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

A PROJECT REPORT

Submitted by

ANDREY FLINTOFF G	**-**	**727822TUCS010**
DEVADHARSHINI S	**-**	**727822TUCS029**
DHARSHINI N	**-**	**727822TUCS032**
KANISHKA M	**-**	**727822TUCS054**

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 **INTERACTIVE CHILDREN'S STORYBOOK AND EDUCATION** is the bonafide work of **DHARSHINI N 727822TUCS032, DEVADHARSHINI S 727822TUCS029, KANISHKA M 727822TUCS054, ANDREY FLINTOFF G** who carried out the project work under my supervision.

SIGNATURE	*SIGNATURE*
DR. R. GNANAKUMARI	**Dr. M. UDHAYAMOORTHI**
SUPERVISOR	**PROGRAM COORDINATOR**
Assistant Professor,	Associate Professor,
Department of Computer Science and Engineering,	Department of Computer Science and Engineering,
Sri Krishna College of Technology,Coimbatore-641042	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 his tireless and relentless support.

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

We thank **Dr. R. Gnanakumari**, 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 Storybooks and Educational Materials application is designed to create an engaging and enriching digital learning experience for young readers. This platform offers a wide array of interactive storybooks and educational resources, incorporating multimedia elements such as text, images, sound, and animations to captivate and stimulate children's imaginations. The content is organized into intuitive categories, making it easy for parents, educators, and children to find appropriate materials tailored to different age groups and learning needs. Detailed descriptions, learning objectives, and user feedback ensure that users can make informed choices about the educational value of each resource. Future enhancements include the integration of augmented reality to bring stories to life, advanced personalization to cater to individual learning paths, and the development of multilingual content to reach a global audience. By leveraging these advanced features, the platform aims to provide a seamless and enjoyable learning journey that adapts to the evolving needs of young learners. The platform is committed to promoting literacy and cognitive development through interactive and immersive learning experiences. By continually updating and expanding its content library, the Interactive Children’s Storybooks and Educational Materials application aspires to be a leading digital destination for early childhood education. The strategic vision includes the introduction of eco-friendly digital practices and partnerships with educational institutions to enhance the overall learning ecosystem.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF ABBREVIATIONS

ABBREVIATION	ACRONYM
HTML	HYPERTEXT MARKUP LANGUAGE
CSS	CASCADING STYLESHEET
JS	JAVASCRIPT
SDLC	SOFTWARE DEVELOPMENT LIFE CYCLE
JWT	JSON WEB TOKENS
SQL	STRUCTURED QUERY LANGUAGE
REST	REPRESENTATIONAL STATE TRANSFER
API	APPLICATION PROGRAMMING INTERFACE
JSX	JAVASCRIPT XML
XML	EXTENSIBLE MARKUP LANGUAGE
VS CODE	VISUAL STUDIO CODE
OS	OPERATING SYSTEM
HTTP	HYPERTEXT TRASFER PROTOCOL
URL	UNIFORM RESOURCE LOCATOR
CRUD	CREATE, READ, UPDATE, DELETE
DTO	DATA TRANSFER OBJECTS
JDBC	JAVA DATABASE CONNECTIVITY

CHAPTER 1

INTRODUCTION

This project aims to deliver an engaging educational experience for children through an interactive storybook platform. It combines dynamic storytelling with educational quizzes and embedded YouTube videos. Built with a React frontend for a responsive user experience, a Spring Boot backend for efficient content management, and MySQL for robust data handling.

1.1 PROBLEM STATEMENT

How can we create an interactive children's storybook platform that integrates dynamic storytelling with educational quizzes and multimedia elements, providing an engaging and educational experience while ensuring user-friendly navigation and content management?

1.2 OBJECTIVE

The primary objective of this project is to create an interactive children's storybook and educational platform that provides an engaging and educational experience. By integrating interactive stories with features like read-aloud options, quizzes the platform aims to enhance learning and make storytelling captivating for children.

1.3 OVERVIEW

In the realm of children's education and interactive storytelling, users often encounter issues such as fragmented content, lack of engaging educational activities, and limited interactive features. This platform will combine captivating stories with educational quizzes, multimedia elements, and read-aloud functionalities, ensuring an engaging, educational, and immersive experience for children while providing intuitive navigation and rich content management

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 supportto 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 storesdata in the form of a key and corresponding value. It is similar to a database table in that itstores 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’scomputer. It is a way of persistently storing data on the client side, which is not sent to theserver 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 abrowser’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 accessedby the website even if the user closes the browser or navigates to another page. It is a greatway 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 eachrequest.

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 isstored 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 storageis also secure, as the data is stored on the user’s machine and not on a server. This meansthat 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

The interactive children's storybook and educational platform is designed to provide an engaging and educational experience for young learners. This system offers a dynamic frontend built with React, enabling interactive storytelling with features like read-aloud options, quizzes, and multimedia content. Children can explore a diverse collection of stories, listen to narrated versions, and participate in interactive quizzes to reinforce their learning.

The platform incorporates an easy-to-use shopping cart functionality, allowing users to purchase books directly. With this feature, users can browse through available books, add their selections to the cart, and complete the purchase through a secure online transaction process. This enhances the user experience by simplifying the process of acquiring educational materials.

From an administrative perspective, the system streamlines content management and user interactions. It automates content updates and ensures efficient management of the platform's features. By leveraging technology to support interactive learning and providing a seamless purchasing experience, the system significantly enhances educational engagement and accessibility for children.

Moreover, the system enhances operational efficiency by automating content management, user interactions, and purchase processes. This ensures that updates and transactions are handled swiftly and accurately, improving the overall user experience. By leveraging advanced technology, the platform delivers a seamless, user-friendly interface while optimizing administrative tasks, resulting in a more effective and enjoyable educational experience for children

3.2 ADVANTAGES

- **3.2.1 ENGAGEMENT:**
- The system offers interactive storybooks and educational games that capture children's attention, making learning both enjoyable and effective. The engaging content fosters a love for reading and enhances knowledge retention by creating a fun learning environment
- **3.2.2 ACCESSIBILITY:**
- With 24/7 access to a diverse range of educational materials and multimedia content, children and parents can engage with the platform from anywhere with internet access. This flexibility accommodates various learning schedules and eliminates geographical constraints, ensuring that quality educational resources are always available.
- **3.2.3 READ-ALOUD FEATURE:**
- Stories come with a read-aloud option, allowing children to listen to narrations while following along with the text. This feature supports language development, improves comprehension, and helps young readers build their reading skills at their own pace.
- **3.2.4 INTERACTIVE QUIZZES:**
- The inclusion of interactive quizzes adds an element of fun and assessment to the learning process. Quizzes reinforce the material covered in the storybooks and educational games, providing immediate feedback and helping children gauge their understanding of the content

3.3 TECHNOLOGIES USED

FRONTEND: REACT WITH MATERIAL-UI:

React.js:

React is a JavaScript library used for building user interfaces, especially single-page applications where performance and responsiveness are critical. It allows developers to create reusable components, manage application state efficiently, and use a virtual DOM to optimize rendering. In your project, React is used to build the interactive and dynamic frontend of the application.

CSS:

Applied for styling and designing the application, ensuring a visually appealing and consistent look across various devices. These subtle effects contribute to a more intuitive and engaging user interface, helping to guide users through their tasks and improving the overall user experience.

JPA (Java Persistence API):

Facilitates efficient management of database operations and interactions, mapping Java objects to database tables. Furthermore, JPA supports various querying techniques, enabling developers to retrieve and manipulate data in a flexible and powerful way

Material-UI:

Material-UI is a popular React component library that implements Google's Material Design guidelines. It provides a set of pre-built components like buttons, sliders, and dialogs, which are fully customizable and optimized for various devices. This helps in creating a consistent and visually appealing user interface with less effort. Material-UI is integrated into your React application to enhance the design and user experience.

BACKEND: SPRING BOOT WITH JWT SECURITY:

Spring Boot:

Spring Boot is an open-source, Java-based framework used to create stand-alone, production-grade Spring-based applications with minimal configuration. It simplifies the process of developing and deploying Java applications by providing a range of pre-configured options. In your project, Spring Boot serves as the backend framework, handling business logic, processing requests, and communicating with the database.

JWT Security:

JSON Web Token (JWT) is a compact, URL-safe means of representing claims to be transferred between two parties. JWTs are commonly used for authentication and securing APIs. In your project, JWT Security is used to implement a stateless authentication mechanism. When a user logs in successfully, the backend generates a JWT, which the frontend stores and sends with each subsequent request to access protected resources, ensuring that only authenticated users can perform certain actions.

DATABASE: MYSQL:

MySQL:

MySQL is a widely-used open-source relational database management system (RDBMS). It stores data in tables, which can be queried using SQL (Structured Query Language). MySQL is known for its reliability, scalability, and performance. In your project, MySQL is used as the database to store and manage all the persistent data, such as user information, product details, and transaction records. The backend (Spring Boot) interacts with the MySQL database to perform CRUD (Create, Read, Update, Delete) operations.

INTEGRATION:

React and Spring Boot:

The frontend (React) communicates with the backend (Spring Boot) through RESTful APIs. The backend processes requests from the frontend, performs the necessary operations (e.g., querying the database), and returns the response, usually in JSON format, to be rendered by the React application.

JWT Authentication Flow:

When a user logs in, the frontend sends the credentials to the backend. If the credentials are valid, the backend generates a JWT and returns it to the frontend. This token is then included in the Authorization header of subsequent requests to secure endpoints. The backend verifies the token's validity before processing the request, ensuring that the user is authenticated.

CHAPTER 4
METHODOLOGIES

This chapter gives a small description about how our system works.

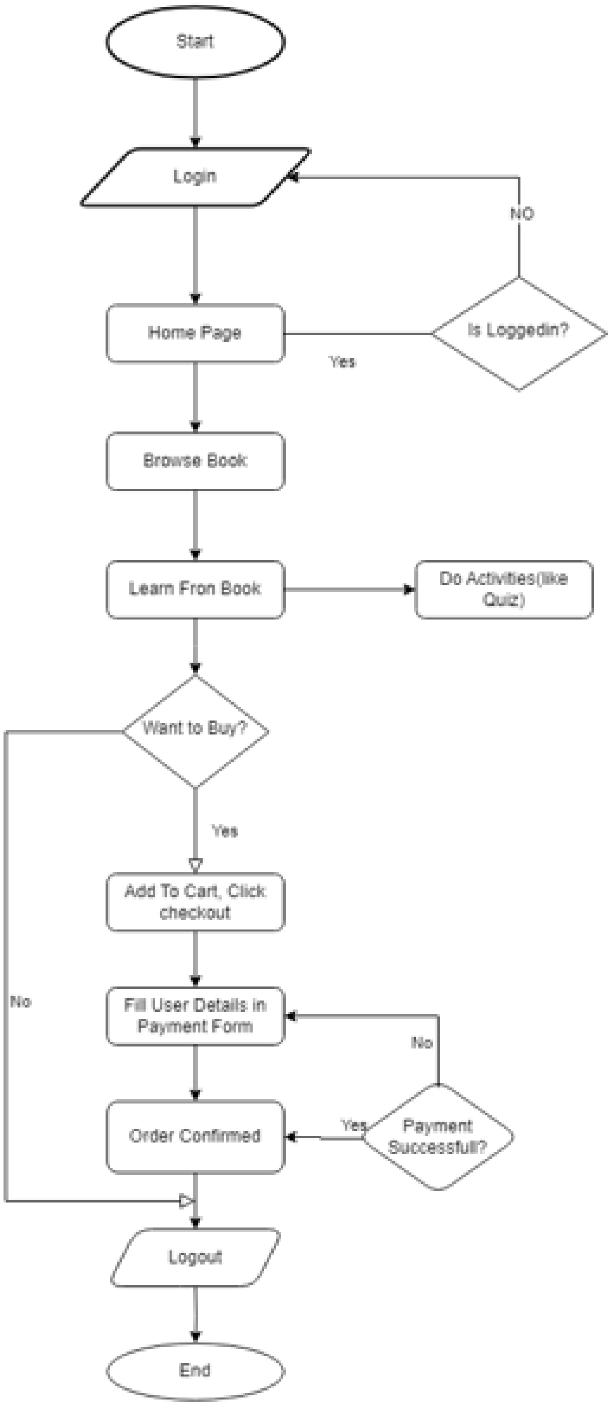


Fig 4.1.Process flow diagram

4.1 STORYBOOK BACKEND SCHEMA

BOOK ENTITY:

- **Purpose:** Manages details of the books, including title, author, genre, and content.
- **Usage:** Stores information about each book available in the storybook application. It serves as the core content that users interact with.

