

Shelfbound
BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING
BY

T. Yogitha	(22501A05I3)
P. Lavanya	(22501A05D4)
P. Sai Advaith	(22501A05D6)
P. Mithil	(22501A05F2)

Under the Guidance of Dr.

K. Jyothsna Devi, Ph.D

Assistant Professor



PRASAD V POTLURI SIDDHARTHA INSTITUTE OF TECHNOLOGY

(Permanently affiliated to JNTU :: Kakinada, Approved by AICTE)

(An NBA & NAAC A+ accredited and ISO 9001:2015 Certified Institution)

Kanuru, Vijayawada - 520007

2024-25

PRASAD V POTLURI

SIDDHARTHA INSTITUTE OF TECHNOLOGY

(Permanently affiliated to JNTU :: Kakinada, Approved by AICTE)

(An NBA & NAAC A+ accredited and ISO 9001:2015 certified institution)

Kanuru, Vijayawada – 520007



CERTIFICATE

This is to certify that the project report title “**SHELFBOUND**” is the bonafide work of **T. Yogitha (22501A05I3), P. Lavanya (22501A05D4), P. Advait (22501A05D6), P. Mithil (22501A05F2)** in partial fulfilment of completing the Academic project in Mern-stack Development during the academic year 2025-26.

Signature of the Incharge

Signature of the HOD

INDEX

S.No.	Content	Page No. (s)
1	Abstract	1
2	SDG – Sustainable Development Goals	2
3	Introduction	3
4	Objectives and Scope of the Project	4
5	Software used - Explanation	5-7
6	Proposed model	8
7	Sample Code	9-11
8	Result/Output Screen shots	12-16
9	Conclusion	17
10	References (web site URLs)	18-19

1. ABSTRACT

In today's digital landscape, book lovers face challenges in organizing their collections, tracking their reading progress, and engaging in meaningful discussions about literature. Many existing platforms lack a seamless way to curate favorite books, share insightful reviews, and connect with other readers in an interactive environment. The absence of a structured yet engaging system for managing book collections often leads to scattered reading experiences and limited user interaction.

To address this, **Shelfbound** is a comprehensive book tracking and review application built using the **MERN stack (MongoDB, Express, React, Node.js)**. It enables users to efficiently manage their book collections by adding favorites, leaving detailed reviews with star ratings, and participating in discussions through a **nested commenting system** powered by Depth-First Search (DFS). The platform fosters a **community-driven approach** to book discovery while maintaining a visually appealing and intuitive interface.

Secure **user authentication** allows individuals to create personalized profiles, ensuring a tailored experience. Additionally, **role-based access control** enables admins to manage book listings, moderate content, and oversee user interactions. **Cloud storage support** for book cover images enhances accessibility, while the React.js-powered UI ensures a **seamless and engaging user experience**.

By integrating **scalability, security, and interactivity**, Shelfbound transforms book tracking into an engaging and organized digital experience, empowering readers to explore, review, and connect with fellow book enthusiasts effortlessly.

2. SUSTAINABLE DEVELOPMENT GOALS(SDG)

SDGs Mapped:

- **SDG 4** – Quality Education
- **SDG 10** – Reduced Inequalities

2.1 How This Project Supports SDG 4 – Quality Education

ShelfBound promotes lifelong learning and organized access to knowledge by:

- **Encouraging Reading Habits:** Motivates users to read regularly, supporting continuous learning.
- **Inclusive Learning:** Offers personalized book lists and wishlists for diverse interests and reading levels.
- **Knowledge Management:** Helps users track progress and reflect on learning journeys.
- **Scalable Tool:** Can be integrated into schools, libraries, and communities to enhance reading engagement globally.

2.2 How This Project Supports SDG 10 – Reduced Inequalities

ShelfBound reduces inequalities in education access by:

- **Democratizing Knowledge:** Provides equal access to book tracking and recommendations.
- **Promoting Diversity:** Highlights inclusive reading lists and underrepresented voices.
- **Bridging Gaps:** Supports learners in remote or under-resourced areas
- **Fostering Inclusive Communities:** Encourages shared learning and discussions for all users.

3. INTRODUCTION

In today's digital age, book lovers and avid readers constantly seek new ways to discover, review, and organize their reading experiences. Whether exploring classic literature, contemporary fiction, or academic resources, managing personal book collections and sharing insights with a like-minded community is crucial. However, traditional methods of tracking books—such as handwritten journals, spreadsheets, or scattered online notes—can be inefficient and unstructured. Shelfbound is designed to solve this problem by providing a centralized platform where users can effortlessly browse books, leave detailed reviews, and manage their reading lists in one place.

