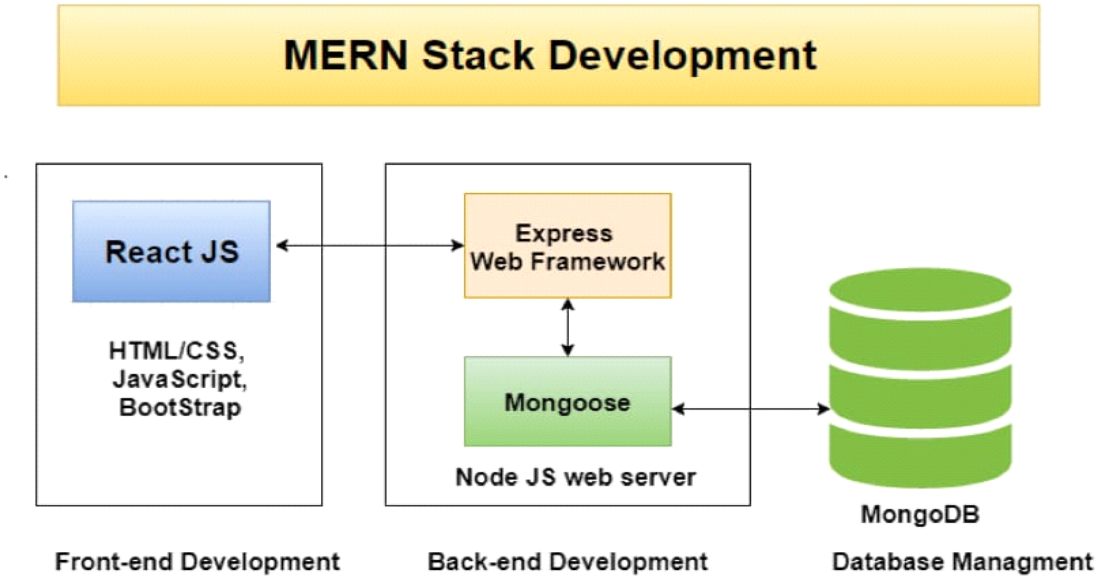
Downloadable/Printable Tickets: Users can download and print their tickets if needed.

Architecture :-

Frontend architecture involves designing the user interface, ensuring smooth interactions, responsiveness, and a seamless user experience using technologies like HTML, CSS, and JavaScript frameworks (e.g., React, Angular, Vue.js).

Backend architecture focuses on the server-side logic, handling data processing, business logic, authentication, and communication with the frontend through APIs. It uses frameworks like Node.js, Express, Django, or Spring.

Database architecture includes the design of structured or unstructured databases (SQL/NoSQL) that store and retrieve data efficiently, ensuring smooth interaction with the backend. This involves tools like MySQL, MongoDB, or PostgreSQL.



Setup Instructions :-

Pre-Requisite

To develop a full-stack food ordering app using React JS, Node.js, and MongoDB, there are several prerequisites you should consider. Here are the key prerequisites for developing such an application:

Installation

Node.js and npm:

Install Node.js, which includes npm (Node Package Manager), on your development machine. Node.js is required to run JavaScript on the server side.

• Download: https://nodejs.org/en/download/

• Installation instructions: https://nodejs.org/en/download/package-manager/

MongoDB: Set up a MongoDB database to store hotel and booking information. Install MongoDB locally or use a cloud-based MongoDB service.

• Download: https://www.mongodb.com/try/download/community

• Installation instructions: https://docs.mongodb.com/manual/installation/

Express.js: Express.js is a web application framework for Node.js. Install Express.js to handle server-side routing,middleware, and API development.

• Installation: Open your command prompt or terminal and run the following command: **npm install express**

React.js: React.js is a popular JavaScript library for building user interfaces. It enables developers to create interactive and reusable UI components, making it easier to build dynamic and responsive web applications. To install React.js, a JavaScript library for building user interfaces, follow the installation guide: https://reactjs.org/docs/create-a-new-react-app.html

HTML, CSS, and JavaScript: Basic knowledge of HTML for creating the structure of your app, CSS for styling,and JavaScript for client-side interactivity is essential.

