A Field Project Report on

ONLINE TICKET BOOKING

Submitted

In partial fulfillment of the requirements for the award of the degree

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE and ENGINEERING

By

P.Sandeep	(231FA04986)
G.Amani	(231FA04A05)
B.Narasimha	(231FA04A07)
A.Hemeshwar	(231FA04A09)

Under the Guidance of
Mr.T.Narasimha Rao ,
Assistant Professor, CSE

VIGNAN'S
FOUNDATION FOR SCIENCE, TECHNOLOGY & RESEARCH
(Deemed to be University) - Estd. u/s 3 of UGC Act 1956

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

SCHOOL OF COMPUTING AND INFORMATICS

VIGNAN'S FOUNDATION FOR SCIENCE, TECHNOLOGY & RESEARCH
(Deemed to be University)
Vadlamudi, Guntur -522213, INDIA.

April, 2025

VIGNAN'S
FOUNDATION FOR SCIENCE, TECHNOLOGY & RESEARCH
(Deemed to be University) - Estd. u/s 3 of UGC Act 1956

CERTIFICATE

This is to certify that the field project entitled *"ONLINE TICKET BOOKING "* is being submitted by **[P.Sandeep]**, **[231FA04986]**, **[G.Amani]**, **[231FA04A05]**, **[B.Narasimha]**, **[231FA04A07]**, and **[A.Hemeshwar]**, **[231FA04A09]** in partial fulfilment of the requirements for the degree of **Bachelor of Technology (B.Tech.) in Computer Science and Engineering** at Vignan's Foundation for Science, Technology and Research (Deemed to be University), Vadlamudi, Guntur District, Andhra Pradesh, India.

This is a bonafide work carried out by the aforementioned students under my guidance and supervision.

[illegible]
Guide

[illegible]
HoD, CSE

HoD
Dept. of Computer Science & Engine
VFSTR Deemed to be Universi
VADLAMUDI - 522 213
Guntur Dist., A.P.

[illegible] 28/4/25
Project Review Committee

VIGNAN'S
FOUNDATION FOR SCIENCE, TECHNOLOGY & RESEARCH
(Deemed to be University) - Estd. u/s 3 of UGC Act 1956

DECLARATION

Date: 26/04/2025

We hereby declare that the work presented in the field project titled "ONLINE TICKET BOOKING " is the result of our own efforts and investigations.

This project is being submitted under the supervision of **Mr.T.Narasimha Rao, Assistant Professor, CSE** in partial fulfillment of the requirements for the Bachelor of Technology (B.Tech.) degree in Computer Science and Engineering at Vignan's Foundation for Science, Technology and Research (Deemed to be University), Vadlamudi, Guntur, Andhra Pradesh, India.

P.Sandeep	(231FA04986)	P. Sandeep.
G.Amani	(231FA04A05)	G. Amani
B.Narasimha	(231FA04A07)	B. Narasimha
A.Hemeshwar	(231FA04A09)	[illegible]

TABLE OF CONTENTS

“ONLINE TICKET BOOKING”

1.Introduction

The aim of the introduction is to provide a clear and concise overview of the **online ticket booking system**, its purpose, and the scope of the project. It sets the context for the problem being addressed, highlights the limitations of existing systems, and introduces the proposed solution. The introduction also outlines the three main types of tickets (movie, train, and stadium) that the system will handle, while emphasizing the need for a centralized, user-friendly, and efficient platform. Ultimately, the introduction aims to establish the significance of the project and its potential to improve the ticket booking experience for users and service providers alike.

1.1 Problem Definition

The traditional ticket booking process often involves long queues, limited availability, and inefficiencies in managing reservations. Customers face challenges such as:

- Difficulty in accessing real-time ticket availability.
- Inconvenience of visiting physical ticket counters.
- Lack of a centralized platform for booking different types of tickets.
- Time-consuming processes for refunds or cancellations.
- For service providers, manual ticket management leads to:
- Increased operational costs.
- Errors in booking and reservation management.
- Limited reach to potential customers.

An online ticket booking system aims to address these issues by providing a seamless, user-friendly, and efficient platform for ticket reservations and management.

1.2 Existing System

The existing ticket booking systems, whether offline or online, often suffer from several limitations:

Offline Systems: Rely on physical ticket counters, leading to long waiting times and limited accessibility.

Basic Online Systems: Some platforms offer online booking but lack integration for multiple ticket types (e.g., movies, trains, and stadium events).

User Experience: Many systems have poor user interfaces, making it difficult for users to navigate and complete bookings.

Limited Features: Existing systems may not support advanced features like seat selection, real-time availability updates, or personalized recommendations.

These shortcomings highlight the need for a more robust and integrated solution.

1.3 Proposed System

The proposed online ticket booking system is designed to overcome the limitations of existing systems by offering the following features:

Centralized Platform: A single platform for booking movie tickets, train tickets, and stadium tickets.

User-Friendly Interface: Intuitive design for easy navigation and quick bookings.

Real-Time Updates: Instant updates on ticket availability, pricing, and schedules.

Advanced Features: Seat selection, multiple payment options, and personalized recommendations based on user preferences.

Efficient Management: Streamlined processes for ticket cancellations, refunds, and rescheduling.

Scalability: The system will be scalable to accommodate future expansions, such as adding new ticket types or integrating with other services.

The proposed system aims to enhance the user experience while reducing operational costs for service providers.

1.4 Literature Review

Several studies and existing systems have been analyzed to identify best practices and areas for improvement in online ticket booking:

Movie Ticket Booking: Platforms like BookMyShow and Fandango have set benchmarks for user experience, offering features like seat selection, trailers, and reviews.

