Event Booking System

Name : Mithilesh Kumar

Roll No : 205122048

Backend Requirements

The backend system implements comprehensive CRUD operations for Users, Events, and Bookings with robust data management capabilities. The core functionality enables event booking by creating booking entries with strict seat limit validation to prevent overbooking scenarios.

The system handles concurrent booking requests effectively to avoid race conditions, ensuring that multiple users attempting to book the same event simultaneously don't cause data inconsistencies. Duplicate booking prevention is implemented to ensure the same user cannot book the same event twice, maintaining data integrity throughout the booking process.

Real-time seat availability tracking provides automatic updates across all system components, while comprehensive validation and error handling ensure data integrity at all levels. The backend is designed with efficient query mechanisms for event availability checking and user booking history retrieval.

Frontend Requirements - Event Booking Dashboard

The frontend provides a comprehensive dashboard for managing users, events, and bookings with real-time connectivity to the backend API. The interface delivers immediate updates for seat availability and booking status changes.

Events Section

The events management interface displays all events with complete information including name, date/time, total seats, booked seats, and current availability status. Users can add new events with comprehensive details including title, description, date/time, and seat capacity. The system supports editing existing event information with built-in validation constraints and allows deletion of events with automatic cascade booking removal.

A sophisticated color-coding system provides immediate visual feedback: green indicators for available events, orange for limited availability, and red for sold-out events. Advanced filtering capabilities allow users to view events by availability status including all events, available only, limited availability, or sold-out events.

Users Section

The user management system provides a complete interface for viewing all registered users with their contact information. New user registration includes name and email validation with duplicate email prevention. The system supports editing existing user profiles while maintaining data integrity and enables user deletion with automatic booking cancellation.

Booking Management

The booking interface allows users to book available events through an intuitive and user-friendly system. Real-time seat availability checking occurs before booking confirmation to prevent overbooking scenarios. The system displays comprehensive booking information including user details, event information, and booking status.

Bonus Features

The application features responsive design optimized for both mobile and desktop viewing experiences. Modal confirmations provide secure user interaction for all critical operations including deletions, bookings, and cancellations. Real-time availability updates occur without requiring page refresh, while professional UI design includes modern styling and smooth animations. Comprehensive error handling provides user-friendly feedback messages, and loading states enhance user experience during API operations.

Technology Stack

The application utilizes React.js with functional components and modern hooks for the frontend development, providing a responsive and interactive user experience. The backend is built using Node.js with the Express.js framework for robust API development. Data storage uses an **in-memory** database system for demonstration purposes, while API communication is handled through Axios with custom interceptors for error handling and request logging.

Modern CSS3 with flexbox and grid layouts ensures responsive design across all devices. Backend validation is implemented using express-validator for comprehensive input validation, while state management utilizes React's built-in useState and useEffect hooks for efficient component state handling.

Sample Code Snippets

1. Backend: Create Event Booking (Node.js Express)

```
app.post('/api/bookings', [
  body('userId').isInt({ min: 1 }).withMessage('Valid user ID is required'),
  body('eventId').isInt({ min: 1 }).withMessage('Valid event ID is required')
], (req, res) => {
  const errors = validationResult(req);
  if (!errors.isEmpty()) {
    return res.status(400).json({ errors: errors.array() });
  }

  const { userId, eventId } = req.body;

  // Check if user already booked this event
  if (hasUserBookedEvent(userId, eventId)) {
    return res.status(400).json({ error: 'User has already booked this event'
});
  }
  // Check seat availability with concurrency protection
  const availableSeats = getAvailableSeats(eventId);
  if (availableSeats <= 0) {
    return res.status(400).json({ error: 'No seats available for this event'
});
  }

  // Create booking and update seat count
  const newBooking = {
    id: nextBookingId++,
    userId,
    eventId
  };

  bookings.push(newBooking);
  const eventIndex = events.findIndex(e => e.id === eventId);
  events[eventIndex].bookedSeats += 1;

  res.status(201).json(newBooking);
});
```