The inspiration behind Shelfbound comes from the need for a structured and interactive way to track and review books. Readers often revisit past reads, seek recommendations, and engage in discussions about their favorite books. However, disorganized review platforms and fragmented book-tracking systems make this process cumbersome. Shelfbound acts as a personalized digital library, ensuring that users can rate, review, and categorize books seamlessly, while also engaging with a community of fellow readers.

Shelfbound is built for casual readers, literary enthusiasts, and researchers who seek an intuitive way to manage their book collections. The platform offers features such as custom book lists, nested comment discussions, and community-driven reviews, making book discovery and engagement effortless. Whether users want to keep track of their to-read list, rate books, or gain insights from in-depth reviews, Shelfbound provides a well-structured, visually appealing interface to enhance their reading experience.

In an era where digital solutions shape how we consume and interact with content, having a dedicated book management and review platform is more important than ever. Shelfbound is not just a book-tracking tool—it is a reading companion that helps users discover, organize, and share their love for books in a meaningful way. Whether you are a casual reader or a dedicated book collector, Shelfbound ensures that your reading journey is always organized, insightful, and engaging.

4. OBJECTIVES AND SCOPE OF THE PROJECT

Objectives

- To provide a **centralized platform** for book lovers to **store, manage, and access** their book collections and reviews efficiently.
- To enhance **reading engagement** by allowing users to **rate, review, and categorize** books effortlessly.
- To offer a **user-friendly and intuitive interface** for organizing personal reading lists and tracking book progress.
- To promote **literary discussions and knowledge sharing** by enabling users to engage in reviews, nested comments, and community-driven recommendations.
- To ensure a **secure and reliable platform** where users can safeguard their reading history, favorite books, and reviews without risk of data loss.

Scope of the Project

- **Shelfbound** allows users to **browse, review, and organize books** in one place, eliminating the need for multiple tracking methods such as notebooks or scattered digital files.
- The platform supports **various book genres and categories**, enabling users to build and maintain a structured reading list.
- **Shelfbound** is designed with **cloud storage support**, allowing users to access their **saved books, reviews, and reading progress** from any device, anywhere.
- The project focuses on creating an **intuitive and visually appealing interface** that ensures **seamless navigation** and **efficient book discovery**.
- **Shelfbound** is useful for **casual readers, book reviewers, educators, and literary enthusiasts**, serving as a **digital repository** for reading insights, recommendations, and interactive discussions.

5. SOFTWARE USED

In the development of the dynamic webpage for **ShelfBound**, the following software technologies were utilized to ensure a smooth, responsive, and user-friendly experience:

Frontend Technologies

HTML (Hypertext Markup Language):

- HTML was used to structure the content of the website, defining the layout for elements like headings, paragraphs, images, and links.
- It ensured that the content was organized and easily accessible, providing the foundational structure for the webpage.

CSS (Cascading Style Sheets):

- CSS was employed to style the website, enhancing the visual appearance of the journal.
- It controlled the layout, colour schemes, fonts, and overall presentation, making the site visually appealing and ensuring it was responsive across devices like desktops, tablets, and smartphones.

JavaScript (JS):

JavaScript was utilized to add interactivity to the website.

- Dynamic features, such as responsive navigation bars, form validation, and user interactions (e.g., click actions), were implemented using JavaScript, enhancing the user experience.

Tailwind CSS:

- Tailwind CSS, a utility-first CSS framework, was used to style the application.
- It provided a flexible and responsive design system with pre-defined utility classes, reducing the need for custom CSS.
- Its mobile-first approach ensured a consistent experience across different screen sizes.

Reference Link - 1

ShadCN:

- ShadCN was integrated for UI components, providing pre-styled and customizable elements.
- It enhanced the overall visual appeal of the application while maintaining consistency in design.

Reference Link - 2

Backend Technologies**Node.js:**

- Node.js served as the runtime environment for executing JavaScript on the server side.
- Its asynchronous, event-driven architecture enabled efficient handling of multiple user requests simultaneously.

Reference Link - 3

Express.js:

- Express.js, a lightweight web framework for Node.js, was used to manage server-side logic and API routing.
- It simplified tasks such as handling authentication, processing user requests, and managing database interactions.

Reference Link - 4

Authentication & Security**JWT (JSON Web Token):**

- JWT was implemented for secure authentication, allowing users to log in and access their personalized book-tracking data.
- It ensured secure and stateless user sessions, reducing the need for repeated authentication requests.

Reference Link - 5

bcrypt:

- bcrypt was used for password hashing, enhancing the security of user credentials.
- It protected against brute-force attacks by adding a hashing and salting mechanism to stored passwords.

Reference Link - 6

CORS (Cross-Origin Resource Sharing):

- CORS was configured to allow secure cross-origin requests between the frontend and backend.
- It ensured that only authorized domains could interact with the backend API, improving security.

