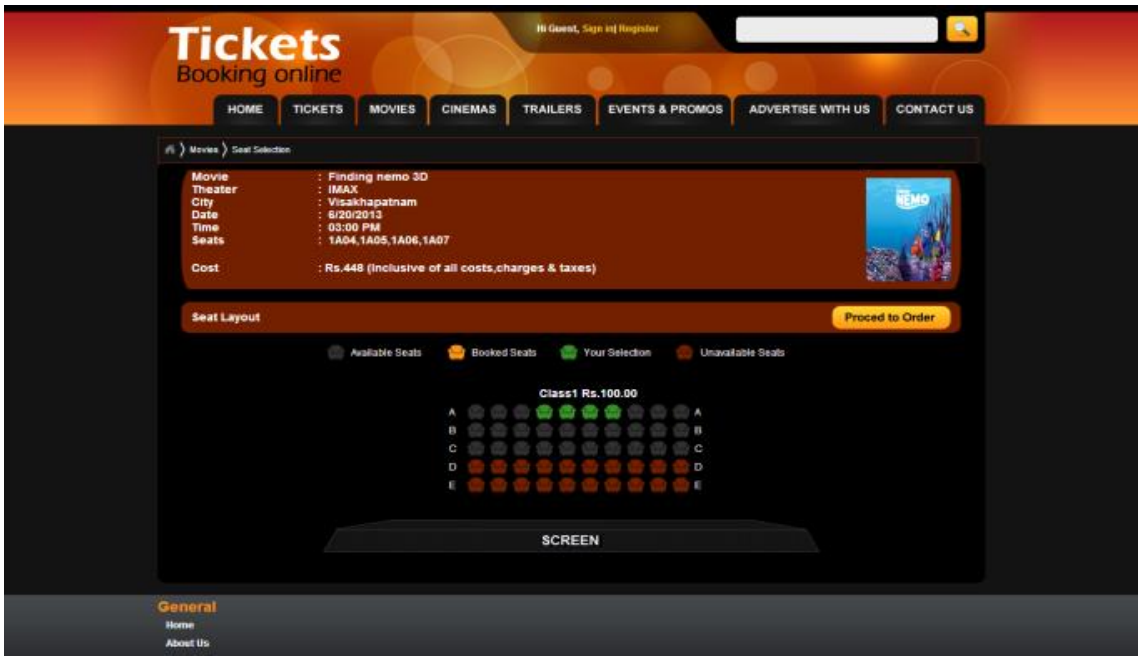
Movie Ticket Booking System



Team Members:

Ganta Karthik Kumar(20481A5415)

Kapalavai Sai Kumar(20481A5426)

Movva Sri Rama Krishna Datta(20481A5435)

K.J.V.N Durga Sai Ram(20485A5404)

1. Introduction

Movie ticket booking refers to the process of reserving seats or purchasing tickets for a film screening at a cinema or movie theater. It has become an essential part of the movie-going experience, offering convenience and flexibility to movie enthusiasts. With the advent of technology, movie ticket booking has transitioned from traditional methods, such as purchasing tickets at the box office or over the phone, to online platforms and mobile applications.

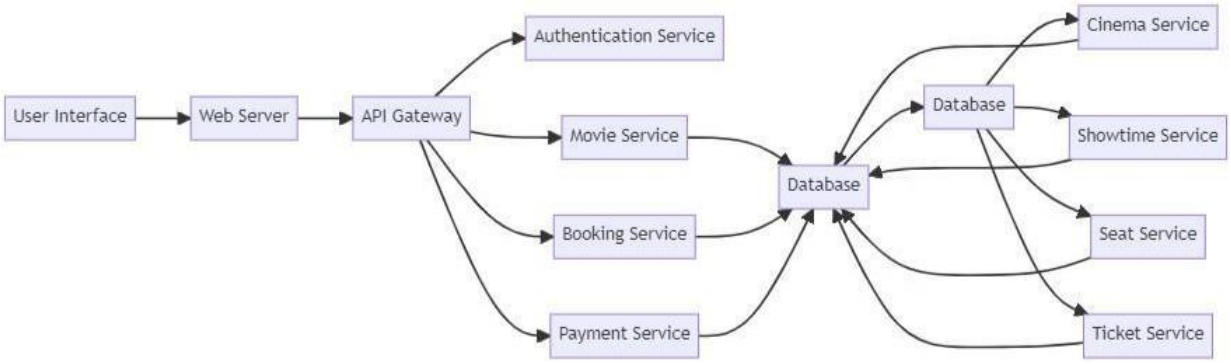
The convenience of booking movie tickets online has revolutionized the industry, making it easier for people to plan their movie outings in advance and secure their preferred seats. Online ticket booking platforms typically provide a user-friendly interface where moviegoers can browse through a list of movies, view show timings, select seats, and complete the transaction seamlessly from the comfort of their homes or on-the-go.