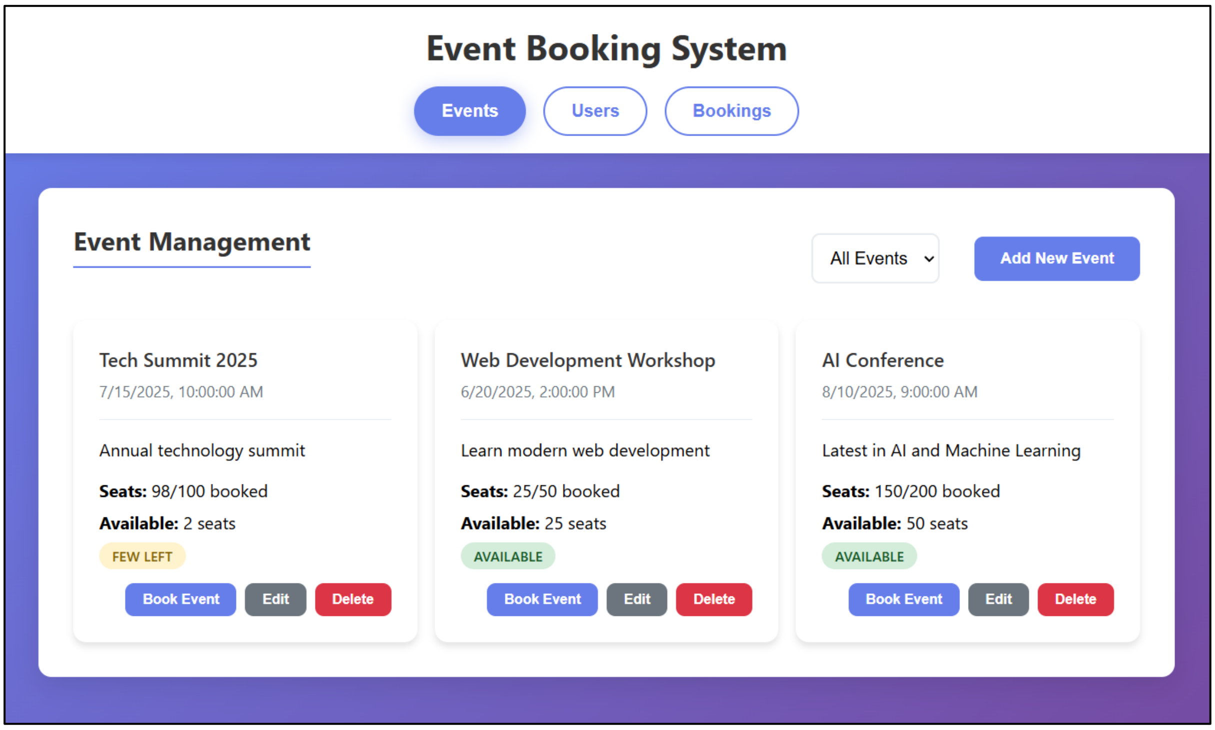
2. Frontend: Event Management Component (React + Axios)

```
import React, { useState, useEffect } from 'react';
import { eventsApi, usersApi, bookingsApi } from '../services/api';

const EventManagement = () => {
  const [events, setEvents] = useState([]);
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    fetchEvents();
  }, []);

  const fetchEvents = async () => {
    try {
      setLoading(true);
      const response = await eventsApi.getAll();
      setEvents(response.data);
    } catch (err) {
      setError('Failed to fetch events');
    } finally {
      setLoading(false);
    }
  };

  const handleBookEvent = async (eventId, userId) => {
    try {
      await bookingsApi.create({ userId, eventId });
      setSuccess('Booking created successfully!');
      fetchEvents(); // Refresh to show updated seat counts
    } catch (err) {
      setError(err.response?.data?.error || 'Failed to create booking');
    }
  };

  return (
    <div className="grid grid-2">
      {events.map(event => (
        <div key={event.id} className="card">
          <div className="card-header">
            <div className="card-title">{event.name}</div>
            <div className="card-
subtitle">{formatDateTime(event.dateTime)}</div>
          </div>
          <div className="card-content">
            <p>Available: {event.availableSeats} seats</p>
            <span className={`status-badge status-${event.status}`}>
              {event.status === 'available' ? 'Available' :
               event.status === 'limited' ? 'Few Left' : 'Sold Out'}
            </span>
          </div>
          <div className="card-actions">
            <button
              onClick={() => handleBookEvent(event.id)}
              disabled={event.availableSeats === 0}
              className="btn btn-primary"
            >
              Book Event
            </button>
          </div>
        </div>
      ))}
    </div>
  );
};
```