FEEDBACK ENTITY:

- **Purpose:** Manages user feedback on books, quizzes, and overall experience with the storybook application.
- **Usage:** Collects and stores user feedback to enhance content quality, improve features, and increase user satisfaction.

ORDER DETAILS ENTITY:

- **Purpose:** Manages information related to user purchases, including book orders and quiz access.
- **Usage:** Tracks purchase history, order status, and payment details, allowing users to review and manage their transactions.

PURCHASE ENTITY:

- **Purpose:** Handles the purchase process, including payment and transaction confirmation.
- **Usage:** Facilitates the purchase of books or access to quizzes and other premium content, ensuring secure and successful transactions.

QUIZ ENTITY:

- **Purpose:** Manages quizzes associated with books, including questions, answers, and scoring.
- **Usage:** Provides users with quizzes to test their comprehension and learning, generating results and insights based on their performance.

BLANKS ENTITY:

- **Purpose:** Manages fill-in-the-blank activities within the books, designed to enhance user engagement and learning.
- **Usage:** Stores and processes blank activities where users can fill in missing words or phrases, promoting active reading and comprehension.

TOKENS ENTITY:

- **Purpose:** Manages user tokens, which can be earned or purchased to access premium content, books, or quizzes.
- **Usage:** Tracks token balance and transaction history, allowing users to redeem tokens for various in-app purchases.

4.2 STORYBOOK BACKEND PROCESS

INPUT DATA:

- Users log in to the application with their credentials to access the storybook, quizzes, and other features.
- Users browse and select books, take quizzes, or participate in interactive activities like fill-in-the-blank exercises.

CONTENT MANAGEMENT:

- Book Management: Users can browse, read, and interact with different books available in the library.
- Quiz Management: The system presents quizzes related to the books, and users can answer questions to test their comprehension.
- Blanks Management: Users can engage in fill-in-the-blank activities, with the system checking the correctness of the filled words.

PURCHASE HANDLING:

- Order Details Management: The system manages user orders, tracking purchase history, order status, and associated details.
- Purchase Process: Users can purchase books or quiz access using tokens or other payment methods, with the system ensuring secure transactions.

FEEDBACK COLLECTION:

- Users can provide feedback on books, quizzes, and overall experience, which is stored in the feedback entity.
- Feedback is analyzed to improve content, user experience, and application features.

SESSION MANAGEMENT:

- Sessions are organized to track user progress, including books read, quizzes taken, and fill-in-the-blank activities completed.
- Users can revisit past sessions to track their progress and review results.

OUTPUT:

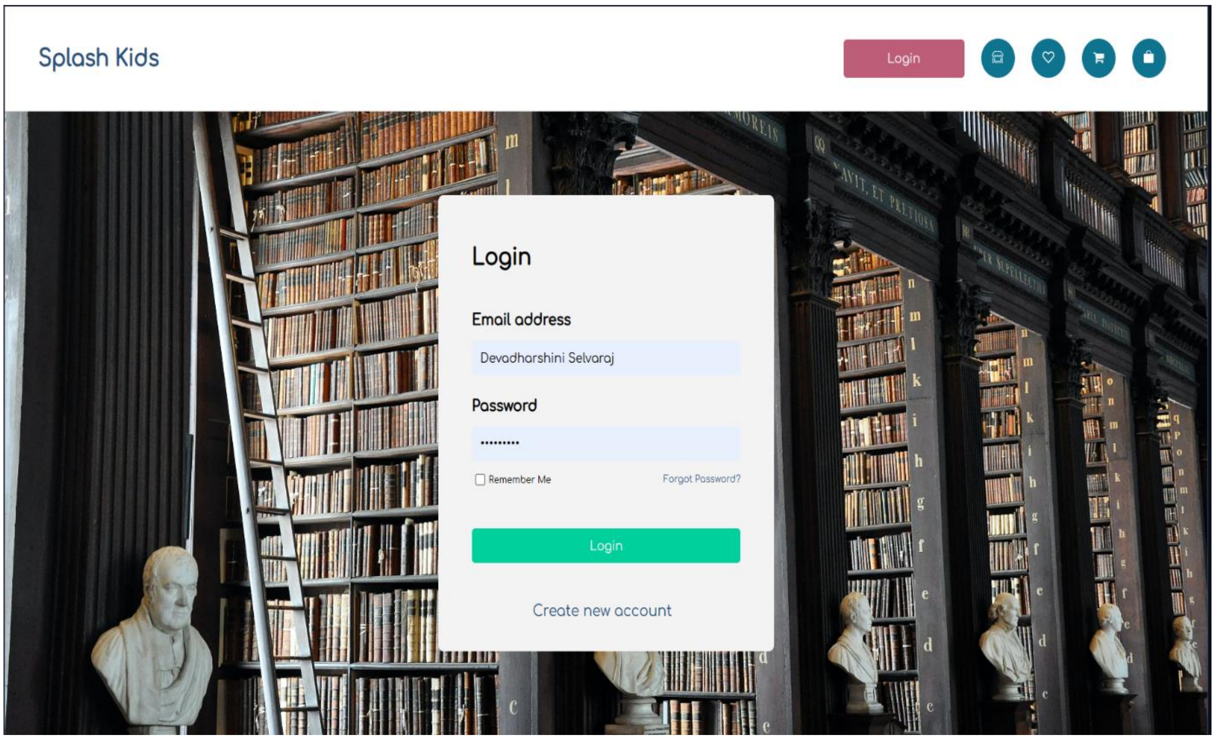
- The final output includes detailed reports on user activity, including quiz scores, books read, and engagement in blank activities.
- Users can generate comprehensive reports on their reading patterns, quiz performance, and overall progress over time.
- The system provides recommendations for further reading, additional quizzes, and personalized learning paths based on user performance.
- Real-time feedback during quizzes and interactive activities allows users to make immediate corrections and improve their understanding.
- Visual dashboards display key metrics such as reading completion, quiz scores, and token usage, providing a clear view of progress and achievements.
- Personalized learning paths guide users through content tailored to their interests and areas for improvement.

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 the user visits our website, they will be directed to a login page where they will be required to enter their login credentials, including their email address and password. Upon submission, the system will authenticate the provided credentials by cross-referencing them with the information stored in our database, which was provided by the user during the account registration process. If the entered details match the stored data, the user will be granted access to their account and the website's features. If the details do not match, the user will receive an error message prompting them to re-enter their credentials or use the password recovery options available. This process ensures that only registered users can access the site, maintaining the security and integrity of the user accounts.



Devadharshini Selvaraj

Fig 5.1 LOGIN PAGE

5.2 USER REGISTER

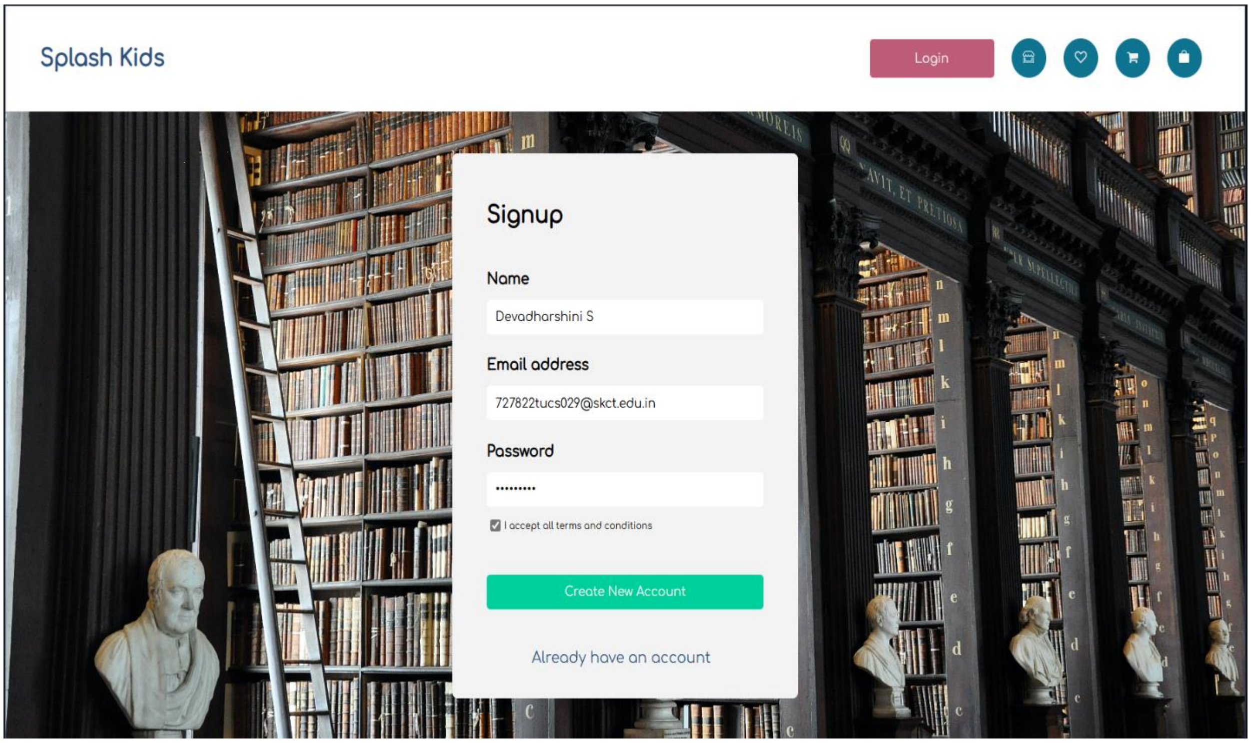
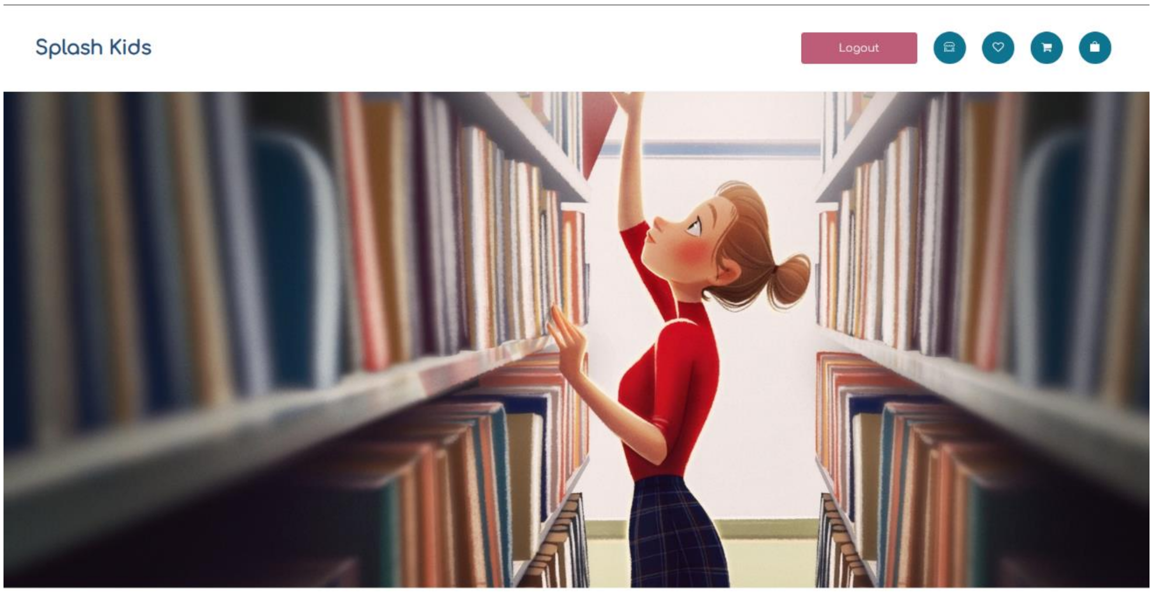


