INT252 – WEB APP DEVELOPMENT WITH REACT JS

BOOK ORIENTED CHAT APP

Project Report CA3 submitted in fulfilment of the requirements for the

Degree of

BACHELOR OF TECHNOLOGY

In

Computer Science and Engineering

By

Ritesh Singh

(12102719)

SUBJECT

SUBMITTED TO

PROF. ASHISH KUMAR

TABLE OF CONTENTS

Introduction

1.1 Problem Statement

In today's digital landscape, communication platforms are not merely tools for exchanging messages but rather dynamic ecosystems that foster collaboration, learning, and exploration. This abstract introduces Book-Rack, a revolutionary chat application that seamlessly integrates the functionality of traditional messaging with the boundless possibilities of literary exploration.

Book-Rack stands out from the crowd with its unique blend of features, chief among them being its diverse array of chat rooms. These rooms cater to a wide spectrum of interests, enabling users to connect with like-minded individuals, engage in lively discussions, and share their expertise on topics ranging from technology and literature to fitness and beyond. Whether you're a seasoned professional seeking industry insights or an avid enthusiast looking to connect with others who share your passions, Book-Rack offers a space for every interest and inclination.

Beyond its social networking capabilities, Book-Rack distinguishes itself with its innovative integration of the Google Books API. This feature empowers users to delve into the vast repository of human knowledge with unparalleled ease. By simply entering a book title, author, or topic of interest, users can instantly access a wealth of information, excerpts, and resources curated from Google's extensive library. Whether you're conducting research, seeking literary inspiration, or simply exploring new avenues of learning, Book-Rack's book search functionality opens doors to a world of intellectual discovery at your fingertips.

Moreover, Book-Rack prioritizes user privacy and security, employing robust encryption protocols to safeguard sensitive data and ensure confidential communication. With its intuitive interface, seamless integration of chat rooms

and book search functionality, and unwavering commitment to user privacy, Book-Rack represents the next evolution in digital communication platforms. Join us on Book-Rack and embark on a journey of connection, collaboration, and enlightenment unlike any other.

1.2 Why Use React, Node.js, and Socket.io?

The choice of React, Node.js, and Socket.io for the project is driven by several key factors:

Robustness: React is a widely adopted JavaScript library renowned for its robustness and performance in building modern web applications. Node.js, as a server-side JavaScript runtime, complements React by providing a robust and scalable backend environment. Socket.io enhances this robustness by facilitating real-time bidirectional communication between clients and the server, ensuring a seamless collaborative code editing experience.

Scalability: Both React and Node.js are known for their scalability, making them ideal choices for applications that need to handle large numbers of concurrent users and growing data demands. Socket.io's scalability features further enhance the application's ability to accommodate increasing user demands and data volume.

Active Developer Community: The React, Node.js, and Socket.io communities are vast and active, offering a wealth of resources, documentation, and support for developers. This ensures the longevity and sustainability of the application, with ongoing updates and improvements provided by the community.

Rapid Development: React's component-based architecture and declarative syntax enable developers to build complex user interfaces quickly and efficiently. Node.js, with its vast ecosystem of libraries and frameworks, allows for rapid backend development. Socket.io's straightforward API and event-driven architecture simplify the implementation of real-time communication features, reducing development time and costs.

By leveraging React, Node.js, and Socket.io, the project aims to deliver a robust, scalable, and efficient app for avid reader to enjoy and bond over.

1.3 Architecture Overview

The architecture of the project follows a client-server model, leveraging React for the frontend and Node.js for the backend. Socket.io facilitates real-time bidirectional communication between clients and the server, enabling seamless communication between database and app, allowing real time chatting.

Frontend (React):

- React serves as the foundation for the frontend, providing a component-based architecture for building user interfaces.

- Components such as the home page and chat page are designed to provide intuitive user experiences and facilitate smooth navigation within the application.

- React Router is utilized for client-side routing, allowing users to navigate between different pages of the application without full-page reloads.

Backend (Node.js):

- Node.js powers the backend server, handling WebSocket communication with clients and managing room-based collaboration.

- Express.js, a minimalist web framework for Node.js, is used to set up routes and handle HTTP requests for serving static files and API endpoints.

