THESIS

SEIFELDIN SALEH
JGDCU1

Debrecen,
2024

UNIVERSITY OF DEBRECEN
FACULTY OF INFORMATICS

Football Booking System

Supervisor :
Dr. Vagner Aniko
Associate Professor

Candidate :
Seifeldin Saleh
Computer Science Bsc.

Debrecen 2024

Table of Contents

Table of Figures

1. Introduction

In the digital realm, web applications, commonly known as web apps, primarily function within web browsers rather than a device's operating system. This approach enables a broader audience to access and interact with the web application via the World Wide Web.

The development of web applications entails creating software programs that can be accessed over the internet via remote servers.

Web applications are designed to be accessed exclusively through internet browsers such as Mozilla Firefox, Internet Explorer, and Google Chrome. They leverage essential technologies like JavaScript, CSS (Cascading Style Sheets), and HTML5 (Hyper Text Markup Language) for development. Unlike traditional software, web apps do not need to be downloaded and installed locally on devices; instead, they operate entirely online, relying on network connections to function seamlessly. This architecture allows users to access and utilize web apps from various devices without the need for extensive storage or installation procedures.

The front end of web applications is built using client-side programming, primarily relying on HTML, CSS, and JavaScript. HTML governs the structure and content display of web pages, while CSS ensures proper styling and layout. JavaScript enhances user interaction by enabling dynamic and responsive elements within the web app interface.
Complementing client-side functionality, server-side programming involves scripting languages such as Ruby, Java, and Python. These languages are instrumental in creating server-side scripts that manage data processing, generate dynamic content, and secure the underlying source code of web applications. Server-side scripting also facilitates the creation of unique user interfaces tailored to specific application requirements.
Furthermore, online applications rely on databases like MySQL or MongoDB for efficient data storage and management. These databases play a crucial role in storing and retrieving information essential for the functioning and persistence of web applications.

2. Technology Overview

2.1. JRE Stack

The project KickCrew uses JRE Stack. Java for backend development using Eclipse IDE, React.js for frontend development with Visual Studio Code, and MySQL for efficient database management. This stack ensures a comprehensive and streamlined approach to full-stack application development.

2.2. MySQL

For efficient querying and manipulation of the KickCrew application, I have used MySql Workbench.

2.3. React JS

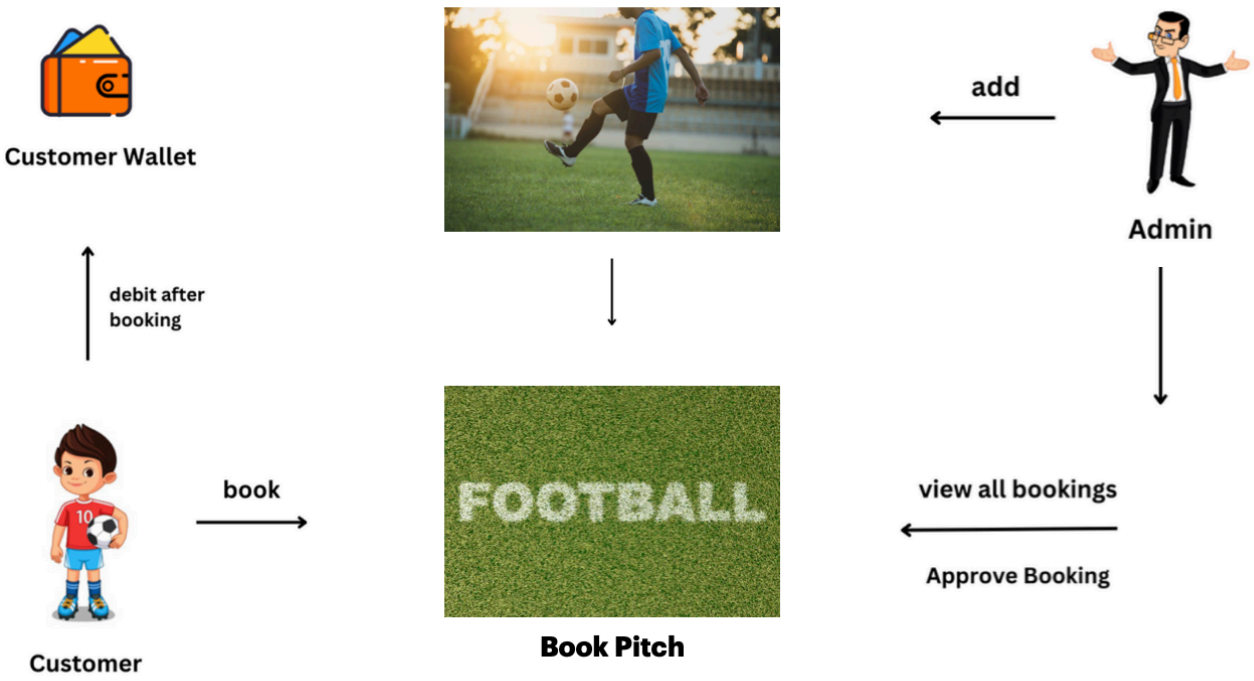
React.js was chosen for frontend development due to it's powerful and flexible nature in user interfaces. It uses a component-based approach which allows it to break down the user interface into smaller components which allows for easier managing and therefore, updates can be made a lot more efficiently. The virtual DOM updates the page a lot faster this way because it refreshes a certain component rather than the whole page. As well as, the simplicity within the React.js syntax that allows for quicker writing and understanding of the code.

2.4. Java

Choosing Java with Eclipse IDE for the backend development brought several benefits to the project. One of Java's strengths is it's platform independence. Which means, that applications that are developed in Java can work on several operating systems without needing major changes. This flexibility simplifies the development process. As well as, Java can handle large amounts of requests at once and in an efficient manner. The reason for Eclipse IDE was due to the user-friendly environment it provides.

By choosing Java with Eclipse IDE for the backend, a reliable and versatile combination was chosen. This supports the complex needs for the task management while still enabling us to build and maintain the application effectively.

Figure 1 Project Overview



KickCrew the Booking System project using React JS and Eclipse is a 2-module project where the administrator can add multiple grounds in the system. After adding the football grounds, the customers are then able to view all the available grounds on the website. However, to book the football ground they must be logged in through the website.

3. Development Lifecycle

3.1. Installment

After setting up my Java backend with everything it needed, I started working on the frontend of my application. Using Visual Studio Code, I created a React application template by running "npx create-react-app" to set up "my react app." Once that was done, I went to the project directory using the terminal and used "npm start" to get the project up and running. From there, I focused on refining the frontend user interface to fit the needs of my project.

4. Front-End

4.1. App.js

In my project's App.js file, I've set up routes for each page of the application. Think of these routes like pathways that direct users to different parts of the app. Each route corresponds to a specific page or feature, like a home page, a profile page, or maybe a page for tasks.
To make these routes work, I've imported components that represent each page. These components are like building blocks that define what users see and interact with. By organizing these routes and components in App.js, it helps keep everything structured and makes it easier for me to manage how the app flows and behaves. This was installed by going in our project terminal and writing nom install react-router-dom f.

```
import "./App.css";

import { Route, Routes } from "react-router-dom";

import AboutUs from "./page/AboutUs";

import ContactUs from "./page/ContactUs";

import Header from “./NavbarComponent/Header";

import HomePage from “./page/HomePage";

import UserRegister from “./UserComponent/UserRegister";

import UserLoginForm from “./UserComponent/UserLoginForm";

import AddGroundForm from “./GroundComponent/AddGroundForm";

import Ground from “./GroundComponent/Ground";

import ViewAllCustomer from “./UserComponent/ViewAllCusomer";

import ViewMyBooking from “./BookingComponent/ViewMyBooking”;

import ViewAllBooking from “./BookingComponent/ViewAllBooking";

import VerifyBooking from “./BookingComponent/VerifyBooking";

import MyWallet from “./UserComponent/MyWallet";

import AddReview from “./ReviewComponent/AddReview";

import ViewAllGround from “./GroundComponent/ViewAllGround";
```

```
import ViewAllTurf from "./GroundComponent/ViewAllTurf";
```