Train Ticket Booking: Systems like IRCTC (Indian Railway Catering and Tourism Corporation) provide real-time availability and e-ticketing but often face issues with user interface and server load during peak times.

Stadium Ticket Booking: Platforms like Ticketmaster and StubHub offer event ticket booking but lack integration with other ticket types.

Unified Systems: Some platforms attempt to integrate multiple ticket types but often lack seamless functionality or fail to provide a consistent user experience.

The literature review highlights the need for a unified, scalable, and user-friendly online ticket booking system that addresses the limitations of existing platforms while incorporating advanced features to meet modern consumer demands.

2 System Requirements

The development and deployment of the online ticket booking system require specific hardware and software resources to ensure optimal performance, scalability, and user satisfaction. Below are the detailed requirements for the system:

2.1 Hardware & Software Requirements

Hardware Requirements:

Category	**Server-Side**	**Client-Side**
Processor	Intel Xeon or equivalent (Quad-core or higher)	Intel i3 or equivalent (minimum)
RAM	16 GB or higher	4 GB or higher
Storage	500 GB SSD (fast access and storage)	100 GB HDD or SSD
Display	--	1024x768 resolution or higher
Network	High-speed internet (min. 100 Mbps bandwidth)	Stable internet connection (min. 10 Mbps bandwidth)
Backup Hardware	External storage devices or cloud-based backup solutions	

Software Requirements:

Operating System	Linux (Ubuntu, CentOS) or Windows Server	Windows, macOS, or Linux
Web Server	Apache or Nginx	-
Database	MySQL, PostgreSQL, or MongoDB	-
Backend Framework	App.test.js, index.css, index.js	-
Programming Languages	JavaScript (Node.js), Python, or Java	-
Web Browser	-	Chrome, Firefox, Safari, or Edge (latest versions)

IDE (Dev Tools)	Visual Studio Code, IntelliJ IDEA, or PyCharm	Visual Studio Code, IntelliJ IDEA, or PyCharm
Version Control	Git and GitHub/GitLab	Git and GitHub/GitLab
Testing Tools	Selenium, JMeter, or Postman	Selenium, JMeter, or Postman
Payment Integration	PayPal, Stripe, or Razorpay	-
Cloud Hosting	AWS, Google Cloud, or Microsoft Azure	-
Security Tools	SSL certificates, firewalls, encryption protocols	-

2.2 Software Requirements Specification (SRS)

The **Software Requirements Specification (SRS)** document outlines the functional and non-functional requirements of the online ticket booking system. It serves as a blueprint for developers, designers, and stakeholders to ensure the system meets user needs and business goals.

Functional Requirements:

User Registration and Authentication:

Users can create accounts and log in using email, phone number, or social media accounts.

Password recovery and two-factor authentication (2FA) for security.

Ticket Booking:

Users can search for and book movie tickets, train tickets, and stadium tickets.

Real-time availability and pricing information.

Seat selection for movies and stadium events.

Booking confirmation via email or SMS.

Payment Integration:

Multiple payment options (credit/debit cards, net banking, UPI, wallets).

Secure payment gateway integration.

Ticket Management:

Users can view, cancel, or reschedule tickets.

Refund processing for canceled tickets.

Admin Panel:

Admin can manage ticket inventory, user accounts, and bookings.

Generate reports on sales, bookings, and user activity.

Notifications:

Real-time notifications for booking confirmations, cancellations, and reminders.

Search and Filters:

Advanced search options for movies, trains, and events.

Filters for date, time, location, price, and availability.

Non-Functional Requirements:

Performance:

The system should handle a minimum of 10,000 concurrent users.

Response time for ticket booking should be less than 3 seconds.

Scalability:

The system should be scalable to accommodate future growth in users and ticket types.

Security:

Data encryption for sensitive information (e.g., payment details, user credentials).

Protection against SQL injection, cross-site scripting (XSS), and other vulnerabilities.

Usability:

Intuitive and user-friendly interface for both desktop and mobile users.

Accessibility compliance (e.g., WCAG standards).

Reliability:

System uptime of 99.9% with minimal downtime.

Backup and recovery mechanisms in case of failures.

Compatibility:

Cross-browser and cross-platform compatibility (desktop, mobile, tablet).

3 System Design

The system design of the online ticket booking platform outlines the architecture, key modules, and workflow. It ensures that all components work together efficiently to provide a seamless ticket booking experience. The system follows a three-tier architecture, comprising:

1. Presentation Layer (Frontend) – User interface for searching, selecting, and purchasing tickets.
2. Application Layer (Backend) – Handles business logic, authentication, and payment processing.
3. Database Layer – Stores user details, ticket records, transaction history, and event schedules.

3.1 Modules of System

The system consists of multiple modules, each performing a specific function to enhance the booking experience:

1. User Management Module

- User registration and login (via email, phone, or social media).
- Profile management and order history tracking.
- Password reset and account security settings.

2. Ticket Booking Module

- **Movie Ticket Booking:** Users can search movies, check availability, and select seats.
- **Train Ticket Booking:** Integration with railway schedules, seat availability, and booking confirmation.
- **Stadium Ticket Booking:** Event listing, seat selection, and dynamic pricing.

3. Payment and Transaction Module

- Secure payment gateways (credit/debit card, UPI, digital wallets).
- Order confirmation and digital ticket generation.
- Refund and cancellation management.

4. Admin Panel Module

- Managing ticket inventory and seat allocations.
- Adding or removing movies, train schedules, and stadium events.
- Monitoring transactions, discounts, and offers.

3.2 UML Diagrams

Online Movie Ticket

A **Use Case Diagram** describes the functionality of a system from a user's perspective, showing how different types of users (called **actors**) interact with the system. In the case of an online ticket booking system, the actors would typically include customers, admins, and payment systems, while the use cases represent actions like browsing events, purchasing tickets, and processing payments.