Project Architecture

1. Frontend (Presentation Layer)

The presentation layer is built using React.js with functional components and modern hooks, providing a responsive and interactive user experience. The component-based architecture ensures reusability and maintainability across the application. A mobile-first responsive design approach utilizes CSS Grid and Flexbox for optimal viewing across all device types. Real-time state management through React hooks enables immediate UI updates, while centralized API integration through Axios provides consistent data communication patterns.


Event Booking System

Events | Users | Bookings

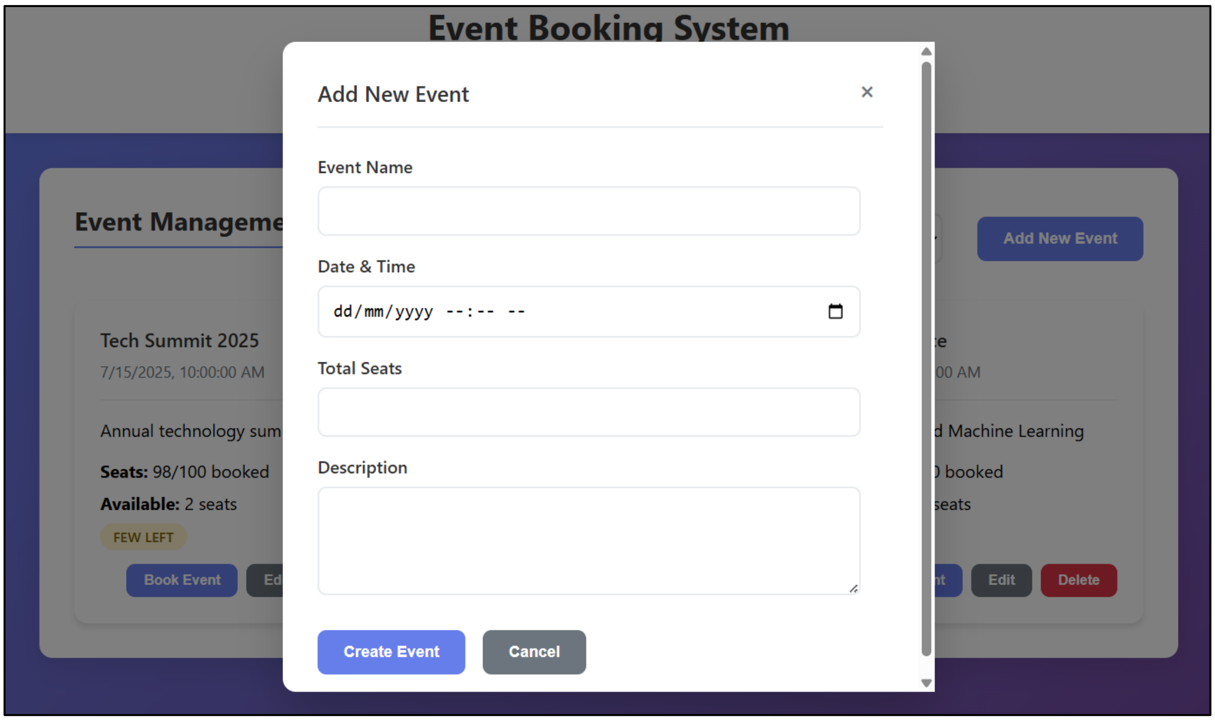
Event Management

All Events | Add New Event

Tech Summit 2025
7/15/2025, 10:00:00 AM
Annual technology summit
Seats: 98/100 booked
Available: 2 seats
FEW LEFT
Book Event | Edit | Delete