Database & Storage

MongoDB:

- MongoDB, a NoSQL database, was used to store book-related data, user information, and tracking details.
- Its flexible schema made it easy to handle different types of data structures, such as user preferences and book metadata.

Reference Link - 7

Testing & API Development

Postman:

- Postman was used to test API endpoints and ensure smooth interaction between the frontend and backend.
- It facilitated debugging, performance testing, and verification of authentication mechanisms.

Reference Link - 8

Deployment

Vercel (Frontend):

- The frontend was deployed using Vercel, ensuring quick and reliable hosting with automatic optimizations for performance.

Reference Link - 9

Render (Backend):

- The backend was deployed on Render, providing a scalable and serverless hosting solution for API endpoints and database interactions.

Reference Link - 10

6. PROPOSED MODEL

Shelfbound provides a structured platform for **storing, organizing, and retrieving books and reviews** with a seamless user experience.

Key Features:

1. **Book Organization & Storage:** Users can categorize books by **genre, author, or tags**, with metadata like cover images and ratings.
2. **User Authentication & Security:** **JWT-based authentication** and **Bcrypt encryption** ensure secure access.
3. **Search & Retrieval:** Users can quickly **find books, reviews, and discussions** using keyword-based search.
4. **Cloud-Based Accessibility:** Books, reviews, and preferences are stored securely, **accessible from any device**.
5. **User-Friendly Interface:** A clean **UI/UX design** enhances book discovery and review submission.
6. **Review & Comment System:** Users can **rate books and leave nested comments**, enabling interactive discussions.
7. **Book Management:** Admins can **update, delete, and manage books and user data** efficiently.

Future Enhancements:

- **AI-powered recommendations** and **offline access**.
- **Social reading clubs** and **personalized challenges** to boost engagement.

With these features, **Shelfbound** offers an **interactive, secure, and user-friendly** book management experience.

7. SAMPLE CODE

Github link : <https://github.com/yogithaaah/ShelfBound>

App.jsx:

```
import Footer from "@components/Footer";

import Header from "@components/Header";

import Sidebar from "@components/Sidebar";

import useUserData from "@hooks/useUserData";

import { useGoogleOneTapLogin } from "@react-
oauth/google";

import axios from "axios";

import { useEffect } from "react";

import { Outlet, useNavigate } from "react-router-dom";

import { toast } from "sonner";

function App() {

  const [isLoggedIn, setIsLoggedIn] = useUserData();

  useEffect(() => {

    const logVisit = async () => {

      const userAgent = navigator.userAgent;

      const { data } = await axios.post(

        `${import.meta.env.VITE_BACKEND_URL}/log-visit,

        {

          userAgent,

        }

      );console.log(Total unique visitors: ${data.totalVisitors}.);

      console.log(You have visited ${data.totalVisits}

time(s).);
```

```

    };

    logVisit();

  }, []);

  useGoogleOneTapLogin({

    onSuccess: async (credentialResponse) => {

      let promise = axios.post(

        ${import.meta.env.VITE_BACKEND_URL}/users/google-auth,

        {

          token: credentialResponse.credential,

          auth_method: "google",

        }

      );

      toast.promise(promise, {

        loading: "Loading...",

        success: (response) => {

          const { token } = response.data;

          localStorage.setItem("token", token);

          setIsLoggedIn(true);

          return response.data.message;

        },

        error: (error) => error.response.data.message,

      });

    },

    auto_select: true,