Fig 5.2 USER REGISTER

5.3 HOME PAGE



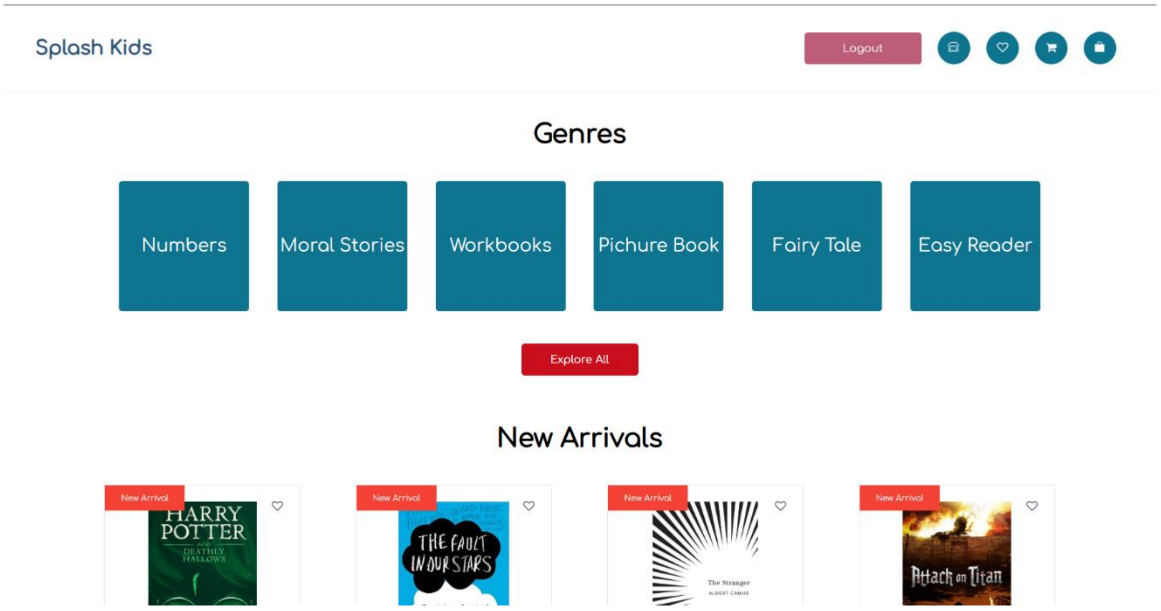


Fig 5.3 HOME PAGE

5.4 STORE PAGE

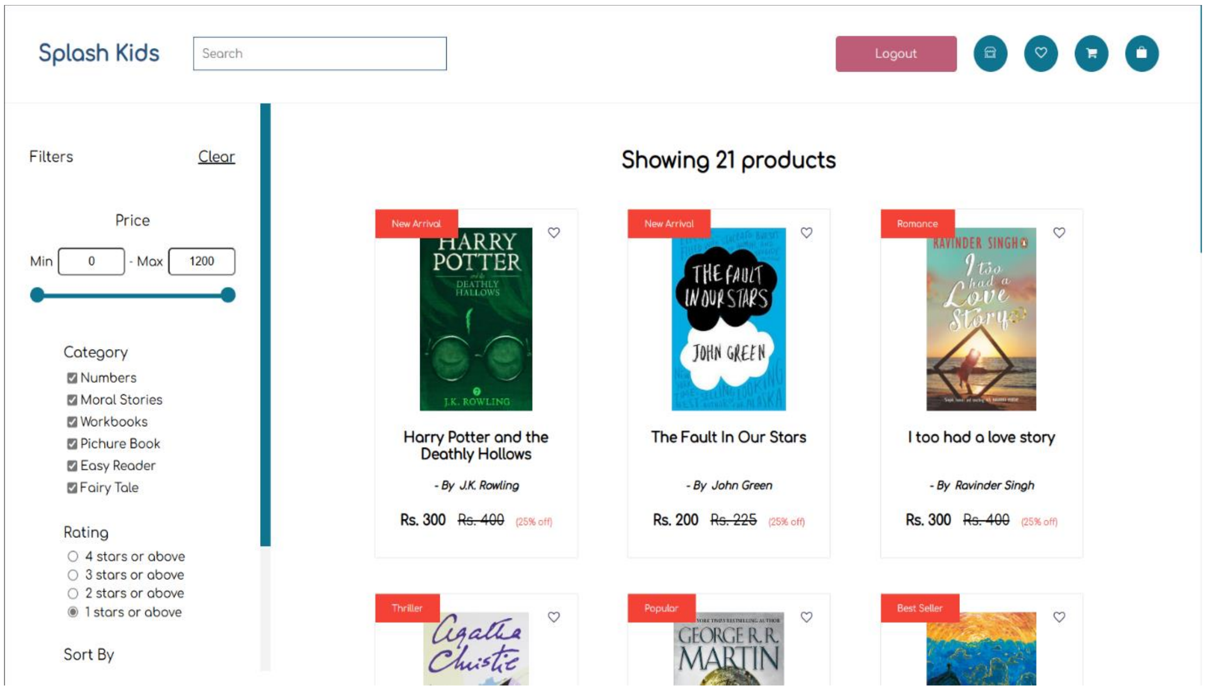


Fig 5.4 STORE PAGE

5.5 STORY DETAILS PAGE

Manage Your Content:

View and track all your interactions with storybooks, educational games, and activities. Monitor your child’s progress, check completion status, and review detailed reports for each learning module.

Interactive Engagement:

Access a wide range of interactive storybooks and educational games that capture children's attention. The engaging content makes learning enjoyable and effective, fostering a love for reading and enhancing knowledge retention by creating a fun and immersive learning environment.

Accessibility:

Enjoy 24/7 access to a diverse collection of educational materials and multimedia content. Children and parents can engage with the platform from anywhere with an internet connection, accommodating various learning schedules and eliminating geographical constraints, ensuring that quality educational resources are always at your fingertips.

Read-Aloud Feature:

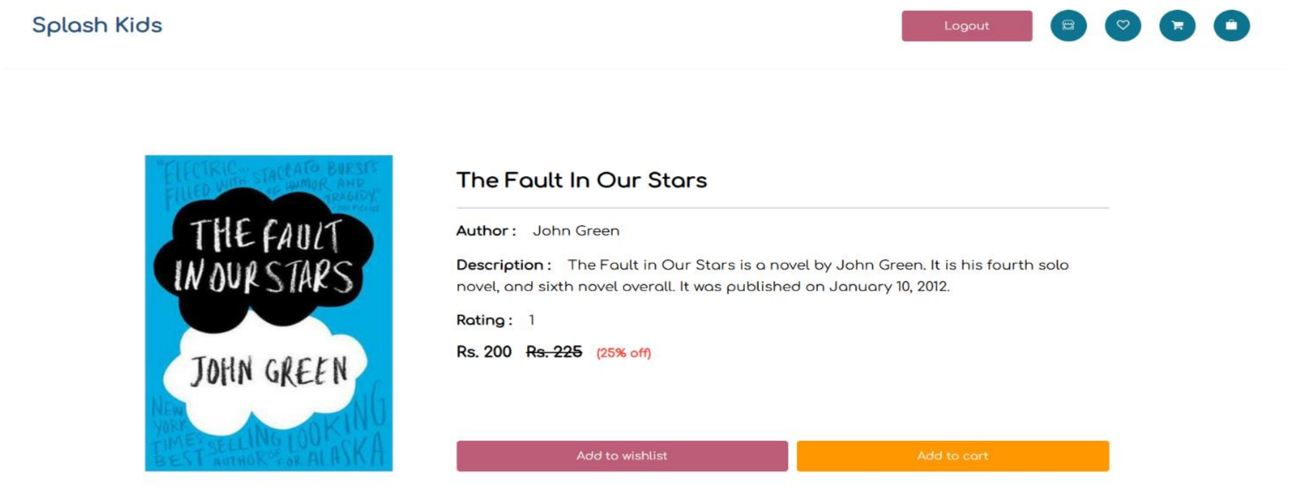
Utilize the read-aloud option in storybooks, allowing children to listen to narrations while following along with the text. This feature supports language development, improves comprehension, and helps young readers build their reading skills at their own pace.

Interactive Quizzes:

Enhance learning with interactive quizzes that add an element of fun and assessment. These quizzes reinforce the material covered in the storybooks and educational games, providing immediate feedback and helping children gauge their understanding of the content.

Interactive Feedback:

Receive actionable insights and recommendations based on your child's performance. Use these insights to guide their learning journey and help them excel in their tasks.


Splash Kids

The Fault In Our Stars

Author : John Green

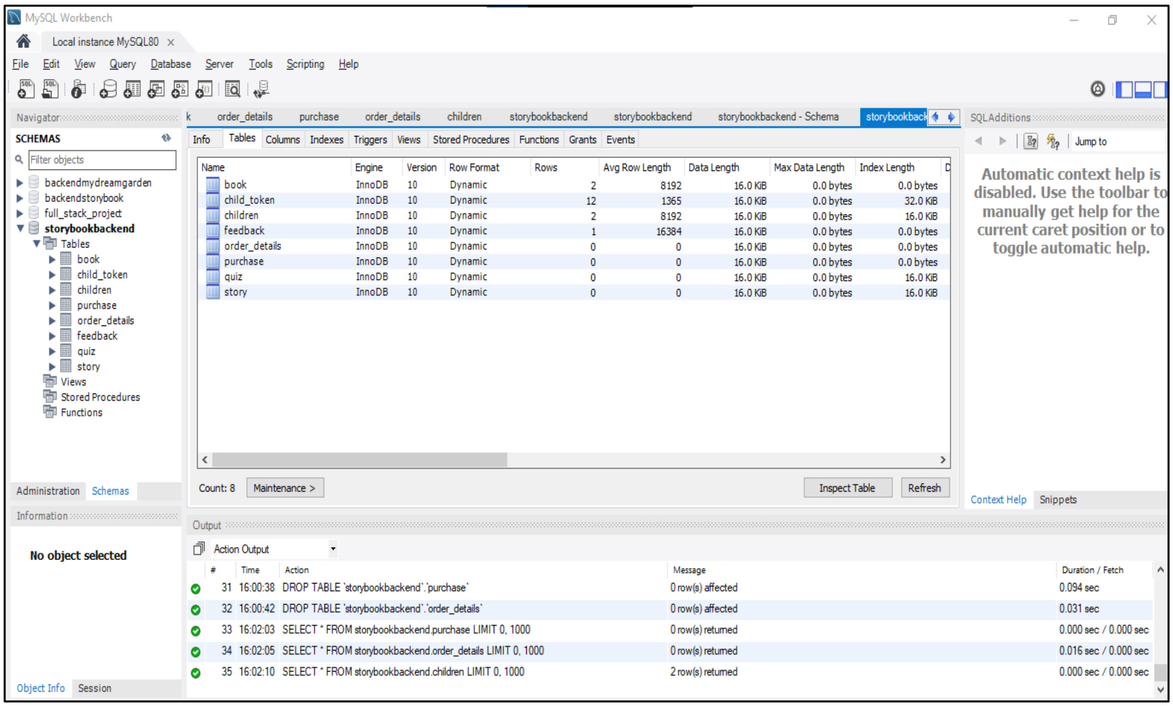
Description : The Fault in Our Stars is a novel by John Green. It is his fourth solo novel, and sixth novel overall. It was published on January 10, 2012.