- Socket.io is integrated into the backend to establish WebSocket connections, enabling real-time event-based communication between the server and clients.

Real-time Communication (Socket.io):

- Socket.io facilitates real-time bidirectional communication between clients and the server, enabling instant synchronization of chats sent between users.

- The server listens for socket events such as 'connection', 'join', ‘send_message, and 'disconnect' to manage room-based communication and chatting.

- Socket.io's event-driven architecture ensures efficient communication and synchronization of data between multiple clients connected to the same room.

2. Technologies Used

1. React: React provides a highly efficient and flexible way to build user interfaces. Its component-based architecture facilitates the creation of reusable UI elements, resulting in faster development cycles and a more maintainable codebase. With React's virtual DOM, updates are rendered efficiently, ensuring a smooth and responsive user experience, ideal for real-time messaging interfaces.

2. Node.js: Node.js is a lightweight and scalable runtime environment, perfect for building server-side applications. Its non-blocking I/O model enables handling multiple connections simultaneously, making it well-suited for chat applications with high concurrency requirements. Additionally, Node.js's vast ecosystem of packages simplifies tasks like handling HTTP requests, managing databases, and integrating with other services.

3. Socket.io: Socket.io is a library that enables real-time, bidirectional communication between web clients and servers. By establishing WebSocket connections, Socket.io facilitates instant data exchange, allowing chat messages to be delivered and received in real-time. Its event-based architecture and support for fallback mechanisms ensure seamless communication across a variety of devices and network conditions, making it an essential component for building responsive and interactive chat applications.

4. Key Features of the App

1. Multi-Room Chat: Users can create and join multiple chat rooms based on their interests, enabling focused discussions and fostering communities around specific topics.

2. Real-Time Messaging: Utilizing Socket.io, messages are delivered instantly, providing a seamless chatting experience with minimal latency.

3. Book Search Integration: Integration with the Google Books API allows users to search for books directly within the app, accessing information, excerpts, and resources from a vast library of literature.

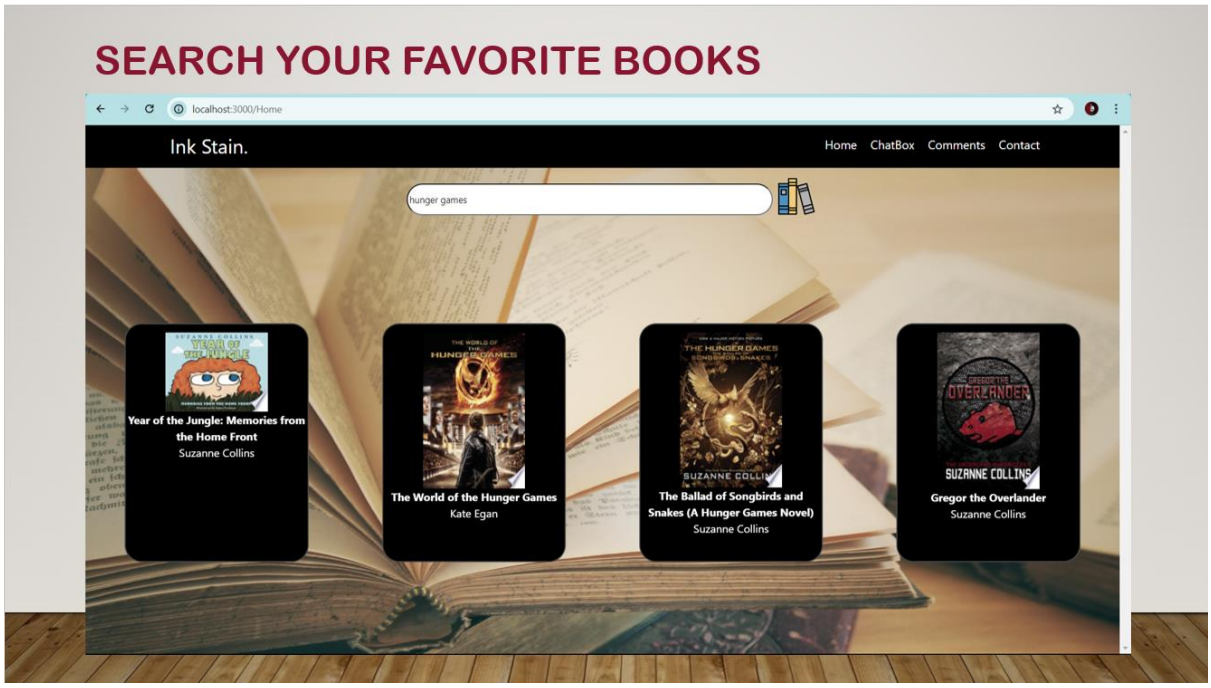
4. User Authentication: Secure user authentication ensures that only authorized individuals can access the chat app, safeguarding privacy and promoting a safe online environment.