These platforms often offer additional features and benefits, such as the ability to choose specific seats, access real-time seating availability, browse through different cinema locations, and even avail of various discounts, promotions, and loyalty programs. They also provide users with the flexibility to cancel or reschedule their bookings if necessary.

Furthermore, movie ticket booking platforms have expanded beyond just ticket sales. They often serve as comprehensive entertainment hubs, offering movie reviews, trailers, ratings, and recommendations to help users make informed decisions about their movie choices. Some platforms may also provide the option to pre-order food and beverages, allowing moviegoers to avoid queues and enjoy a hassle-free experience.

Overall, movie ticket booking has evolved to provide a convenient and efficient way for individuals to plan their cinema visits, reserve seats, and enjoy the latest movies on the big screen. By leveraging technology, these platforms have transformed the movie-watching experience, making it more accessible, personalized, and enjoyable for audiences worldwide.

1.1 Technical Architecture

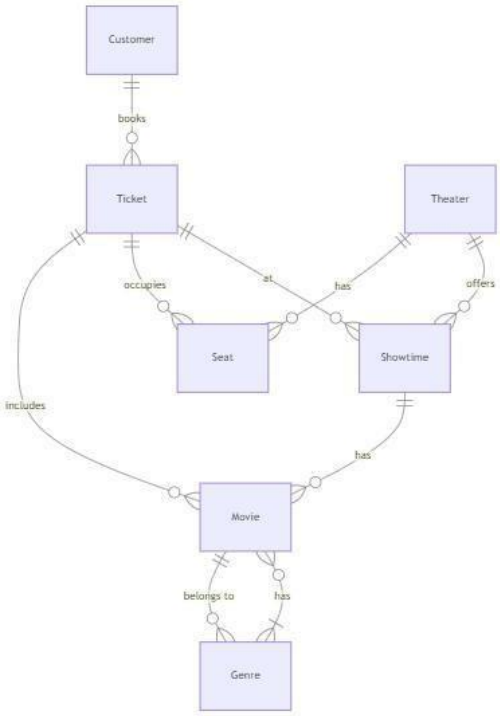


In this technical architecture, the movie ticket booking system consists of several components:

- User Interface: Represents the web interface through which users interact with the system to search for movies, select seats, and make bookings.
- Web Server: Hosts the user interface and serves the web pages to the users.
- API Gateway: Acts as a single entry point for client requests and routes them to the respective services.
- Authentication Service: Handles user authentication and authorization for secure access to the system.
- Movie Service: Manages movie-related information such as movie details, ratings, and reviews.
- Booking Service: Handles the process of creating and managing movie bookings.
- Payment Service: Integrates with payment gateways to process ticket payments securely.
- Database: Stores persistent data related to movies, bookings, payments, and other system entities.

- Cinema Service: Manages information about cinemas, their locations, and amenities.
- Showtime Service: Handles showtime schedules for different movies in various cinemas.
- Seat Service: Manages seat availability, selection, and reservation for movie screenings.
- Ticket Service: Generates and manages electronic tickets for users.

1.2 ER-Diagram



1.2.1 User:

- UserID (Primary Key)
- Name
- Email
- Phone

1.2.2 Movie:

- MovieID (Primary Key)
- Title
- Genre
- ReleaseDate

1.2.3 Cinema:

- CinemaID (Primary Key)
- Name
- Address
- City
- State
- ZipCode

1.2.4 Showtime:

- ShowtimeID (Primary Key)
- MovieID (Foreign Key)
- CinemaID (Foreign Key)
- DateTime

1.2.5 Booking:

- BookingID (Primary Key)
- UserID (Foreign Key)
- ShowtimeID (Foreign Key)
- NumTickets
- TotalAmount

In this diagram, we have four main entities: User, Movie, Cinema, and Showtime.

- User represents the information about the users who book movie tickets.
- Movie represents the details of each movie, such as its title, genre, and release date.
- Cinema represents the different cinemas where movies are screened, along with their location details.
- Showtime represents a specific showtime for a movie in a particular cinema, with the date and time of the show.

There is also an additional entity called Booking, which represents the booking made by a user for a specific showtime. It includes details like the number of tickets booked and the total amount.

