



SRI KRISHNA COLLEGE OF TECHNOLOGY
(An Autonomous Institution)
Approved by AICTE | Affiliated to Anna University Chennai|
Accredited by NBA - AICTE| Accredited by NAAC with 'A' Grade
KOVAIPUDUR, COIMBATORE 641042



INTERACTIVE CHILDREN'S STORYBOOK AND EDUCATION APPLICATION

A PROJECT REPORT

Submitted by

SHRIJITH R

-

727822TUCS218

In partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

JULY 2024

BONAFIDE CERTIFICATE

Certified that this project report **STAFF SCHEDULING SYSTEM** is the bonafide work **SHRIJITH R 727822TUCS218** who carried out the project work under my supervision.

SIGNATURE

Ms. A. GOMATHY

SUPERVISOR

Assistant Professor,

Department of Computer Science
and Engineering,

Sri Krishna College of

Technology, Coimbatore-641042

SIGNATURE

Dr. M. UDHAYAMURTHI

HEAD OF THE DEPARTMENT

Associate Professor,

Department of Computer Science and
Engineering,

Sri Krishna College of

Technology, Coimbatore-641042

Certified that the candidate was examined by me in the Project Work Viva Voce examination held on _____ at Sri Krishna College of Technology, Coimbatore-641042.

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

First and foremost we thank the **Almighty** for being our light and for showering his gracious blessings throughout the course of this project.

We express our gratitude to our beloved Principal, **Dr. M.G. Sumithra**, for providing all facilities.

We are grateful to our beloved Head, Computing Sciences **Dr.T. Senthilnathan**, for her tireless and relentless support.

With the grateful heart, our sincere thanks to our Head of the Department **Dr. M. Udhayamurthi**, Department of Computer Science and Engineering for the motivation and all support to complete the project work.

We thank **Ms. A. Gomathy**, Assistant Professor, Department of Computer Science and Engineering, for his motivation and support.

We are thankful to all the **Teaching and Non-Teaching Staff** of Department of Computer Science and Engineering and to all those who have directly and indirectly extended their help to us in completing this project work successfully.

We extend our sincere thanks to our **family members** and our beloved **friends**, who had been strongly supporting us in all our endeavour

ABSTRACT

The Interactive Children's Storybook and Education Application leverages cutting-edge technologies such as React JS, Node.js, REST API, and MongoDB to deliver a dynamic and engaging learning experience for young readers. This innovative platform combines interactive storytelling with educational content, providing an immersive environment where children can explore a variety of interactive stories and educational activities. Key features include interactive story elements, gamified learning modules, personalized content recommendations, and progress tracking. Parents and educators benefit from detailed dashboards that allow them to monitor children's progress, customize learning paths, and access insightful reports on educational development. The application's responsive design ensures an intuitive user experience across devices, while real-time updates and notifications keep users informed about new content and achievements. By harnessing React JS's flexible frontend capabilities, Node.js's scalable backend architecture, and MongoDB's efficient data handling, the application fosters an engaging and educational environment that adapts to individual learning needs. Its scalability and user-centric design ensure a fun and effective learning journey for children, while providing valuable tools for parents and educators to support and enhance the educational experience.

TABLE OF CONTENT

CHAPTER.NO	TITLE	PAGE NO
1	INTRODUCTION	1
	1.1 PROJECT OVERVIEW	1
	1.2 SCOPE OF THE PROJECT	1
	1.3 OBJECTIVE	1
2	SYSTEM SPECIFICATIONS	2
3	SOFTWARE DEVELOPMENT LIFE CYCLE	4
	3.1 PHASES OF SDLC	4
	3.2 OVERVIEW OF SDLC	4
4	METHODOLOGIES	6
5	IMPLEMENTATION AND RESULT	15
6	CONCLUSION AND FUTURE SCOPE	46
	6.1 CONCLUSION	46
	6.2 FUTURE SCOPE	46

LIST OF FIGURES

Figure No	TITLE	Page No
4.1	Process-Flow Diagram	6
4.2	Use Case Diagram	8
4.3	Class Diagram	9
4.4	Sequence Diagram	10
5.1	Login Page	11
5.2	User Register	12
5.3	Admin Dashboard	14
5.4	Admin Management	14
5.5	User Dashboard	15
5.6	User Management	16
5.7	Book Page	16

LIST OF ABBREVIATIONS

Abbreviation

Acronym

HTML

HYPertext MARKUP LANGUAGE

CSS

CASCADING STYLESHEET

JS

JAVASCRIPT

CHAPTER 1

INTRODUCTION

This project aims to deliver a dynamic and engaging solution for creating interactive storybooks and educational experiences for children through an online platform. In this chapter, we will explore the problem statement, provide an overview of the application, and outline the main objectives of the interactive storybook and educational app.

1.1 PROBLEM STATEMENT

How can we develop an interactive storybook and educational app that allows children to engage with interactive stories and educational content, while providing parents and educators with tools to monitor progress and customize learning experiences? The app should feature an intuitive user interface and seamlessly integrate interactive elements and educational modules.

1.2 OVERVIEW

In the domain of children's education and interactive storytelling, developers often encounter challenges such as engaging young readers, delivering educational content effectively, and providing valuable insights to parents and educators. Traditional methods can be fragmented and lack interactive elements, resulting in reduced engagement and less effective learning experiences. To address these challenges, we propose the development of an Interactive Storybook and Educational App. This app will harness modern technologies to offer a captivating, user-friendly platform that combines interactive storytelling with educational activities. By integrating gamified learning modules, personalized content, and progress tracking, the app aims to enhance learning outcomes, keep children engaged, and provide actionable insights for parents and educators.

1.3 OBJECTIVE

The primary objective of this project is to create an Interactive Storybook and Educational App that offers children an engaging and educational experience while providing parents and educators with effective tools for monitoring and enhancing learning.

CHAPTER 2

SYSTEM SPECIFICATION

In this chapter, we are going to see the software that we have used to build the website. This chapter gives you a small description about the software used in the project.

2.1 VS CODE

Visual Studio Code is a source code editor developed by Microsoft for Windows, Linux, and macOS. It includes support for debugging, embedded Git control, syntax highlighting, intelligent code completion, snippets, and code refactoring. It is also customizable, so users can change the editor's theme, keyboard shortcuts, and preferences.

VS Code is an excellent code editor for React projects. It is lightweight, customizable, and has a wide range of features that make it ideal for React development. It has built-in support for JavaScript, JSX, and TypeScript, and enables developers to quickly move between files and view detailed type definitions. It also has a built-in terminal for running tasks. Additionally, VS Code has an extensive library of extensions that allow developers to quickly add features like code snippets, debugging tools, and linting support to their projects.

2.2 LOCAL STORAGE

Local storage is a type of web storage for storing data on the client side of a web browser. It allows websites to store data on a user's computer, which can then be accessed by the website again when the user returns. Local storage is a more secure alternative to cookies because it allows websites to store data without having to send it back and forth with each request. Local storage is a key-value pair storage mechanism, meaning it stores data in the form of a key and corresponding value. It is similar to a database table in that it stores data in columns and rows, except that local storage stores the data in the browser rather than in a database. Local storage is often used to store user information such as

preferences and settings, or to store data that is not meant to be shared with other websites. It is also used to cache data to improve the performance of a website. Local storage is supported by all modern web browsers, including Chrome,

Firefox, Safari, and Edge. It is accessible through the browser's JavaScript API. Local storage is a powerful tool for websites to store data on the client side. It is secure, efficient, and can be used to store data that does not need to be shared with other websites.

Local Storage is a great way to improve the performance of a website by caching data. Local storage in web browsers allows website data to be stored locally on the user's computer. It is a way of persistently storing data on the client side, which is not sent to the server with each request. This allows users to store data such as preferences, login information, and form data without needing to send it to a server. It is typically stored in a browser's cookie file, but it can also be stored in other locations such as HTML5 Local Storage and Indexed DB. The data stored in local storage is persistent and can be accessed by the website even if the user closes the browser or navigates to another page. It is a great way for websites to store user-specific data, as it is secure, reliable, and fast. It is also a great way for developers to store data that does not need to be sent to the server with each request.

One of the key benefits of using local storage is its reliability. Unlike server-side storage, which can be affected by network outages or other server issues, local storage is stored locally on the user's machine, and so is not affected by these issues. Another advantage of local storage is its speed. Because the data is stored locally, it is accessed quickly, as there is no need to send requests to a server. This makes it ideal for storing data that needs to be accessed quickly, such as user preferences or session data. Local storage is also secure, as the data is stored on the user's machine and not on a server. This means that the data is not accessible by anyone other than the user, making it a good choice for storing sensitive information.

CHAPTER 3

PROPOSED SYSTEM

This chapter gives a small description about the proposed idea behind the development of our website

3.1 PROPOSED SYSTEM

This app offers numerous benefits from multiple perspectives. The Interactive Storybook and Educational App empowers children to engage with captivating stories and educational content, while providing parents and educators with valuable tools for monitoring and enhancing learning. The app simplifies the educational experience, allowing children to interact with stories and complete educational activities easily, while parents and educators can track progress and customize learning paths online.