Database Connectivity: Use a MongoDB driver or an Object-Document Mapping (ODM) library like Mongoose to connect your Node.js server with the MongoDB database and perform CRUD (Create, Read, Update, Delete) operations.

Front-end Framework: Utilize Angular to build the user-facing part of the application, including product listings, booking forms, and user interfaces for the admin dashboard.

Version Control: Use Git for version control, enabling collaboration and tracking changes throughout the development process. Platforms like GitHub or Bitbucket can host your repository.

• Git: Download and installation instructions can be found at: https://gitscm.com/downloads

Development Environment: Choose a code editor or Integrated Development Environment (IDE) that suits your preferences, such as Visual Studio Code, Sublime Text, or WebStorm.

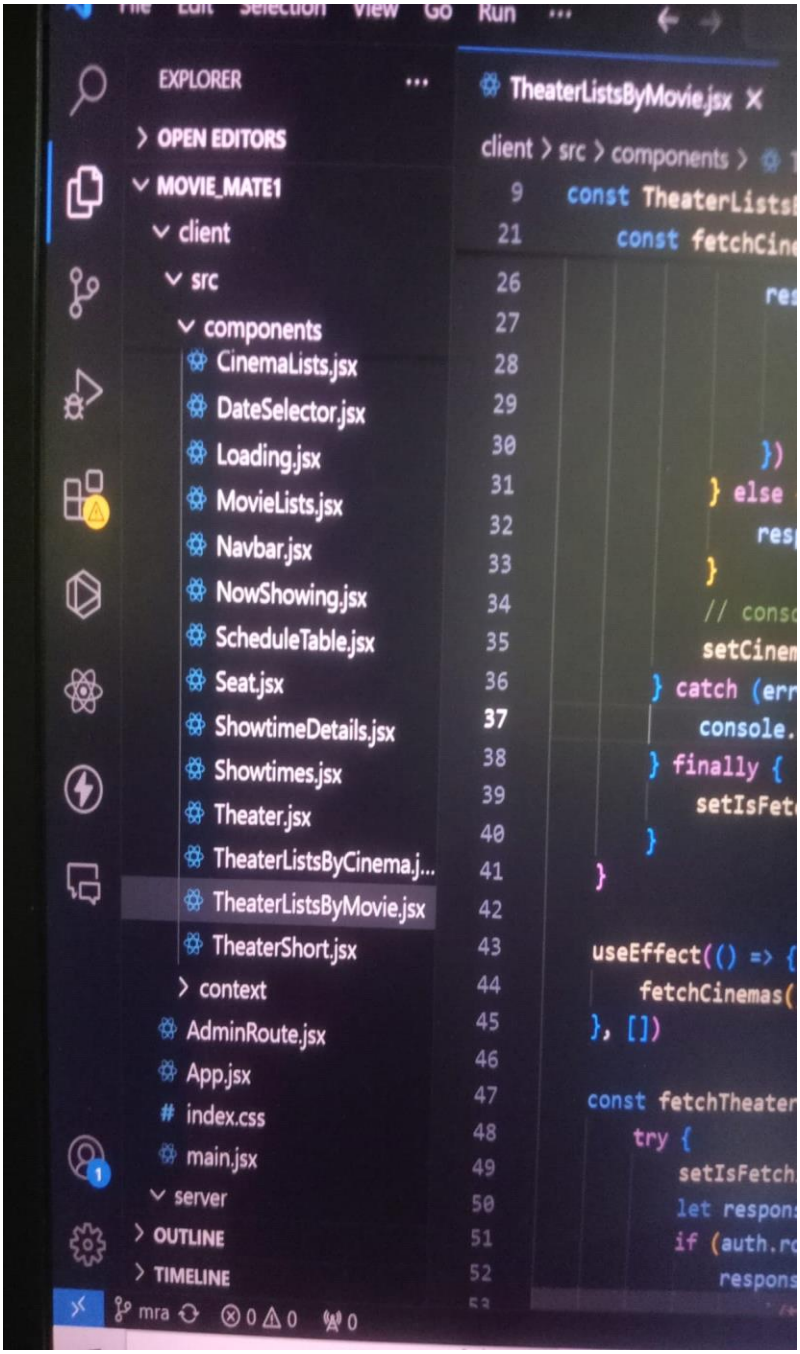
• Visual Studio Code: Download from https://code.visualstudio.com/download