```

```

    disabled: isLoggedIn,

  });

  return (
    <div className="flex min-h-dvh w-full flex-col
dark:bg-zinc-950 dark:text-zinc-50">
      <Sidebar />

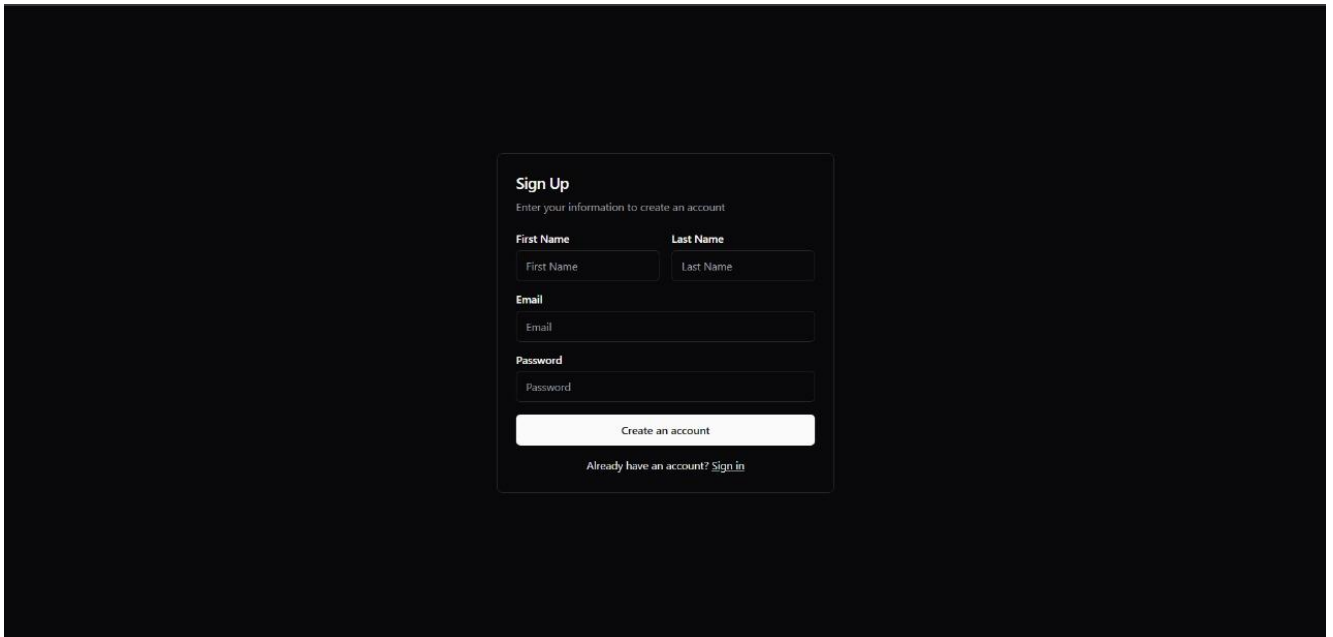
      <div className="flex flex-col min-h-dvh sm:pl-14">
        <Header />
        <Outlet />

        <div className="flex-1"></div>
      </div>
    </div>
  );
}

export default App;

```

8. RESULT / OUTPUT



Sign Up
Enter your information to create an account

First Name

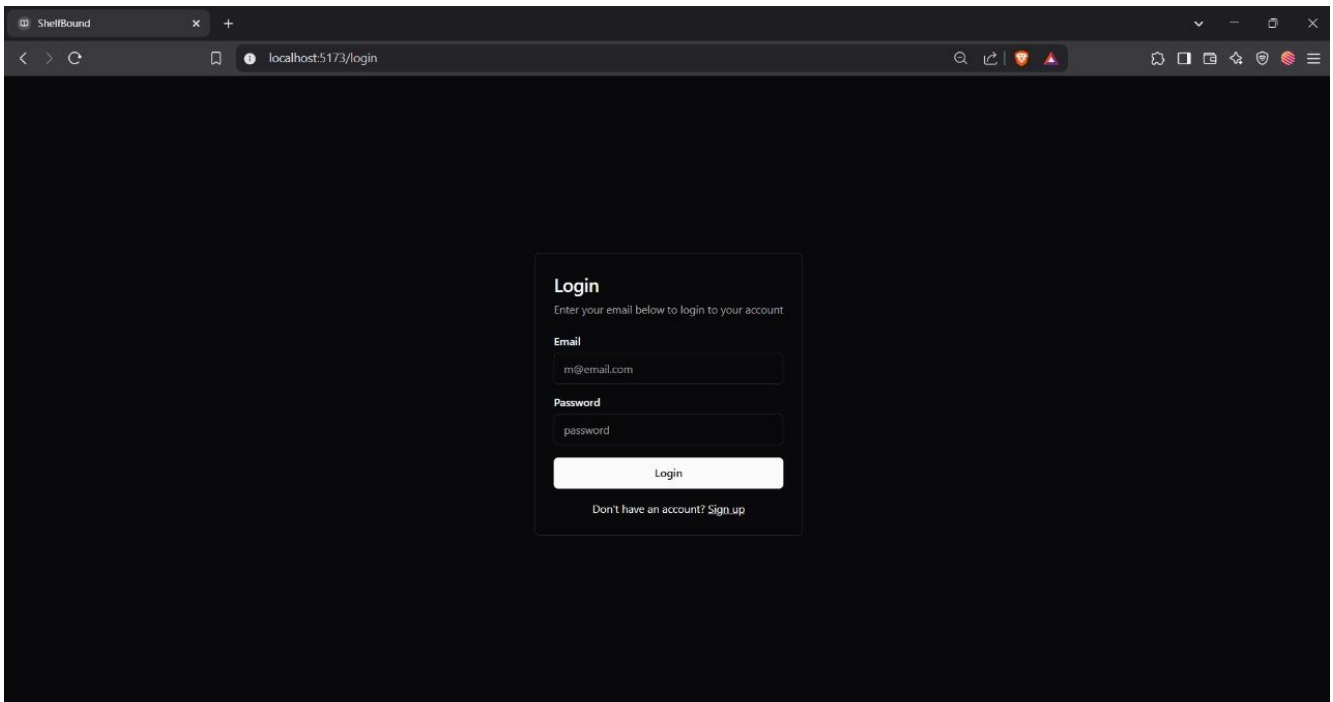
Last Name

Email

Password

Already have an account? [Sign in](#)