5. Responsive Design: A responsive user interface ensures a seamless experience across devices, including desktops, tablets, and smartphones, enabling users to chat on the go.

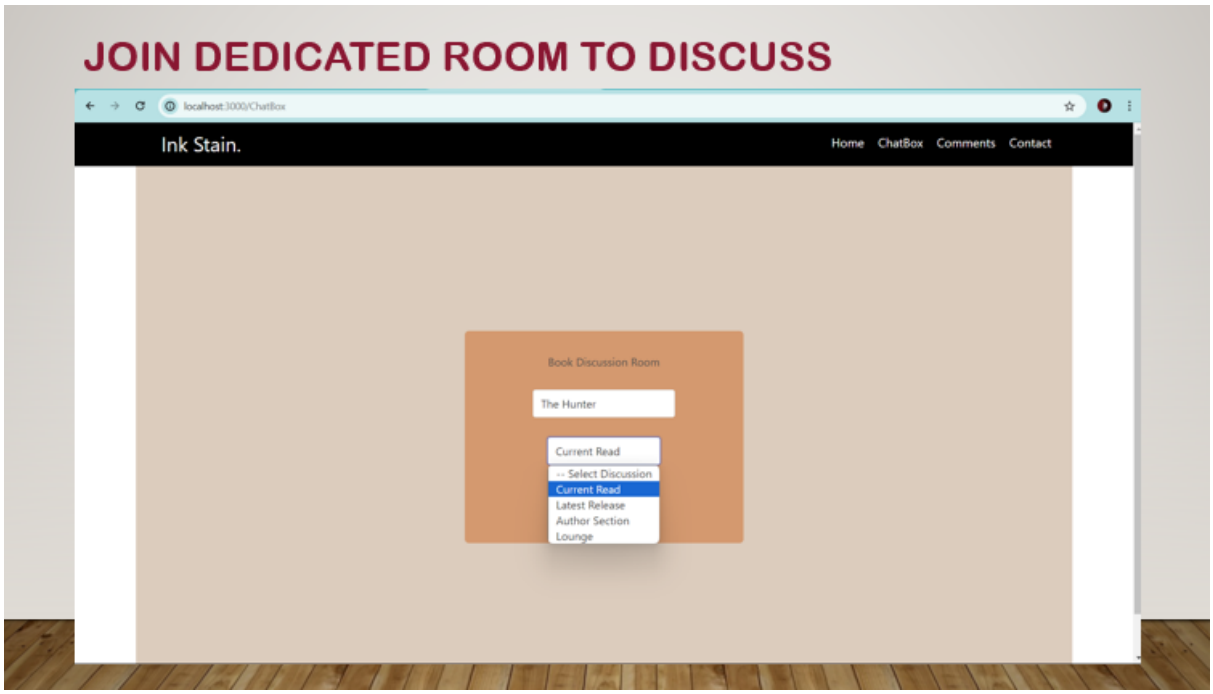
selection of themes or customizing colors, fonts, and other visual elements to suit their preferences.