• Sublime Text: Download from https://www.sublimetext.com/download

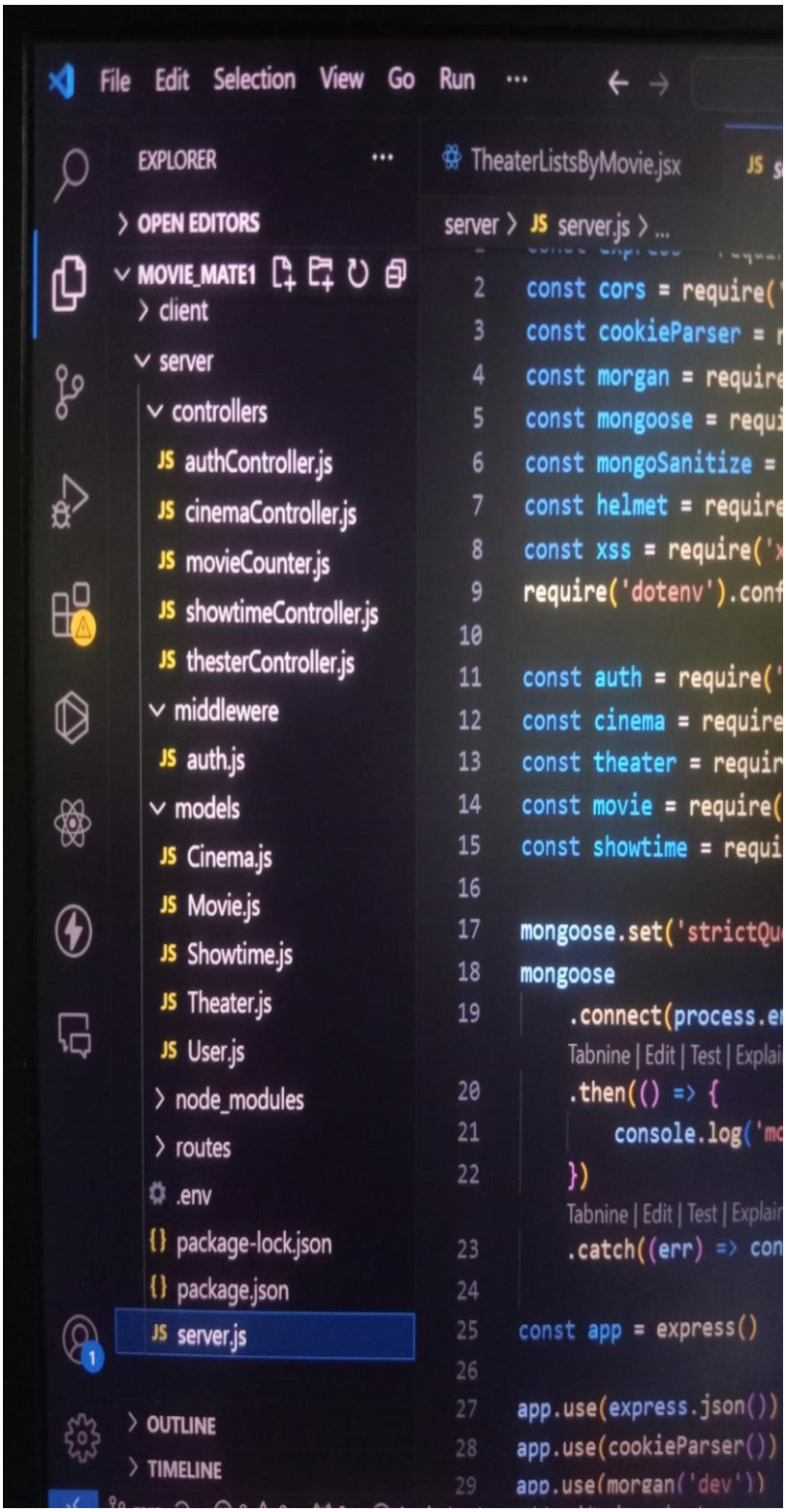
• WebStorm: Download from https://www.jetbrains.com/webstorm/download