Fig.1 Signup Page



Login
Enter your email below to login to your account

Email

Password

Don't have an account? [Sign up](#)

Fig.2 Login Page

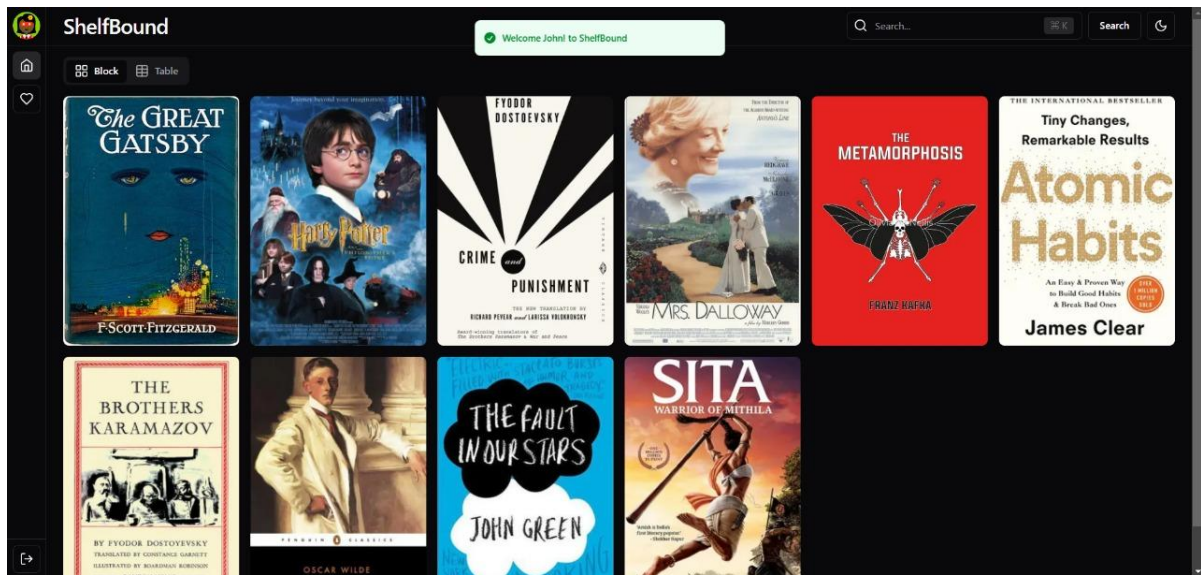


Fig.3 Home Page

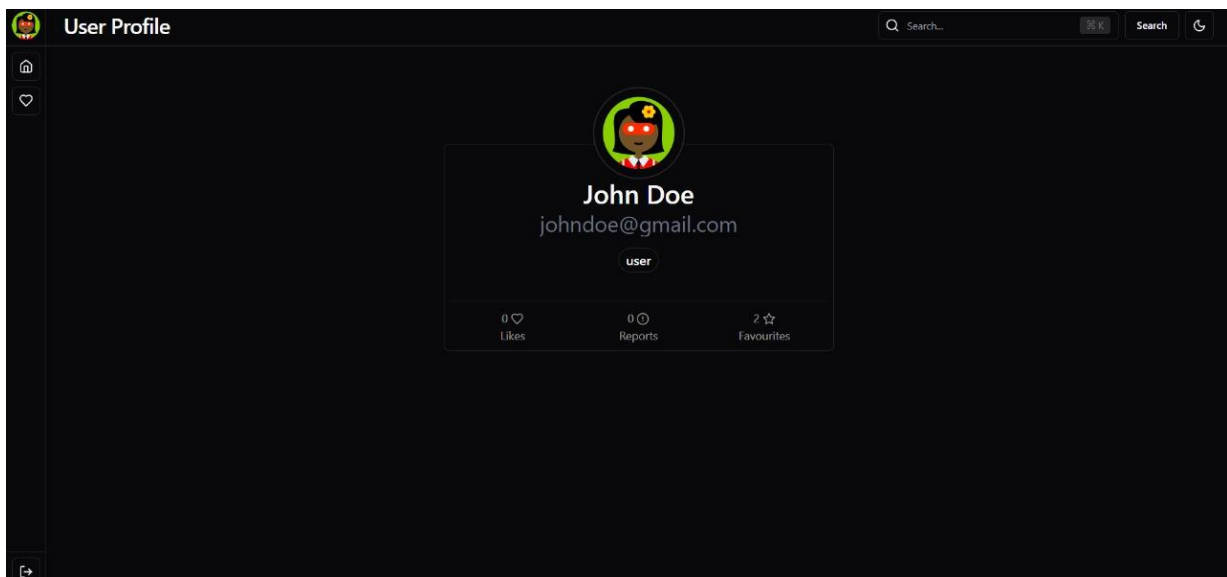