Once a learning module is completed, progress and achievements are recorded and accessible through the app. The app ensures that educational content is up-to-date and relevant, with the ability to adjust learning paths and update content based on user feedback and progress. Children receive real-time updates and notifications about new content and achievements, making learning engaging and dynamic.

The app significantly enhances the learning experience by eliminating traditional barriers such as fragmented educational resources and lack of engagement. It mitigates challenges associated with static learning materials and limited interaction by providing a seamless, interactive platform. Children can directly engage with content and track their progress, avoiding potential delays or disengagement from traditional methods.

Additionally, the app improves educational efficiency by leveraging technology to automate content delivery, progress tracking, and notifications. This results in a more engaging and effective learning journey for children, while providing parents and educators with actionable insights and tools to support and enhance the educational experience.

3.2 ADVANTAGES

- **Efficiency:** The app allows children to interact with engaging storybooks and educational content from anywhere with internet access. This dynamic approach enhances learning experiences and keeps children motivated, reducing the need for traditional, static educational materials.
- **Personalization:** Children can explore a variety of stories and educational modules tailored to their interests and learning needs. This flexibility supports different learning styles and preferences, enhancing engagement and making learning more enjoyable.
- **Progress Tracking:** The app automatically tracks and updates learning progress, providing insights into achievements and areas for improvement. This feature helps parents and educators monitor development and adjust learning paths as needed, ensuring that educational goals are met.
- **Accessibility:** The app is accessible from any device with internet connectivity, allowing children to learn and explore content remotely. This accessibility supports continuous learning and makes it easy for children to engage with educational materials anytime, anywhere.
- **Transparency:** The app provides clear and detailed information about learning progress, achievements, and upcoming content. This transparency helps parents and educators stay informed about children's educational journeys, facilitating better support and communication.
- **Real-Time Updates:** Automated notifications keep children and parents informed about new content, progress updates, and important reminders. This feature ensures timely information delivery, enhancing the overall learning experience and helping to keep both children and educators engaged and informed.

CHAPTER 4

METHODOLOGIES

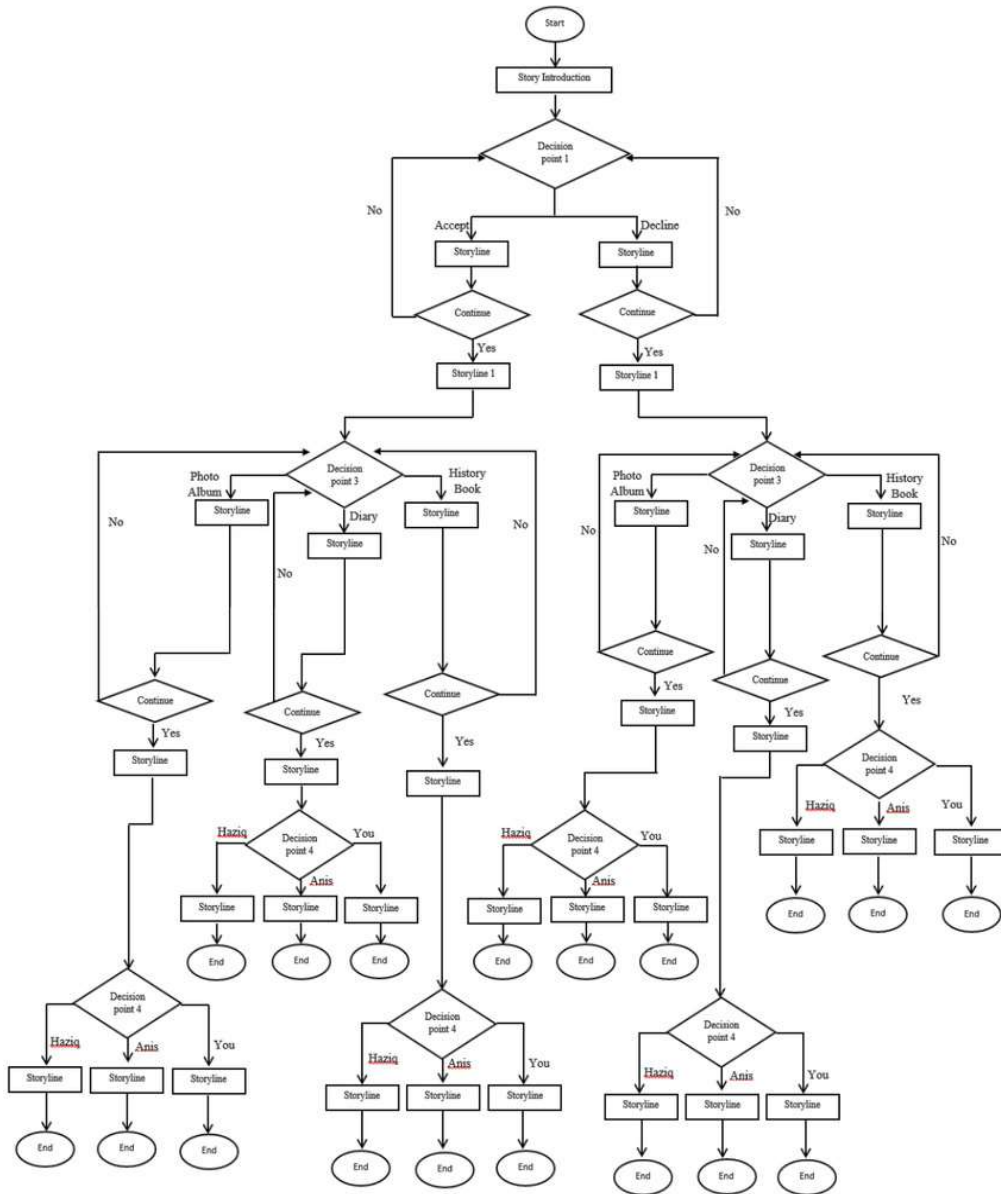


Fig 4.1.Process flow diagram

User Registration:

Children and educators register on the app by providing essential details such as name, email address, and age. Optional information like interests and educational goals may be included to personalize the learning experience.

Role-Based Access:

Upon registration/login, users are assigned roles such as student, parent, or educator, which determine their access permissions and functionalities within the app.

Schedule Exploration:

Users can explore available storybooks and educational modules by browsing through different categories or using search functionality to find content suited to their interests or educational needs.

Schedule Request:

Users can request new stories or educational modules or provide feedback on existing content. Administrators review these requests and consider them for future updates or adjustments.

Schedule Processing:

The app updates and integrates new content or changes in real-time. Automated systems ensure that all content is accurately updated and categorized without errors.

Notification:

After updates, the app generates notifications about new content, changes, or upcoming educational activities. Users receive alerts about new stories or learning modules.

Schedule Confirmation:

Administrators review and confirm the final schedules.

The system updates the schedule and notifies all relevant users about their confirmed assignments.

Real-Time Updates:

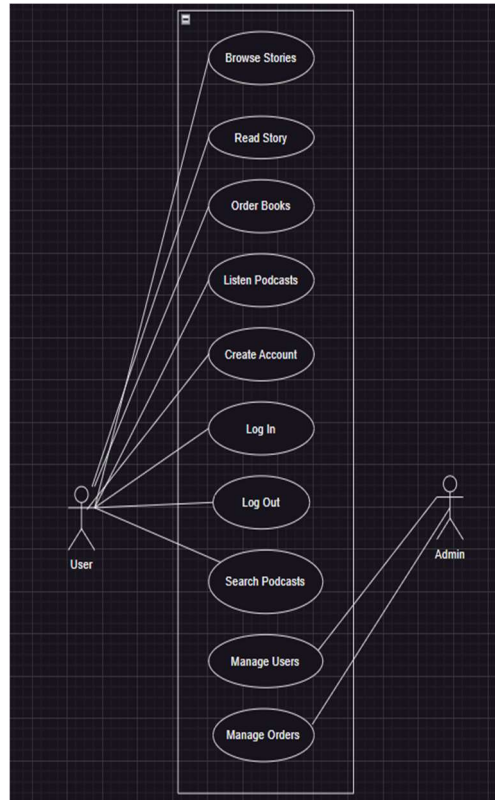
As users engage with the app, they receive reminders and real-time updates about new content or upcoming educational events.

Feedback and Adjustments:

After engaging with content, users can provide feedback on their experience. This feedback is reviewed for potential improvements or adjustments to enhance the overall learning experience.

Use Case Diagram:

The use case diagram illustrates the Interactive Storybook and Educational App, depicting various user roles including Students, Parents, Educators, Administrators, Content Creators, and IT Support. Each role has specific functionalities such as exploring content, managing learning paths, providing feedback, handling notifications, and accessing educational resources. The diagram highlights interaction points for roles to perform tasks like content customization, progress tracking, and activity management. It effectively showcases the workflow, demonstrating the interplay between different users and system features to achieve an engaging and educational experience.