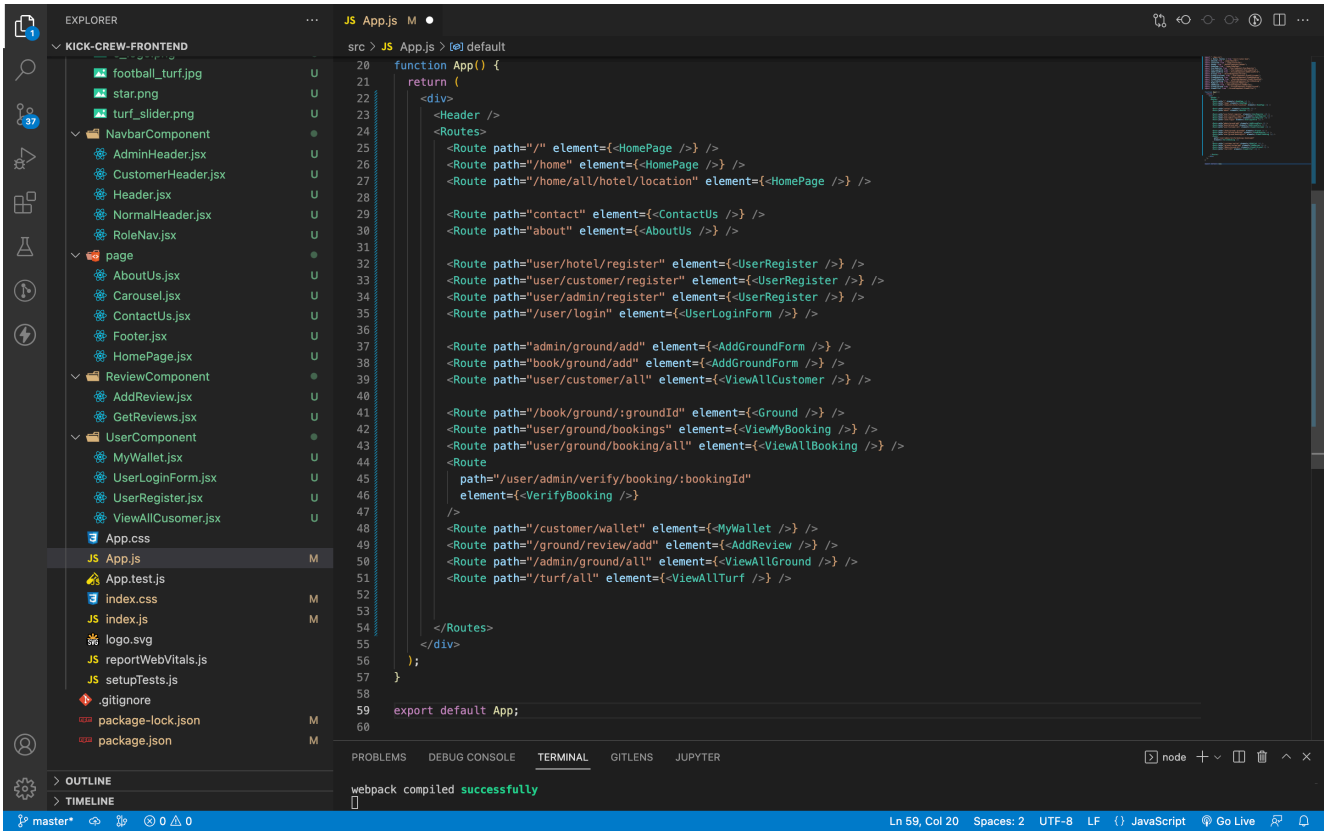
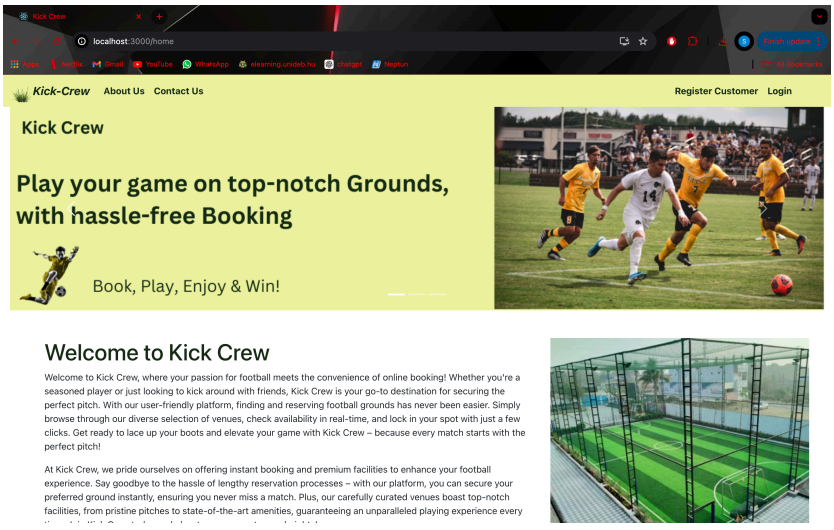


Figure 2 App.js

4.2. Home Page

Below is a snippet of the home page which shows a react component that is simply the Front page of the football pitch booking website. In the Component, we have what can be similar to a slide show. In the first slide theres a text saying (“Play your game on top-notch Grounds, with hassle-free Booking”). In the future this can be used to shuffle between advertisements or football news.

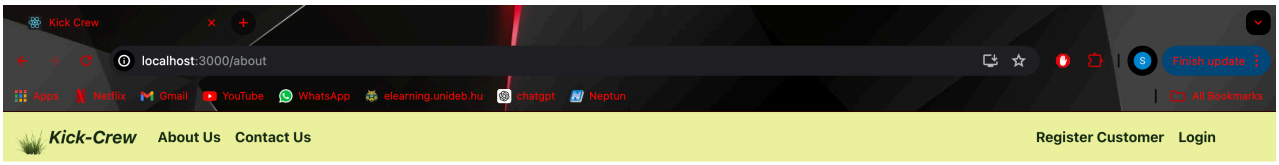


Below that component is a Welcome to KickCrew, and just below it is a short description of what the application aims to do. And the benefits of booking through KickCrew rather than the old fashioned phone call.

Figure 3 Home Page

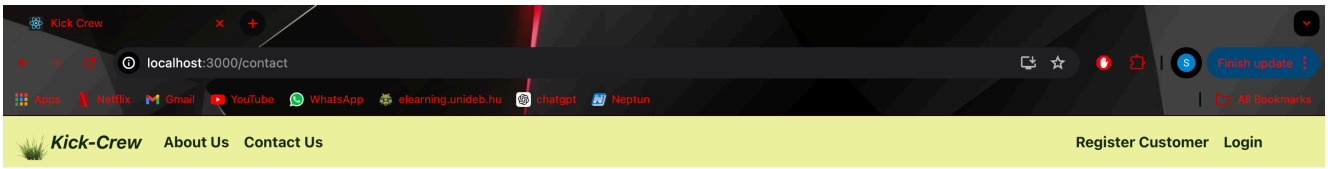
Figure 4 KickCrew HomePage links

Below the text, there is the Get Started button. Which takes you to the page that shows all the football grounds.



At Kick Crew, we're passionate about bringing football enthusiasts together and making the beautiful game more accessible than ever before. Founded with a vision to revolutionize the way football grounds are booked, Kick Crew is dedicated to providing a seamless and convenient platform for players of all levels to find and reserve pitches effortlessly. Our team is committed to delivering excellence, ensuring that every aspect of your experience with Kick Crew – from browsing available venues to stepping onto the field – is nothing short of exceptional. With a focus on instant booking and premium facilities, we're here to empower you to pursue your passion for football with ease and excitement. Join us on this journey and discover a whole new world of footballing possibilities with Kick Crew!

Figure 6 About Us



We're here to assist you every step of the way! Whether you have a question, feedback, or need support, our team at Kick Crew is ready to help.You can reach us through the following channels:

Email: info@kickcrew.com
Phone: +1 (123) 456-7890
Address: 123 Main Street, Cityville, State, Zip Code

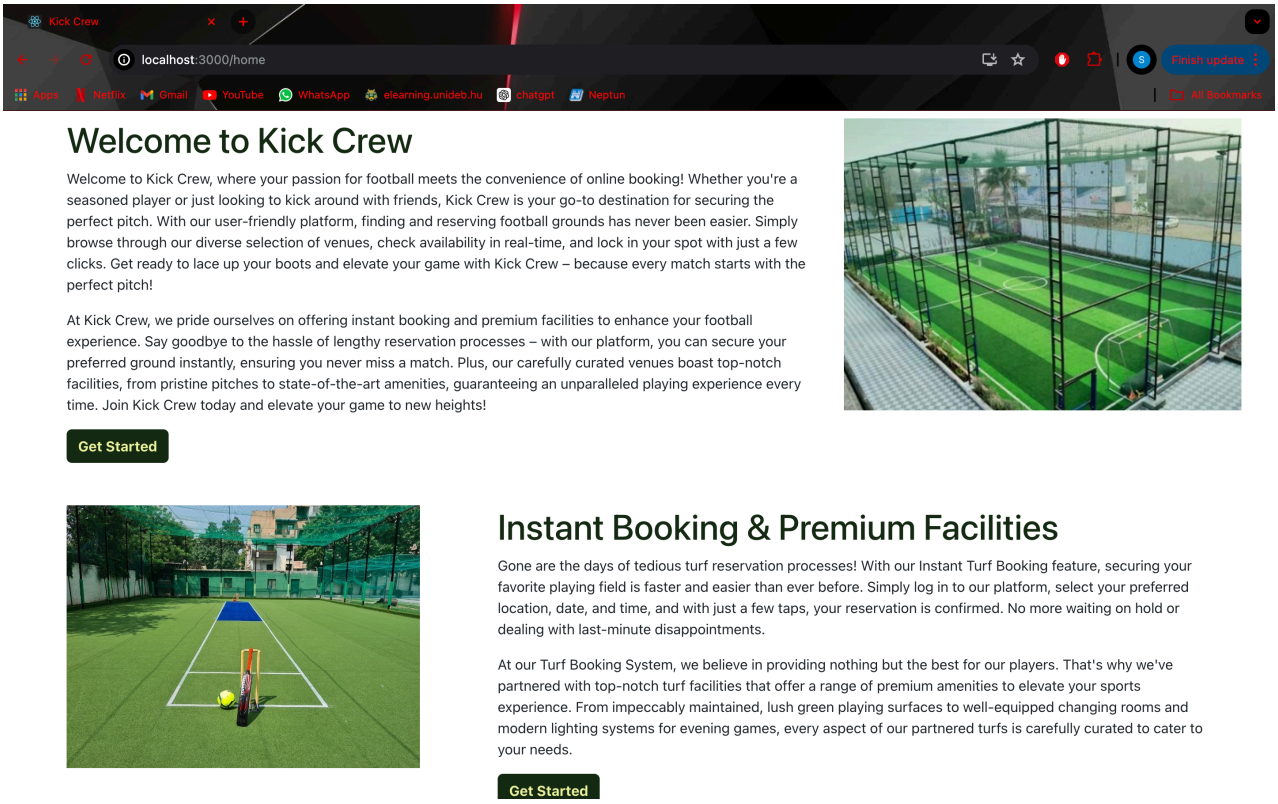
Connect with us on social media:
Facebook: @kickcrewofficial
Twitter: @kickcrew_tweets
Instagram: @kickcrewpics

Don't hesitate to get in touch – we're always eager to hear from fellow football enthusiasts like you!

localhost:3000/contact

Figure 7 Contact Us.