The relationships between the entities are as follows:

- User has a one-to-many relationship with Booking (one user can make multiple bookings).
- Movie has a one-to-many relationship with Showtime (one movie can have multiple showtimes).

- Cinema has a one-to-many relationship with Showtime (one cinema can have multiple showtimes).
- Showtime has a one-to-many relationship with Booking (one showtime can have multiple bookings).

The foreign key relationships are established between the entities using the primary keys from the respective tables.

Please note that this is just an example, and you can modify or expand it based on your specific requirements for the movie ticket booking system.

1.3 Key Features:

- User Registration and Authentication: Allow users to create accounts, log in, and authenticate their identity to access the booking system.
- Movie Listings: Display a list of available movies with details such as title, genre, release date, and showtimes.
- Cinema Selection: Provide a selection of cinemas where the movies are being screened, including their location details.
- Seat Selection: Allow users to choose their preferred seats from a seating layout for the selected showtime.
- Booking Management: Enable users to book tickets for a specific showtime, specifying the number of tickets required.
- Payment Integration: Integrate with a payment gateway to facilitate secure online payment for the booked tickets.
- Booking Confirmation: Provide users with a booking confirmation page or email containing the details of the booking, including the ticket information and booking ID.
- Booking History: Allow users to view their past and upcoming bookings, along with the option to cancel or modify bookings if allowed.
- Seat Availability Tracking: Keep track of the availability of seats for each showtime and update in real-time as seats are booked or canceled.

- Ratings and Reviews: Allow users to rate and leave reviews for movies they have watched, providing valuable feedback to other users.
- Notifications: Send email or SMS notifications to users for booking confirmations, reminders, or any changes related to their bookings.

- Admin Dashboard: Provide an administrative interface to manage movies, showtimes, cinemas, seat availability, and user bookings.
- Reporting and Analytics: Generate reports and analytics on ticket sales, popular movies, user demographics, and other relevant metrics to gain insights into the business.
- Integration with External APIs: Integrate with third-party APIs for services like movie information, showtime updates, or seat availability for seamless data synchronization.

These are just a few key features, and you can customize and expand them based on your specific requirements and the scale of the movie ticket booking system you are building.

2. PRE REQUISITES:

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

2.1 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/

2.2 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/

2.3 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**

2.4 Angular: Angular is a JavaScript framework for building client-side applications. Install AngularCLI (Command Line Interface) globally to create and manage your Angular project.

Install Angular CLI:

- Angular provides a command-line interface (CLI) tool that helps with project setup and development.

- Install the Angular CLI globally by running the following command:
  npm install -g @angular/cli

  Verify the Angular CLI installation:
- Run the following command to verify that the Angular CLI is installed correctly:
  ng version

You should see the version of the Angular CLI printed in the terminal if the installation was successful.

Create a new Angular project:

- Choose or create a directory where you want to set up your Angular project.
- Open your terminal or command prompt.
- Navigate to the selected directory using the cd command.
- Create a new Angular project by running the following command: ng new client Wait for the project to be created:
- The Angular CLI will generate the basic project structure and install the necessary dependencies

Navigate into the project directory:

- After the project creation is complete, navigate into the project directory by running the following command**: cd client**

Start the development server:

- To launch the development server and see your Angular app in the browser, run the following command: **ng serve / npm start**
- The Angular CLI will compile your app and start the development server.
- Open your web browser and navigate to http://localhost:4200 to see your Angular app running.

You have successfully set up Angular on your machine and created a new Angular project. You can now start building your app by modifying the generated project files in the src directory.

Please note that these instructions provide a basic setup for Angular. You can explore more advanced configurations and features by referring to the official Angular documentation: https://angular.io

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 products 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://git-scm.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

To Connect the Database with Node JS go through the below provided link:

- Link: https://www.section.io/engineering-education/nodejs- mongoosejs-mongodb/

To run the existing Expense Tracker App project downloaded from github:

Follow below steps:

1. Clone the Repository:

- Open your terminal or command prompt.
- Navigate to the directory where you want to store the movie ticket booking app.
- Execute the following command to clone the repository:

  git clone https://github.com/ganta-karthik/movie-ticket-booking

2. Install Dependencies:

- Navigate into the cloned repository directory:
  ```
  cd movie-ticket-booking
  ```
- Install the required dependencies by running the following command:
  ```
  npm instal
  ```

3. Start the Development Server:

- To start the development server, execute the following command:

  npm run dev or npm run start
- The e-commerce app will be accessible at http://localhost:5100 by default. You can change the port configuration in the .env file if needed.