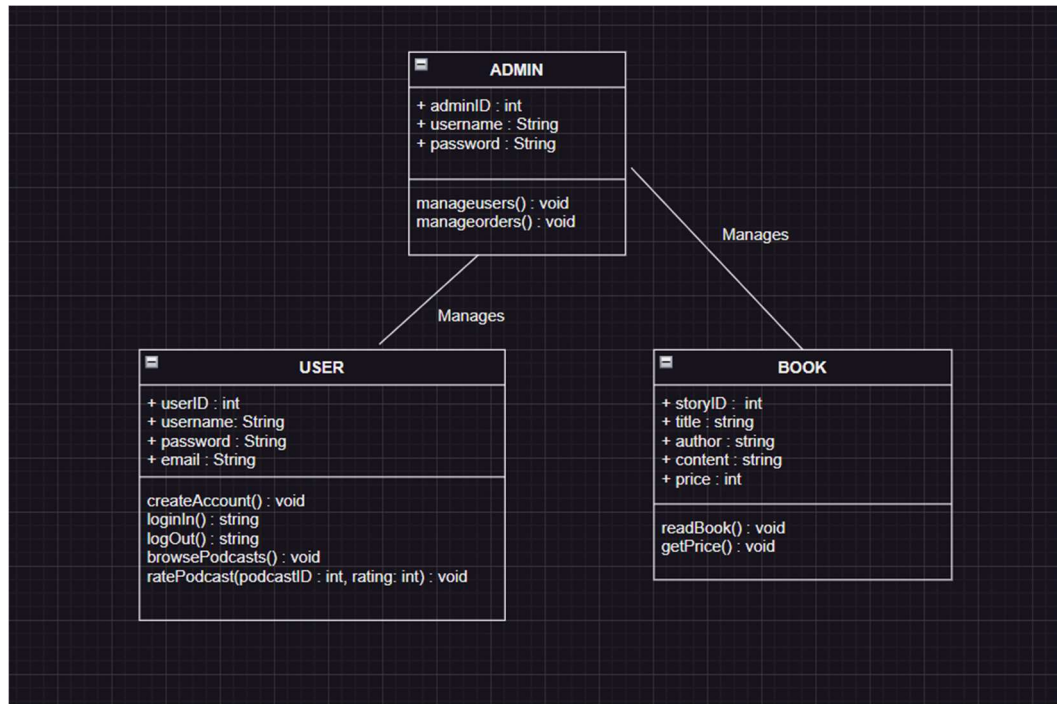


4.2 Use Case Diagram

Class Diagram:

The class diagram outlines the architecture of the Interactive Storybook and Educational App, detailing the roles and their specific functionalities. Key classes include Student, Parent, Educator, Administrator, Content Creator, and IT Support, each interacting with shared entities such as User,

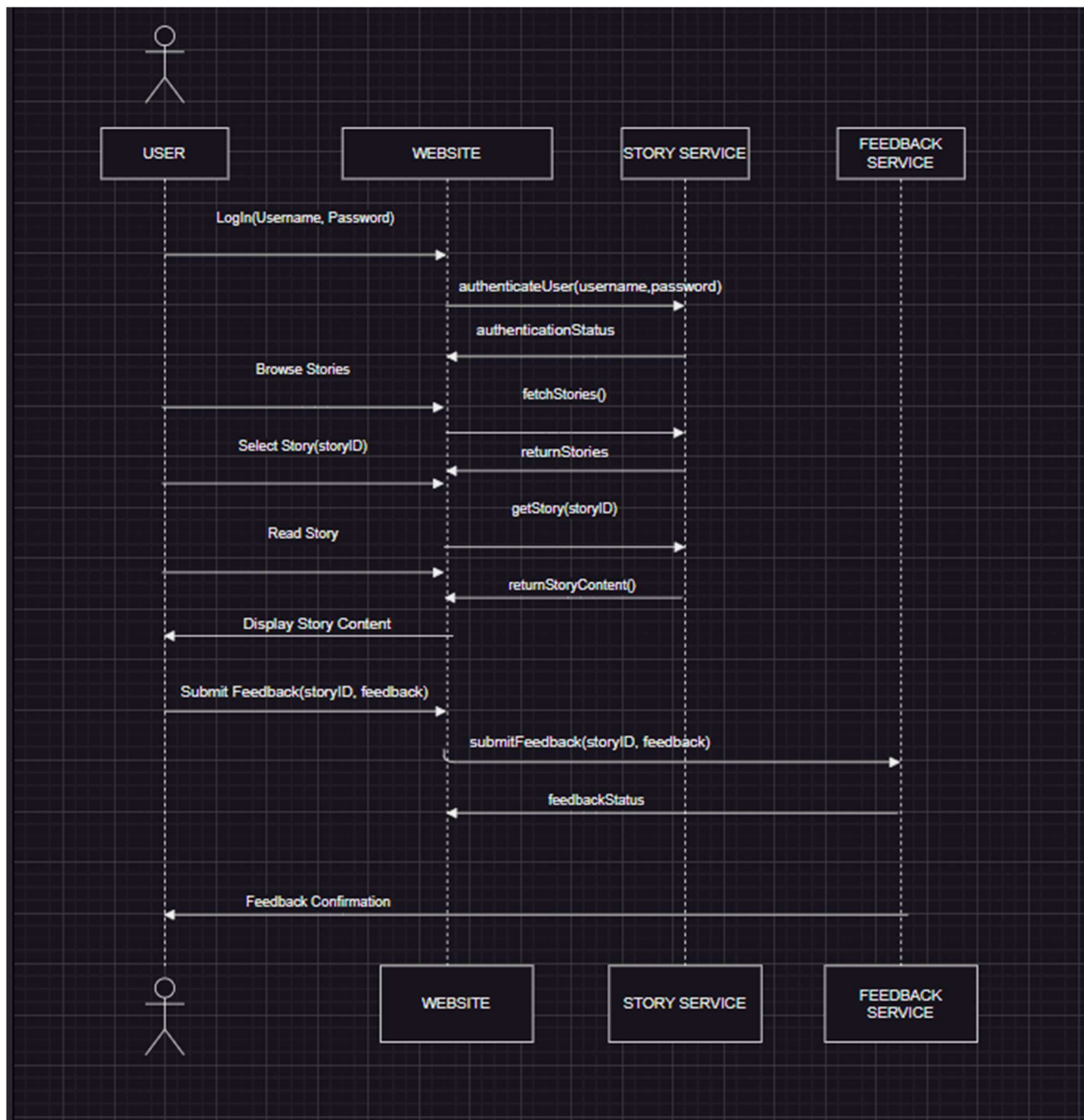
Content, Learning Path, and Notification. This diagram highlights methods for managing user profiles, educational content, learning progress, and notifications, illustrating how each role contributes to the app's overall functionality and user experience.



4.3 Class Diagram

Sequence Diagram:

The Content Interaction Sequence Diagram illustrates the process for a student to engage with educational content on the app. It shows the interaction between the User, Student, System, and Content components to select, access, and interact with storybooks or educational modules, and receive feedback or progress updates. The diagram details how the system processes content requests, updates learning progress, and provides notifications or recommendations based on user interactions.



4.4 Sequence Diagram

CHAPTER 5

IMPLEMENTATION AND RESULT

This chapter gives a description about the output that we produced by developing the website of our idea.

5.1 LOGIN

When User enters our website he will be asked about his login details like email id and password. The login details will be verified with the details given while the user creates an account.