Rating : 1

Rs. 200 ~~Rs. 225~~ (25% off)

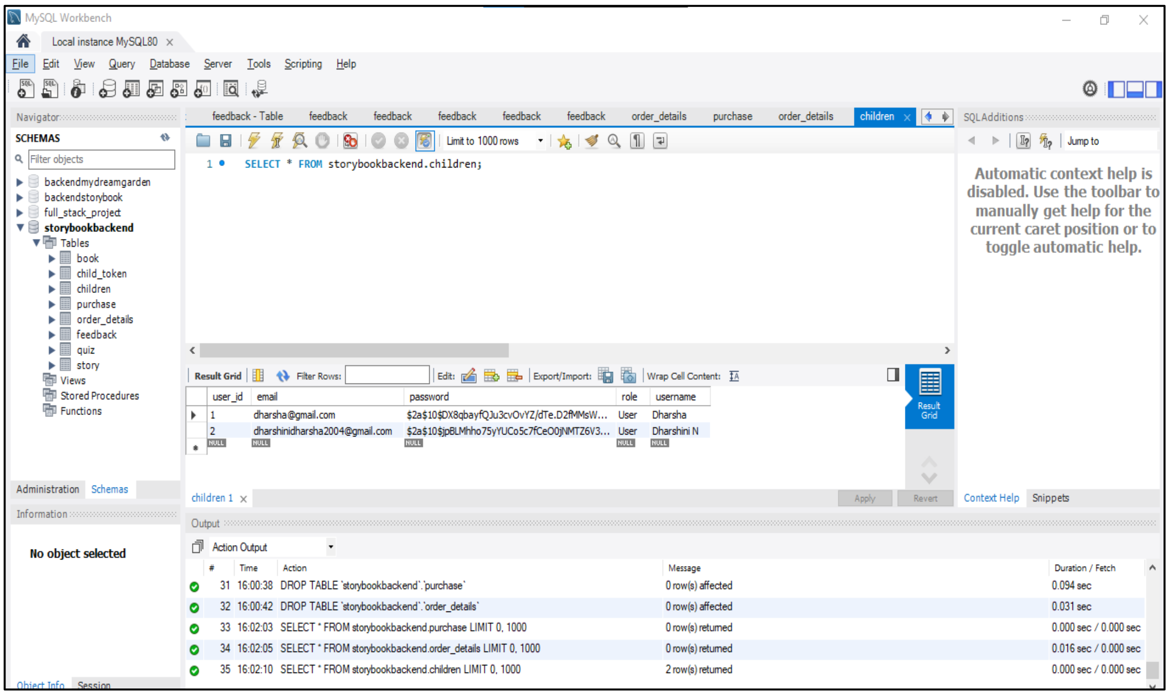
Fig 5.5 STORY DETAILS PAGE

5.6 DATABASE



Name	Engine	Version	Row Format	Rows	Avg Row Length	Data Length	Max Data Length	Index Length
book	InnoDB	10	Dynamic	2	8192	16.0 KiB	0.0 bytes	0.0 bytes
child_token	InnoDB	10	Dynamic	12	1365	16.0 KiB	0.0 bytes	32.0 KiB
children	InnoDB	10	Dynamic	2	8192	16.0 KiB	0.0 bytes	16.0 KiB
feedback	InnoDB	10	Dynamic	1	16384	16.0 KiB	0.0 bytes	0.0 bytes
order_details	InnoDB	10	Dynamic	0	0	16.0 KiB	0.0 bytes	0.0 bytes
purchase	InnoDB	10	Dynamic	0	0	16.0 KiB	0.0 bytes	0.0 bytes
quiz	InnoDB	10	Dynamic	0	0	16.0 KiB	0.0 bytes	16.0 KiB
story	InnoDB	10	Dynamic	0	0	16.0 KiB	0.0 bytes	16.0 KiB

```
#   Time      Action                                                    Message             Duration / Fetch
31  16:00:38  DROP TABLE 'storybookbackend'.'purchase'                  0 row(s) affected   0.094 sec
32  16:00:42  DROP TABLE 'storybookbackend'.'order_details'             0 row(s) affected   0.031 sec
33  16:02:03  SELECT * FROM storybookbackend.purchase LIMIT 0, 1000     0 row(s) returned   0.000 sec / 0.000 sec
34  16:02:05  SELECT * FROM storybookbackend.order_details LIMIT 0, 1000 0 row(s) returned  0.016 sec / 0.000 sec
35  16:02:10  SELECT * FROM storybookbackend.children LIMIT 0, 1000     2 row(s) returned   0.000 sec / 0.000 sec
```



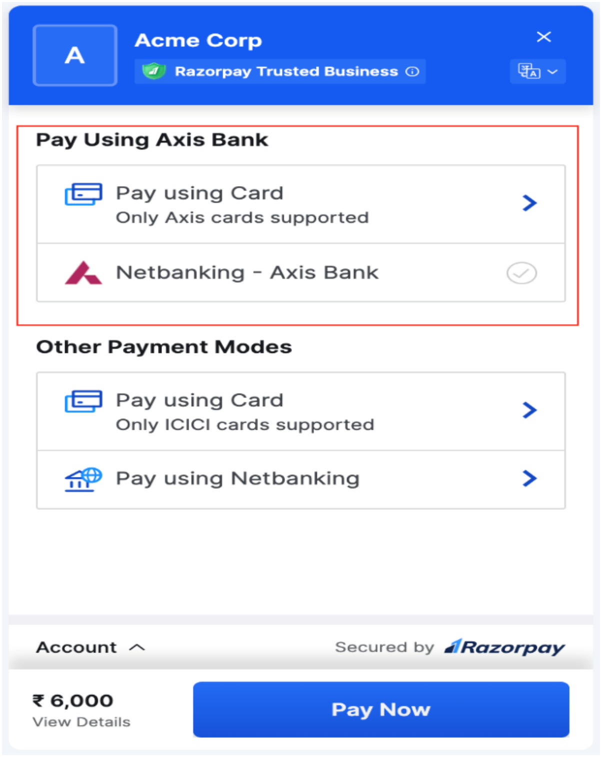
```
SELECT * FROM storybookbackend.children;
```

user_id	email	password	role	username
1	dharsha@gmail.com	$2a$10$DX8qbayfQJu3cvOvYZ/dTe.D2fMMsW...	User	Dharsha
2	dharshinidharsha2004@gmail.com	$2a$10$jpBLMhho75yYUCo5c7fCeO0jNMTZ6V3...	User	Dharshini N
NULL	NULL	NULL	NULL	NULL

```
#   Time      Action                                                   Message              Duration / Fetch
31  16:00:38  DROP TABLE `storybookbackend`.`purchase`                 0 row(s) affected    0.094 sec
32  16:00:42  DROP TABLE `storybookbackend`.`order_details`            0 row(s) affected    0.031 sec
33  16:02:03  SELECT * FROM storybookbackend.purchase LIMIT 0, 1000    0 row(s) returned    0.000 sec / 0.000 sec
34  16:02:05  SELECT * FROM storybookbackend.order_details LIMIT 0, 1000  0 row(s) returned  0.016 sec / 0.000 sec
35  16:02:10  SELECT * FROM storybookbackend.children LIMIT 0, 1000    2 row(s) returned    0.000 sec / 0.000 sec
```

Fig 5.6 DATABASE

5.7 PAYMENT PAGE



Acme Corp

Razorpay Trusted Business

Pay Using Axis Bank

Pay using Card
Only Axis cards supported

Netbanking - Axis Bank

Other Payment Modes

Pay using Card
Only ICICI cards supported

Pay using Netbanking

Account

Secured by Razorpay

₹ 6,000
View Details

Pay Now

Fig 5.7 PAYMENT PAGE

5.8 CODING

5.8.1 LOGIN PAGE :

```
import React, { useRef, useState } from 'react';
import { Box, TextField, InputAdornment, Button, Avatar, Typography } from
'@mui/material';
import AccountCircle from '@mui/icons-material/AccountCircle';
import PasswordIcon from '@mui/icons-material/Password';
import LockRoundedIcon from '@mui/icons-material/LockRounded';
import { useNavigate } from 'react-router-dom';
import { useDispatch } from 'react-redux';
import axios from 'axios';
import { login } from '../feature/user/UserSlice';

const linearGradient = 'linear-gradient(to right, #f569a3, #f79ed1)';
export default function Login() {
  // Data for backend
  const [formData, setFormData] = useState({
    email: '',
    password: '',
  });
  const emailRef = useRef(null);
  const passwordRef = useRef(null);
  const [error, setError] = useState({ email: false, password: false });
  const dispatch = useDispatch();
  const navigate = useNavigate();
  const handleChange = (event) => {
    const { name, value } = event.target;
    setFormData({ ...formData, [name]: value });
  };
  const handleSubmit = async (event) => {
    event.preventDefault();
```

```
const email = formData.email;
const password = formData.password;
try {
  const response = await axios.post(
    'http://localhost:8080/api/auth/login',
    formData
  );
  console.log(response.data);
  const { accessToken, role,userId } = response.data;
  localStorage.setItem('token', accessToken);
  localStorage.setItem('role', role);
  localStorage.setItem('userId', userId);
  // localStorage.setItem('username', username);

  console.log('Token:', localStorage.getItem('token'));
  // alert('Login Success.!');
  if (role === 'ADMIN') {
    navigate('/admin');
  } else {
    navigate('/home');
  }
} catch (error) {
  console.error(error);
  alert('Invalid Credentials.!');
}

if (email && password) {
  setError({ email: false, password: false });
  // dispatch(login(emailRef.current.value));
} else {
  if (!email) setError((prev) => ({ ...prev, email: true }));
```

```
    if (!password) setError((prev) => ({ ...prev, password: true }));
  }
};

return (
  <form onSubmit={handleSubmit}>
    <Box
      display="flex"
      flexDirection="column"
      maxWidth={600}
      minWidth={400}
      alignItems="center"
      justifyContent="center"
      margin="auto"
      marginTop={5}
      padding={3}
      borderRadius={5}
      boxShadow="5px 5px 10px #ccc"
      sx={{
        ':hover': {
          boxShadow: '10px 10px 20px #ccc',
        },
        backgroundImage: linearGradient,
        alignItems: 'center',
        color: 'black',
      }}
    >
      <Avatar id="avatar">
        <LockRoundedIcon />
      </Avatar>
      <Typography variant="h4" padding={3} textAlign="center">
  Login
</Typography>
<Box>
  <TextField
    id="input-with-icon-textfield"
    label="Email"
    name="email"
    inputRef={emailRef}
    value={formData.email}
    onChange={handleChange}
    variant="outlined"
    margin="normal"
    InputProps={{
      startAdornment: (
        <InputAdornment position="start">
          <AccountCircle />
        </InputAdornment>
      ),
    }}
    error={error.email}
    helperText={error.email ? 'Fill the valid email' : ''}
  />
</Box>
<Box>
  <TextField
    id="password-with-icon-textfield"
    label="Password"
    type="password"
    name="password"
    inputRef={passwordRef}
    value={formData.password}
```

```
          onChange={handleChange}
          variant="outlined"
          margin="normal"
          fullWidth
          InputProps={{
            startAdornment: (
              <InputAdornment position="start">
                <PasswordIcon />
              </InputAdornment>
            ),
          }}
          error={error.password}
          helperText={error.password ? 'Fill the password' : ''}
        />
      </Box>
      <Button
        sx={{ marginTop: 3, borderRadius: 3, backgroundColor: '#d6067c', ':hover': {
backgroundColor: '#ed5192' } }}
        variant="contained"
        type="submit"
      >
        Submit
      </Button>
      <Typography variant="body1" component="span" style={{ marginTop: '10px'
}}>
        Not registered yet?{' '}
        <span style={{ color: '#d6067c', cursor: 'pointer' }} onClick={() =>
navigate('/register')}>
          Register
        </span>
      </Typography>
      </Box>
    </form>
  );
}
```