4.3. Navbar



The Navigation bar below is the home navigation bar before any admin or customers log in. It has the website's name which is a link at any moment to go back to the home page. An about us page. And a Contact Us page with has the email, phone number, and the address of KickCrew. As well as, the accounts on social media on applications such as: FaceBook, Twitter and Instagram.

Figure 5 NavBar Home

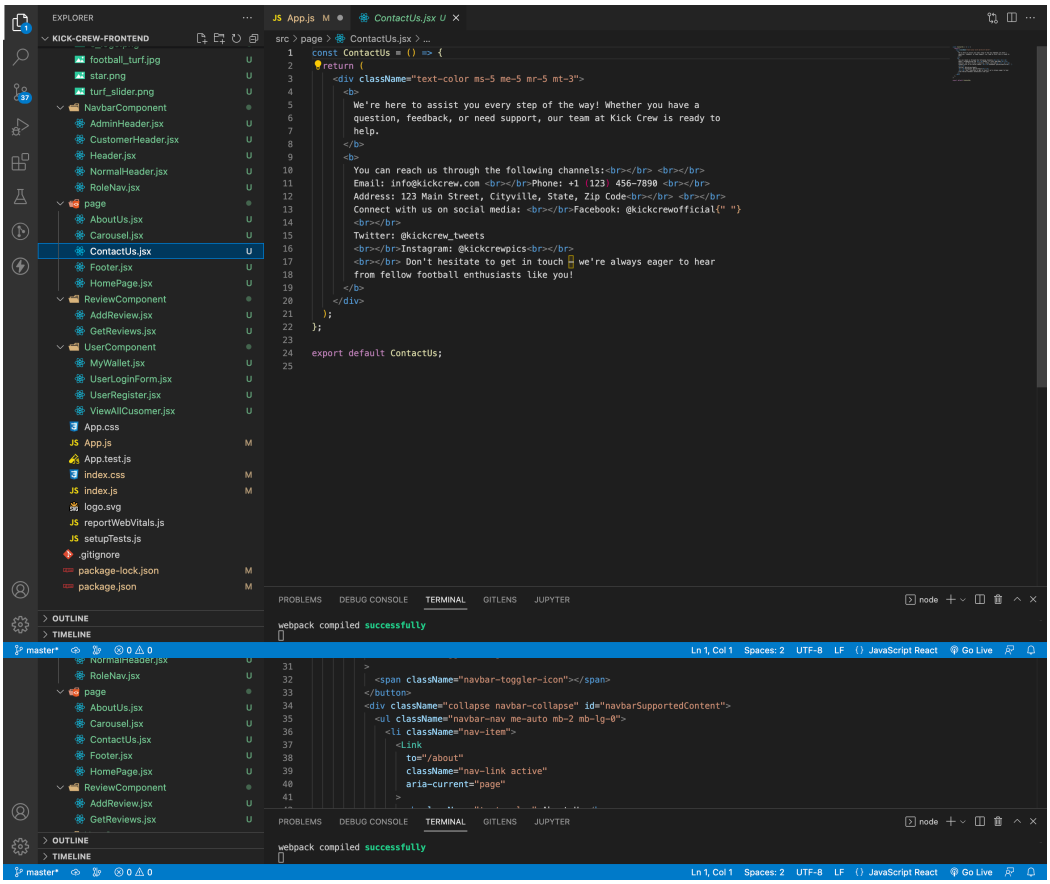


Figure 8
Header.jsx.

The Header in the NavBar Component is the general NavBar. It is used to it's purpose generally. However, this changes depending on the user logged in. Depends on if it is a user or an admin.

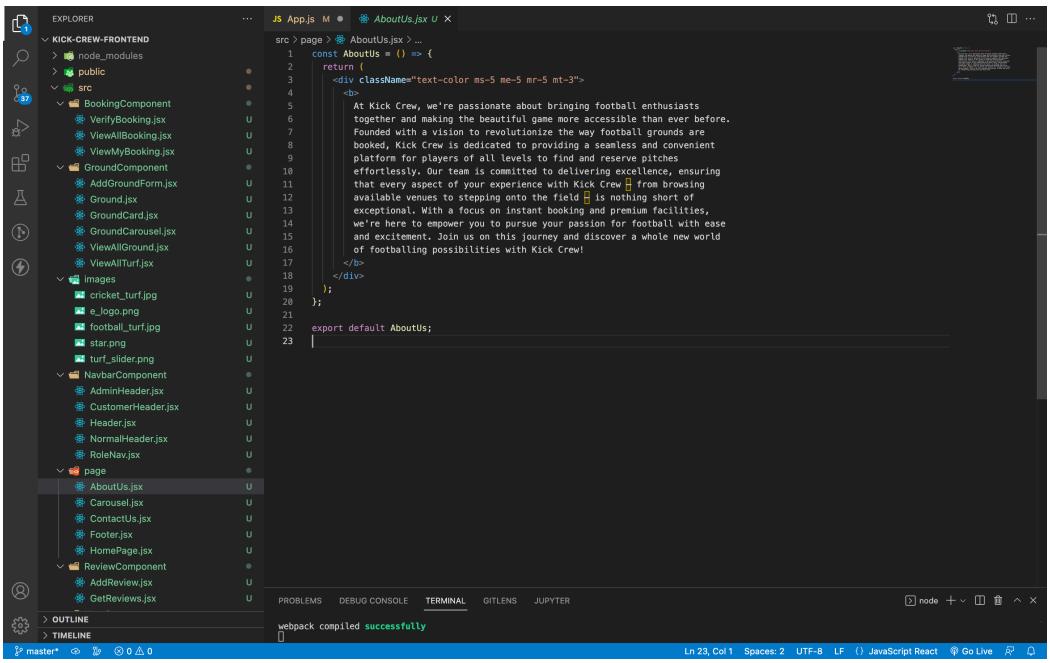


Figure 9
AboutUs.jsx

The About Us page can be accessed using any NavBar and can be accessed at any moment excluding registration and login pages.

Figure 10 ContactUs.jsx

Figure 4.4. Log in Link

Shown below is the bottom of the homepage which includes various links. The Get Started link works as a way to view all the grounds. However, not much can be done since there is no user logged in. At the bottom the Log in link directs you directly to the user login.

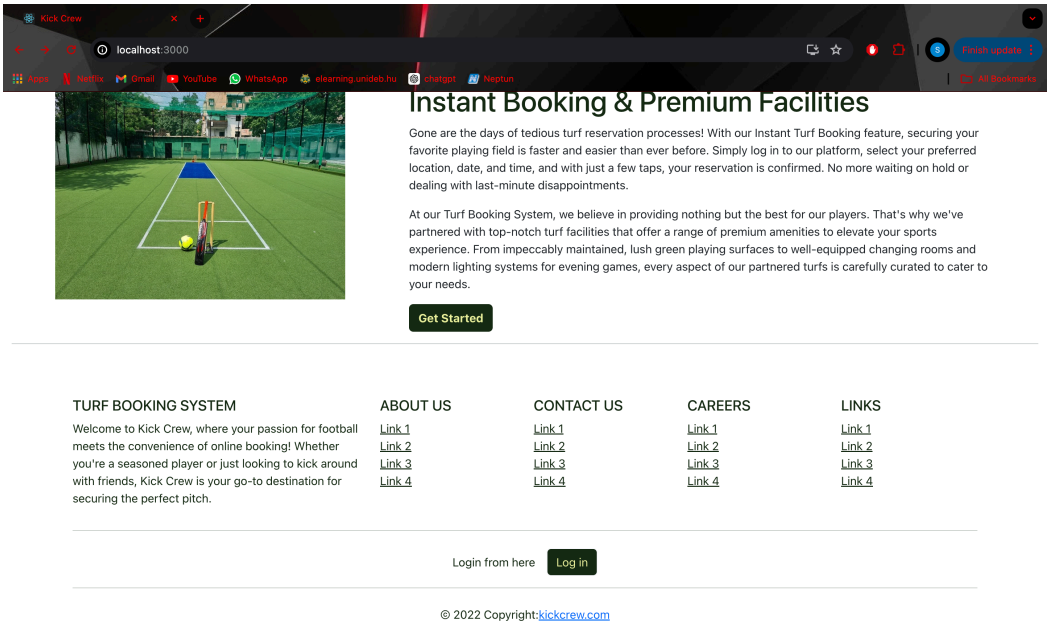


Figure 11
Login link.

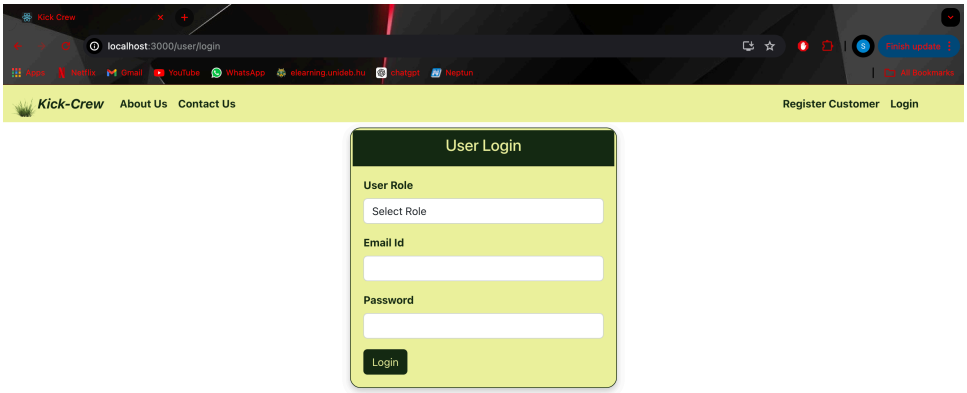


Figure 12 User Login.