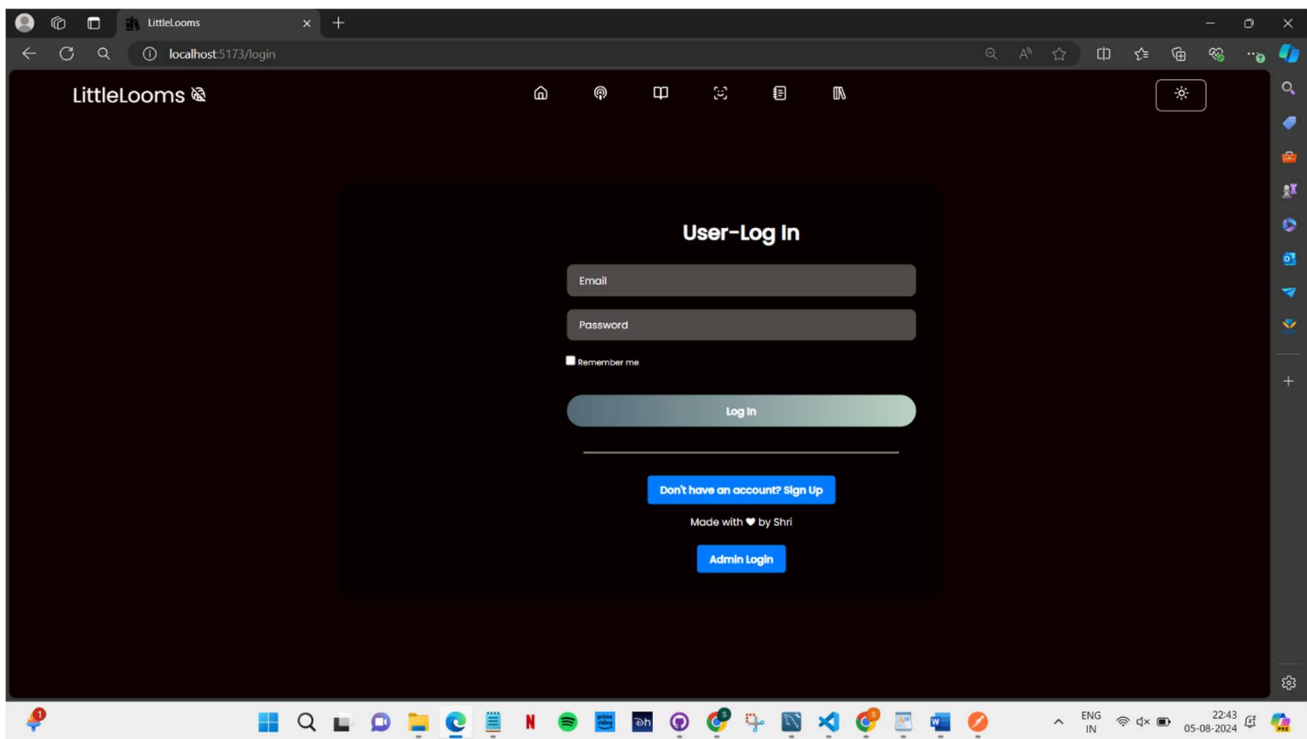
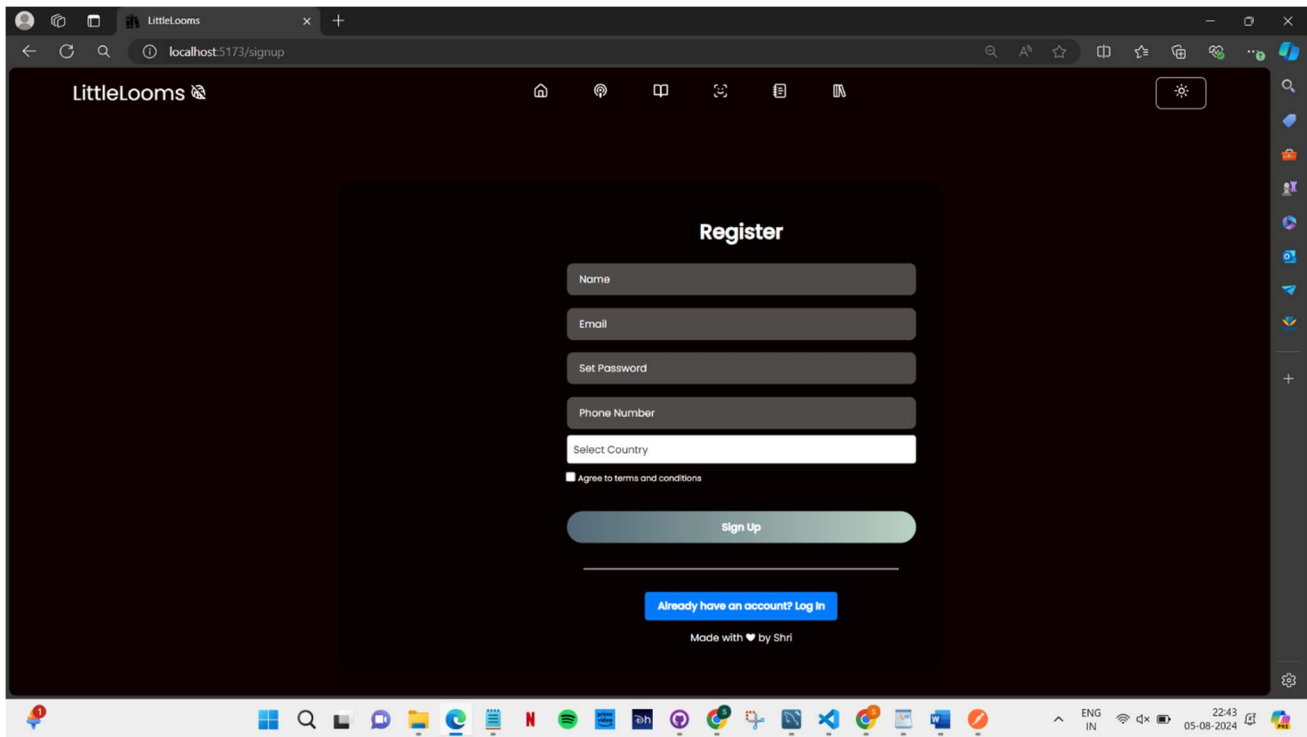


Fig 5.1 LOGIN PAGE

5.2 USER REGISTER



The screenshot shows a web browser window with the address bar displaying "localhost:5173/signup". The page title is "LittleLooms". The main content area has a dark background and features a "Register" form. The form includes the following fields: "Name", "Email", "Set Password", "Phone Number", and "Select Country". Below these fields is a checkbox labeled "Agree to terms and conditions". A "Sign Up" button is positioned below the checkbox. At the bottom of the form, there is a link that says "Already have an account? Log In". The footer of the page reads "Made with ❤️ by Shri". The browser's taskbar at the bottom shows various application icons, including Windows, Search, File Explorer, and several development tools. The system tray on the right indicates the language is "ENG IN", the date is "05-08-2024", and the time is "22:43".

LittleLooms

Register

Name

Email

Set Password

Phone Number

Select Country

☐ Agree to terms and conditions

Sign Up

[Already have an account? Log In](#)