Actors:

- **Customer**: The person who interacts with the system to browse available events, select tickets, and make payments.
- **Admin**: Manages the event details, tickets, and user accounts.
- **Payment Gateway**: Processes payments for the tickets.
- **System**: Manages ticket availability, event details, and user interaction.

Use Cases:

1. **Browse Events**: The customer can view available tickets for different events (movies, trains, or stadiums).
2. **Select Tickets**: The customer selects the type of tickets they want to purchase.
3. **Make Payment**: The customer enters payment details, and the system processes the transaction via the payment gateway.
4. **View Ticket**: After payment is completed, the customer can view and download.

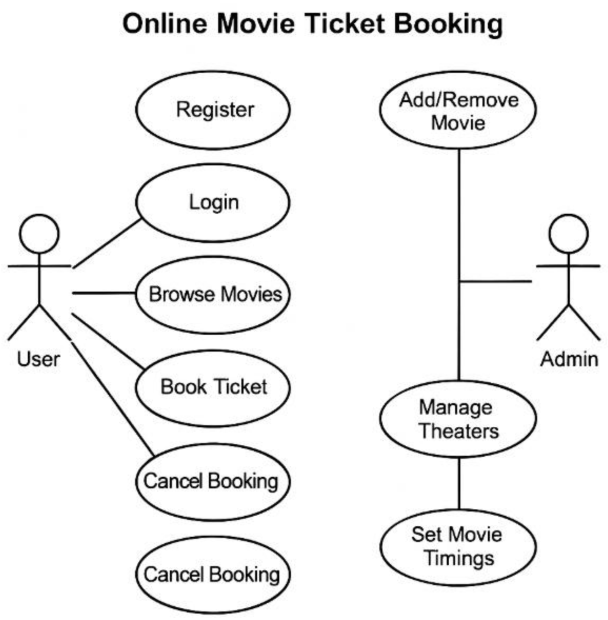


Fig-3.2.1

Train Tickets

An **Activity Diagram** describes the flow of activities or actions within a process. It is used to model the dynamic behavior of a system. For booking a train ticket, it shows the sequence of actions from starting the booking process to receiving the final ticket.

Workflow for Booking Train Tickets:

1. **Start**: The customer initiates the process.
2. **Select Train**: The customer chooses a train (based on destination, time, etc.).
3. **Select Seats**: The customer selects the available seats for the journey.
4. **Provide Payment Details**: The customer enters their payment information.
5. **Process Payment**: The payment gateway verifies and processes the payment.
6. **Confirm Ticket**: If the payment is successful, the system confirms the booking.
7. **End**: The customer receives their train ticket, either digitally or physically.

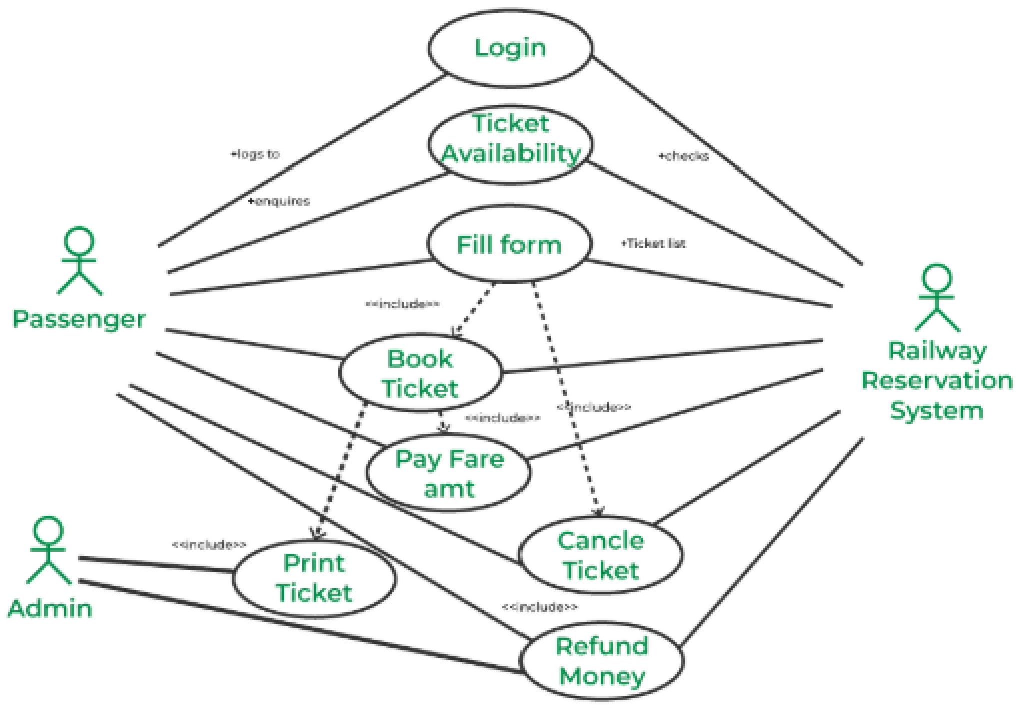


Fig-3.2.1

Stadium Ticket Booking System

A **Use Case Diagram** represents the interaction between the **actors** (users) and the **system**. For a stadium ticket booking system, we can have different actors like customers, administrators, and payment systems.

Actors:

- **Customer**: A person who browses the available events, selects tickets, and makes payments.
- **Payment Gateway**: An external service that processes payments made by the customer.
- **System**: The platform that manages the booking process, seat availability, and ticket generation.