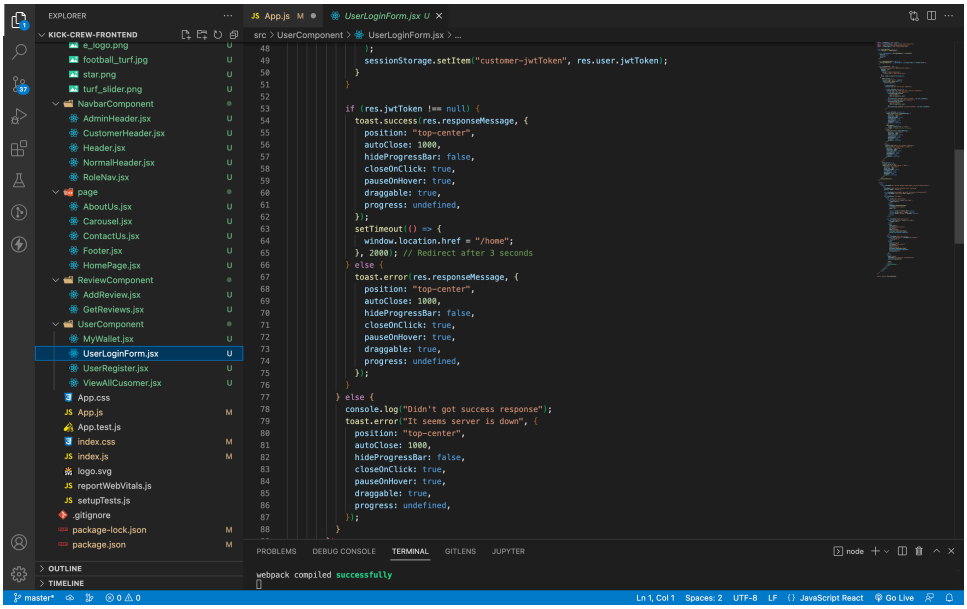


Figure 13 UserLoginForm.jsx

The 'UserLoginForm' component is a React-based form designed for user authentication in a web application. It includes fields for selecting user role (admin or customer), entering email and password credentials. When users submit their login details, the form sends a POST request to 'http://localhost: 8080/api/user/login' using the 'fetch' API. If authentication succeeds (*res. success*), users are redirected to '/home' after storing role-specific session data ('active-admin' or 'active-customer') and JWT tokens ('admin-jwtToken' or 'customer-jwtToken*) in 'sessionStorage'. The component also displays toast notifications using ' react-toastify for success or error messages based on server responses.

Styled with Bootstrap classes for a responsive layout, the 'UserLoginForm' integrates seamlessly into the application's login page or authentication flow, offering a straightforward user experience. Developers should ensure the backend API endpoint is correctly implemented to handle authentication and provide responses in the expected format (‘success' ,‘jwtToken’, ‘responseMessage').

4.5. Customer

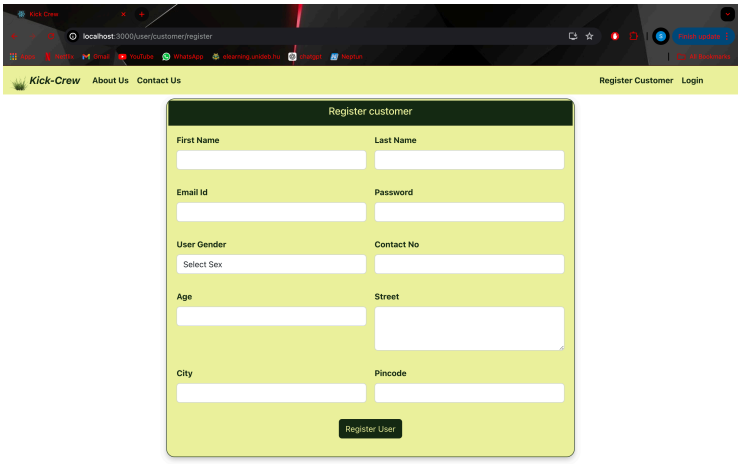


Figure 14 Customer Registration

The customer register page asks for. First Name, Last Name, Email ID, Password, User gender, Contact No Age, Street, City, Code.

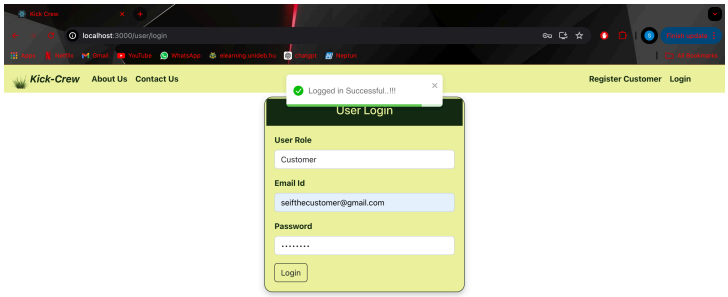


Figure 15 Customer Login

After the customer registration is complete, the user is straight away directed to the login. In there once the credentials are verified the user is informed with a pop up that says “Logged in Successful”

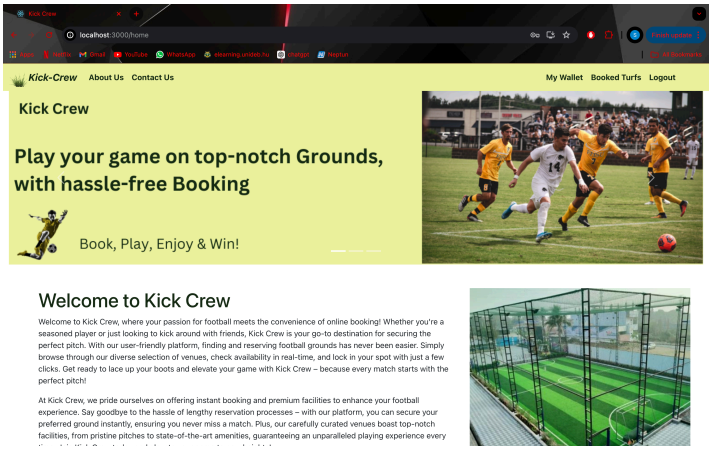


Figure 16 Customer HomePage

As mentioned previously the NavBar changes when the customer is logged in.