Made with ❤️ by Shri

Fig 5.2 USER REGISTER

5.3 ADMIN DASHBOARD

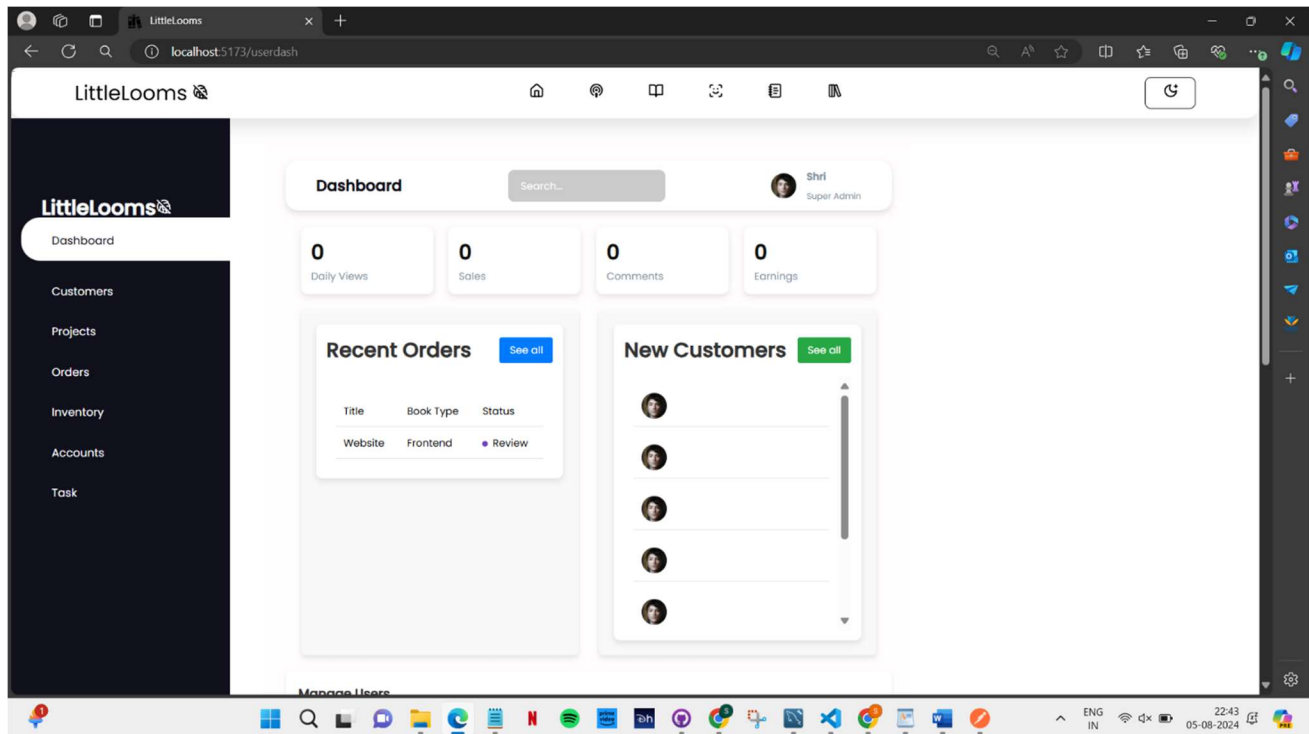


Fig 5.3 Admin Dashboard

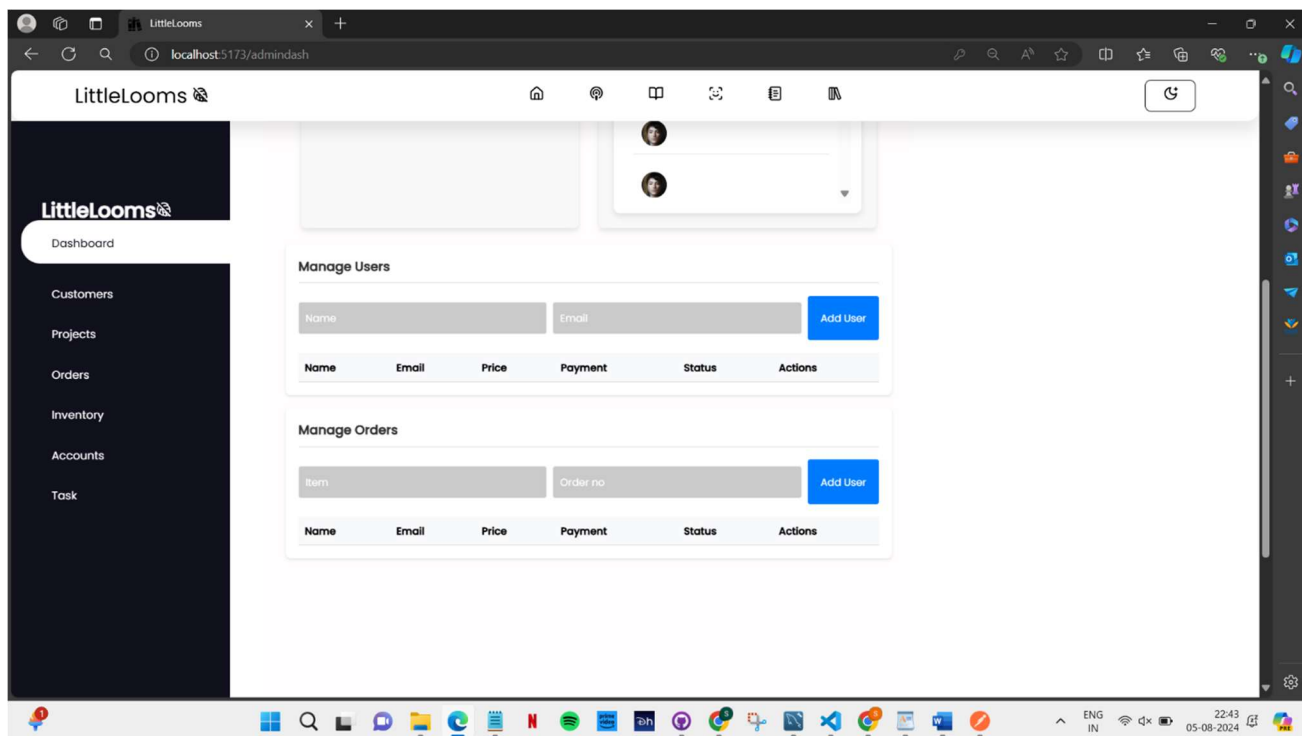


Fig 5.4 Admin Management

5.4 USER DASHBOARD

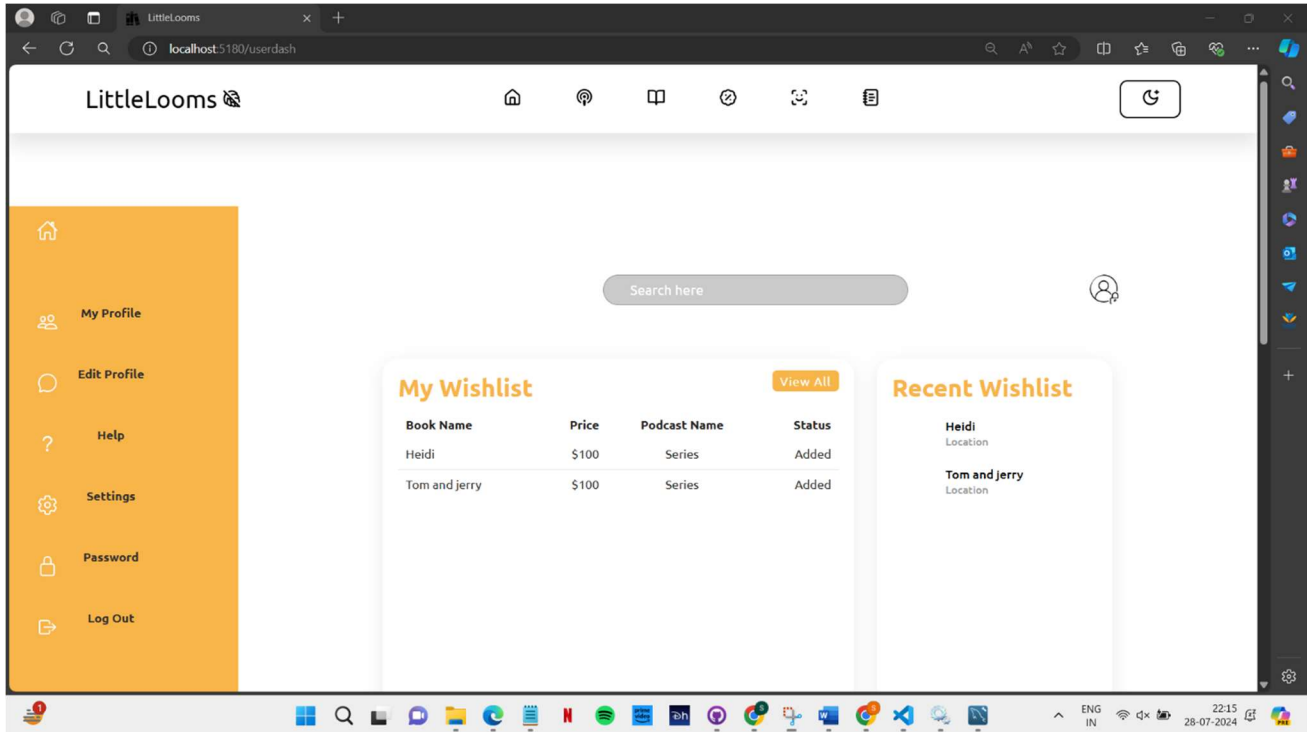


Fig 5.5 User Dashboard

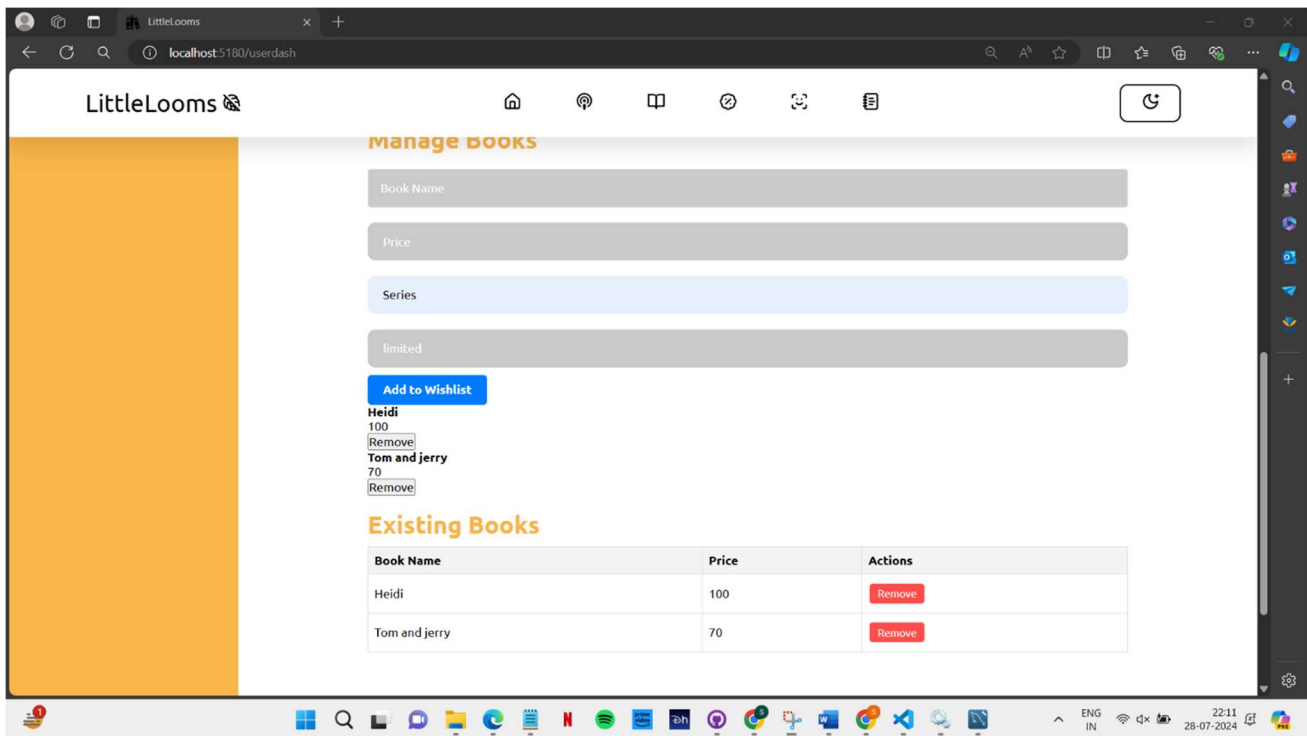


Fig 5.6 User Management

5.5 BOOK PAGE

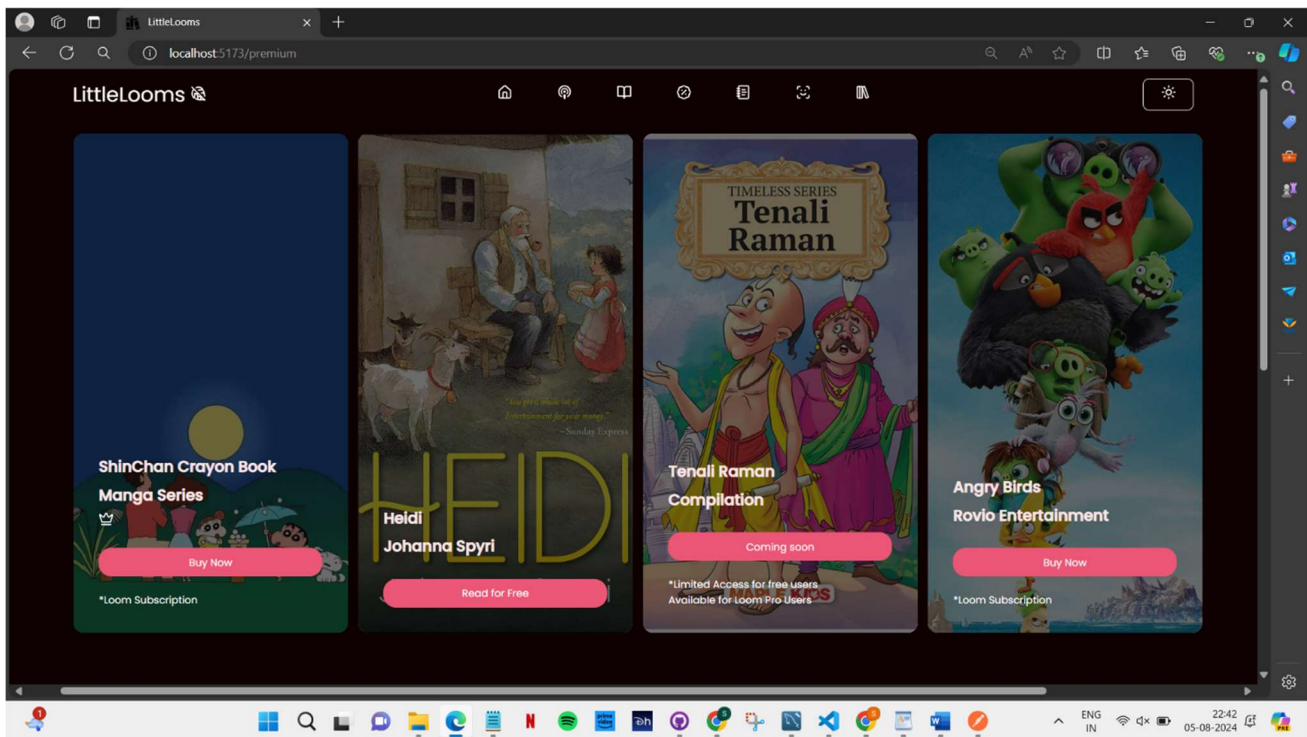


Fig 5.7 Book Page

5.6 CODING

Frontend:

Register:

```
import React from 'react';
import { useNavigate, Link } from 'react-router-dom'; // Import useNavigate and Link
import '../pages/css/Signup.css';
import { ToastContainer, toast } from 'react-toastify';
import 'react-toastify/dist/ReactToastify.css';

const Signup = () => {
  const navigate = useNavigate(); // Initialize useNavigate
```



```

const handleSubmit = (e) => {
  e.preventDefault();

  // Display the toast notification
  toast.success('Registered successfully!', {
    position: "bottom-right",
    autoClose: 2000,
    className: 'toast-custom',
    bodyClassName: 'toast-custom-body',
    progressClassName: 'toast-custom-progress',
  });

  // Navigate to Admindash.jsx
  setTimeout(() => {
    navigate('/admindash'); // Update with your actual route
  }, 2000); // Delay to match toast duration
};

return (
  <div className="signup-background">
    <video
      className="signup-background__video"
      src="/files/video-bg.mp4"
      autoPlay
      muted
      loop
    ></video>
    <main className="signup-main">
      <div className="signup-card__image-container">
        <div className="signup-slide-images">
          
  
  
</div>
</div>
<form className="signup-form" action="#" autoComplete="off"
onSubmit={handleSubmit}>
  <h2 className="signup-heading">Register</h2>
  <div className="signup-input-group">
    <i className="far fa-user fa-lg signup-icon"></i>
    <input
      type="text"
      name="name"
      id="signup-name"
      placeholder="Name"
      required
    />
  </div>
  <div className="signup-input-group">
    <i className="far fa-envelope fa-lg signup-icon"></i>
    <input
      type="email"

```

```

    name="email"
    id="signup-email"
    placeholder="Email"
    required
  />
</div>
<div className="signup-input-group">
  <i className="fas fa-unlock-alt fa-lg signup-icon"></i>
  <input
    type="password"
    name="password"
    id="signup-password"
    placeholder="Set Password"
    required
  />
</div>
<div className="signup-input-group">
  <i className="fas fa-phone fa-lg signup-icon"></i>
  <input
    type="tel"
    name="phone"
    id="signup-phone"
    placeholder="Phone Number"
    required
  />
</div>
<div className="signup-input-group">
  <i className="fas fa-globe fa-lg signup-icon"></i>
  <select
    name="country"
    id="signup-country"
    className="signup-select"

```

```

    required
  >
    <option value="" disabled selected>Select Country</option>
    <option value="usa">United States</option>
    <option value="canada">Canada</option>
    <option value="uk">United Kingdom</option>
    <option value="australia">Australia</option>
    <option value="germany">Germany</option>
    <option value="france">France</option>
    <option value="india">India</option>
    <option value="china">China</option>