Folder Structure :-

- Frontend


EXPLORER
OPEN EDITORS
MOVIE_MATE1
client
src
components
CinemaLists.jsx
DateSelector.jsx
Loading.jsx
MovieLists.jsx
Navbar.jsx
NowShowing.jsx
ScheduleTable.jsx
Seat.jsx
ShowtimeDetails.jsx
Showtimes.jsx
Theater.jsx
TheaterListsByCinema.j...
TheaterListsByMovie.jsx
TheaterShort.jsx
context
AdminRoute.jsx
App.jsx
index.css
main.jsx
server
OUTLINE
TIMELINE
TheaterListsByMovie.jsx
client > src > components >
const TheaterLists
const fetchCine
} else
} catch (err
console.
} finally {
setIsFet
}
useEffect(() => {
fetchCinemas(
}, [])
const fetchTheater
try {
setIsFetch
let respons
if (auth.r
respons

- Backend


File Edit Selection View Go Run ···
EXPLORER
TheaterListsByMovie.jsx
OPEN EDITORS
server > server.js > ...
MOVIE_MATE1
client
server
controllers
authController.js
cinemaController.js
movieCounter.js
showtimeController.js
thesterController.js
middlewere
auth.js
models
Cinema.js
Movie.js
Showtime.js
Theater.js
User.js
node_modules
routes
.env
package-lock.json
package.json
server.js
OUTLINE
TIMELINE
const cors = require(
const cookieParser =
const morgan = require
const mongoose = requi
const mongoSanitize =
const helmet = require
const xss = require('
require('dotenv').conf
const auth = require('
const cinema = require
const theater = requir
const movie = require(
const showtime = requi
mongoose.set('strictQu
mongoose
.connect(process.e
Tabnine | Edit | Test | Explai
.then(() => {
console.log('mo
})
Tabnine | Edit | Test | Explai
.catch((err) => con
const app = express()
app.use(express.json())
app.use(cookieParser())
app.use(morgan('dev'))

Running the Application :-

Frontend:

Navigate to the frontend directory (if separate from the backend).

Install necessary dependencies using a package manager like npm (for JavaScript/Node.js-based frameworks):

npm start

This runs the frontend application, usually on a development server like http://localhost:3000 (for React)

Backend:

Navigate to the backend directory.

Install backend dependencies (if using Node.js)

npm run dev (for development mode)

This typically runs the backend server on a different port, like http://localhost:5000.

API Documentation :-

router.post('/register', register)

router.post('/login', login)

router.get('/logout', logout)

router.get('/me', protect, getMe)

router.get('/tickets', protect, getTickets)

router.put('/user/:id', protect, updateUser)

router.get('/user', protect, authorize('admin'), getAll)

router.delete('/user/:id', protect, authorize('admin'), deleteUser)

Authentication :-

1. **User Credentials**: Users provide credentials, typically a username and password.

2. **Secure Storage**: Passwords are securely stored using hashing algorithms (like bcrypt or Argon2) to prevent exposure of sensitive information.