4. Access the App:

- Open your web browser and navigate to http://localhost:5100.
- You should see the movie-ticket-booking app's homepage, indicating that the installation and setup were successful.

Project Repository Link: https://github.com/ganta-karthik/movie-ticket-booking

Congratulations! You have successfully installed and set up the movie-ticket-booking app on your local machine. You can now proceed with further customization, development, and testing as needed.

3.Roles and Responsibility

3.1 User:

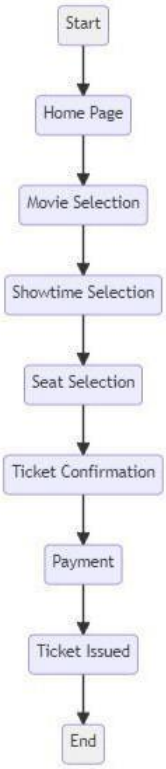
- Registration: Users are responsible for creating an account on the booking system by providing necessary details like name, email, and phone number.
- Movie Search and Selection: Users can search for movies, view movie details such as title, genre, and release date, and select the desired movie for booking.
- Cinema Selection: Users can choose the cinema where they want to watch the movie, based on the available options.
- Showtime Selection: Users can select a specific showtime for the chosen movie and cinema.
- Seat Selection: Users are responsible for selecting their preferred seats from the seating layout for the chosen showtime.
- Booking and Payment: Users can proceed with booking the selected seats and make the required payment for the tickets.
- Booking Management: Users can view their booking history, upcoming bookings, and cancel or modify their bookings if allowed by the system.
- Rating and Reviews: Users can provide ratings and leave reviews for movies they have watched, sharing their experiences with others.

3.2 Admin:

- System Management: The admin is responsible for managing the overall functioning and configuration of the movie ticket booking system.
- Movie Management: The admin can add, update, or remove movies from the system, including details like title, genre, and release date.
- Seat Management: The admin is responsible for configuring and maintaining the seating layouts for each cinema, ensuring accurate seat availability information.
- User Management: The admin can manage user accounts, including user registration, authentication, and handling user-related issues or inquiries.

These roles and responsibilities can vary based on the specific requirements and scope of the movie ticket booking system.

3.3 User Flow



In this user flow, the process starts at the "Start" node, then moves to the "Home Page" where the user can browse movies. From there, the user selects a movie and proceeds to the "Movie Selection" node. After selecting a movie, the user chooses a showtime at the "Showtime Selection" node.

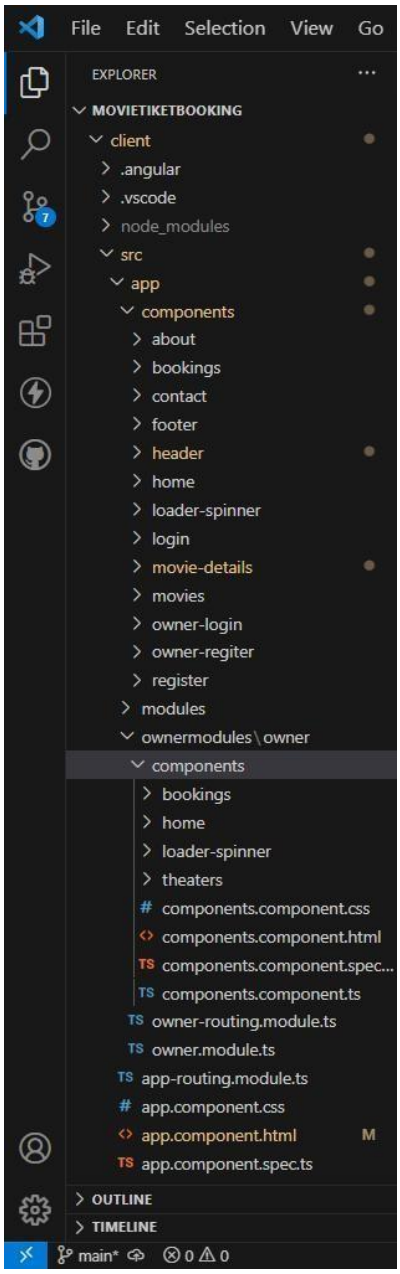
Next, the user proceeds to the "Seat Selection" node to choose their preferred seats. Once the seats are selected, the user moves to the "Ticket Confirmation" node where they can review their booking details.

From there, the user moves to the "Payment" node to complete the transaction. After successful payment, the system issues the ticket at the "Ticket Issued" node. Finally, the process ends at the "End" node.