    <option value="japan">Japan</option>
    <option value="brazil">Brazil</option>
    { /* Add more countries as needed */ }
  </select>
</div>
<div className="signup-terms">
  <input
    type="checkbox"
    name="agree"
    id="signup-agree"
    required
  />
  <label htmlFor="signup-agree">
    Agree to terms and conditions
  </label>
</div>
<button type="submit" className="signup-button">Sign Up</button>
<hr className="signup-divider" />
<Link to="/login" className="signup-login-button">Already have an account? Log
In</Link>
<center>

```

```

    <p className="signup-creator">Made with <span>❤️</span> by Shri</p>
  </center>
</form>
</main>
<ToastContainer
  position="bottom-right"
  autoClose={2000}
/>
</div>
);
};

export default Signup;

```

AdminDashboard:

```

import React, { useState } from 'react';
import { HopOff } from 'lucide-react';
import './admin/Admindash.css';

const Admindash = () => {
  const [users, setUsers] = useState([]);
  const [metrics, setMetrics] = useState({
    dailyViews: 0,
    sales: 0,
    comments: 0,
    earnings: 0
  });
  const [searchQuery, setSearchQuery] = useState("");
  const [filteredUsers, setFilteredUsers] = useState([]);

```

```

// Function to handle search input changes
function handleSearchChange(event) {
  const query = event.target.value;
  setSearchQuery(query);
  // Filter users based on the search query
  setFilteredUsers(users.filter(user =>
    user.name.toLowerCase().includes(query.toLowerCase())
  ));
}

// Function to add a new user
function addUser() {
  const name = document.getElementById('userName').value;
  const email = document.getElementById('userEmail').value;

  if (name && email) {
    const newUser = {
      name,
      email,
      price: '$50', // Default price
      payment: 'Paid', // Default payment status
      status: 'New' // Default status
    };
    setUsers(prevUsers => {
      const updatedUsers = [...prevUsers, newUser];
      // Filter the updated list based on the search query
      setFilteredUsers(updatedUsers.filter(user =>
        user.name.toLowerCase().includes(searchQuery.toLowerCase())
      ));
      return updatedUsers;
    });
    setMetrics(prevMetrics => ({

```

```

    ...prevMetrics,
    dailyViews: prevMetrics.dailyViews + 1,
    sales: prevMetrics.sales + 1,
    comments: prevMetrics.comments + 1
  }));
document.getElementById('userName').value = "";
document.getElementById('userEmail').value = "";
} else {
  alert('Please fill in all fields.');
```

```

}
}

// Function to handle user actions (e.g., remove user)
function handleUserAction(index, action) {
  setUsers(prevUsers => {
    const updatedUsers = prevUsers.filter((_, i) => i !== index);
    // Filter the updated list based on the search query
    setFilteredUsers(updatedUsers.filter(user =>
      user.name.toLowerCase().includes(searchQuery.toLowerCase())
    ));
    return updatedUsers;
  });
  setMetrics(prevMetrics => ({
    ...prevMetrics,
    comments: action === 'remove' ? prevMetrics.comments - 1 : prevMetrics.comments
  }));
}

function updateUserList(index, action) {
  if (action === 'edit') {
    const newName = prompt("Enter new name:", users[index].name);
    const newEmail = prompt("Enter new email:", users[index].email);