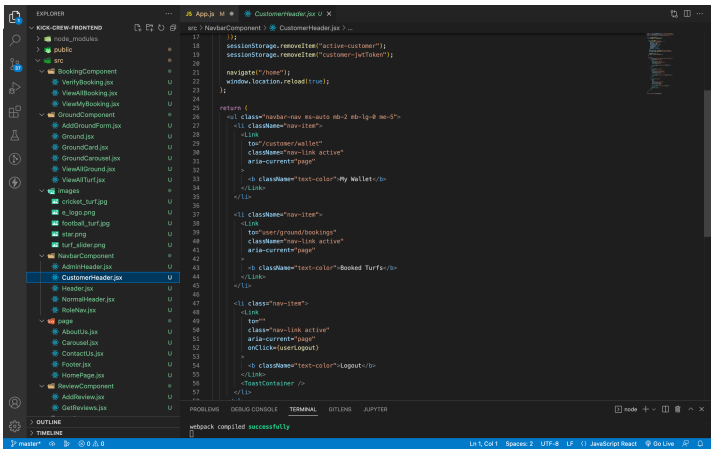


Figure 17 CustomerHeader.jsx

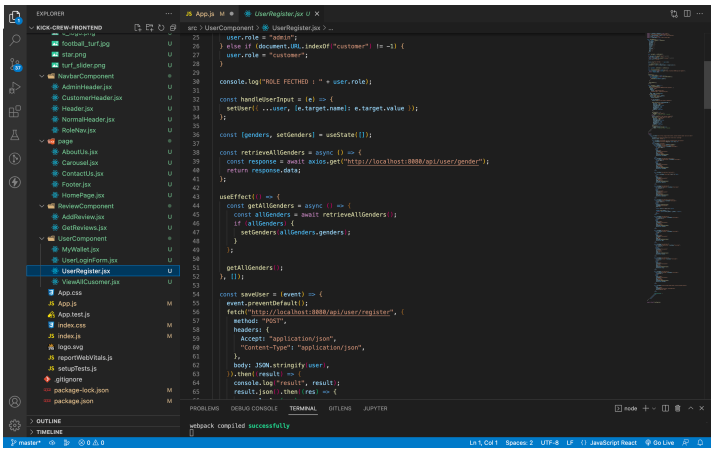


Figure 18 UserRegister.jsx

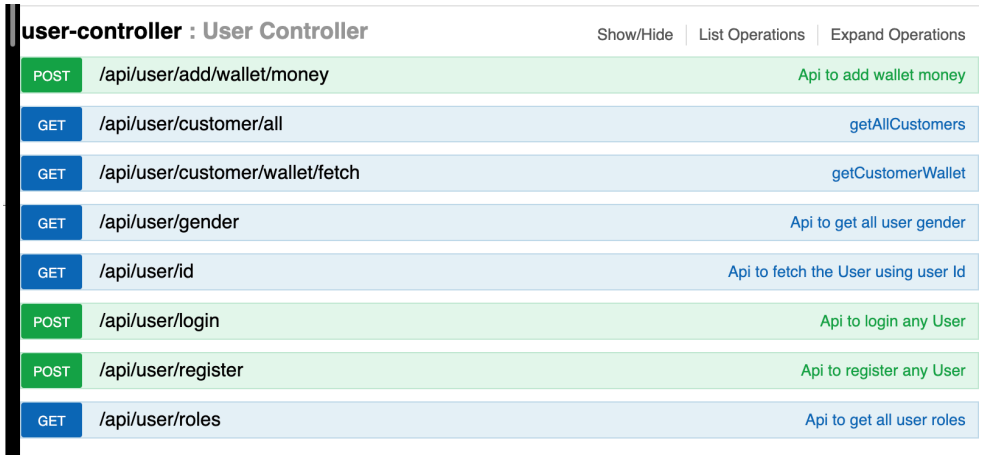


Figure 19 User Apis

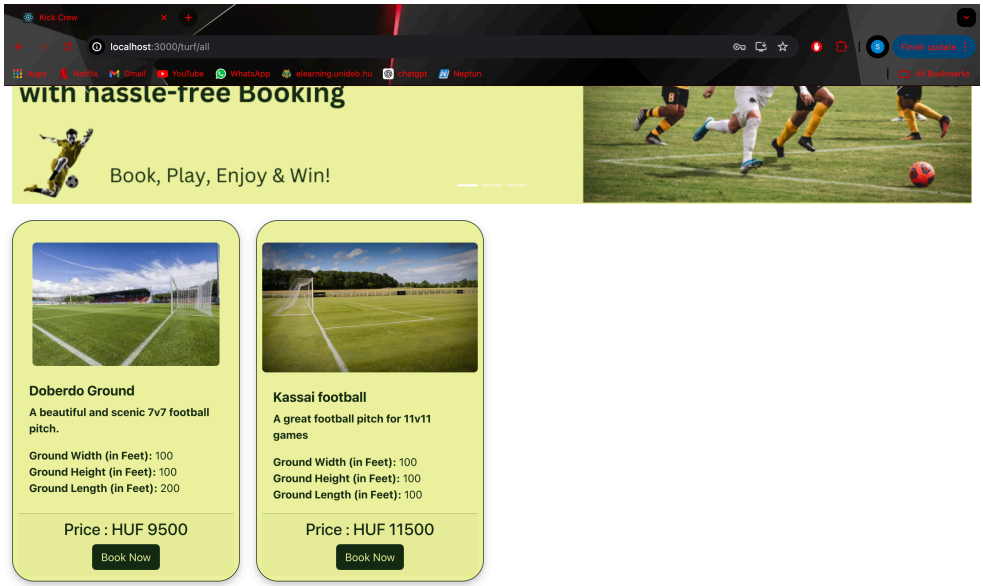


Figure 20 Customer All grounds

Shows the customer all the available pitches. This is provided by the admin. And a vibrant Book Now button is presented with the price above.

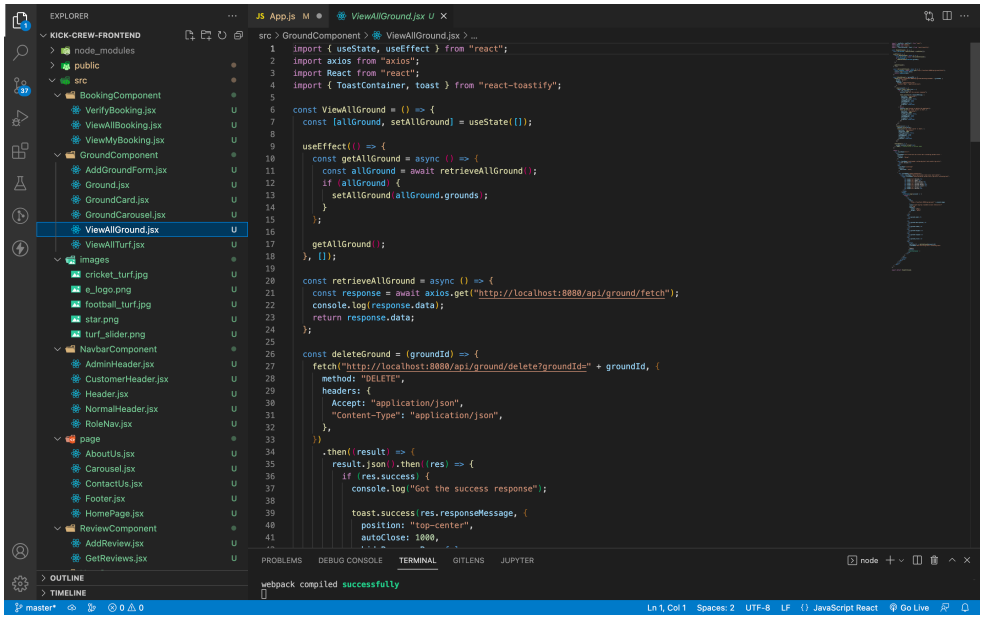


Figure 21 ViewAllGround.jsx

4.6. Wallet

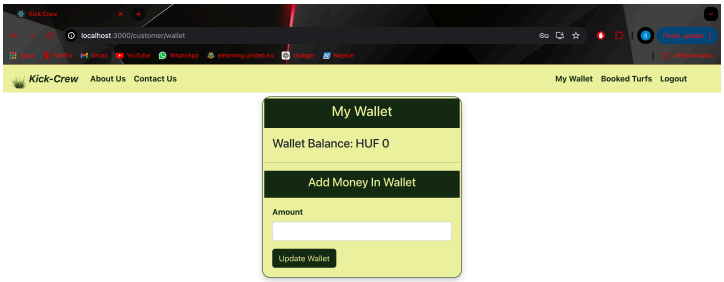


Figure 22 Wallet

Before the user can book. He needs to add money to the wallet by writing an amount in the 'amount' are and clicking update wallet.

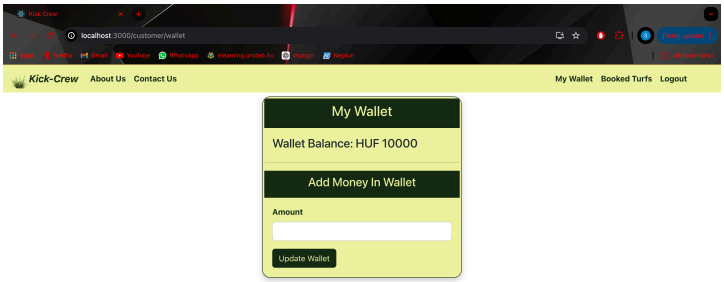


Figure 23 Wallet Balance

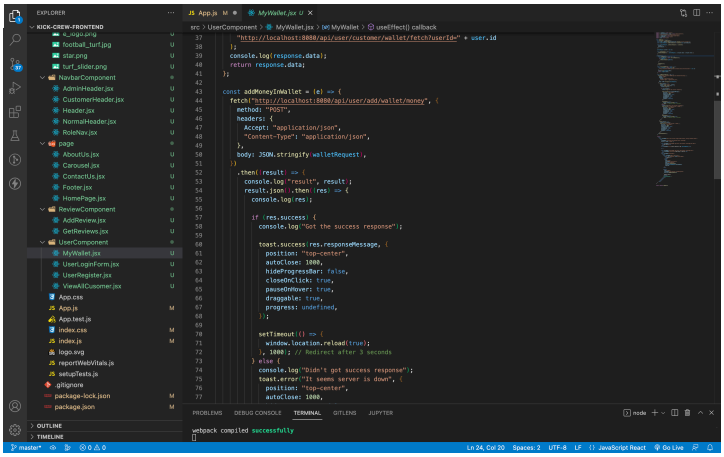


Figure 24 MyWallet.jsx

The MyWallet component manages the user's wallet, allowing them to view their current balance and add money. Upon mounting, it fetches the wallet balance from the backend using an API call and displays it. Users can input an amount to add to their wallet, which is then sent to the server when they submit the form. The response is handled with appropriate success or error messages using react-toastify, and the page refreshes to show the updated balance.

This component leverages React hooks for state management and lifecycle methods, and axios for making HTTP requests. It also handles user interactions and provides feedback through toast notifications, ensuring a smooth and responsive user experience.