Fig .4 User's Profile

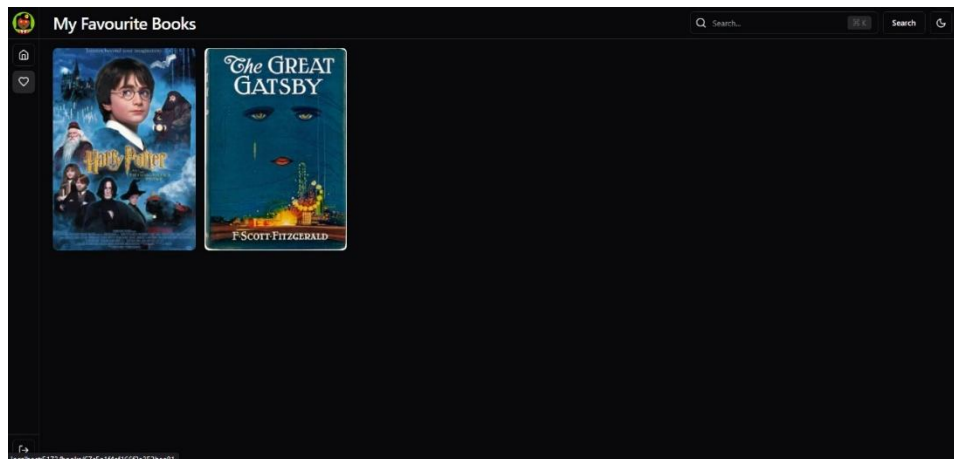


Fig.5 Favorite books



Fig.6 Book Details

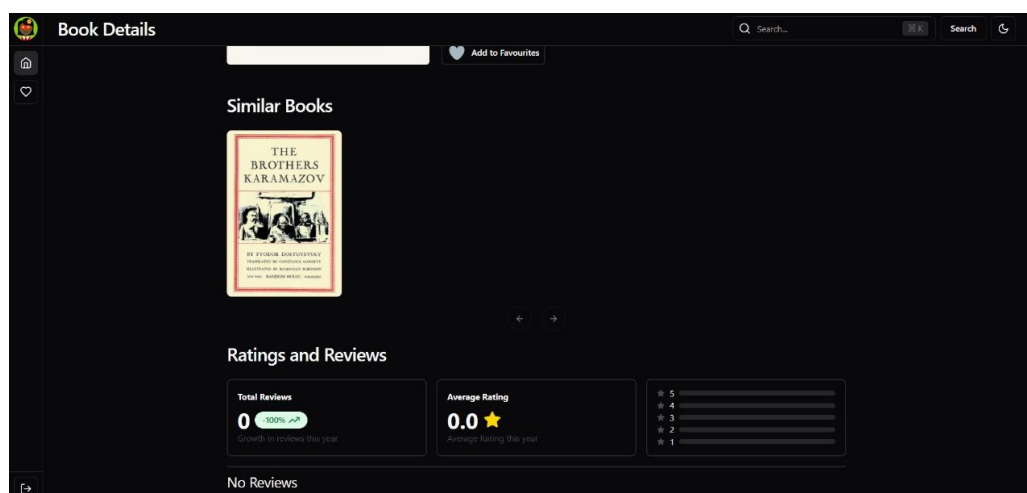


Fig.7 Reviews & Ratings

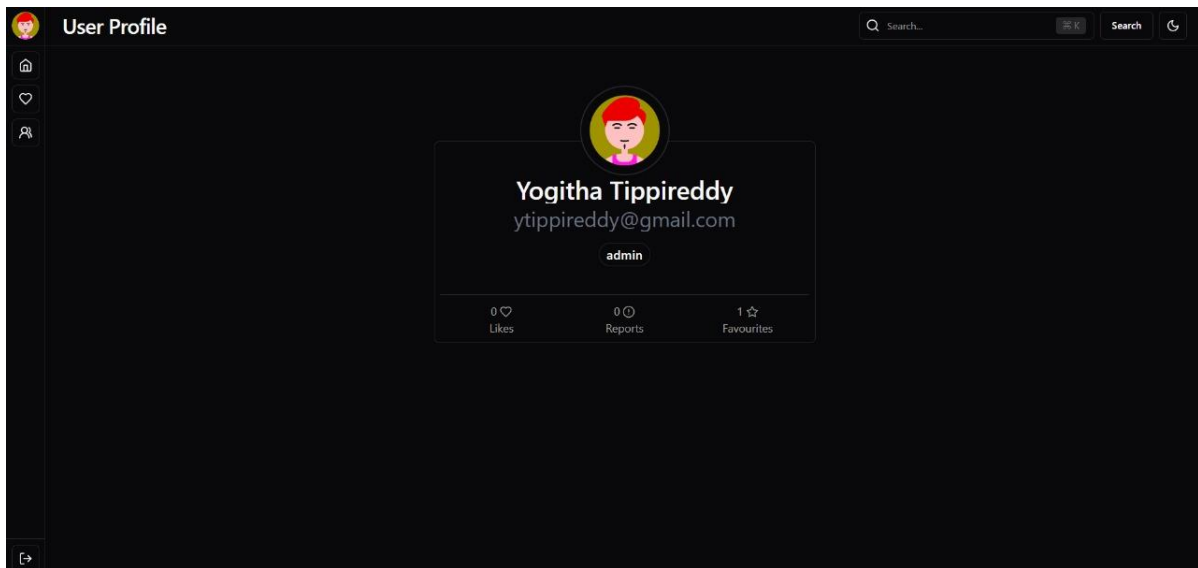