5. Project Snapshots

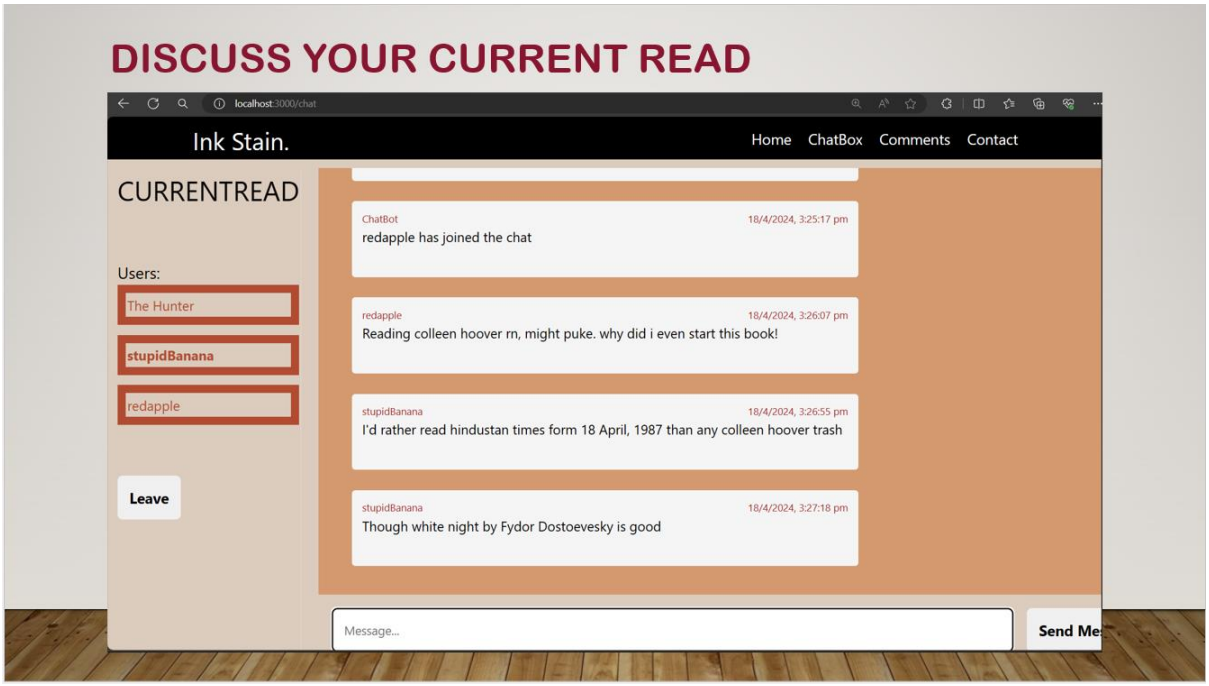
1. Book Search (Google Books API)