4.7. Booking

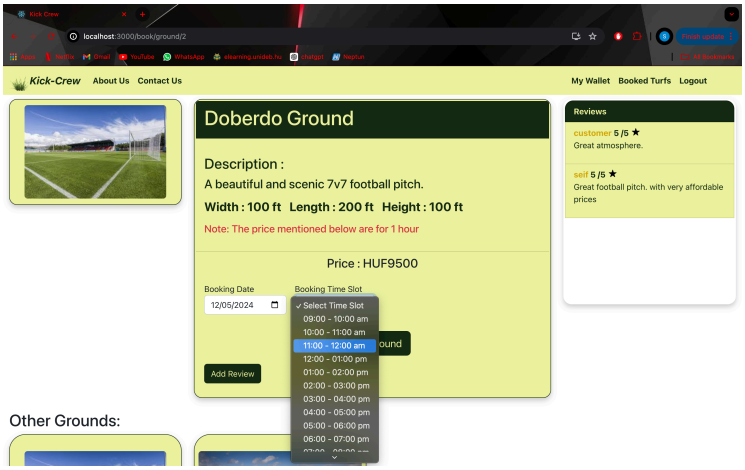


Figure 25 Booking Ground

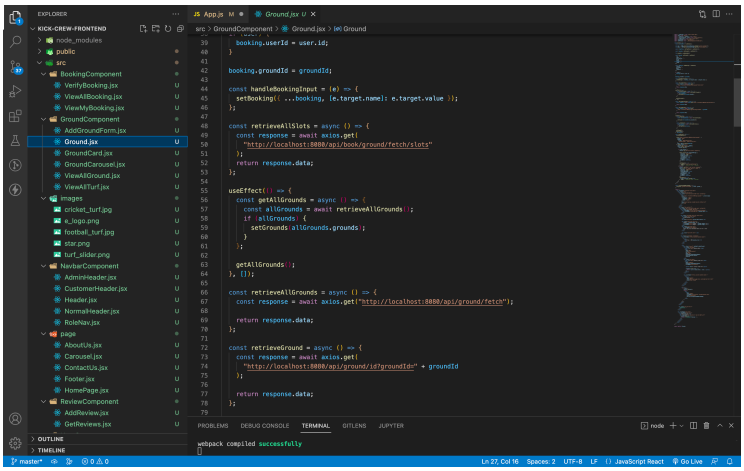
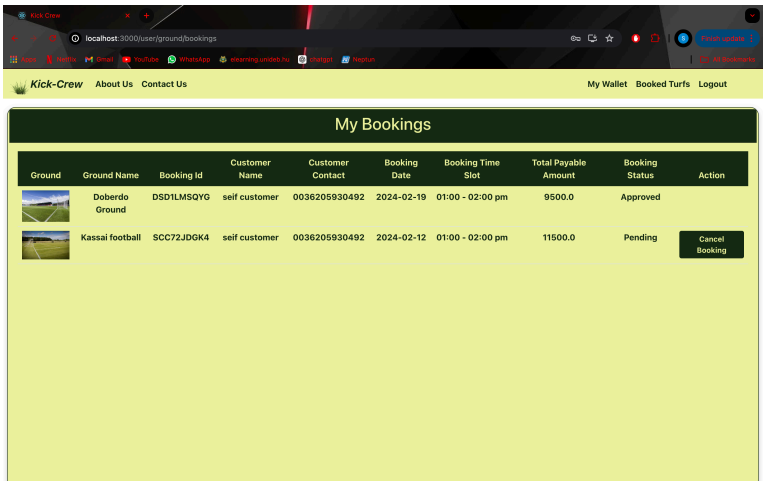


Figure 26 Ground.jsx

The Ground component showcases detailed information about a specific sports ground, allowing users to book it for a certain date and time and leave reviews. When the component loads, it fetches the ground details, available time slots, and other grounds from the server using axios. Users can select a date and time slot to book the ground, and the booking info is sent to the backend. The component handles responses with success or error messages using react-toastify.
In addition to booking, the component displays the ground's image, name, description, and dimensions. Logged-in users see an option to add a review for the ground. There's also a section listing other available grounds for users to explore, making it easy to find and book sports grounds while also sharing their experiences.



4.8. Admin

Figure 27 My Bookings

After the customer submits a booking, they are automatically redirected to the "My Bookings" page. Here, they can view the ground name, a unique booking ID, customer details, and the booking information, including the price. The booking status is initially marked as "pending" until an admin logs in and approves the request.

The UserRegister component simplifies the sign-up process for admins. To register, admins should visit: http://localhost:3000/user/admin/register. When accessed, the form automatically sets the role to 'admin' based on the URL. It collects essential details like name, email, password, contact info, and address. Additionally, gender options are dynamically fetched from the server to ensure up-to-date choices. After filling out the form, admins can submit their information, which is then sent to the server. If the registration is successful, a toast notification confirms it, and they are redirected to the login page.

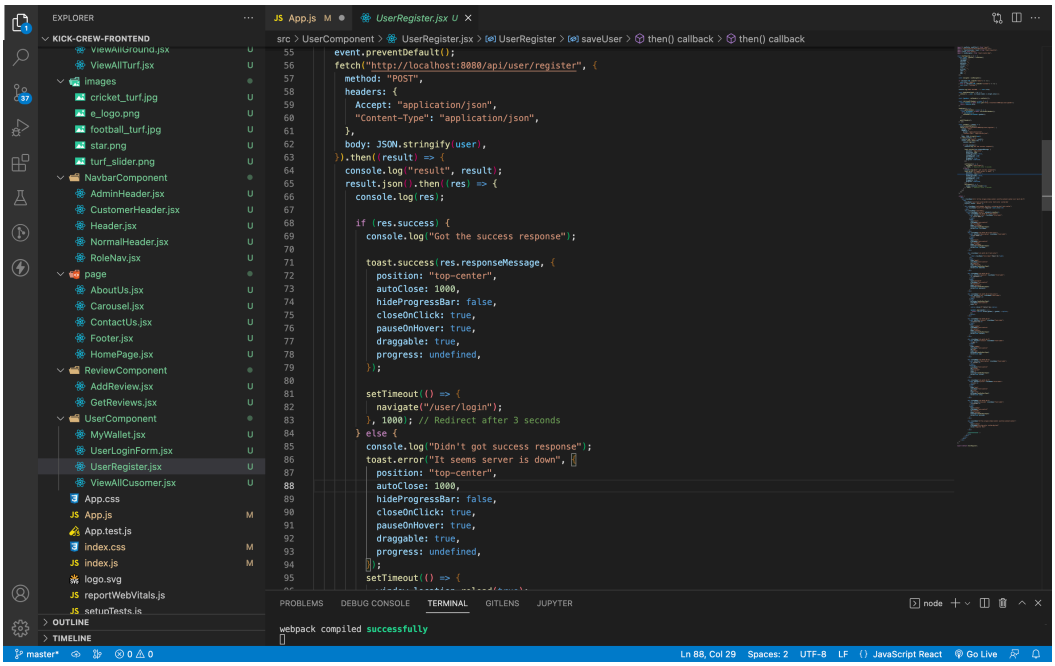


Figure 28 UserRegister.jsx

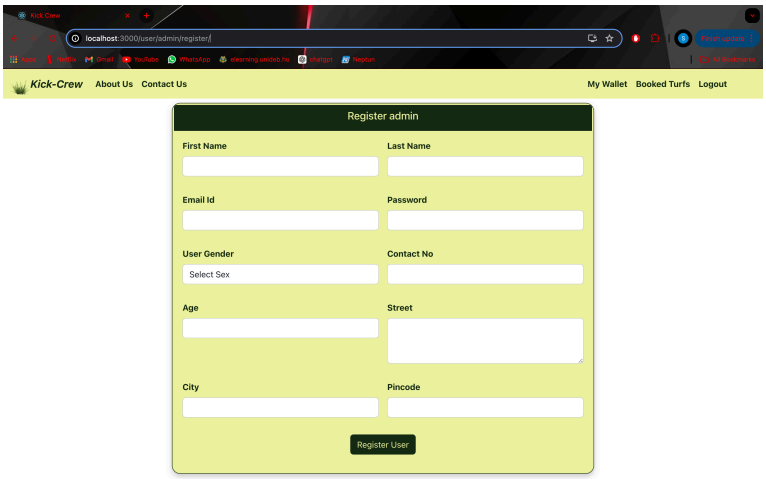


Figure 29 Admin Register

The UserLoginForm component handles user authentication and includes functionality specifically for admin users. Upon a successful login, the system checks the user's role. If the user is an admin, their details are stored in the session storage under "active-admin" along with a JWT token labeled "admin-jwtToken". This allows the admin to have an authenticated session with appropriate access levels.
In the event of a successful login, the admin is greeted with a success toast notification and redirected to the home page. In order to register a new admin. The user/admin must go to the link : http://localhost:3000/user/admin/register

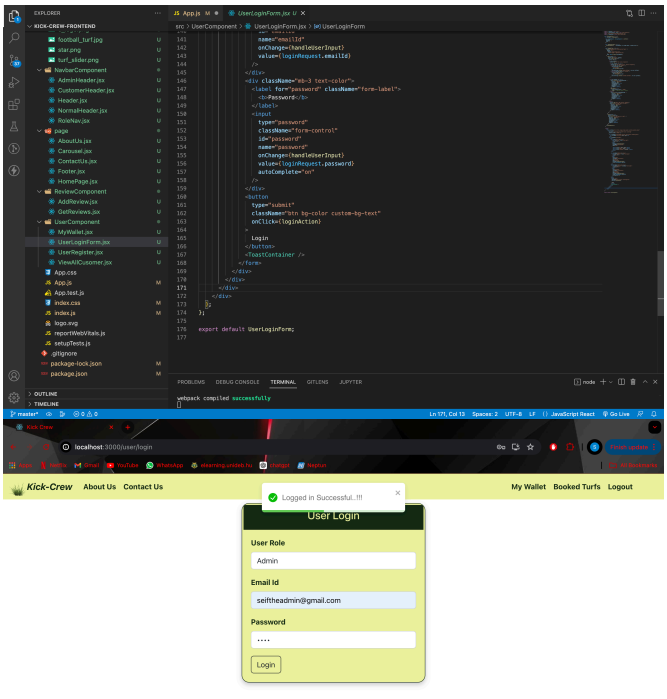


Figure 30 UserLoginForm.jsx

Figure 31 Admin Login

4.9. Admin NavBar

The AdminHeader component is designed to provide easy navigation for admins within the application. Upon logging in, admins can see options to add a new ground, view all grounds, view all customers, and view all bookings. These options are displayed as navigation links, making it simple for admins to manage the platform efficiently. The component also fetches the currently active admin from the session storage to personalize the experience.
When an admin decides to log out, they can simply click the "Logout" link. This triggers the adminLogout function, which displays a success toast notification, clears the admin session data, reloads the page, and navigates back to the home page. This ensures a smooth and user-friendly logout process. The ToastContainer is included to handle the display of toast notifications, providing instant feedback to the admin for actions taken.