Use Cases:

1. **Browse Events**: The customer can browse upcoming events (sports, concerts, etc.) at the stadium.
2. **Select Seats**: The customer selects available seats for the event they wish to attend.
3. **Make Payment**: The customer enters their payment details to purchase the tickets.
4. **View/Download Ticket**: The customer can view or download the ticket after payment.
5. **Manage Events**: Admin can create, update, or remove events, including adding or removing tickets for those events.
6. **Manage Tickets**: Admin manages the availability, pricing, and seating for each event.

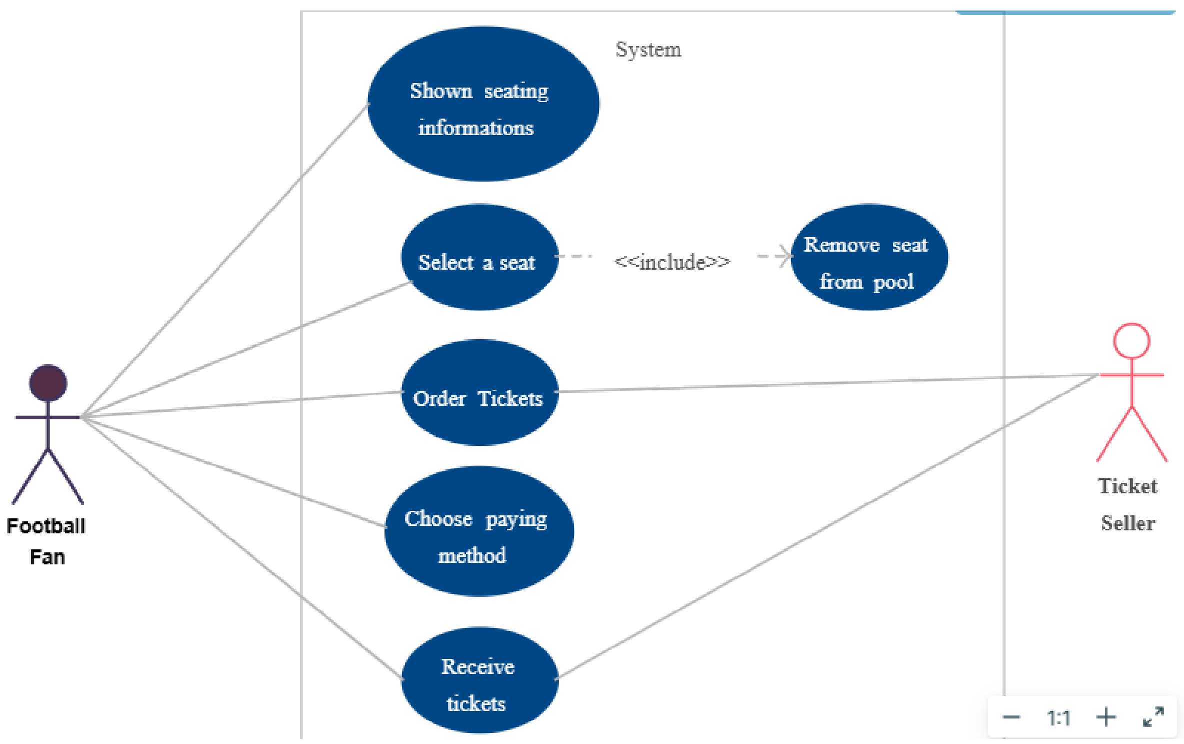


Fig-3.2.3

4 Implementation

Sample Code

App.css

```
.App {
  text-align: center;
}
.App-logo {
  height: 40vmin;
  pointer-events: none;
}
@media (prefers-reduced-motion: no-preference) {
  .App-logo {
    animation: App-logo-spin infinite 20s linear;
  }
}
.App-header {
  background-color: #282c34;
  min-height: 100vh;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  font-size: calc(10px + 2vmin);
  color: white;
}
.App-link {
  color: #61dafb;
}
@keyframes App-logo-spin {
  from {
    transform: rotate(0deg);
  }
  to {
    transform: rotate(360deg);
  }
}
```

App.js

```
import { BrowserRouter as Router, Route, Routes } from "react-router-dom";
import HomePage from "./comonents/home";
import Dashboard from "./comonents/Dashboard";
import MovieBooking from "./comonents/movie";
import StadiumBooking from "./comonents/stadium";
import TrainBooking from "./comonents/trains";
function App() {
  return (
    <Router>
      <Routes>
        <Route path="/" element={<HomePage />} />
        <Route path="/dashboard" element={<Dashboard />} />
        <Route path="/movies" element={<MovieBooking />} />
        <Route path="/stadiums" element={<StadiumBooking />} />
        <Route path="/trains" element={<TrainBooking/>} />
      </Routes>
    </Router>
  );
}
export default App;
```

App.test.js

```
import { render, screen } from '@testing-library/react';
import App from './App';
```

```
test('renders learn react link', () => {
  render(<App />);
  const linkElement = screen.getByText(/learn react/i);
  expect(linkElement).toBeInTheDocument();
});
```

index.css

```
body {
  margin: 0;
  font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto', 'Oxygen',
    'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue',
    sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}
code {
  font-family: source-code-pro, Menlo, Monaco, Consolas, 'Courier New',
    monospace;
}
```

index.js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);
```

```
// If you want to start measuring performance in your app, pass a function
// to log results (for example: reportWebVitals(console.log))
// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
reportWebVitals();
```

reportWebVitals.js

```
const reportWebVitals = onPerfEntry => {
  if (onPerfEntry && onPerfEntry instanceof Function) {
    import('web-vitals').then(({ getCLS, getFID, getFCP, getLCP, getTTFB }) => {
      getCLS(onPerfEntry);
      getFID(onPerfEntry);
      getFCP(onPerfEntry);
      getLCP(onPerfEntry);
      getTTFB(onPerfEntry);
    });
  }
};
export default reportWebVitals;
```

setupTests.js