Web Development Workshop
6/20/2025, 2:00:00 PM
Learn modern web development
Seats: 25/50 booked
Available: 25 seats
AVAILABLE
Book Event | Edit | Delete

AI Conference
8/10/2025, 9:00:00 AM
Latest in AI and Machine Learning
Seats: 150/200 booked
Available: 50 seats
AVAILABLE
Book Event | Edit | Delete


Add New Event ×

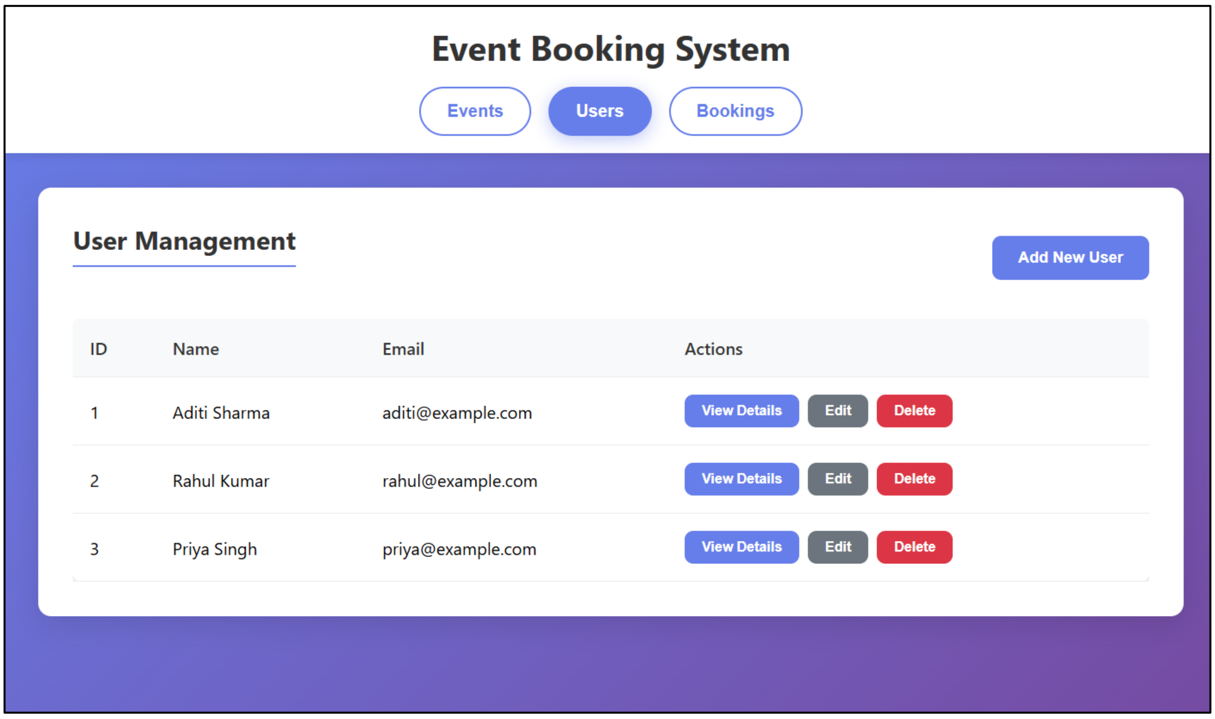
Event Name

Date & Time
dd/mm/yyyy --:-- --

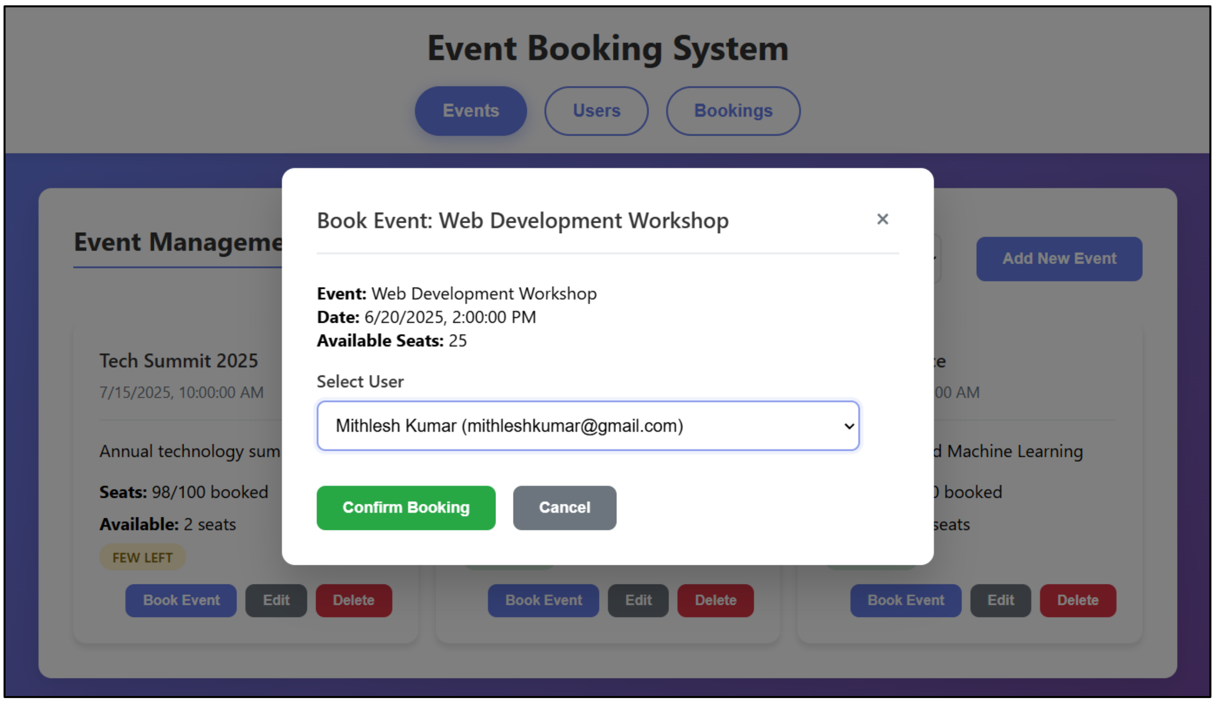
Total Seats