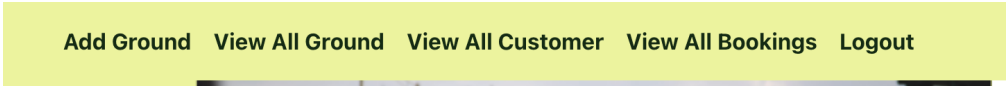


Figure 32 Admin Navbar

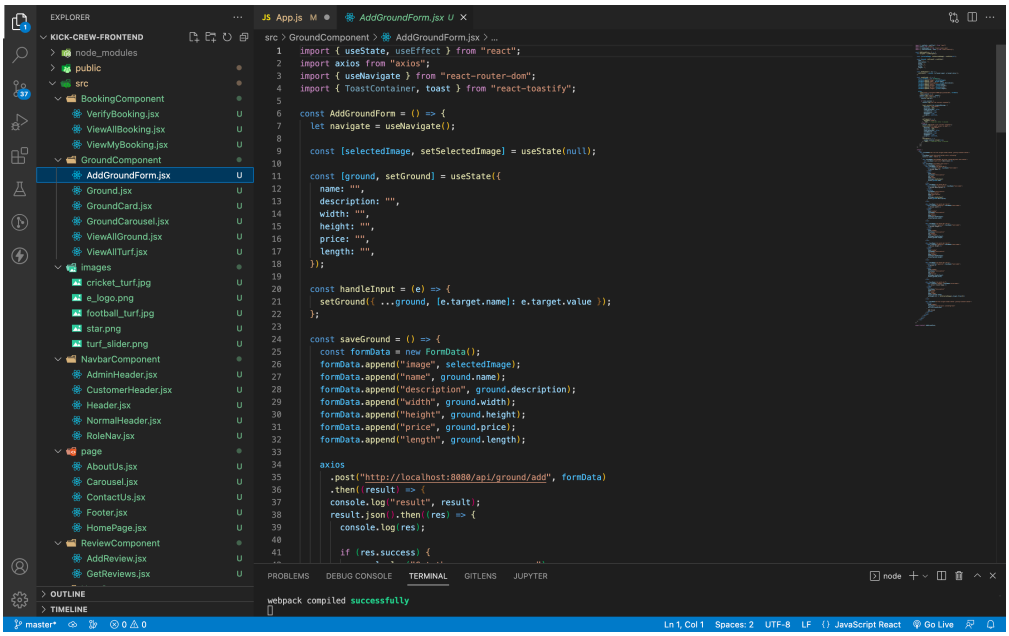


Figure 33 AddGroundForm.jsx

4.10. Add Ground

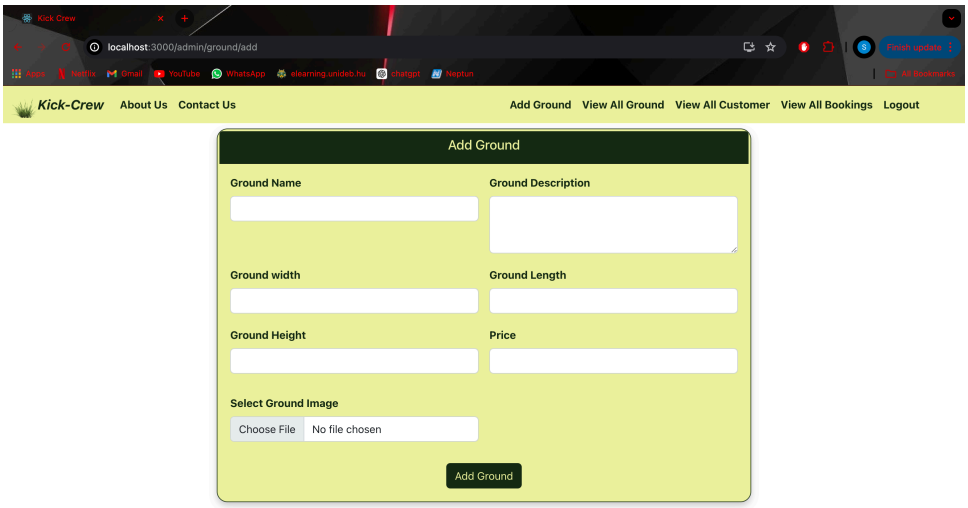


Figure 34 Add Ground

The AddGroundForm component helps administrators add new grounds easily. It includes fields for the ground's name, description, width, height, length, and price. Admins can also upload an image of the ground. The form data is handled using state, and when submitted, it's sent to the server with axios. If the submission is successful, a toast notification pops up to confirm, and the admin is redirected to the home page.
Admins will find this form straightforward to use. It captures all the essential details needed for a new ground and provides immediate feedback through toast notifications. This makes it simple to add new grounds and manage the information effectively.

4.11 View All Grounds

Figure 35 View All Ground

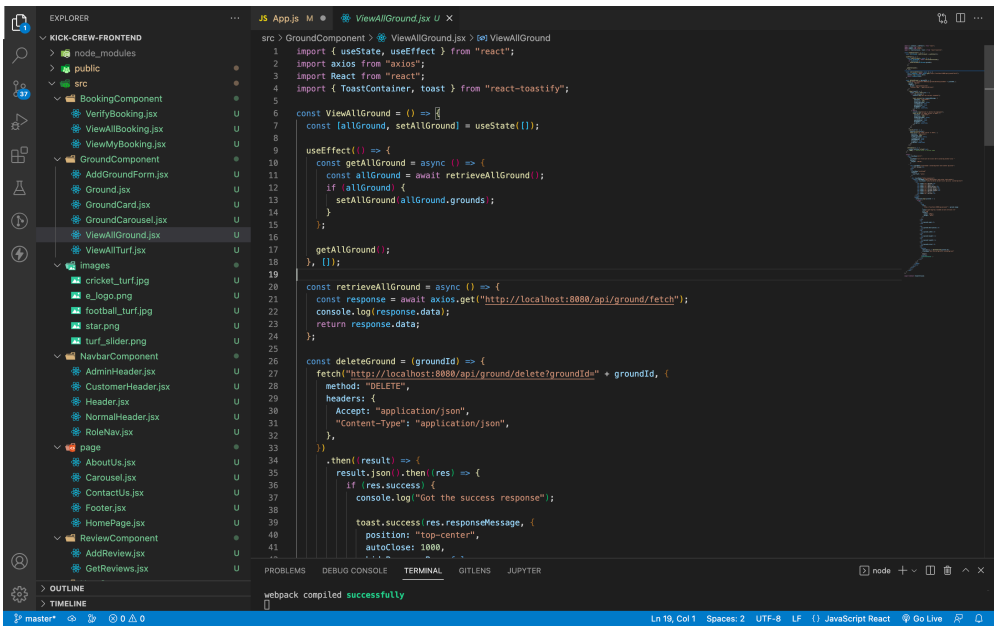


Figure 36. ViewAllGround.jsx

The ViewAllGround component displays a list of all grounds currently registered in the system. It fetches this data from the server upon loading using a call with axios. Each ground's details such as name, description, dimensions, and price are presented in a table format. For each ground entry, administrators have the option to delete it directly from the list.

When deleting a ground, the system sends a request to the server with the ground's ID, and upon successful deletion, a toast notification confirms the action. If there are any issues during the

deletion process or if the server is unreachable, an error notification appears instead. The interface provides a seamless experience for administrators to manage grounds efficiently.
This component ensures administrators can quickly view all grounds, perform necessary actions, and receive immediate feedback through notifications, enhancing their workflow management.