Fig.8 Admin Profile

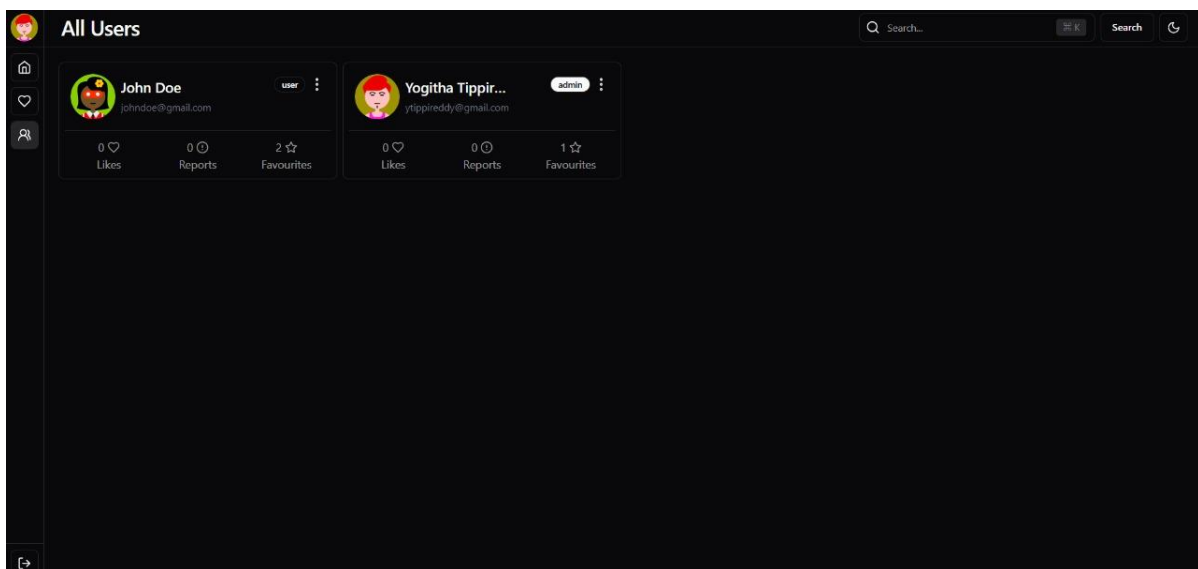


Fig.9 All users Page

Add Book

Search...

Add Book

Title
Title of the Book

Author
Author of the Book

Year
2025

Description
Enter a brief summary of the book...