5.8.2 HOME PAGE:

```
import React from 'react'
import homegif from "../assets/homegif.gif";
import { Box, Grid, Typography } from '@mui/material';
import 'bootstrap/dist/css/bootstrap.min.css';
import Carousalcomp from '../components/Carouselcomp';
import Footer from '../components/Footer';
import { useNavigate } from 'react-router-dom';

function Home() {
  const navigate=useNavigate();
  return (
    <>
    <div
    >
    <Typography variant='h3'
    marginTop="80px"
        sx={maintext}>
    Explore our curated collection of new and popular books to find your
    next literary adventure.
    </Typography>
    </div>
    <img src={homegif} alt="gifimage" ></img>
    <Typography
      variant="h3"
      sx={
        maintext
       }
     >
       Explore the Categories!
     </Typography>
     <Box
      sx={{ flexGrow: 1 }}>
       <Grid container spacing={5}
       sx={{ justifyContent: 'center',
          margin: "0 auto",
          width:"100%"
        }}
       >
         <Grid item xs={6} sm={3}>
           <img src="https://www.shutterstock.com/image-vector/reading-interesting-
book-concept-immersion-600nw-2296479725.jpg" alt="Picture Books"
           style={{ width: '100%',
             height: '200px',
             objectFit: 'cover',
             }}
             onClick={() => {
                 navigate("/picturebook");
               }}></img>
             <h4>Picture Books</h4>
         </Grid>
       </Grid>
     </Box>
     <Typography
       variant="h3"
       sx={
        maintext
      }
    >
      Book Showcase!
    </Typography>
    <Box >
      <Carousalcomp/>
    </Box>

    <Footer/>
    </>
  )
}

export default Home
```

5.8.3 LIST OF BOOKS:

```
// src/components/PictureBooks.jsx
import React, { useEffect, useState } from 'react';
import BookCard from '../../../components/BookCard';
import PictureItem from "./PictureItem";
import { Container, Grid, Typography } from '@mui/material';
import axios from 'axios';

const maintext={
    textAlign: "center",
    color: "#FF5722", // Bright color
    fontSize: 40,
    fontFamily: "'Comic Sans MS', cursive, sans-serif", // Playful font
    fontWeight: 700,
    padding: "20px 20px",
    background: "linear-gradient(90deg, rgba(255,215,0,1) 0%, rgba(255,0,150,1) 100%)"
```

```
}
function PictureBooks() {
  // const books = PictureItem;
  const [books, setBook] = useState([]);
 useEffect(() => {
  const fetchBook = async () => {
   try {
    const token = localStorage.getItem("token");
    if (!token) {
     console.error("Token not found in localStorage");
     return;
    }

    const response = await axios.get(
     "http://localhost:8080/api/books/get",
     {
      headers: {
       Authorization: `Bearer ${token}`,
      },
     }
    );
    setBook(response.data);
   } catch (error) {
    console.error(error);
   }
  };
  fetchBook();
 }, []);

  return (
    <Container>
```

```
            <Typography variant="h3" sx={{ textAlign: 'center', my: 4 }} style={maintext}>
                Explore Our Collection of Picture Books!
            </Typography>
            <Grid container spacing={4}>
                {books.map(book => (
                    <Grid item xs={12} sm={6} md={4} key={book.bookId}>
                        <BookCard book={book} />
                    </Grid>
                ))}
            </Grid>
        </Container> )}
export default PictureBooks;
```

CHAPTER 6

BACKEND SYSTEM SPECIFICATION

In this chapter, the content discusses the software employed for constructing the website. This chapter provides a brief description of the software utilized in the project.

6.1 SQL:

Tables in backend2
Book
Feedback
Order Details
Purchase
Quiz
Tokens

Fig 6.1 Tables in MySQL

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. 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 JavaScriptAPI. 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. 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.

6.2 REST API:

A REST API (Representational State Transfer Application Programming Interface) is a popular architectural style for designing networked applications. It is based on a set of principles and constraints that allow for scalability, simplicity, and interoperability between systems.

6.2.1 CLIENT-SERVER:

Separated entities communicate over HTTP or a similar protocol, with distinct responsibilities and the ability to evolve independently.

6.2.2 STATELESS:

Each request from the client to the server must contain all the necessary information to understand and process the request. The server does not maintain any client state between requests.

6.2.3 UNIFORM INTERFACE:

The API exposes a uniform interface, typically using HTTP methods (GET, POST, PUT, DELETE) to perform operations on resources. Resources are identified by URLs (Uniform Resource Locators).

6.2.4 CACHEABLE:

Responses can be cached by the client or intermediaries to improve performance and reduce the load on the server.

6.2.5 LAYERED SYSTEM:

Intermediary servers can be placed between the client and server to provide additional functionality, such as load balancing, caching, or security.

6.3 SPRINGBOOT:

Spring Boot is an open-source Java framework that simplifies the development of standalone, production-ready applications. It offers several advantages for building robust and scalable applications.

6.3.1 SIMPLIFIED CONFIGURATION:

Spring Boot eliminates the need for complex XML configuration files by leveraging sensible default configurations and annotations.

6.3.2 EMBEDDED SERVER:

Spring Boot includes an embedded server (e.g., Apache Tomcat, Jetty) that allows developers to create self-contained applications. This eliminates the need for external server installation and configuration, making it easier to package and deploy the application.

6.3.3 DEPENDENCY MANAGEMENT:

Spring Boot incorporates the concept of starter dependencies, which are curated sets of libraries that provide commonly used functionalities. It simplifies dependency management and ensures that all required dependencies are included automatically, reducing configuration issues and potential conflicts.

6.3.4 AUTO-CONFIGURATION:

Spring Boot's auto-configuration feature analyzes the class path and automatically configures the application based on the detected dependencies. It saves developers from writing boilerplate configuration code, resulting in faster development and reduced code clutter.

6.3.5 ACTUATOR:

Spring Boot Actuator provides out-of-the-box monitoring and management endpoints for the application. It offers metrics, health checks, logging, and other management features, making it easier to monitor and manage the application in production environments.

6.3.6 DEVOPS FRIENDLINESS:

Spring Boot's emphasis on simplicity and ease of use makes it DevOps friendly. It supports various deployment options, including traditional servers, cloud platforms, and containerization technologies like Docker. It also provides features for externalized configuration, making it easier to manage different environments.

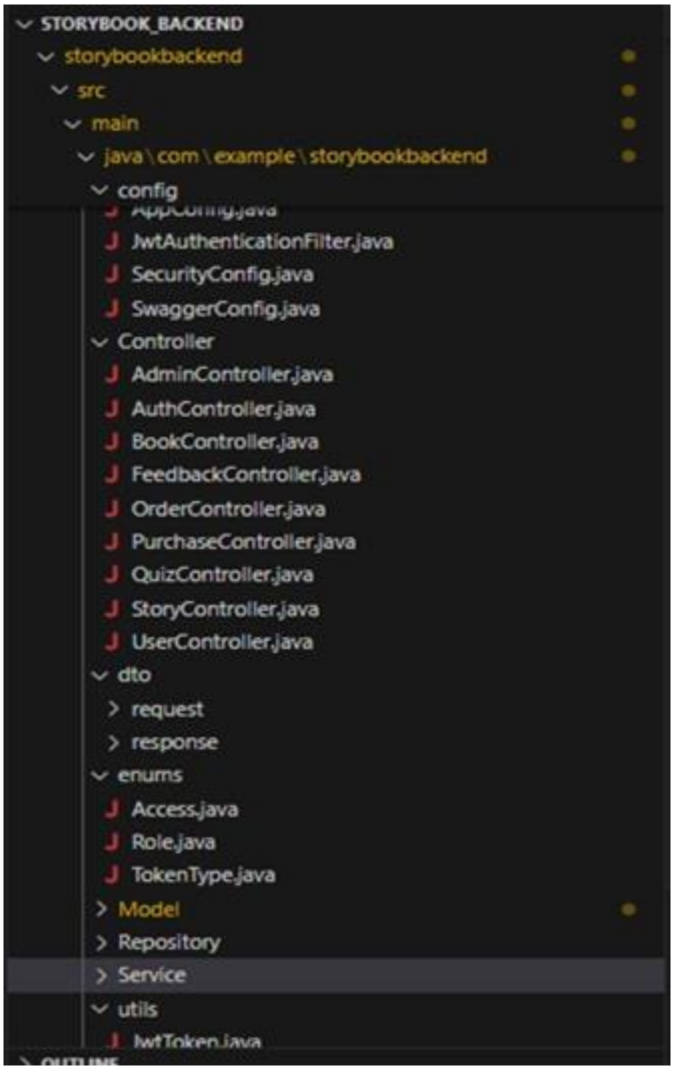
6.4 DATA JPA

Spring Data JPA handles most of the complexity of JDBC-based database access and ORM (Object Relational Mapping). It reduces the boilerplate code required by JPA. It makes the implementation of your persistence layer easier and faster

Spring Data JPA aims to improve the implementation of data access layers by reducing the effort to the amount that is needed.