4.12. View All Customers

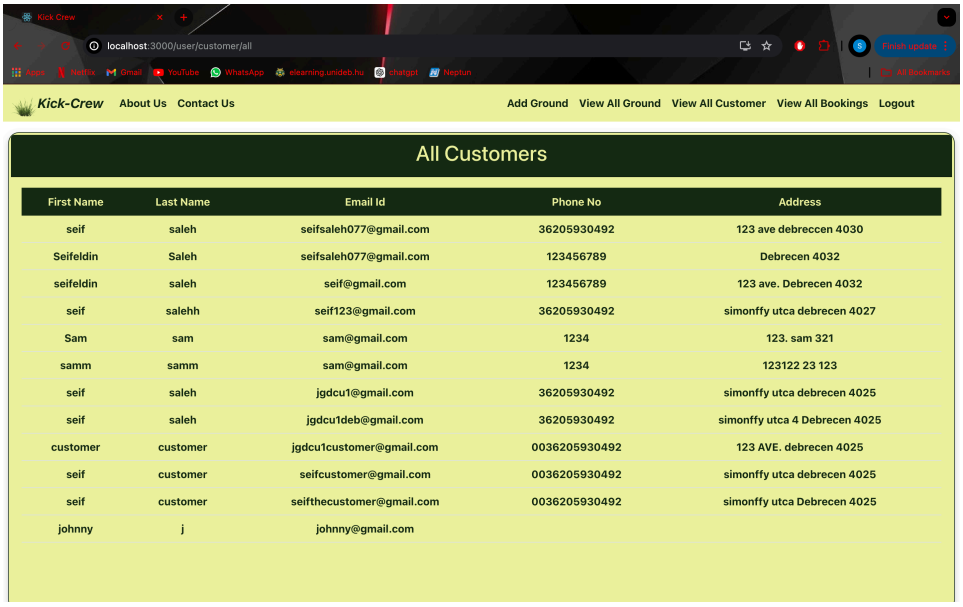


Figure 37 View All Customers

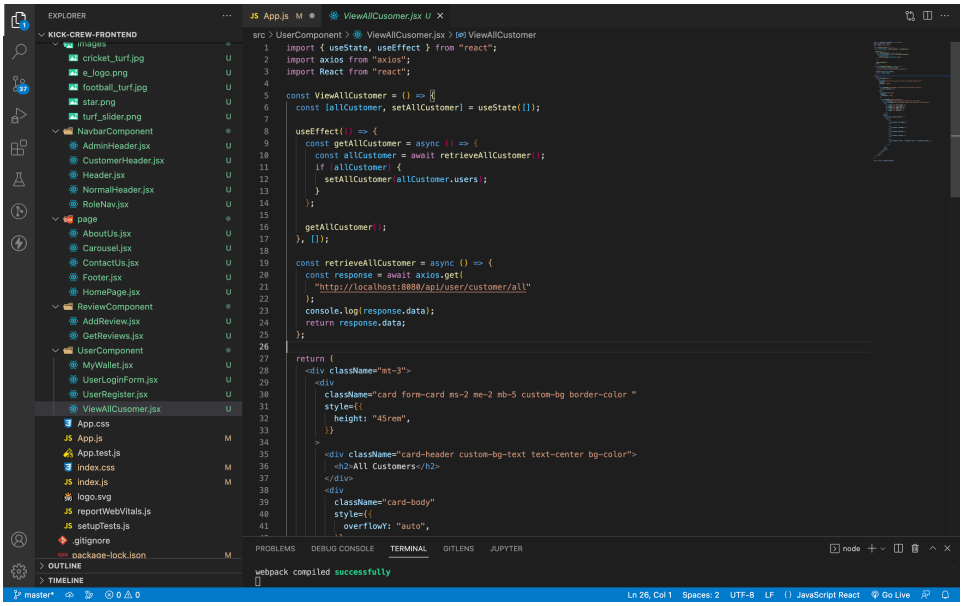


Figure 38 ViewAllCustomer.jsx

The ViewAllCustomer component is designed to display a comprehensive list of all registered customers in a straightforward manner. It fetches customer data from the server using axios within the useEffect hook upon component rendering. This ensures that the displayed information, which includes essential details like first name, last name, email ID, phone number, and address for each customer, is always up-to-date.

The layout of the component utilizes a table format, making it easy for administrators to quickly scan and review customer details. This structured presentation enhances clarity and usability, facilitating efficient management of user accounts within the system. Overall, ViewAllCustomer aims to support administrators in effectively overseeing and managing customer information, promoting streamlined operations and user management workflows.

4.13. View All Bookings

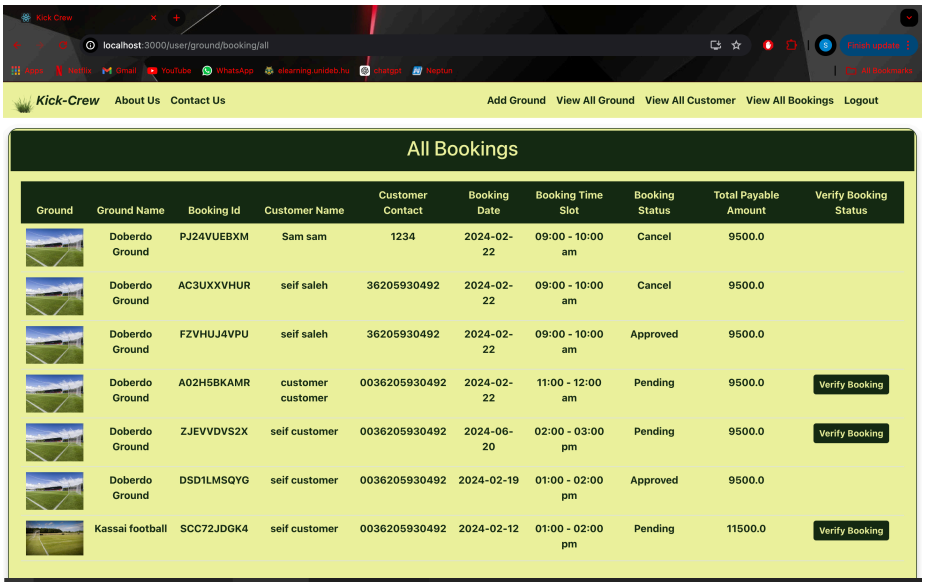


Figure 39 All bookings

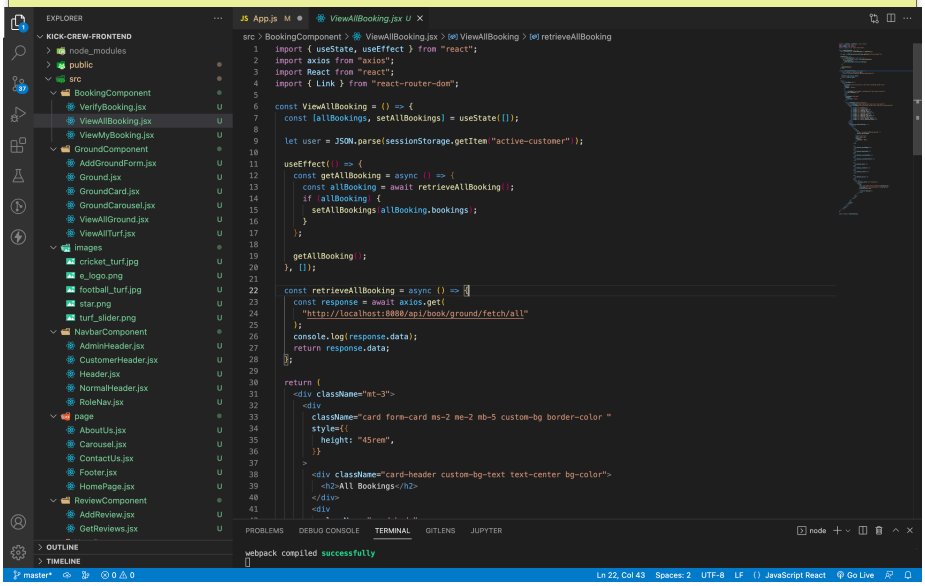
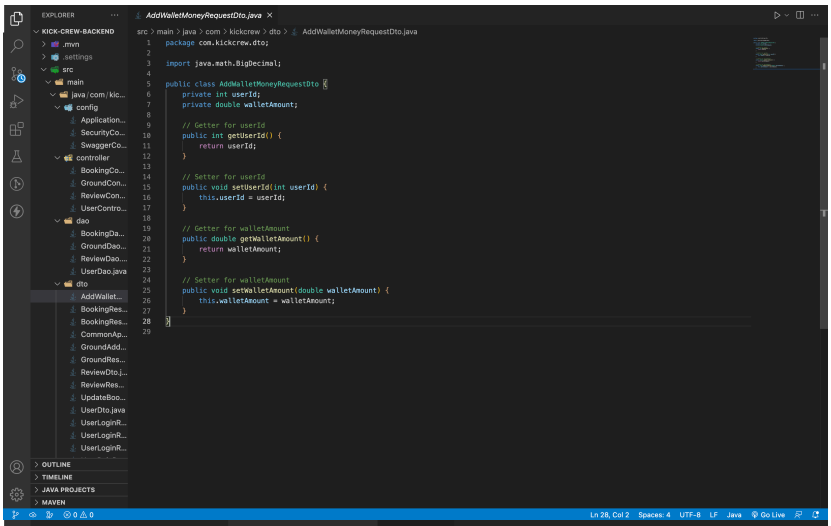


Figure 40 ViewAllBooking.jsx

The ViewAllBooking component is designed to present a comprehensive list of all bookings made within the system. It fetches this data using an asynchronous call to the server upon component initialization, ensuring the displayed information is always current. Each booking entry includes essential details such as the ground image, ground name, booking ID, customer name, contact information, booking date, time slot, booking status, total payable amount, and an option to verify the booking status if it's pending. When an admin approves a booking, the money is then deducted



from the customer's wallet. This structured presentation in a table format allows administrators to efficiently monitor and manage booking activities, facilitating streamlined operational oversight and customer service management.

Figure 41 AddWalletMoneyRequestDTO.java

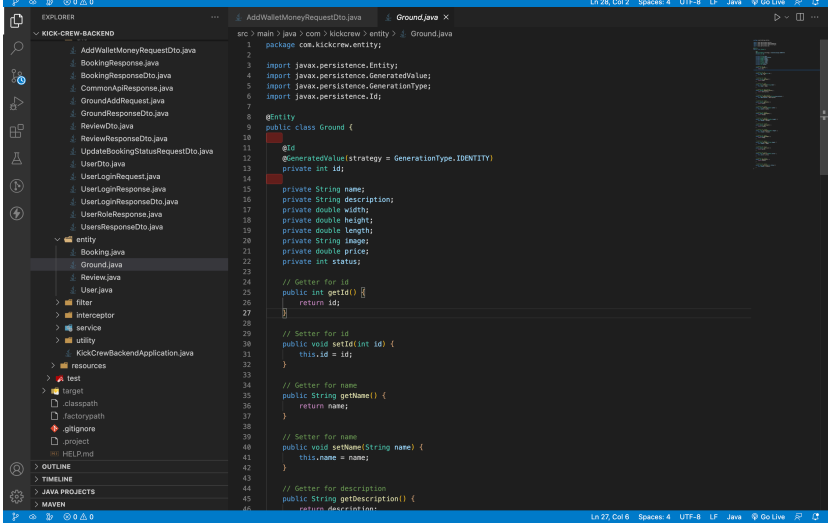


Figure 42 Ground.java

The Ground class in Java serves as a fundamental component for managing football pitch bookings within a web application. It encapsulates crucial attributes like name, description, and physical dimensions (width, height, length) that define each football ground. This class is annotated with @Entity, marking it as a persistent entity in a database, typically managed through JPA (Java Persistence API). This setup ensures that instances of Ground can be seamlessly stored and retrieved from a relational database, facilitating efficient data management and retrieval for booking operations