```
// jest-dom adds custom jest matchers for asserting on DOM nodes.
// allows you to do things like:
// expect(element).toHaveTextContent(/react/i)
// learn more: https://github.com/testing-library/jest-dom
import '@testing-library/jest-dom';
```

gitignore

```
# See https://help.github.com/articles/ignoring-files/ for more about ignoring files.
# dependencies
/node_modules
/.pnp
.pnp.js
# testing
```

```
/coverage
# production
/build
# misc
.DS_Store
.env.local
.env.development.local
.env.test.local
.env.production.local
npm-debug.log*
yarn-debug.log*
yarn-error.log*
```

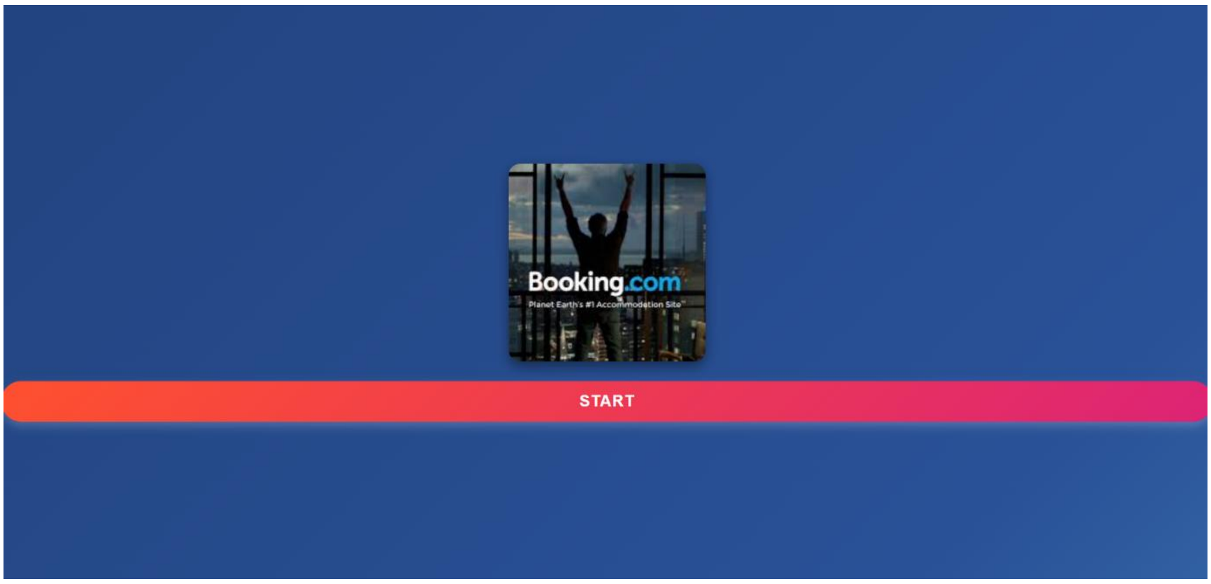
package.json