```

```

    if (newName && newEmail) {
      const updatedUser = { ...users[index], name: newName, email: newEmail };
      setUsers(prevUsers => {
        const updatedUsers = [...prevUsers];
        updatedUsers[index] = updatedUser;
        // Filter the updated list based on the search query
        setFilteredUsers(updatedUsers.filter(user =>
          user.name.toLowerCase().includes(searchQuery.toLowerCase())
        ));
        return updatedUsers;
      });
    }
  } else if (action === 'remove') {
    setUsers(prevUsers => {
      const updatedUsers = prevUsers.filter((_, i) => i !== index);
      // Filter the updated list based on the search query
      setFilteredUsers(updatedUsers.filter(user =>
        user.name.toLowerCase().includes(searchQuery.toLowerCase())
      ));
      return updatedUsers;
    });
    setMetrics(prevMetrics => ({
      ...prevMetrics,
      comments: prevMetrics.comments - 1
    }));
  }
}

return (
  <>
  <input type="checkbox" id="nav-toggle" />

```



```

<div className="dashboard-sidebar">
  <div className="sidebar-header">
    <h1>
      <span className="fab fa-asymmetrik"></span>
      LittleLooms<HopOff />
    </h1>
  </div>

  <div className="sidebar-navigation">
    <ul>
      <li>
        <a href="#" className="active-link">
          <span className="fas fa-tachometer-alt"></span>
          <span>Dashboard</span>
        </a>
      </li>
      <li>
        <a href="#">
          <span className="fas fa-users"></span>
          <span>Customers</span>
        </a>
      </li>
      <li>
        <a href="#">
          <span className="fas fa-stream"></span>
          <span>Projects</span>
        </a>
      </li>
      <li>
        <a href="#">
          <span className="fas fa-shopping-cart"></span>
          <span>Orders</span>
        </a>
      </li>
    </ul>
  </div>
</div>

```

```

    </a>
  </li>
  <li>
    <a href="#">
      <span className="fas fa-boxes"></span>
      <span>Inventory</span>
    </a>
  </li>
  <li>
    <a href="#">
      <span className="fas fa-user-circle"></span>
      <span>Accounts</span>
    </a>
  </li>
  <li>
    <a href="#">
      <span className="fas fa-tasks"></span>
      <span>Task</span>
    </a>
  </li>
</ul>
</div>
</div>

```

```

<div className="content-area">
  <header>
    <h2>
      <label htmlFor="nav-toggle">
        <span className="fas fa-bars"></span>
      </label>
      Dashboard
    </h2>

```

```

<div className="search-container">
  <span className="fas fa-search"></span>
  <input type="search" placeholder="Search..." value={searchQuery}
onChange={handleSearchChange} />
</div>

<div className="user-info">
  
  <div>
    <h4>Shri</h4>
    <small>Super Admin</small>
  </div>
</div>
</header>

<main>
  <div className="info-cards">
    <div className="info-card">
      <div>
        <h1>{metrics.dailyViews}</h1>
        <span>Daily Views</span>
      </div>
      <div>
        <span className="fas fa-eye"></span>
      </div>
    </div>
    <div className="info-card">
      <div>
        <h1>{metrics.sales}</h1>
        <span>Sales</span>
      </div>
    </div>
  </div>

```

```

    <div>
      <span className="fas fa-dollar-sign"></span>
    </div>
  </div>
<div className="info-card">
  <div>
    <h1>{metrics.comments}</h1>
    <span>Comments</span>
  </div>
  <div>
    <span className="fas fa-comments"></span>
  </div>
</div>
<div className="info-card">
  <div>
    <h1>{metrics.earnings}</h1>
    <span>Earnings</span>
  </div>
  <div>
    <span className="fas fa-wallet"></span>
  </div>
</div>
</div>

<div className="recent-activity">
  <div className="projects-section">
    <div className="card-container">
      <div className="card-header">
        <h2>Recent Orders</h2>
        <button>
          See all <span className="fas fa-arrow-right"></span>
        </button>
      </div>
    </div>
  </div>
</div>

```

```

</div>
<div className="card-content">
  <div className="table-container">
    <table width="100%">
      <thead>
        <tr>
          <td>Title</td>
          <td>Book Type</td>
          <td>Status</td>
        </tr>
      </thead>
      <tbody>
        { /* Existing project rows */ }
        <tr>
          <td>Website</td>
          <td>Frontend</td>
          <td>
            <span className="status-indicator purple"></span>
            Review
          </td>
        </tr>
        { /* Add more project rows as needed */ }
      </tbody>
    </table>
  </div>
</div>
</div>
</div>
<div className="customers-section">
  <div className="card-container">
    <div className="card-header">
      <h2>New Customers</h2>

```

```

    <button>
      See all <span className="fas fa-arrow-right"></span>
    </button>
  </div>
  <div className="card-content">
    {[...Array(7)].map((_, i) => (
      <div className="customer-item" key={i}>
        <div className="customer-details">
          
          <div>
            { /* <h4>Malik Abushabab</h4> */}
            { /* <small>CEO</small> */}
          </div>
        </div>
        <div>
          <div className="customer-actions">
            <span className="fas fa-user-circle"></span>
            <span className="fas fa-comment"></span>
            <span className="fas fa-phone-alt"></span>
          </div>
        </div>
      )]}
    </div>
  </div>
</div>

{ /* User Management Section */}
<div className="userManagement">
  <div className="card-header">
    <h2>Manage Users</h2>
  </div>
  <div className="form">

```

```

<input type="text" id="userName" placeholder="Name" />
<input type="email" id="userEmail" placeholder="Email" />
<button onClick={addUser}>Add User</button>
</div>
<div className="table-container">
  <table width="100%">
    <thead>
      <tr>
        <th>Name</th>
        <th>Email</th>
        <th>Price</th>
        <th>Payment</th>
        <th>Status</th>
        <th>Actions</th>
      </tr>
    </thead>
    <tbody>
      {filteredUsers.map((user, index) => (
        <tr key={index}>
          <td>{user.name}</td>
          <td>{user.email}</td>
          <td>{user.price}</td>
          <td>{user.payment}</td>
          <td>{user.status}</td>
          <td>
            <button onClick={() => handleUserAction(index, 'remove')}>Remove</button>
            <br></br>
            <button onClick={() => updateUserList(index, 'edit')}>Edit</button>
          </td>
        </tr>
      )
    )
  )}

```

```

    </tbody>
  </table>
</div>
</div>

```

```

<div className="userManagement">
  <div className="card-header">
    <h2>Manage Orders</h2>
  </div>
  <div className="form">
    <input type="text" id="userName" placeholder="Item" />
    <input type="email" id="userEmail" placeholder="Order no" />
    <button onClick={addUser}>Add User</button>
  </div>
  <div className="table-container">
    <table width="100%">
      <thead>
        <tr>
          <th>Name</th>
          <th>Email</th>
          <th>Price</th>
          <th>Payment</th>
          <th>Status</th>
          <th>Actions</th>
        </tr>
      </thead>
      <tbody>
        {filteredUsers.map((user, index) => (
          <tr key={index}>
            <td>{user.name}</td>
            <td>{user.email}</td>

```



```

        <td>{user.price}</td>
        <td>{user.payment}</td>
        <td>{user.status}</td>
        <td>
            <button onClick={() => handleUserAction(index, 'remove')}>Remove</button>
            <br></br>
            <br></br>
            <button onClick={() => updateUserList(index, 'edit')}>Edit</button>
        </td>
    </tr>
    )})
</tbody>
</table>
</div>
</div>
</main>
</div>
</>

);
}

```

```
export default Admindash;
```

BACKEND:

Config:

```
@Configuration
```

```
@RequiredArgsConstructor
```

```
public class AppConfig {
```

```
    private final UserRepo userRepo;
```

@Bean

```
public UserDetailsService userDetailsService() {
    return username -> userRepo.findByEmail(username)
        .orElseThrow(() -> new UsernameNotFoundException("User not found."));
}
```

@Bean

```
public PasswordEncoder passwordEncoder() {
    return new BCryptPasswordEncoder();
}
```

@Bean

```
public AuthenticationManager authenticationManager(AuthenticationConfiguration
authenticationConfiguration)
    throws Exception {
    return authenticationConfiguration.getAuthenticationManager();
}
}
```

@Configuration

@EnableWebSecurity

@EnableMethodSecurity

@RequiredArgsConstructor

public class SecurityConfig {

private final UserDetailsService userDetailsService;

private final JwtAuthenticationFilter jwtAuthenticationFilter;

private final LogoutHandler logoutHandler;

private static final String[] PublicEndpoints = {

```

"/api/auth/**",
"/api/web/sites",
"/swagger-ui/**",
"/swagger-ui.html/**",
"/api/admin/default",
"/v3/api-docs/**"
};