The style statements at the bottom are optional and can be used to customize the appearance of specific nodes. In this example, the "Start" and "End" nodes have a light gray background.

Feel free to modify the code as per your specific requirements or add additional nodes as needed.

Project Structure:



The project structure may vary depending on the specific framework, programming language, or development approach used. It's essential to organize the files and directories in a logical and consistent manner to improve code maintainability and collaboration among developers.

app/app.component.scss, src/app/app.component.spec.ts: These files are part of the main AppComponent, which serves as the root component for the Angular app. The component handles the overall layout and includes the router outlet for loading different components based on the current route.

4. PROJECT FLOW:

Milestone 4.1: Project Setup and Configuration:

1. Install required tools and software:

- Node.js.
- MongoDB.
- Angular CLI.

2. Create project folders and files:

- Client folders.
- Server folders.

Milestone 4.2: Backend Development:

Setup express server:

- Install express.
- Create app.js file.
- Define API's

Configure MongoDB:

- Install Mongoose.
- Create database connection.
- Create Models.

Implement API end points:

- Implement CRUD operations.
- Test API endpoints.

Milestone 4.3: Web Development:

1. Setup Angular Application:

- Create Angular application using angular CLI.
- Configure Routing.
- Install required libraries.

2. Design UI components:

- Create Components.
- Implement layout and styling.
- Add navigation.

3. Implement frontend logic:

- Integration with API endpoints.
- Implement data binding.

To Setup the frontend development and to connect node.js with MongoDB Database Go through this video link: -

https://drive.google.com/file/d/1b5bMvnqmASXLnSZ74B2t3EzNjuWHj63g/view?usp=drive_link

4.4 Backend:

Server: The backend system starts with a server that handles incoming requests from the client-side (e.g., web browser or mobile app). The server receives the user's actions and processes them accordingly.

API Endpoints: The server exposes a set of API endpoints that allow the client-side to interact with the backend. These endpoints receive requests related to movie selection, showtime selection, seat selection, payment, and ticket issuance.

Database: The backend typically utilizes a database to store and retrieve movie data, showtimes, seat availability, user information, and ticket details. The server communicates with the database to perform CRUD operations (Create, Read, Update, Delete) as needed.

Movie Selection: When the user selects a movie, the server fetches the relevant movie information from the database and sends it back to the client-side. This may include details like movie title, synopsis, duration, genre, cast, and poster image.

Showtime Selection: Once the user chooses a movie, the server retrieves the available showtimes for that movie from the database. It sends the showtime options back to the client-side, allowing the user to select a preferred showtime.

Seat Selection: After the showtime is selected, the server checks the availability of seats for that particular showtime. It retrieves the seating layout and seat status (available, booked, or reserved) from the database and sends it to the client-side. The user can then choose their desired seats.

Ticket Confirmation: Once the user confirms their seat selection, the server generates a booking confirmation. It calculates the total ticket price based on the selected seats and any additional charges (e.g., convenience fee, taxes). The confirmation details, including the selected movie, showtime, seats, and total price, are sent to the client-side for review.

Payment: The server integrates with a payment gateway or third-party payment service to handle the payment transaction securely. The client-side sends the payment details to the server, which forwards the information to the payment service. The server receives the payment status (success or failure) from the payment service and proceeds accordingly.

Ticket Issuance: Upon successful payment, the server generates the ticket(s) for the user. It stores the ticket details in the database and sends the ticket information (e.g., ticket ID, movie details, showtime, seat numbers) back to the client-side. The user can then view or download their ticket.

Throughout the backend process, the server performs necessary validations, error handling, and business logic to ensure data integrity and a smooth user experience.

Please note that the backend implementation may vary based on the specific technologies, frameworks, and architectural choices used. This explanation provides a general overview of how the backend components and processes could be structured for a movie ticket booking system.

Conclusion

The ticket booking system has also significantly benefited cinema owners and management. The comprehensive backend management tools have improved operational efficiency, enabling theatre owners to optimize resources and monitor performance effectively. Real-time data analytics and insights have empowered decision-makers to make informed choices, leading to increased profitability and sustained growth. Moreover, the movie ticket booking system has facilitated a seamless transition to digital cinema, promoting eco-friendly practices through reduced paper usage and encouraging a more sustainable industry. While the system has undoubtedly achieved remarkable success, continuous improvement and innovation will be crucial to stay ahead in the dynamic digital landscape. Embracing emerging technologies, ensuring cybersecurity measures, and focusing on customer feedback will play a pivotal role in maintaining its position as the preferred method for movie ticket reservations.