Description

Create Event | Cancel



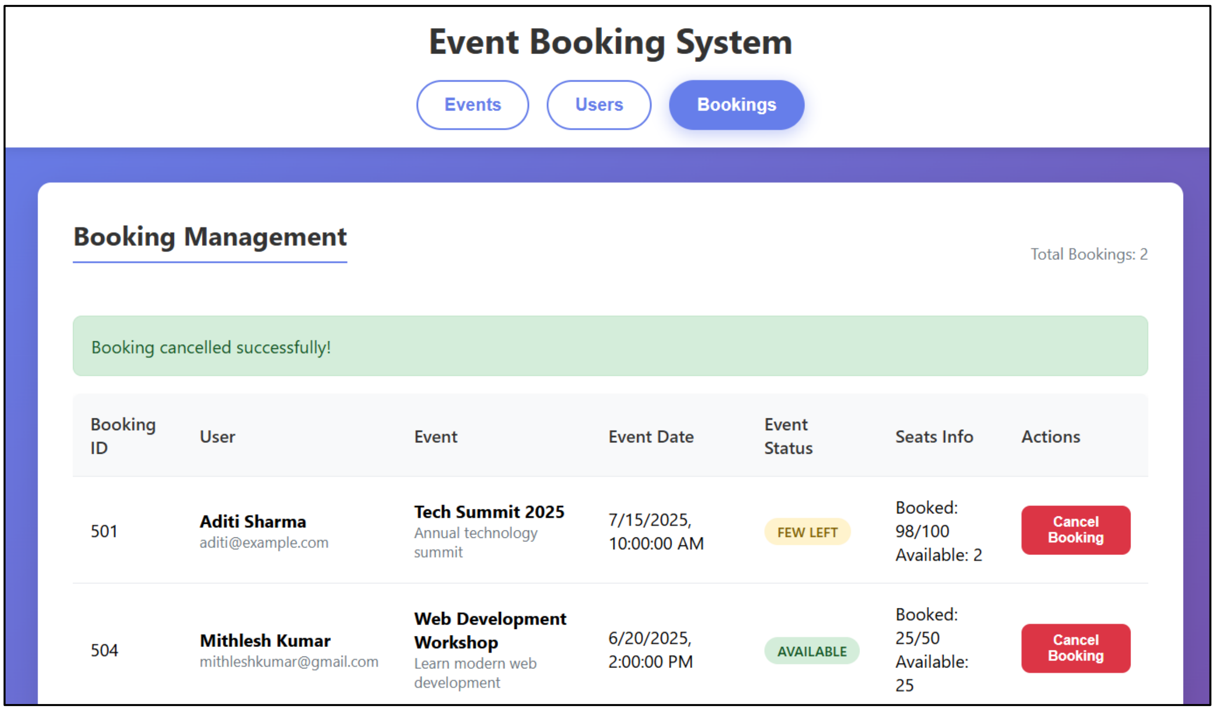
ID	Name	Email	Actions
1	Aditi Sharma	aditi@example.com	View Details Edit Delete
2	Rahul Kumar	rahul@example.com	View Details Edit Delete
3	Priya Singh	priya@example.com	View Details Edit Delete



Book Event: Web Development Workshop

Event: Web Development Workshop
Date: 6/20/2025, 2:00:00 PM
Available Seats: 25

Select User
Mithlesh Kumar (mithleshkumar@gmail.com)