```

@Bean

```

public SecurityFilterChain securityFilterChain(HttpSecurity httpSecurity) throws Exception
{
    return httpSecurity
        .csrf(AbstractHttpConfigurer::disable)
        .cors(cors -> cors.configurationSource(corsConfigurationSource()))
        .authorizeHttpRequests(
            authorize ->
            authorize.requestMatchers(PublicEndPoints).permitAll()
                .anyRequest().authenticated())
        .sessionManagement(session ->
            session.sessionCreationPolicy(STATELESS))
        .userDetailsService(userDetailsService)
        .addFilterBefore(jwtAuthenticationFilter,
            UsernamePasswordAuthenticationFilter.class)
        .logout(logout -> logout.logoutUrl("/api/auth/logout")
            .addLogoutHandler(logoutHandler)
            .logoutSuccessHandler((request, response,
                authentication) -> SecurityContextHolder
                    .clearContext()))
        .build();
}

```

@Bean

```

public CorsConfigurationSource corsConfigurationSource() {
    CorsConfiguration corsConfiguration = new CorsConfiguration();
    // Note : Replace with server url/ip in production
    corsConfiguration.setAllowedOrigins(Arrays.asList("http://localhost:5173"));
    corsConfiguration.setAllowedHeaders(Arrays.asList(AUTHORIZATION,
CONTENT_TYPE));
    corsConfiguration.setAllowedMethods(Arrays.asList(GET.name(), POST.name(),
PUT.name(), PATCH.name(),
        DELETE.name(), HEAD.name(), OPTIONS.name()));
    corsConfiguration.setAllowCredentials(true);
    UrlBasedCorsConfigurationSource source = new UrlBasedCorsConfigurationSource();
    source.registerCorsConfiguration("/*", corsConfiguration);
    return source;
}
}

```

@Component

@RequiredArgsConstructor

```

public class JwtAuthenticationFilter extends OncePerRequestFilter {

```

```

    private final JwtToken jwtTokenUtil;

```

```

    private final UserDetailsService userDetailsService;

```

```

    private final JwtRepo jwtRepo;

```

@Override

```

protected void doFilterInternal(@NonNull HttpServletRequest request, @NonNull
HttpServletResponse response,

```

```

    @NonNull FilterChain filterChain) throws ServletException, IOException {

```

```

    final String authHeader = request.getHeader("Authorization");

```

```

    final String token;

```

```

    final String username;

```

```

    if (authHeader == null || !authHeader.startsWith("Bearer ")) {
        filterChain.doFilter(request, response);
        return;
    }
    token = authHeader.substring(7);
    username = jwtTokenUtil.extractUsername(token);
    if (username != null && SecurityContextHolder.getContext().getAuthentication() == null)
    {
        UserDetails userDetails = this.userDetailsService.loadUserByUsername(username);
        var isValidToken = jwtRepo.findByToken(token).map(t -> !t.isExpired() &&
!t.isRevoked()).orElse(false);

        if (jwtTokenUtil.isValidToken(token, userDetails) && isValidToken) {
            UsernamePasswordAuthenticationToken authToken = new
UsernamePasswordAuthenticationToken(
                userDetails, null, userDetails.getAuthorities());
            authToken.setDetails(new WebAuthenticationDetailsSource().buildDetails(request));
            SecurityContextHolder.getContext().setAuthentication(authToken);
        }
    }
    filterChain.doFilter(request, response);
}
}

```

@Service

```

public class JwtToken {
    private String EKey =
"EbeEsh7VhXpHMAkLz7Xb3TYm7a4KLMlYn0Kr1NJEhTIOeU9HJsv3t2bMa5OjoiaD";
    private int Duration = 60 * 60 * 24 * 7; // 7 days

    public String extractUsername(String token) {
        return extraClaim(token, Claims::getSubject);
    }
}

```

```
}
```

```
private Date extractExpiration(String token) {
    return extraClaim(token, Claims::getExpiration);
}
```

```
private <T> T extraClaim(String token, Function<Claims, T> claimsResolver) {
    final Claims claims = extractAllClaims(token);
    return claimsResolver.apply(claims);
}
```

```
private Claims extractAllClaims(String token) {
    return
    Jwts.parserBuilder().setSigningKey(getSigningkey()).build().parseClaimsJws(token).getBody();
}
```

```
private Key getSigningkey() {
    byte keyBytes[] = Decoders.BASE64.decode(EKey);
    return Keys.hmacShaKeyFor(keyBytes);
}
```

```
public String generateToken(Map<String, Object> extraClaims, UserDetails userDetails) {
    return buildToken(extraClaims, userDetails, Duration);
}
```

```
private String buildToken(Map<String, Object> extraClaims, UserDetails userDetails, long
expiration) {
    return Jwts.builder()
        .setClaims(extraClaims)
        .setSubject(userDetails.getUsername())
        .setIssuedAt(new Date(System.currentTimeMillis()))
        .setExpiration(new Date(System.currentTimeMillis() + expiration))
```

```

        .signWith(getSigningkey(), SignatureAlgorithm.HS256)
        .compact();
    }

    public boolean isValidToken(String token, UserDetails userDetails) {
        final String username = extractUsername(token);
        return (username.equals(userDetails.getUsername())) && !isTokenExpired(token);
    }

    private boolean isTokenExpired(String token) {
        return extractExpiration(token).before(new Date());
    }
}

@Service
@RequiredArgsConstructor
public class LogoutUtil implements LogoutHandler {

    private final JwtRepo jwtRepo;

    @Override
    public void logout(HttpServletRequest request, HttpServletResponse response, Authentication
authentication) {
        final String authHeader = request.getHeader("Authorization");
        final String token;
        if (authHeader == null || !authHeader.startsWith("Bearer ")) {
            return;
        }
        token = authHeader.substring(7);
        var storedToken = jwtRepo.findByToken(token).orElse(null);
        if (storedToken != null) {
            storedToken.setExpired(true);

```

```

        storedToken.setRevoked(true);
        jwtRepo.save(storedToken);
        SecurityContextHolder.clearContext();
    }
}
}

```

Home page:

```
public class Users implements UserDetails {
```

```

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long uid;
    private String name;
    private String email;
    private String password;

```

```

    @Enumerated(EnumType.STRING)
    private Role role;

```

```

    public enum Role {
        USER, ADMIN
    }

```

```

    @OneToMany(mappedBy = "user")
    private List<Token> tokens;

```

```

    @Override
    public Collection<? extends GrantedAuthority> getAuthorities() {
        return List.of(new SimpleGrantedAuthority(role.name()));
    }

```



```
@Override
public String getUsername() {
    // NOTE : return username, if you are using username for login instead of email
    return email;
}
```

```
@Override
public boolean isAccountNonExpired() {
    return true;
}
```

```
@Override
public boolean isAccountNonLocked() {
    return true;
}
```

```
@Override
public boolean isCredentialsNonExpired() {
    return true;
}
```

```
@Override
public boolean isEnabled() {
    return true;
}
}
```

```
@Builder
@NoArgsConstructor
@AllArgsConstructor
@Entity
```

```
public class Token {  
    @Id  
    @GeneratedValue(strategy = GenerationType.UUID)  
    private String id;  
  
    @Column(unique = true)  
    private String token;  
  
    private boolean revoked;  
    private boolean expired;  
  
    @ManyToOne(fetch = FetchType.LAZY, cascade = CascadeType.ALL)  
    @JoinColumn(name = "user_uid")  
    private Users user;  
  
}
```

CHAPTER 6

CONCLUSION

This chapter tells about the conclusion that anyone can drive from the project and the learning we learnt by taking over this project.

6.1 CONCLUSION

In conclusion, the proposed Interactive Storybook and Educational App is designed to enrich the learning experience and streamline educational content delivery for users including students, parents, educators, and administrators. This app is scalable, accommodating the educational needs of various user groups, from young learners to educators in diverse settings. It simplifies content management, enhances engagement with interactive features, and ensures secure handling of user data. With real-time updates and adaptability to different learning styles, the app empowers users to explore educational materials, track progress, and receive tailored recommendations. By fostering interactive learning and providing comprehensive educational tools, this app improves user satisfaction and supports effective learning and teaching practices.

6.2 FUTURE SCOPE

Integration of AI and Machine Learning:

Implementing AI algorithms to enhance the app's capabilities, such as personalizing story recommendations, adapting educational content based on user interactions, and predicting learning needs based on user performance and preferences.

Enhanced Security Measures:

Continual improving security protocols to protect user data, ensure privacy, and prevent unauthorized access or breaches, thereby maintaining a safe learning environment for all users.

Mobile Application Development:

Developing a mobile app version of the platform to offer greater accessibility and convenience, allowing users to engage with interactive storybooks and educational content anytime, anywhere.

Analytics and Reporting:

Creating robust analytics tools to provide insights into user engagement, learning progress, and content effectiveness. This data can help educators and parents make informed decisions and improve the educational experience.

Integration with Learning Management Systems (LMS):

Integrating with existing LMS platforms used by educational institutions to synchronize user progress, content, and achievements, creating a cohesive and unified learning experience.

Adaptive Exam Models:

Implementing adaptive learning features that adjust the difficulty of educational content based on user performance, providing a tailored and effective learning journey for each child.

Collaborations with Exam Boards:

Partnering with educational publishers and content creators to expand the range of interactive stories and educational modules available on the platform, offering diverse and high-quality resources.

Feedback Mechanisms:

Incorporating feedback tools for users to share their experiences, suggestions, and concerns, facilitating continuous improvement and ensuring the app meets the needs.