```
{
  "name": "onlines",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@testing-library/dom": "^10.4.0",
    "@testing-library/jest-dom": "^6.6.3",
    "@testing-library/react": "^16.2.0",
    "@testing-library/user-event": "^13.5.0",
    "axios": "^1.8.2",
    "react": "^19.0.0",
    "react-dom": "^19.0.0",
    "react-icons": "^5.5.0",
    "react-router-dom": "^7.3.0",
    "react-scripts": "5.0.1",
    "web-vitals": "^2.1.4"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  },
  "eslintConfig": {
    "extends": [
      "react-app",
      "react-app/jest"
    ]
  },
  "browserslist": {
    "production": [
      ">0.2%",
      "not dead",
      "not op_mini all"
    ],
    "development": [
      "last 1 chrome version",
      "last 1 firefox version",
      "last 1 safari version"
    ]
  }
}
```

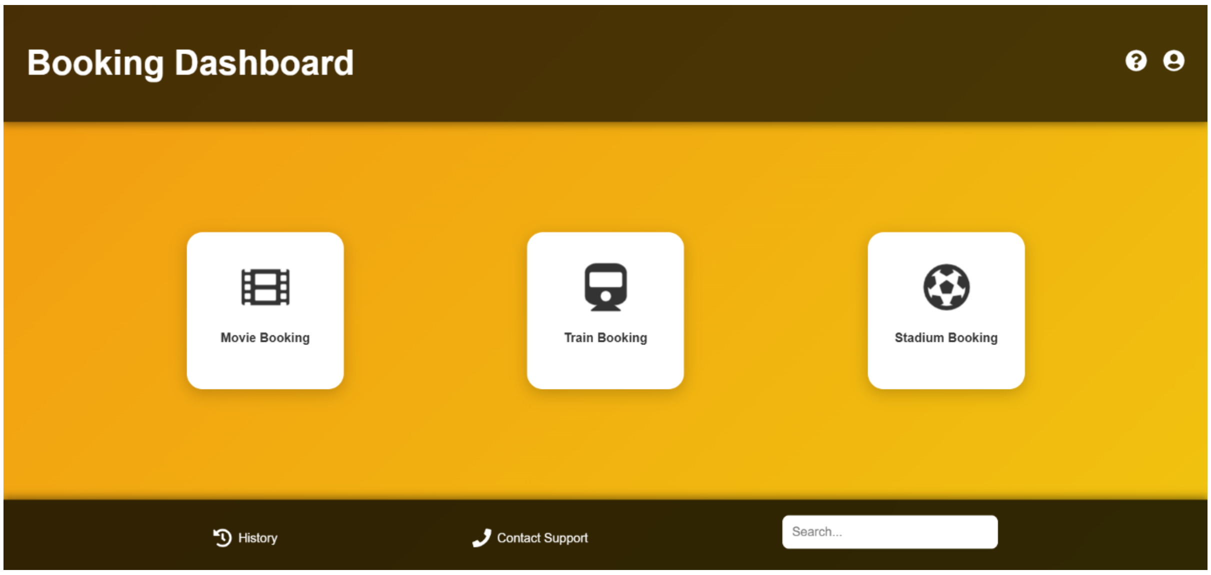
5 Results

5.1 Output Screens

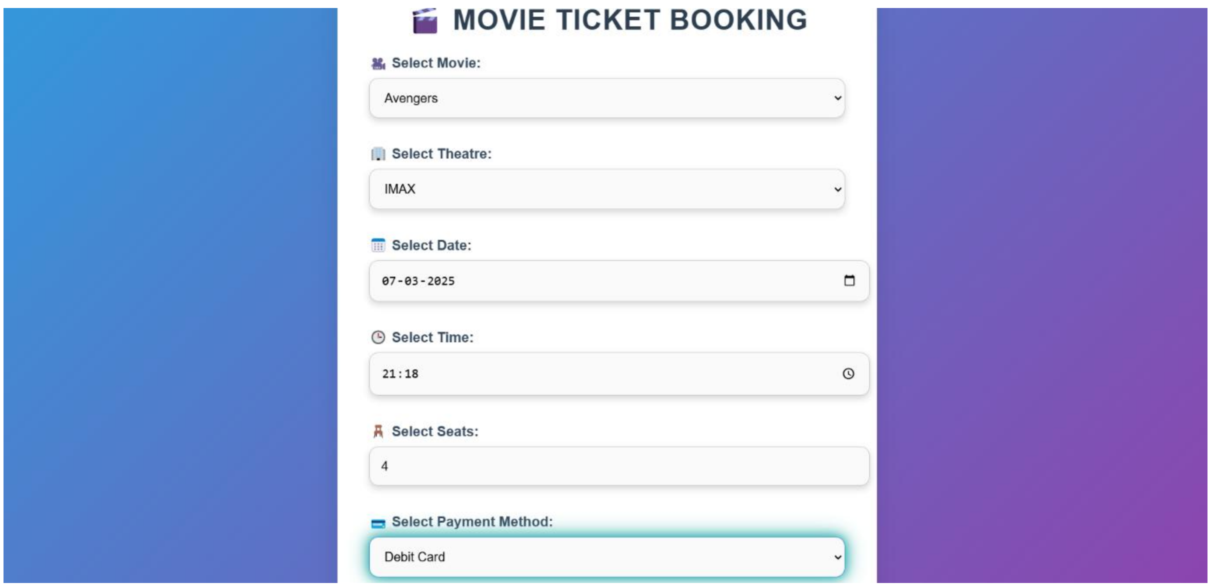
Login Page



Home page



Book Now Page:



MOVIE TICKET BOOKING

Select Movie: Avengers

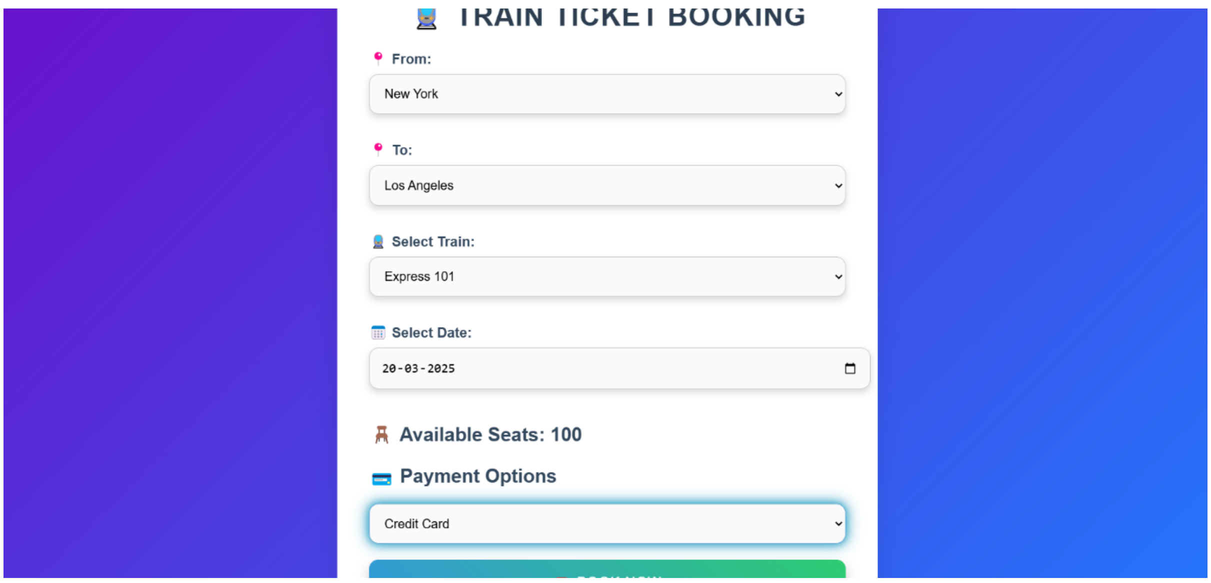
Select Theatre: IMAX

Select Date: 07-03-2025

Select Time: 21:18

Select Seats: 4

Select Payment Method: Debit Card



TRAIN TICKET BOOKING

From: New York

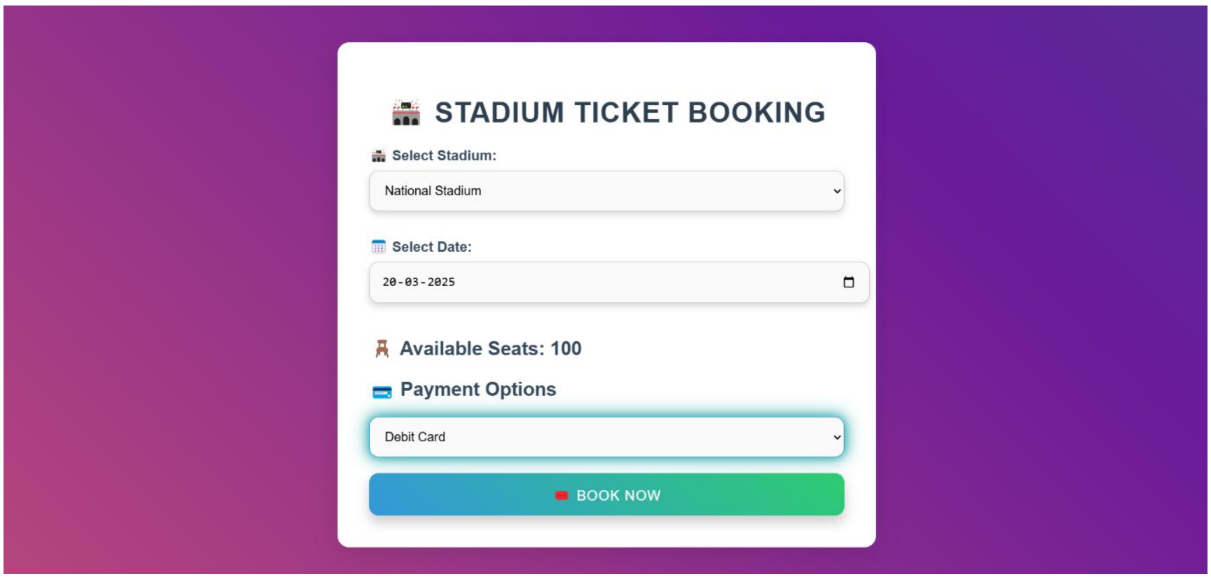
To: Los Angeles

Select Train: Express 101

Select Date: 20-03-2025

Available Seats: 100

Payment Options: Credit Card



STADIUM TICKET BOOKING

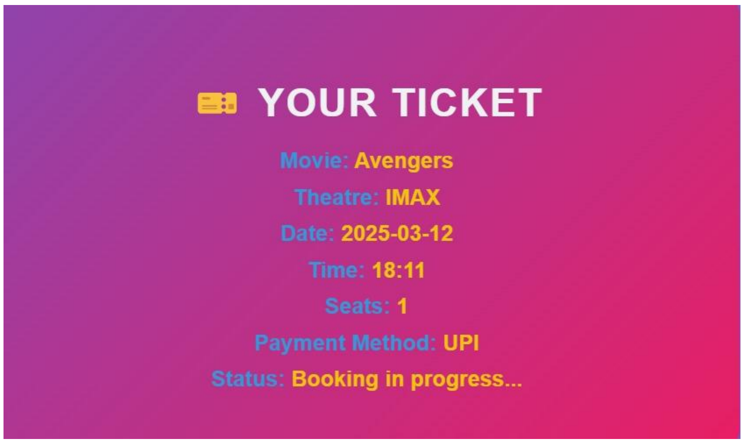
Select Stadium: National Stadium

Select Date: 20-03-2025

Available Seats: 100

Payment Options: Debit Card

Ticket Confirmation:



YOUR TICKET

Movie: Avengers

Theatre: IMAX

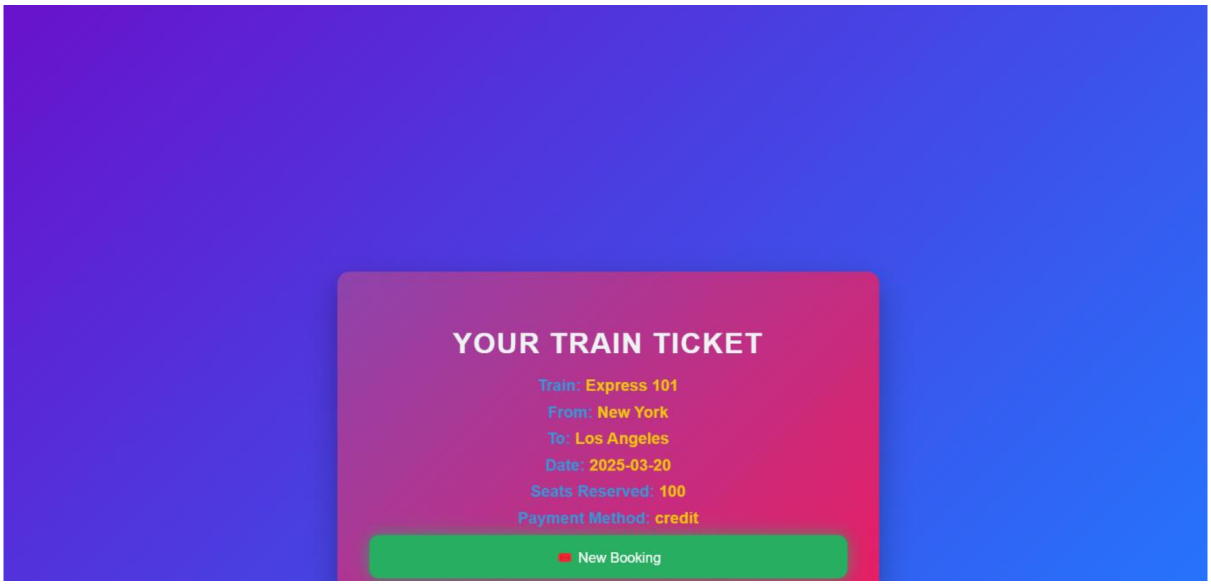
Date: 2025-03-12

Time: 18:11

Seats: 1

Payment Method: UPI