6.5 SYSTEM ARCHITECTURE

The application adopts a contemporary and scalable three-tier architecture. It comprises the frontend layer, backend layer, and the database layer. Each of these layers fulfills a pivotal role in the application's comprehensive functionality, facilitating seamless communication and efficient data management.



```
v STORYBOOK_BACKEND
  v storybookbackend
    v src
      v main
        v java \ com \ example \ storybookbackend
          v config
            J AppConfig.java
            J JwtAuthenticationFilter.java
            J SecurityConfig.java
            J SwaggerConfig.java
          v Controller
            J AdminController.java
            J AuthController.java
            J BookController.java
            J FeedbackController.java
            J OrderController.java
            J PurchaseController.java
            J QuizController.java
            J StoryController.java
            J UserController.java
          v dto
            > request
            > response
          v enums
            J Access.java
            J Role.java
            J TokenType.java
          > Model
          > Repository
          > Service
          v utils
            J JwtToken.java
> OUTLINE
```

Fig 6.2 Backend System Architecture

The backend layer of the Interactive Children Story Book and Educational web platform is built upon the Spring Boot framework, a Java-based solution renowned for its ability to streamline the development of resilient and scalable web applications. Spring Boot provides a comprehensive suite of features and libraries, enabling seamless management of HTTP requests, data persistence, implementation of robust security measures, and integration with external systems. In this application, the backend's primary role is to craft RESTful APIs that facilitate CRUD (Create, Read, Update, Delete) operations, ensuring the platform is user-friendly and efficient. Additionally, the backend manages user interactions and authentication, creating a secure and personalized experience for both educators and young learners. The backend is architected following the principles of Spring Boot to enhance security and modularity.

6.6 SPRING BOOT STRUCTURE:

Spring Boot is a Java framework that simplifies building enterprise-grade applications. It provides a strong set of features and conventions for developing backend systems, including dependency management, configuration, and automatic setup. Spring Boot follows the principle of convention over configuration, which reduces the amount of boilerplate code needed.

6.6.1 REST API:

The backend of the Interactive Children Story Book and Educational web platform exposes a RESTful API that allows the frontend to communicate with the server. REST (Representational State Transfer) is an architectural style for designing networked applications, utilizing standard HTTP methods (GET, POST, PUT, DELETE) to perform CRUD (Create, Read, Update, Delete) operations on resources. The API endpoints define the URLs and request/response formats for interacting with the system.

6.6.2 CONTROLLER:

In this platform, controllers play a vital role in handling incoming HTTP requests. They map these requests to the appropriate methods within the system. API endpoints are defined by controllers, orchestrating the processing logic for incoming requests. Controllers act as the bridge between the frontend and backend, receiving user inputs,

6.6.3 SERVICES:

Services within the application encapsulate the essential business logic. They orchestrate complex operations and facilitate interactions between different system components, including data retrieval, validation, transformation, and storage. These services ensure that the educational content and interactive stories are managed efficiently and accurately.

6.6.4 REPOSITORIES:

In the application, repositories serve as an abstraction layer for interacting with the database. They define the methods required for executing CRUD (Create, Read, Update, Delete) operations and querying the database using SQL or Object-Relational Mapping (ORM) frameworks like Hibernate.

6.6.5 DATA TRANSFER OBJECTS (DTOS):

Data Transfer Objects (DTOs) play a pivotal role in enabling data exchange between the frontend and backend layers of the application. These objects define the structure and format of data shared in API requests and responses. DTOs are employed to represent user details, event booking, updating, deletion, and other relevant data that is transferred between the frontend and backend, ensuring seamless communication.

6.6.6 SECURITY:

Security measures are a top priority within the application. Authentication and authorization protocols are diligently implemented to safeguard user data and system integrity. A robust security framework, such as Spring Security, is employed to manage user authentication and access control. It offers features such as user registration, login, password hashing, and role-based permissions, contributing to a secure and reliable application.

6.7 UML DIAGRAMS :

6.7.1 USE CASE DIAGRAM:

The Interactive Children Storybook and Education Platform enables young learners to engage with educational content in a fun and interactive way. Children can browse a wide selection of storybooks, participate in educational activities, and take quizzes that reinforce learning. The platform offers personalized learning experiences by adapting content to the child’s progress and preferences. Parents can monitor their child's development and progress, while administrators manage content, track usage, and analyze educational outcomes, ensuring an enriching and tailored educational journey for each child.

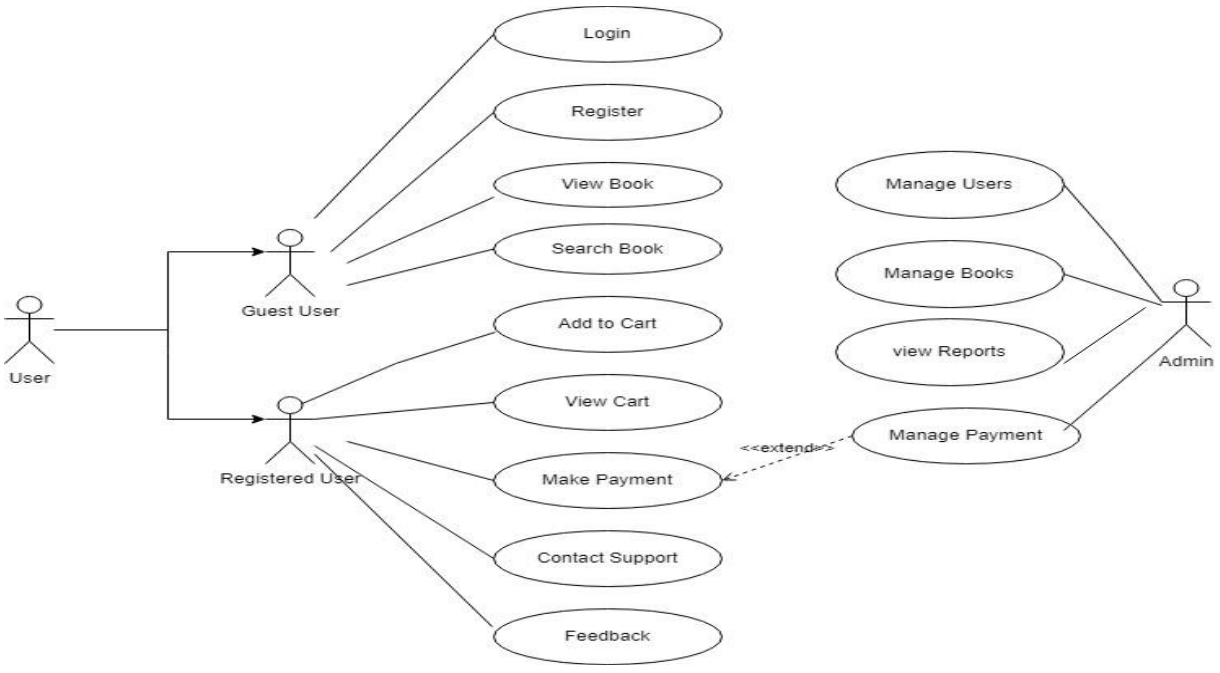


Fig 6.3 Use Case Diagram

6.7.2 CLASS DIAGRAM:

The Interactive Children Storybook and Education System is composed of classes such as User, Book, Story, Quiz, Feedback, and Order Book. Users can browse books, read stories, participate in quizzes, and provide feedback. The Book class manages information about the storybooks, while the Story class handles the content of the stories themselves. Quizzes are associated with books and are designed to reinforce learning. Users can order books through the Order Book class, which tracks orders and manages the checkout process. Feedback from users is managed through the Feedback class, allowing users to share their thoughts on the content. This diagram illustrates the core functionalities and relationships within the system, ensuring an engaging and educational experience for children.

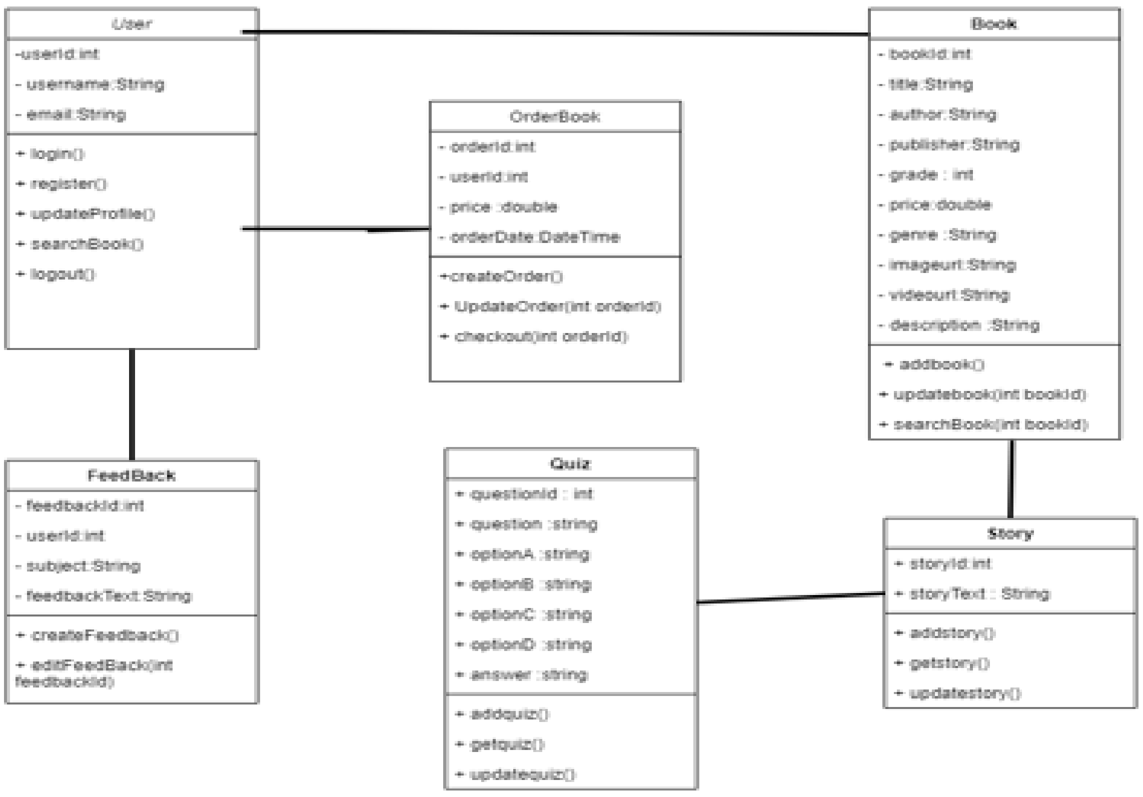


Fig 6.4 Class Diagram

6.7.3 SEQUENCE DIAGRAM:

The sequence diagram for the Online Exam Registration System illustrates the process where Candidates initiate registration by providing personal details and selecting exams, followed by payment processing and admission ticket generation. Admins then verify registrations by reviewing details and updating statuses accordingly. This diagram succinctly outlines the interactions between Candidates, the system, and Admins during the exam registration process, from registration initiation to Admin verification.