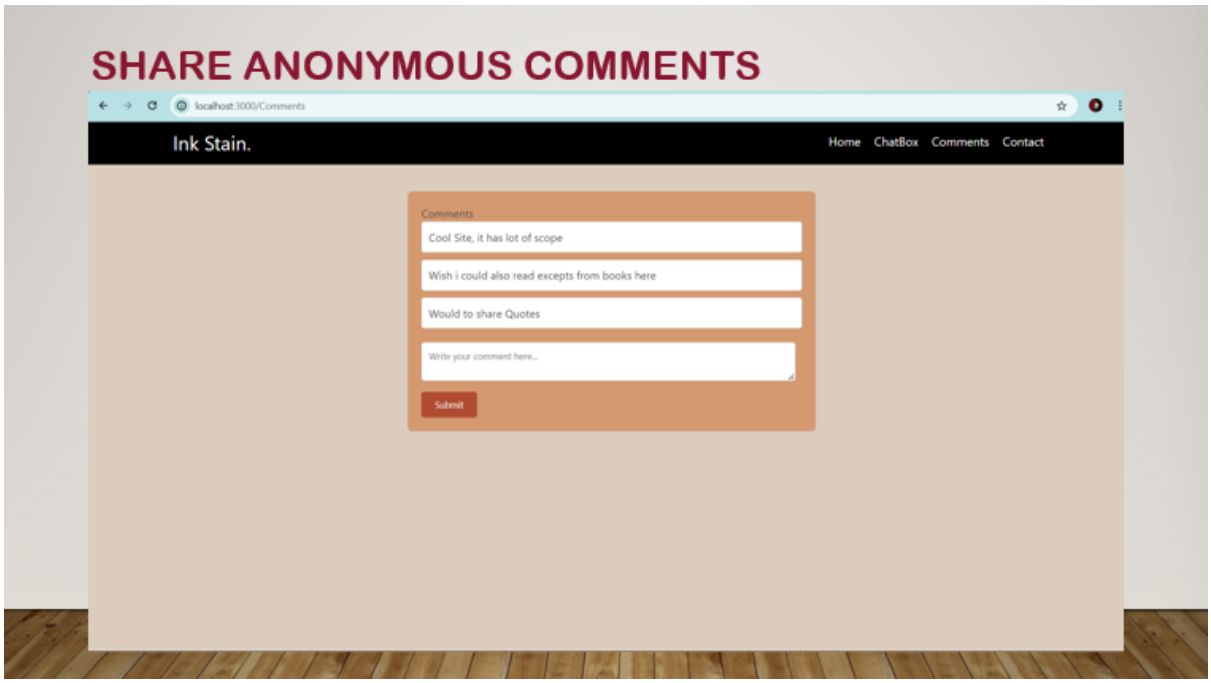
2.Chat room selection



3.Chat room using Socket.io



4. Comment Section



5. Code Snapsots

```jsx
import { BrowserRouter as Router,Routes, Route } from 'react-router-dom';
import io from 'socket.io-client';
import { useState } from 'react';

const socket = io.connect('http://localhost:4000');

function App() {
  const [username,setUsername] = useState('');
  const [room,setRoom] = useState("");

  return (
    <div>
    <Router>
      <Navbar></Navbar>
      <Routes>
        <Route path="*" element={<Home/>}/>
        <Route exact path='/Home' element={<Home></Home>}>Home</Route>
        <Route
        exact path='/ChatBox'
        element={
        <ChatBox
                username={username}
                setUsername={setUsername}
                room={room}
                setRoom={setRoom}
                socket={socket}
        />}>ChatBox</Route>
        <Route
            exact path='/chat'
            element={<Chat username={username} room={room} socket={socket} />}
          />
```

App.js

```jsx
frontend > src > components > JS ChatBox.js > ...
      You, 5 days ago | 1 author (You)
1   import React from 'react'
2   import './styles/ChatBox.css'        You, 5 days ago • hehe
3   import { useNavigate } from 'react-router-dom'
4
5
6   function ChatBox({username, setUsername, room, setRoom, socket}) {
7     const navigate = useNavigate();
8
9     const joinRoom = () =>{
10      if(room!== any  username!== ' '){
11        socket.emit('join_room',{username,room})
12      }
13      navigate('/chat', { replace: true });
14    }
15
16
17    return (
18      <div className="container">
19        <div className="formContainer">
20          <div className="chatbox-groupname">
21              Book Discussion Room
```

Chatroom.js

Conclusion

In conclusion, Book-Rack presents a powerful and versatile platform that seamlessly integrates real-time messaging with the exploration of literature. With its multi-room chat functionality, users can connect with others who share their interests and engage in meaningful conversations. The integration of the Google Books API adds a unique dimension, allowing users to discover, discuss, and delve into a wealth of literary content right within the app.

Looking ahead, there are several exciting avenues for future development and enhancement. One potential direction is to further expand the book search capabilities, integrating additional sources and providing advanced features such as personalized recommendations based on user preferences. Additionally, enhancing the user experience with multimedia support, including images, videos, and audio clips, could enrich communication within the app.

Furthermore, exploring opportunities for community building, such as user-generated content, discussion forums, or virtual book clubs, could foster deeper engagement and connections among users. Collaboration tools, document sharing features, and integration with productivity apps could also broaden the app's appeal beyond casual chatting to include professional and educational use cases.

As technology evolves and user needs evolve with it, Book-Rack remains poised to adapt and innovate, continuously striving to provide a dynamic and enriching platform for communication, collaboration, and literary exploration. With a commitment to user satisfaction and a dedication to staying at the forefront of digital trends, Book-Rack looks forward to a future filled with exciting possibilities and continued growth.

GitHub Link: Electric-Shade/int-252-react-project (github.com)