Status: Booking in progress...



YOUR TRAIN TICKET

Train: Express 101

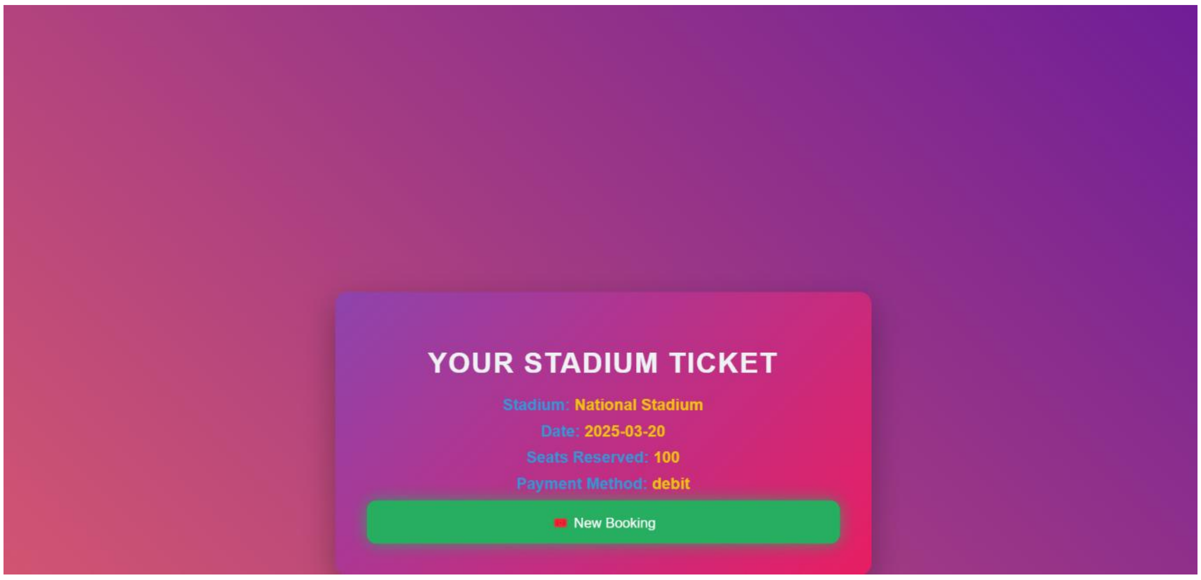
From: New York

To: Los Angeles

Date: 2025-03-20

Seats Reserved: 100

Payment Method: credit



YOUR STADIUM TICKET

Stadium: National Stadium

Date: 2025-03-20

Seats Reserved: 100

Payment Method: debit

6.Conclusion

ToDeveloping a unified, efficient platform for movie, train, and stadium ticket booking is crucial in addressing the increasing demand for convenience and simplicity. By integrating features like real-time availability, secure payment processing, and easy cancellation, we can significantly improve the user experience and reduce frustrations associated with managing multiple platforms. Technologies like machine learning for personalized recommendations, cloud solutions for scalability, and modern front-end frameworks will play a vital role in enhancing the platform's functionality. The goal is to offer a seamless, efficient booking experience that serves all user needs under one roof.

Important Topics and Main Points:

Unified Ticketing Platform: A single system to manage bookings for movies, trains, and stadiums.

Real-Time Availability: Integration of APIs for live data updates to ensure users see accurate availability.

Secure Payment Systems: Implementation of payment gateways like **Stripe** and **PayPal** to ensure secure transactions.

Personalized Recommendations: Using **machine learning** to analyze user behavior and suggest relevant events or tickets.

User-Friendly Interface: Simple, intuitive design using modern frameworks like **React** or **Vue.js** for a seamless user experience.

Cross-Platform Accessibility: Ensuring the platform is accessible on desktops and mobile devices through responsive web design or mobile apps.