Genre
Genre

Image
Choose File No file chosen

Submit

Fig.10 Add Book Page

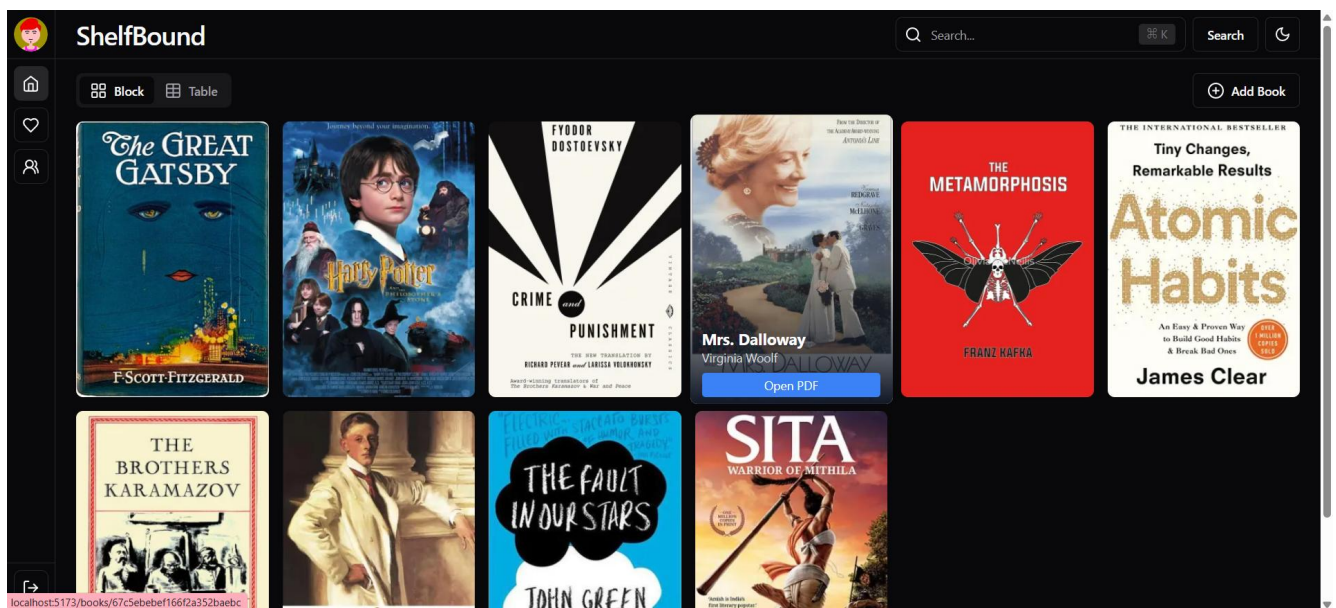


Fig.11 PDF Viewing

9. CONCLUSION

The Shelfbound project serves as a powerful platform designed to help book enthusiasts discover, review, and manage their reading collections in a structured and user-friendly environment. By leveraging modern technologies like React for the frontend, MongoDB for the backend, and JWT for authentication, Shelfbound ensures a seamless experience for users who need a centralized space to track and interact with books. Through this project, we aimed to simplify book management by providing features such as search functionality, nested commenting, and cloud-based storage, enabling users to focus on their reading journey without organizational hassles.

Additionally, Shelfbound enhances engagement by offering a streamlined interface for users to browse, rate, and discuss books efficiently. The project emphasizes a clutter-free experience, ensuring that readers can easily access and manage their book lists. As we continue to develop Shelfbound, our goal is to incorporate user feedback and new features such as AI-powered book recommendations, offline access, and social book clubs to further enrich the user experience.

In conclusion, Shelfbound is not just a book-tracking tool—it is a reading companion that enhances book discovery, community engagement, and literary exploration. Whether you're a casual reader or a dedicated book collector, Shelfbound provides a reliable and interactive way to organize, review, and enjoy books, making your reading journey more structured, engaging, and enjoyable.

10.REFERENCES

1. Tailwind CSS:

- Tailwind CSS is a utility-first CSS framework that provides pre-defined classes for rapid UI development, ensuring a consistent and responsive design.

Reference Link: [Tailwind CSS Official Documentation](#)

2. ShadCN:

- ShadCN offers beautifully designed, accessible, and customizable UI components that can be seamlessly integrated into applications.

Reference Link: [ShadCN UI Official Website](#)

Backend Technologies:

3. Node.js:

- Node.js is a runtime environment that allows the execution of JavaScript on the server side, featuring an asynchronous, event-driven architecture for efficient request handling.

Reference Link: [Node.js Official Website](#)

4. Express.js:

- Express.js is a lightweight web framework for Node.js, simplifying server-side logic and API routing, and facilitating tasks like authentication and database interactions.

Reference Link: [Express.js Official Website](#)

Authentication & Security:

5. JWT (JSON Web Token):

- JWT is implemented for secure authentication, allowing users to log in and access personalized data through stateless user sessions.

Reference Link: [JWT Introduction](#)

6. bcrypt:

- bcrypt is used for password hashing, enhancing the security of user credentials by adding hashing and salting mechanisms to stored passwords.

Reference Link: [bcrypt NPM Package](#)

Database & Storage:

7. MongoDB:

- MongoDB is a NoSQL database used to store data like user information and tracking details, offering a flexible schema for various data structures.

Reference Link: [MongoDB Official Website](#)

Testing & API Development:

8. Postman:

- Postman is used to test API endpoints, facilitating debugging, performance testing, and verification of authentication mechanisms.

Reference Link: [Postman Official Website](#)

Deployment:

9. Vercel (Frontend):

- Vercel is used for deploying the frontend, offering quick and reliable hosting with automatic performance optimizations.

Reference Link: [Vercel Official Website](#)

10. Render (Backend):

- Render is used for deploying the backend, providing scalable and serverless hosting solutions for API endpoints and database interactions.

Reference Link: [Render Official Website](#)