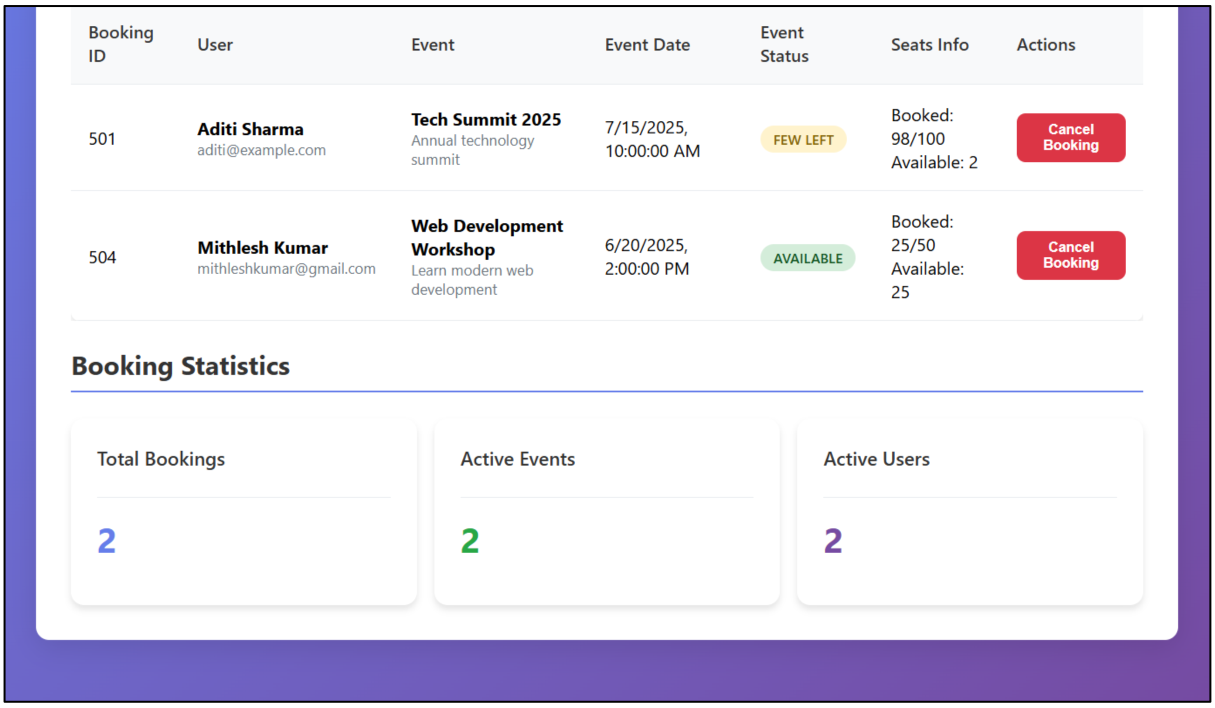
Event Booking System

Booking Management

Total Bookings: 2

Booking cancelled successfully!

Booking ID	User	Event	Event Date	Event Status	Seats Info	Actions
501	**Aditi Sharma** aditi@example.com	**Tech Summit 2025** Annual technology summit	7/15/2025, 10:00:00 AM	FEW LEFT	Booked: 98/100 Available: 2	Cancel Booking
504	**Mithlesh Kumar** mithleshkumar@gmail.com	**Web Development Workshop** Learn modern web development	6/20/2025, 2:00:00 PM	AVAILABLE	Booked: 25/50 Available: 25	Cancel Booking



Booking ID	User	Event	Event Date	Event Status	Seats Info	Actions
501	**Aditi Sharma** aditi@example.com	**Tech Summit 2025** Annual technology summit	7/15/2025, 10:00:00 AM	FEW LEFT	Booked: 98/100 Available: 2	Cancel Booking
504	**Mithlesh Kumar** mithleshkumar@gmail.com	**Web Development Workshop** Learn modern web development	6/20/2025, 2:00:00 PM	AVAILABLE	Booked: 25/50 Available: 25	Cancel Booking

Booking Statistics

Total Bookings	Active Events	Active Users
2	2	2

2.Backend (Application Layer)

The application layer implements a RESTful API using Node.js and Express.js framework, providing robust and scalable server-side functionality. Input validation through express-validator ensures data integrity and security at the API level. Advanced concurrency handling prevents race conditions during simultaneous booking operations, while comprehensive error handling provides proper HTTP status codes and descriptive error messages. CORS configuration enables secure cross-origin resource sharing for frontend-backend communication.