3. **Login Process**:

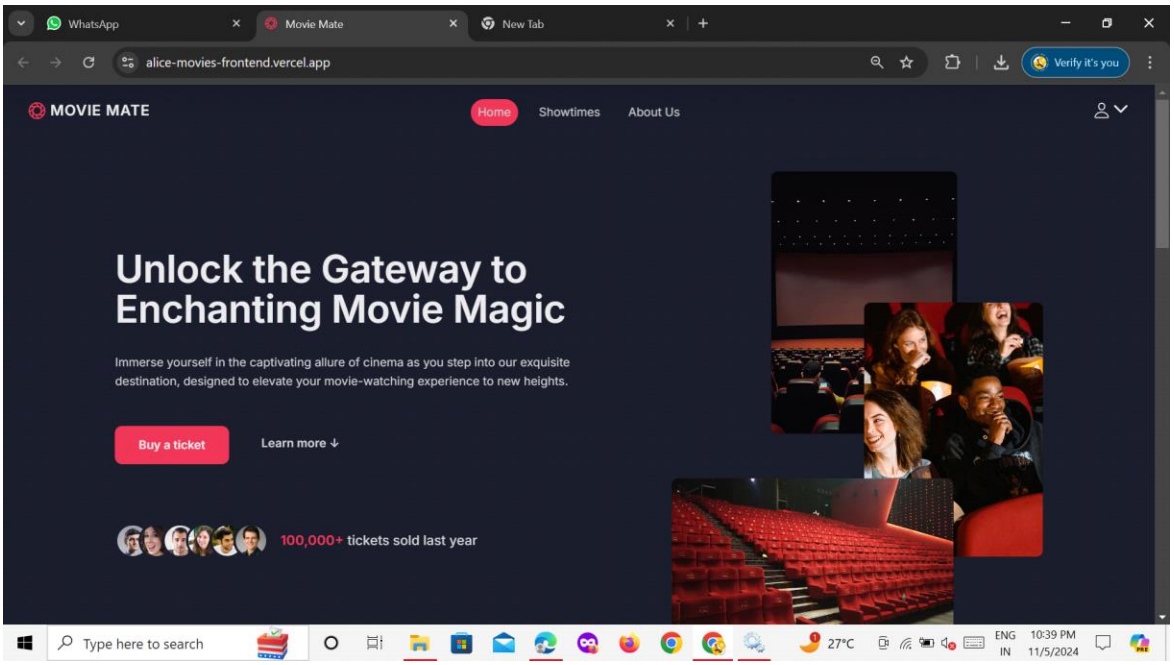
- When a user attempts to log in, the provided credentials are sent to the server.
- The server hashes the provided password and compares it to the stored hash.
- If they match, the user is authenticated.

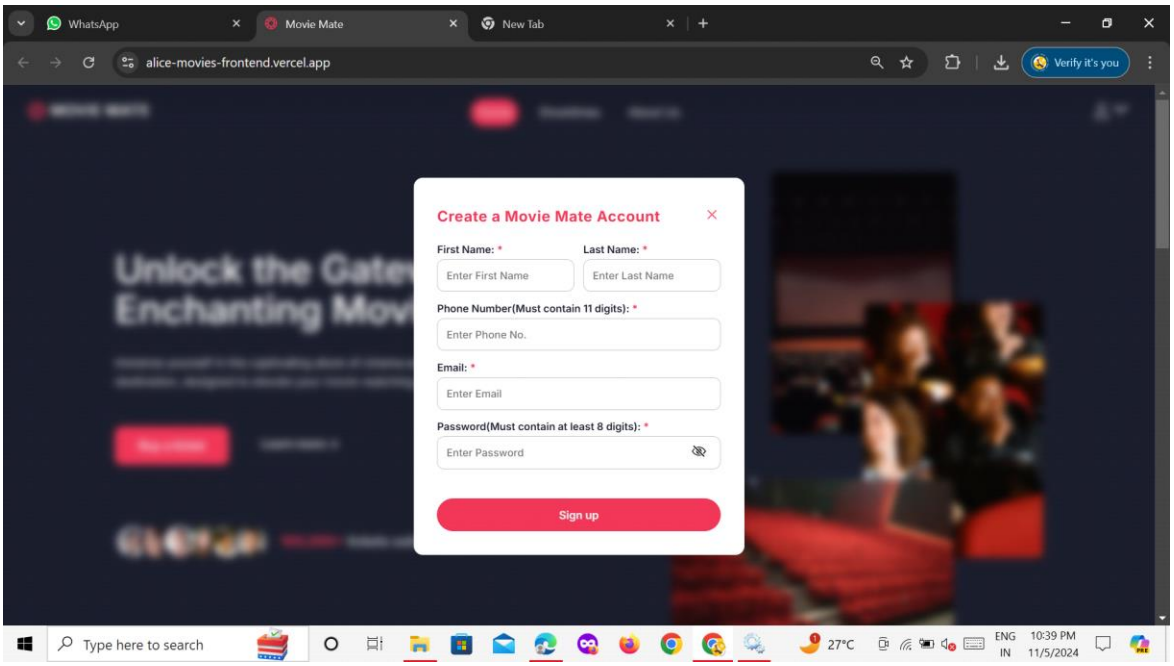
4. **Authentication Flow**:

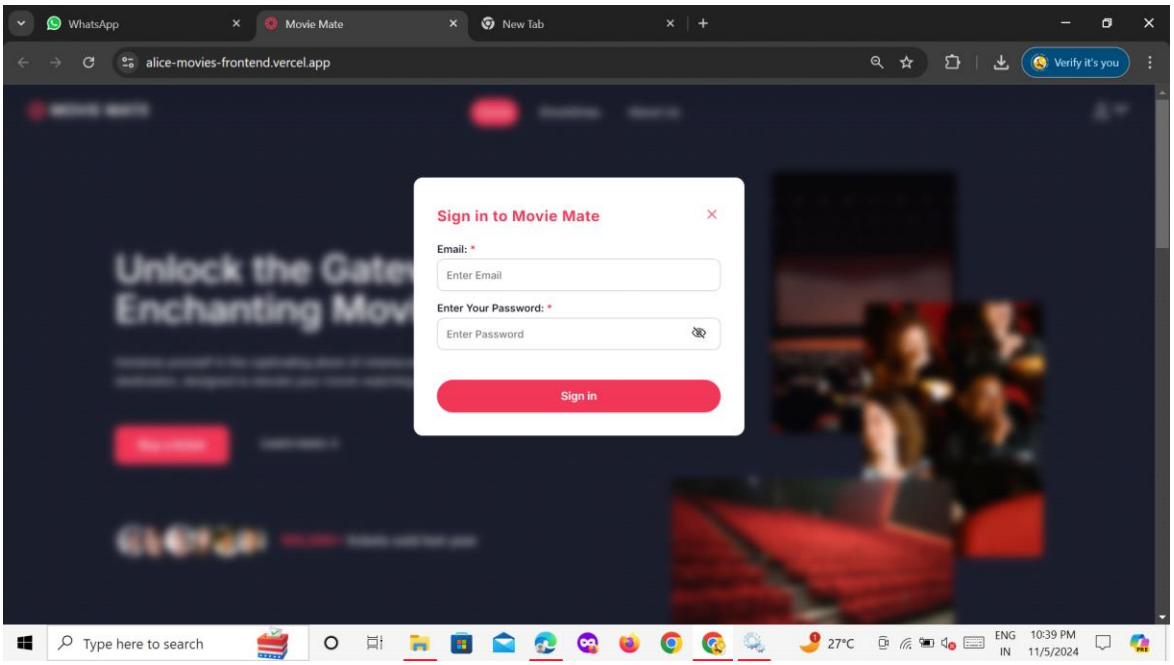
- User submits login form.
- Server checks credentials.
- On success, server generates a JWT (JSON Web Token) and sends it back to the client.