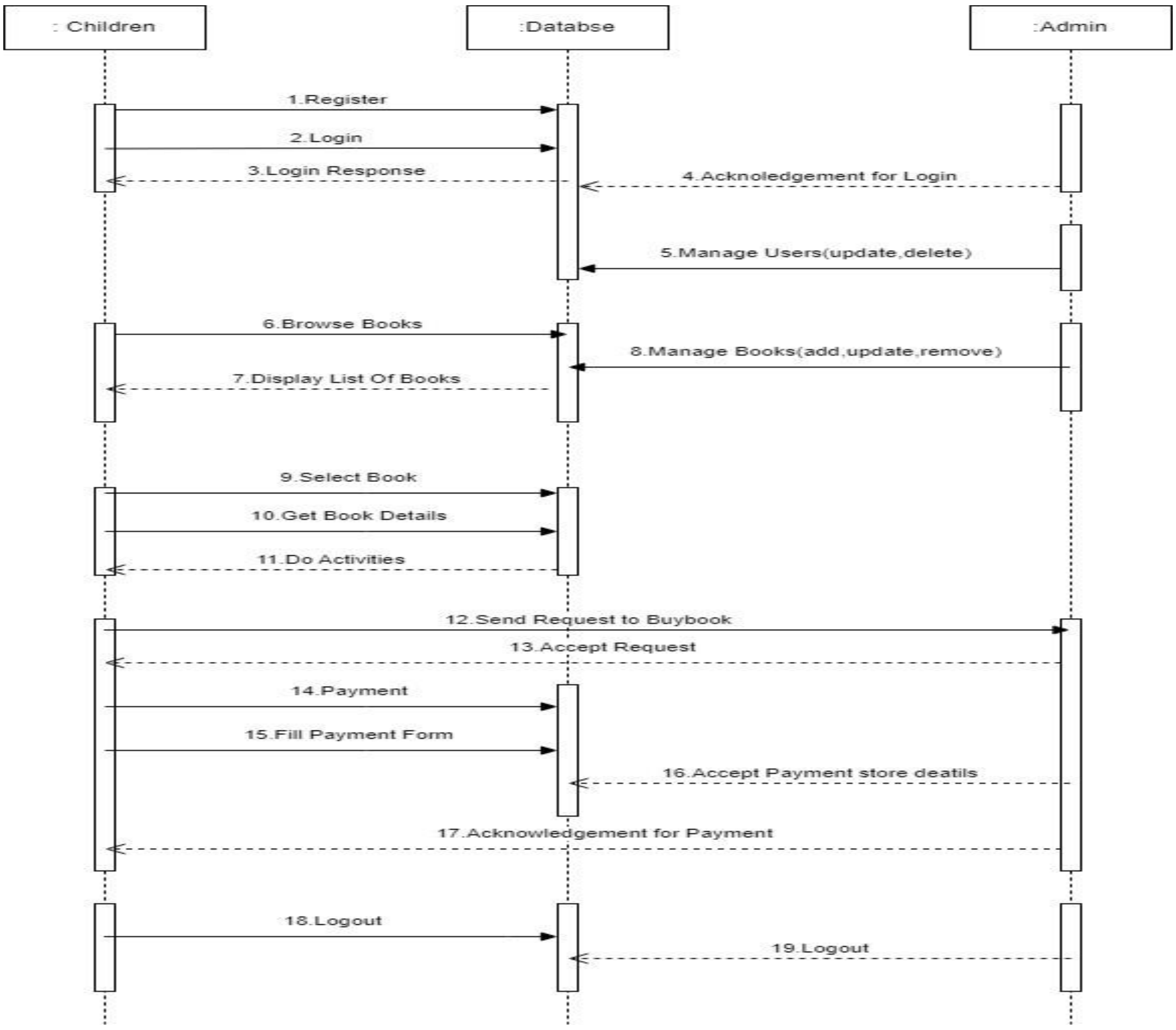


Fig 6.5 Sequence Diagram

6.7.4 COLLABORATION DIAGRAM:

The collaboration diagram for the Online Exam Registration System illustrates the interactions where Candidates input their details and exam choices via the RegistrationForm object, which then communicates with the system for verification. Upon successful verification, the system collaborates with the PaymentGateway for payment processing, generates an AdmissionTicket object for the Candidate, and updates the Registration object. Admins interact with the AdminPanel object to review and approve/reject registrations, updating statuses accordingly in the system. This diagram succinctly captures the collaboration between objects in the exam registration process, from Candidate input to Admin verification and system updates.

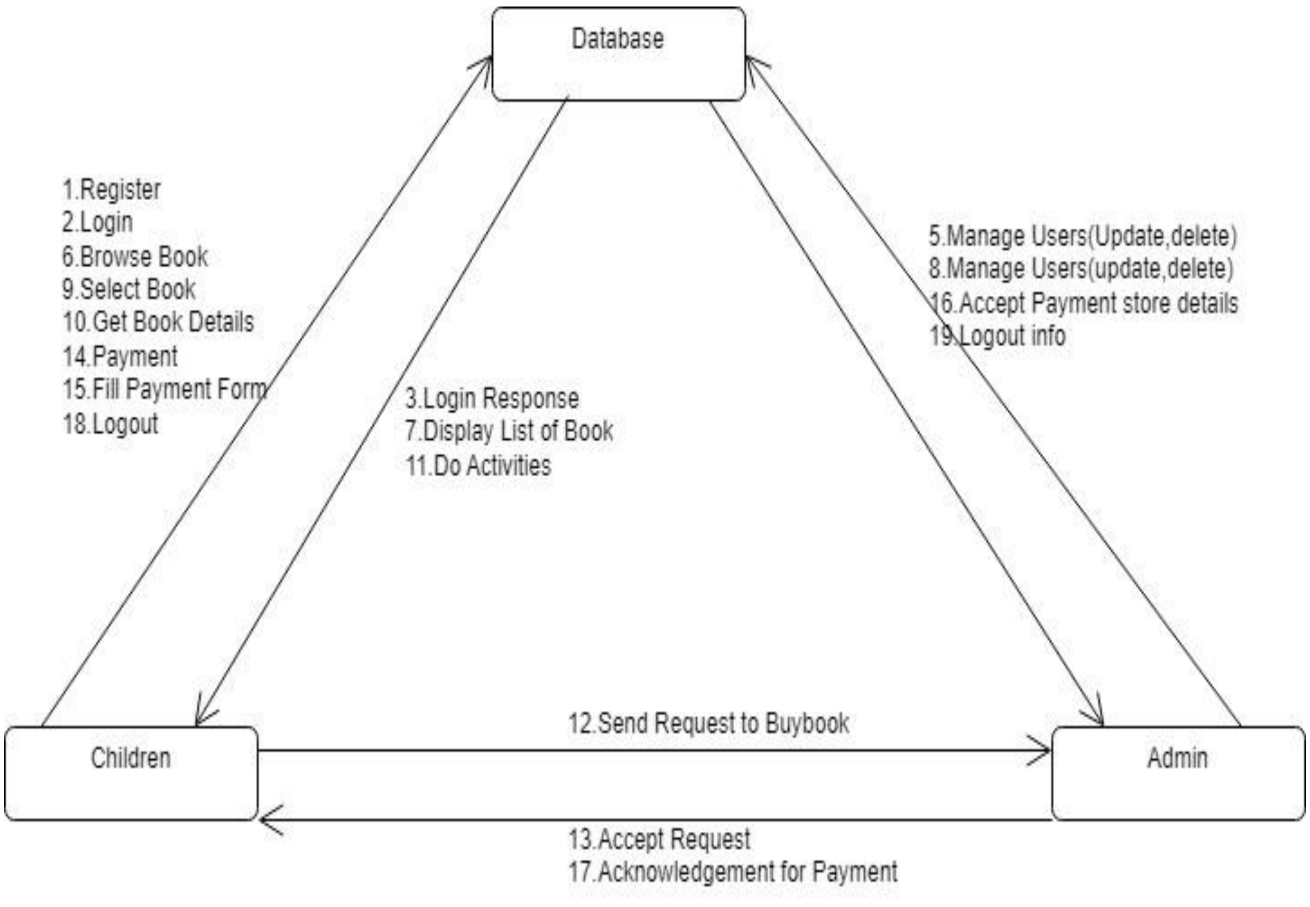
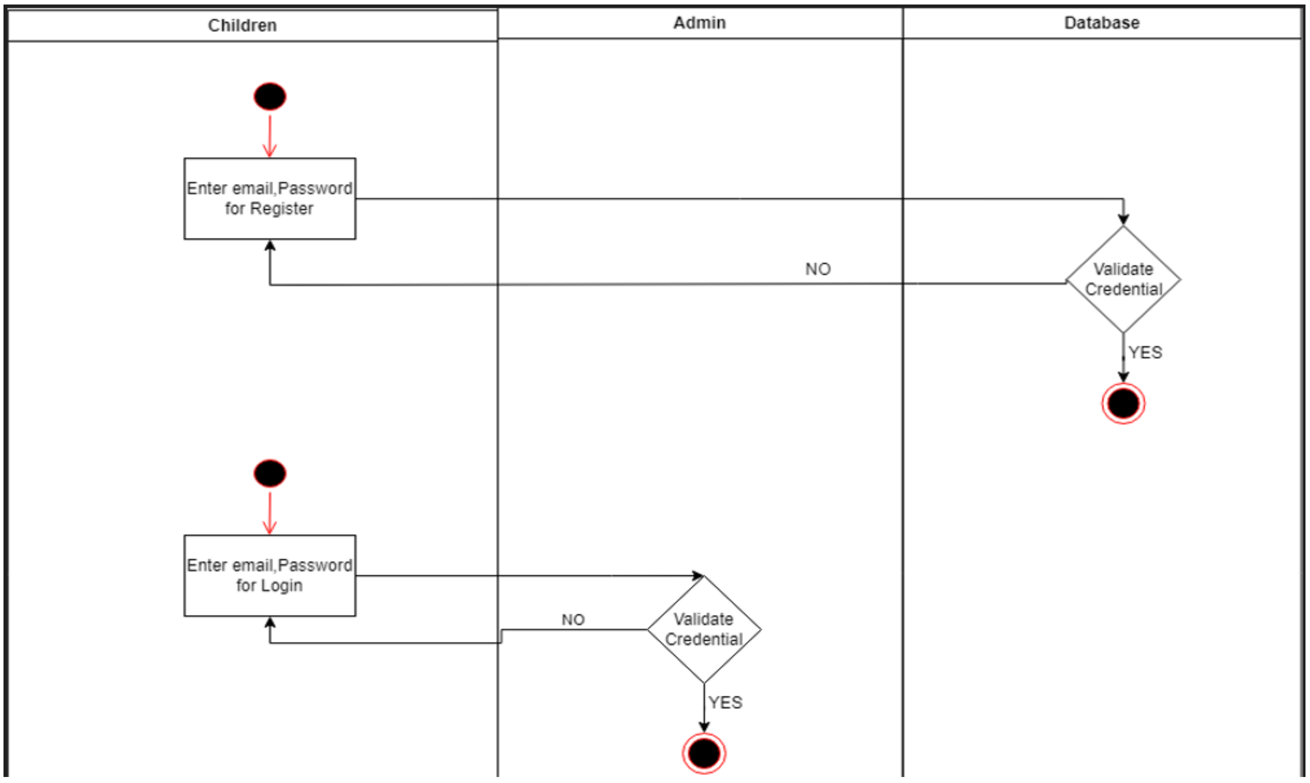
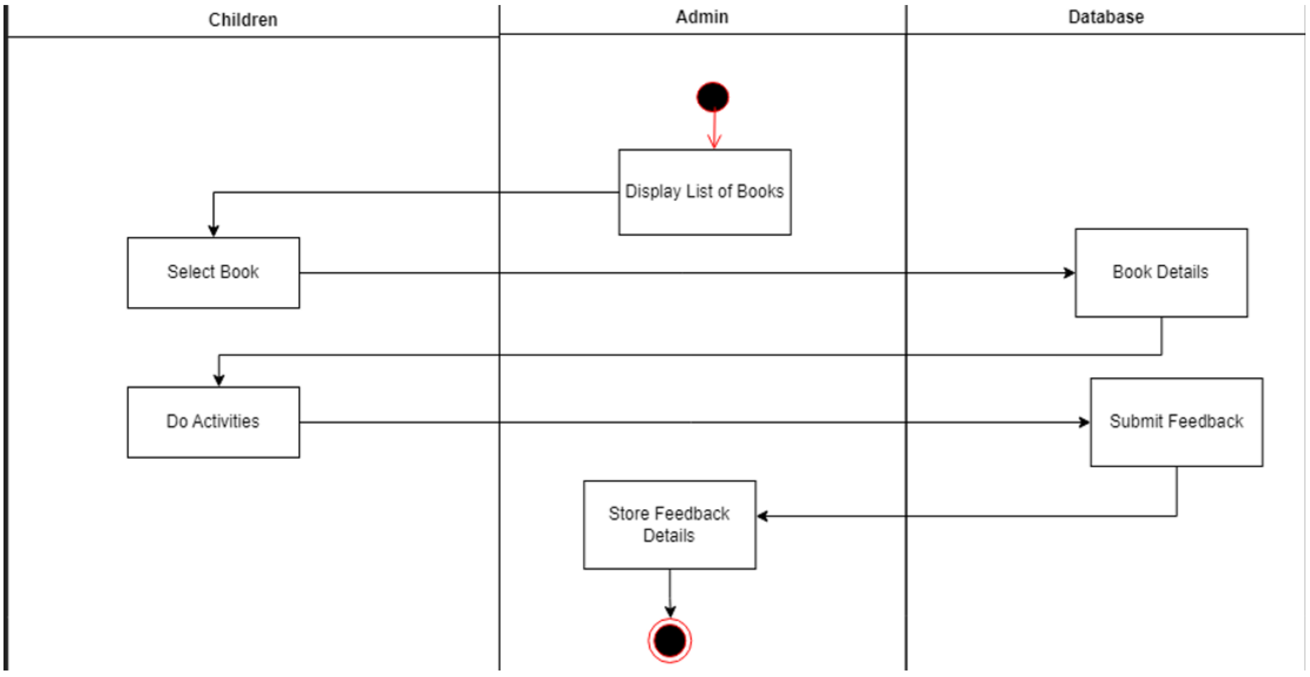