3.Data Layer

The data layer utilizes in-memory storage through JavaScript arrays for demonstration purposes, providing immediate data access without external database dependencies. Simulated database operations implement proper CRUD functionality with data relationship management through foreign key references. Real-time calculations for seat availability and booking statistics provide immediate feedback, while data persistence during runtime includes automatic ID generation and relationship maintenance.

```
// In-memory database simulation
let users = [
  { id: 1, name: "Aditi Sharma", email: "aditi@example.com" },
  { id: 2, name: "Rahul Kumar", email: "rahul@example.com" },
  { id: 3, name: "Priya Singh", email: "priya@example.com" }
];

let events = [
  {
    id: 101,
    name: "Tech Summit 2025",
    dateTime: "2025-07-15T10:00:00",
    totalSeats: 100,
    bookedSeats: 98,
    description: "Annual technology summit"
  },
  {
    id: 102,
    name: "Web Development Workshop",
    dateTime: "2025-06-20T14:00:00",
    totalSeats: 50,
    bookedSeats: 25,
    description: "Learn modern web development"
  },
  {
    id: 103,
    name: "AI Conference",
    dateTime: "2025-08-10T09:00:00",
    totalSeats: 200,
    bookedSeats: 150,
    description: "Latest in AI and Machine Learning"
  }
];
```

```
let bookings = [
  { id: 501, userId: 1, eventId: 101 },
  { id: 502, userId: 2, eventId: 102 },
  { id: 503, userId: 3, eventId: 103 }
];
```

This separation of concerns ensures scalability, maintainability, and clear responsibility across layers.