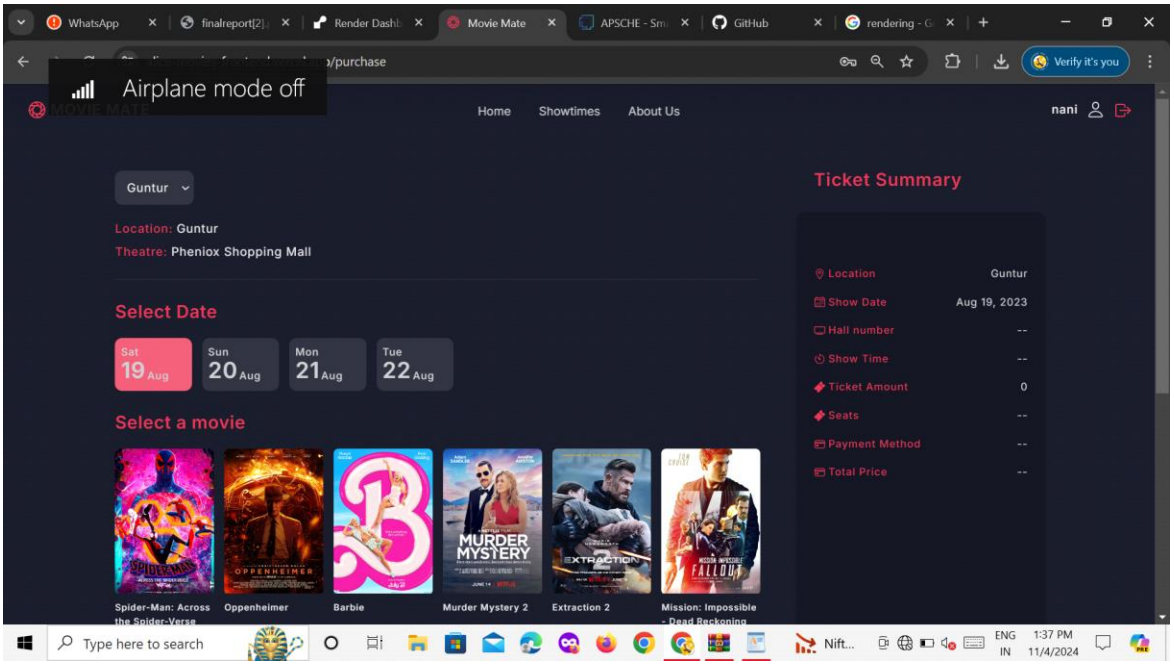
User Interface:-

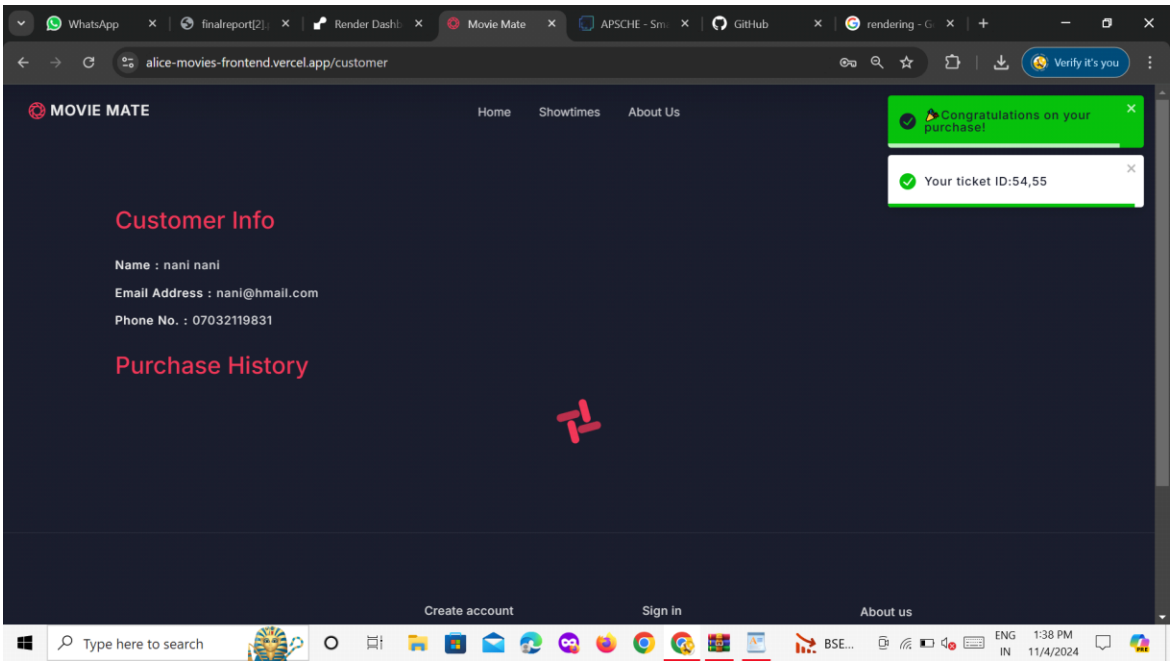
Screenshots or Demo


WhatsApp
Movie Mate
New Tab
alice-movies-frontend.vercel.app
Verify it's you
MOVIE MATE
Home
Showtimes
About Us
Unlock the Gateway to Enchanting Movie Magic
Immerse yourself in the captivating allure of cinema as you step into our exquisite destination, designed to elevate your movie-watching experience to new heights.
Buy a ticket
Learn more ↓
100,000+ tickets sold last year
Type here to search
27°C
ENG IN
10:39 PM 11/5/2024


WhatsApp
Movie Mate
New Tab
alice-movies-frontend.vercel.app
Verify it's you
Create a Movie Mate Account
First Name: *
Last Name: *
Enter First Name
Enter Last Name
Phone Number(Must contain 11 digits): *
Enter Phone No.
Email: *
Enter Email
Password(Must contain at least 8 digits): *
Enter Password
Sign up
Type here to search
27°C
ENG IN
10:39 PM 11/5/2024


WhatsApp
Movie Mate
New Tab
alice-movies-frontend.vercel.app
Verify it's you
Sign in to Movie Mate
Email: *
Enter Email
Enter Your Password: *
Enter Password
Sign in
Type here to search
27°C
ENG
IN
10:39 PM
11/5/2024


WhatsApp
finalreport[2]
Render Dashb
Movie Mate
APSCHE - Sm
GitHub
rendering - G
/purchase
Verify it's you
Airplane mode off
Home
Showtimes
About Us
nani
Guntur
Location: Guntur
Theatre: Pheniox Shopping Mall
Select Date
Sat 19 Aug
Sun 20 Aug
Mon 21 Aug
Tue 22 Aug
Select a movie
Spider-Man: Across the Spider-Verse
Oppenheimer
Barbie
Murder Mystery 2
Extraction 2
Mission: Impossible - Dead Reckoning
Ticket Summary
Location Guntur
Show Date Aug 19, 2023
Hall number --
Show Time --
Ticket Amount 0
Seats --
Payment Method --
Total Price --
Type here to search
Nift...
ENG
IN
1:37 PM
11/4/2024



Testing :-

Outputs and Reports Testing

Outputs testing involves verifying that the system produces the correct results based on the inputs provided. This is especially important in areas like user registration, ticket booking, cart operations, and order processing.