Easy Cancellation Process: Simplified refund and cancellation workflows to enhance user satisfaction.

Scalability: Using cloud solutions like **AWS** or **Google Cloud** to scale the platform during high traffic.

Data Privacy and Security: Ensuring all user data, especially payment information, is encrypted and protected.

Streamlined Booking Process: Minimizing steps to complete a booking, reducing complexity and time spent by users.

Real-Time Notifications: Sending users updates about their bookings, cancellations, or changes to events.

Database Management: Efficient storage and management of user and booking data using **SQL** or **NoSQL** databases.

API Integration: Connecting with third-party services for real-time data like seat availability, pricing, and event schedules.

7.References

references for your online ticket booking project, you can include sources related to **web development, ticket booking systems, online payment security, and user experience design**. Here are some possible references you can use:

Books & Research Papers:

Laudon, K. C., & Traver, C. G. (2020). *E-Commerce 2020: Business, Technology, and Society*. Pearson.

Sharma, S., & Singh, P. (2019). *A Study on Online Ticket Booking Systems and Their Impact on the Market.* International Journal of Computer Science & Information Technology, 11(2), 45-58.

Web Development & UI/UX Design:

W3Schools. (2024). *HTML, CSS, and JavaScript for Web Development.* Retrieved from https://www.w3schools.com

MDN Web Docs. (2024). *Building Accessible and User-Friendly Interfaces.* Retrieved from https://developer.mozilla.org

Project Link:

https://github.com/NarasimhaReddy969-web/fieldproject.git