GreenStitch

GreenStitch Technologies PVT. LTD. (CIN: U62013KA2023PTC172387)

8th Floor, iSprout - Shilpitha Tech Park, ORR, Bellandur, Bengaluru, Karnataka 560103

Frontend Technical Assessment - *Seat Booking System*

Introduction

Thank you for taking the time to interview with us at **GreenStitch**! As part of the interview process, we would like you to complete a frontend technical assessment.

You can find all the files necessary for the assignment in the **GreenStitch Frontend Assessment Materials** folder. Feel free to make any changes to the provided files, including adding new files, installing new packages, and modifying any provided code. Please use **JavaScript/React** for this assessment.

Project Overview

A seat booking system with UI implemented.

Your job: Implement the booking logic.

What's Provided:

- Complete UI with 8×10 seat grid
- Three seat states: Available (grey), Selected (green), Booked (red)
- Pricing tiers: Premium (₹1000), Standard (₹750), Economy (₹500)
- State management structure

Requirements

Functional Requirements:

➢ Seat Selection

Users must be able to click on available seats to select them for booking. Selected seats should be visually distinct from available and booked seats.

➢ Seat Pricing

The system implements tiered pricing based on seat location:

- Premium seats (Rows A-C): ₹1000 per seat

- Standard seats (Rows D-F): ₹750 per seat
- Economy seats (Rows G-H): ₹500 per seat

➢ Price Calculation

The system must display the total price of all currently selected seats in real-time as users select or deselect seats.

➢ Seat Counters

The interface must show live counts of:

- Available seats (not selected, not booked)
- Selected seats (currently selected by user)
- Booked seats (confirmed bookings)

➢ Booking Validation

- Users cannot book more than 8 seats in a single transaction
- Attempting to book more than 8 seats must show an error message
- Booked seats cannot be selected or modified

➢ Seat Selection Continuity Rule

- Users may not leave any **available seat** isolated between their selected seats.
- Seat selection must not create a pattern like:

```
[Selected/Booked] [Available] [Selected/Booked]
```

- However, **gaps caused by already booked seats are allowed**.
- Non-continuous selections are permitted only when the gap is due to booked seats, not available ones.

➢ Booking Confirmation

Before confirming a booking, the system must:

- Show the number of seats being booked
- Display the total price
- Request user confirmation
- Only proceed with booking if confirmed

➢ Booking Persistence

All confirmed bookings must persist across page refreshes. When users reload the page, their booked seats must remain booked.

➢ Clear Selection

Users must be able to clear their current seat selection without affecting already booked seats.

➢ System Reset

The system must provide functionality to reset all seats to the available state and clear all persisted booking data.

Technical Implementation

➢ Getting Started:

Navigate to the project directory and run:

- npm install – Install dependencies
- npm start – Start development server
- Open http://localhost:3000 in your browser

➢ Code Structure

The main implementation file is src/SeatBooking.js. You will find:

- Seat state constants and pricing tiers are already defined
- UI components and layout already implemented
- Functions marked with TODO that need implementation
- The seat grid structure (8 rows × 10 seats) was initialised

➢ Implementation Guidelines

- Maintain React best practices (immutable state updates)
- Use React hooks appropriately (useState, useEffect)
- Implement proper error handling
- Ensure the UI updates correctly after each action
- Test all edge cases before submission

➢ Creativity & Innovation

Going above and beyond is encouraged. While meeting the core requirements is essential, we highly value creativity and original thinking. Feel free to add features, improve the user experience, or implement innovative solutions beyond the base requirements. There are no limitations to creativity.

Submission Requirements

Your submission must include the following components:

1. Code Implementation
   - Complete implementation of all three required functions
   - All functionality is working as specified
   - No console errors or warnings
   - Clean, well-organised code

3. Git History
   - Initialise a git repository if not already done: git init
   - Make **meaningful commits** as you work on different parts
   - Write clear, descriptive commit messages
   - We want to see your development process through your commits

4. Submission Format
   - Create a ZIP file containing:

a. The complete project folder (including .git directory for git history)
b. A link to your video walkthrough (in a VIDEO_LINK.txt file)
c. **OR** include the video file directly if the file size is reasonable

- **DO NOT** upload your code to GitHub or any public repository
- Name your ZIP file: GreenStitch-Assessment-[YourName].zip
- Ensure the ZIP includes the .git folder so we can review your commit history
- Completed submissions should be emailed directly to sonu.maria@greenstitch.io

Important Notes for Candidates

The provided UI contains several id, className, and data-* attributes that are used by our automated Playwright test suite.

You may freely add new elements, attributes, or classes if needed for your implementation.

Do NOT remove, rename, or modify any existing:

- data-testid
- data-* attributes
- id attributes
- CSS class names that are already present in the files

Modifying or deleting these identifiers will cause the automated evaluation tests to fail, even if the UI appears correct.

Ensure all existing structural and test-related attributes remain intact throughout your implementation.

Good luck, and we look forward to reviewing your submission!