Unit Testing:

Test individual functions or components like registration, login, and ticket booking to ensure they return the expected outputs.

For example, after booking a ticket, verify that the correct ticket details are returned in the response.

API Response Testing:

Ensure that all API responses have the correct status codes and data formats (JSON,etc.).

Test various endpoints like /api/user/me or /api/order to ensure the expected output is returned for both successful and failed requests (e.g., ticket details for valid users or appropriate error messages for unauthenticated users).

Conclusion:-

In developing an online movie ticket booking system, we have established a structured approach encompassing various essential components, including architecture, API endpoints, authentication, and comprehensive testing. By implementing a robust architecture that clearly separates frontend and backend functionalities, we enable seamless user interactions and efficient data handling. The API documentation provides clarity on the interactions available, facilitating easier integration and understanding for developers.

Authentication is a crucial element, ensuring that user data and operations remain secure while allowing for a personalized experience. With user registration, login, and protected routes, we can confidently manage access to sensitive functionalities.

Moreover, thorough testing—including outputs and reports testing—ensures the application functions correctly, provides accurate data, and meets user expectations. Through unit and integration testing, we can validate that individual components and their interactions yield the expected results. Additionally, reports testing guarantees that generated outputs, such as sales data and user activity logs, are accurate and reflect real-time data.