GraphQL

This project is a seat booking application built using HTML, JavaScript, Express.js, and GraphQL. The HTML file provides a user interface where users can view available seats and book them. Users can click on available seats to select them for booking. Once a seat is selected, the "Book Ticket" button becomes enabled, allowing users to confirm the booking.

The JavaScript code handles seat selection and booking functionality. It sends a GraphQL mutation request to the server when a user clicks the "Book Ticket" button. The Express.js server uses GraphQL to manage seat data and bookings. It defines a GraphQL schema with queries and mutations for fetching seats and booking seats, respectively.

Overall, this application provides a simple and intuitive interface for users to browse available seats and book them in real-time.

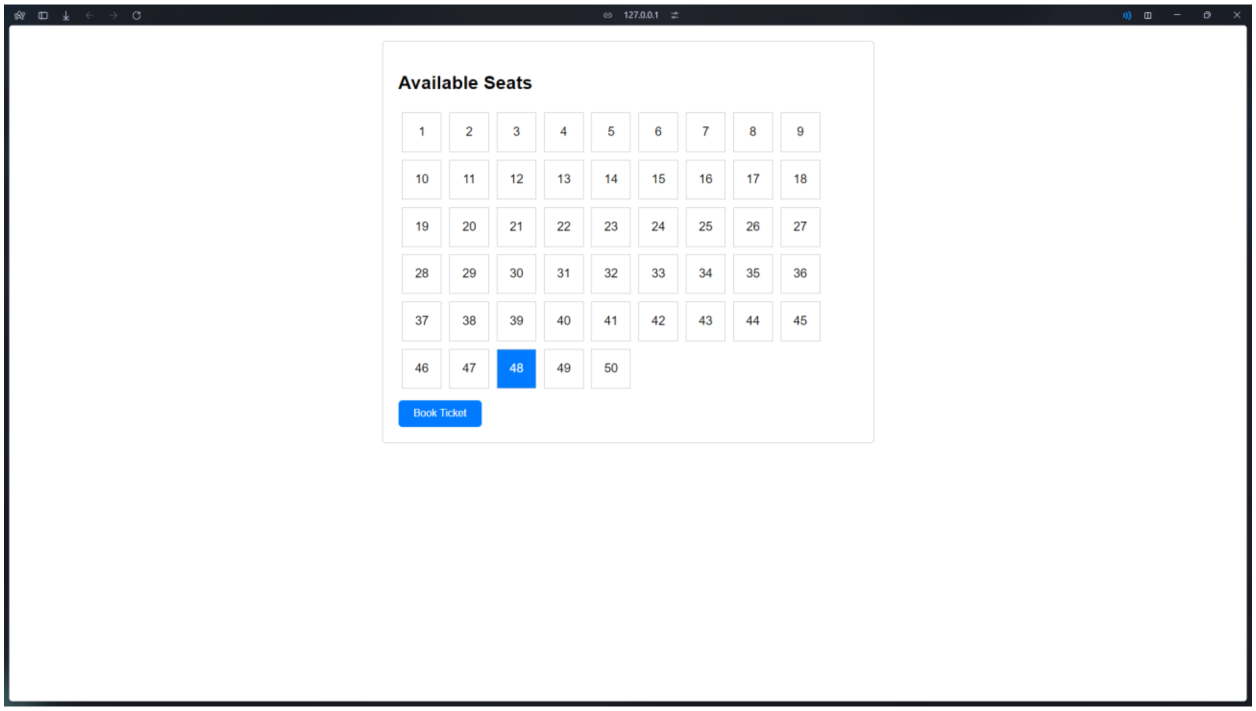
App.js

```
const express = require('express');
const { graphqlHTTP } = require('express-graphql');
const { buildSchema } = require('graphql');
const cors = require('cors');

// Sample data - seats and bookings
// Sample data - seats and bookings
let seats = Array(50)
  .fill()
  .map((_, index) => ({ id: String(index + 1), booked: false }));

let bookings = [];

// Construct a schema, using GraphQL schema language
const schema = buildSchema(`
  type Query {
    seats: [Seat]
  }

  type Seat {
    id: ID!
    booked: Boolean!
  }
  type Booking {
    id: ID!
    seatId: ID!
    userId: ID!
    createdAt: String!
  }

  type Mutation {
    bookSeat(seatId: ID!): Booking
  }
`);

// The root provides a resolver function for each API endpoint
const root = {
  seats: () => seats,
  bookSeat: ({ seatId }) => {
    const seatIndex = seats.findIndex(seat => seat.id === seatId);
    if (seatIndex === -1 || seats[seatIndex].booked) {
      throw new Error('Seat not available');
    }
    const booking = { id: String(bookings.length + 1), seatId, userId: 'user123',
createdAt: new Date().toISOString() };
    seats[seatIndex].booked = true;
    bookings.push(booking);
    return booking;
  },
};

// Create an express server and a GraphQL endpoint
const app = express();
app.use(cors)
app.use('/graphql', graphqlHTTP({
  schema: schema,
  rootValue: root,
  graphiql: true, // Enable GraphiQL for easy testing
}));

const port = 4000;
app.listen(port, () => {
  console.log(`Server is running on http://localhost:${port}/graphql`);
});
```

Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Seat Booking</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      margin: 0;
      padding: 0;
    }
    .container {
      max-width: 600px;
      margin: 20px auto;
      padding: 20px;
      border: 1px solid #ccc;
      border-radius: 5px;
    }
    .seat {
      display: inline-block;
      width: 50px;
      height: 50px;
      margin: 5px;
      border: 1px solid #ccc;
      text-align: center;
      line-height: 50px;
      cursor: pointer;
    }
    .booked {
      background-color: #f00;
      color: #fff;
      cursor: not-allowed;
    }
    .selected {
      background-color: #007bff;
      color: #fff;
    }
    .book-btn {
      display: block;
      margin-top: 10px;
      padding: 10px 20px;
      background-color: #007bff;
      color: #fff;
      border: none;
      border-radius: 5px;
      cursor: pointer;
    }
  </style>
</head>
<body>
  <div class="container">
    <h2>Available Seats</h2>
    <div id="seatsContainer"></div>
    <button id="bookBtn" class="book-btn" disabled>Book Ticket</button>
  </div>

  <script>
    document.addEventListener("DOMContentLoaded", function() {
      const seatsContainer = document.getElementById('seatsContainer');
      const bookBtn = document.getElementById('bookBtn');
      let selectedSeatId = null;

      // Create containers for all seats beforehand
      for (let i = 1; i <= 50; i++) {
        const seatElement = document.createElement('div');
        seatElement.className = 'seat';
        seatElement.textContent = i;
        seatElement.setAttribute('data-seat-id', i);
        seatElement.addEventListener('click', () => selectSeat(i));
        seatsContainer.appendChild(seatElement);
      }

      function selectSeat(seatId) {
        const selectedSeatElement = document.querySelector(`[data-seat-
id="${seatId}"]`);
        if (!selectedSeatElement.classList.contains('booked')) {
          if (selectedSeatId) {
            const previousSelectedSeatElement = document.querySelector(`[data-
seat-id="${selectedSeatId}"]`);
            previousSelectedSeatElement.classList.remove('selected');
          }
          selectedSeatElement.classList.add('selected');
          selectedSeatId = seatId;
          bookBtn.removeAttribute('disabled');
        }
      }
      bookBtn.addEventListener('click', () => bookTicket());

      function bookTicket() {
        if (selectedSeatId) {
          fetch('http://localhost:4000/graphql', {
            method: 'POST',
            headers: { 'Content-Type': 'application/json' },
            body: JSON.stringify({ query: `mutation { bookSeat(seatId:
"${selectedSeatId}") { id } }` }),
          })
          .then(res => res.json())
          .then(data => {
            if (data.errors) {
              alert(data.errors[0].message);
            } else {
              alert(`Seat ${selectedSeatId} booked successfully!`);
              document.querySelector(`[data-seat-
id="${selectedSeatId}"]`).classList.add('booked');
              document.querySelector(`[data-seat-
id="${selectedSeatId}"]`).classList.remove('selected');
              selectedSeatId = null;
              bookBtn.setAttribute('disabled', true);
            }
          })
          .catch(error => console.error('Error booking seat:', error));
        } else {
          alert('Please select a seat before booking.');
        }
      }
    });
  </script>
</body>
</html>
```

```
  "dependencies": {
    "cors": "^2.8.5",
    "express": "^4.18.2",
    "express-graphql": "^0.12.0",
    "graphql": "^15.8.0"
  }
```


127.0.0.1
Available Seats
1 2 3 4 5 6 7 8 9
10 11 12 13 14 15 16 17 18
19 20 21 22 23 24 25 26 27
28 29 30 31 32 33 34 35 36
37 38 39 40 41 42 43 44 45
46 47 48 49 50
Book Ticket