Fig 6.6 Collaboration Diagram

6.7.5 ACTIVITY DIAGRAM:





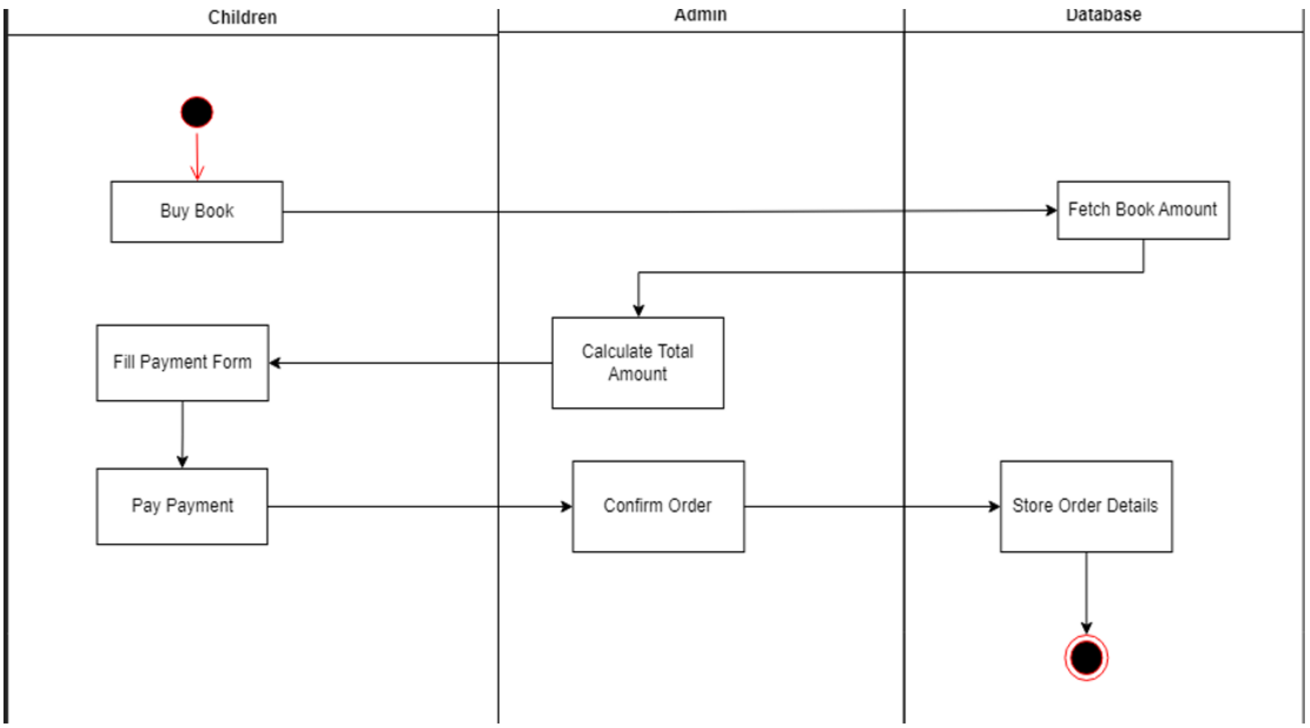


Fig 6.7 Activity Diagram

6.7.6 STATE DIAGRAM:

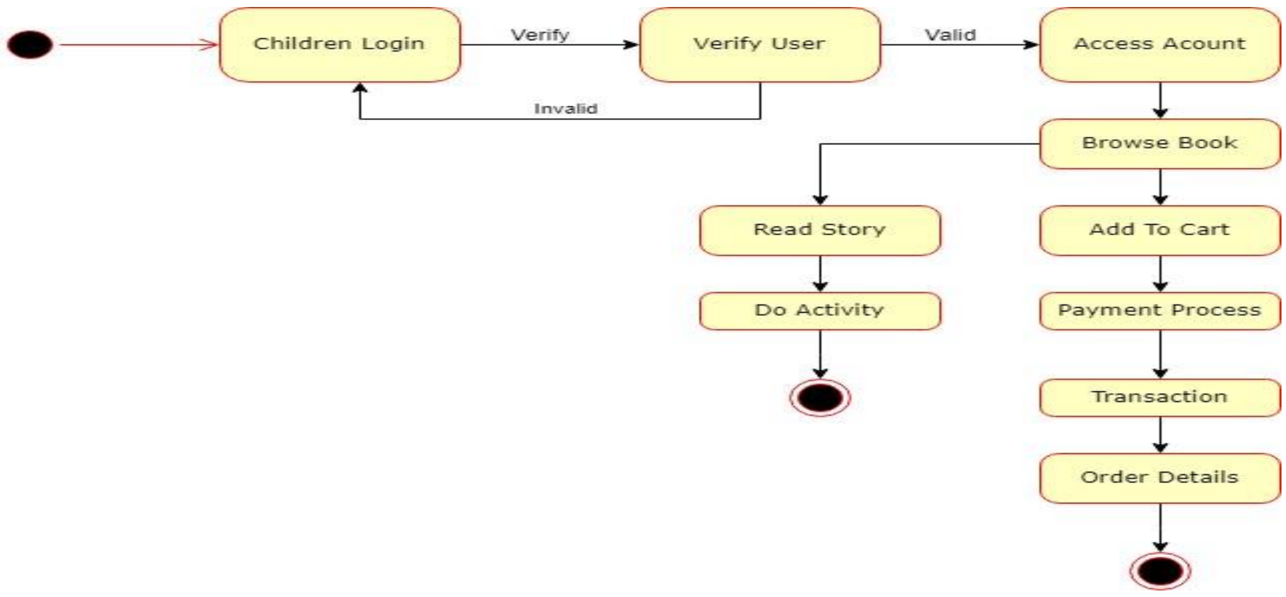


Fig 6.8 State Diagram

6.7.7 PACKAGE DIAGRAM:

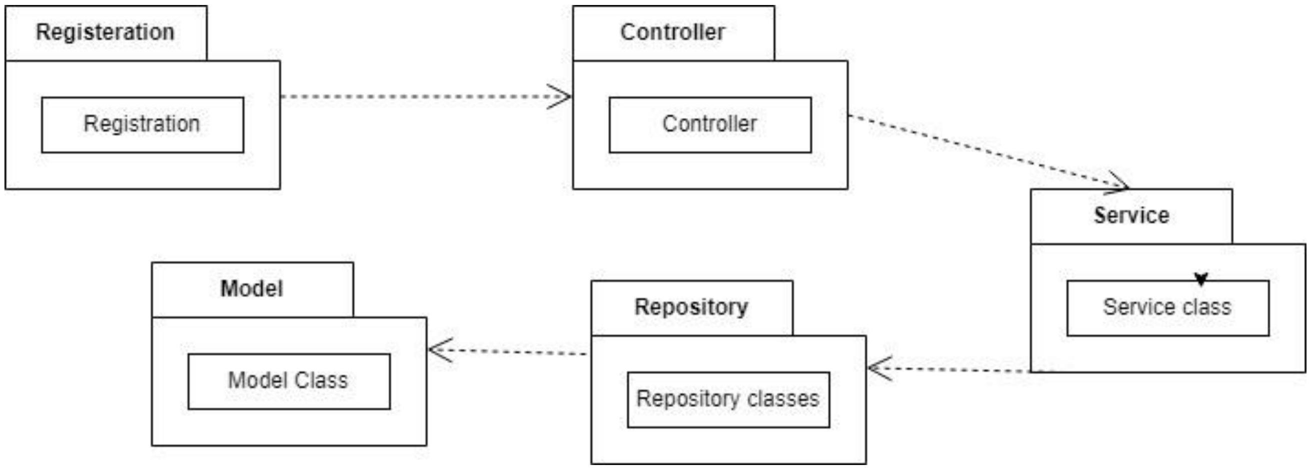


Fig 6.9 Package Diagram

6.8 CODING

6.8.1 USERCONTROLLER:

```
package com.example.storybookbackend.Controller;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;
import com.example.storybookbackend.Model.User;
import com.example.storybookbackend.Service.UserService;

@RestController
public class UserController {
   @Autowired
   UserService userDetailsService;

   //post
   @PostMapping("/adduser")
   public ResponseEntity<User> addUserData( @RequestBody User userDetails)
   {
      User ud=userDetailsService.createUserDetails(userDetails);
      return new ResponseEntity<>(ud,HttpStatus.CREATED);
   }
   //get
   @GetMapping("/getuserdata")
   public ResponseEntity<java.util.List<User>> showuserData()
   {
      return new
ResponseEntity<>(userDetailsService.getUserDetails(),HttpStatus.OK);
   }

   //UPDATE
   @PutMapping("/putuserdata/{id}")
   public ResponseEntity<User> updateUserDetails(@PathVariable("id") int
id,@RequestBody User userDetails )
   {
      if(userDetailsService.updateUserDetails(id, userDetails)==true)
      {
        return new ResponseEntity<>(userDetails,HttpStatus.OK);
      }
      return new ResponseEntity<>(null,HttpStatus.NOT_FOUND);
    }

    //DELETE
    @DeleteMapping("/deleteuserdata/{id}")
    public ResponseEntity<Boolean> deleteuserdata(@PathVariable("id") int id)
    {
      if(userDetailsService.deleteUserDetails(id)==true)
      {
        return new ResponseEntity<>(true,HttpStatus.OK);
      }
      return new ResponseEntity<>(false,HttpStatus.NOT_FOUND);
    }}
```

6.8.2 USERSERVICE:

```
 package com.example.storybookbackend.Service;
 import java.util.List;
 import org.springframework.beans.factory.annotation.Autowired;
 import org.springframework.stereotype.Service;
 import com.example.storybookbackend.Model.User;
 import com.example.storybookbackend.Repository.UserRepo;

 @Service
 public class UserService {
   @Autowired
   UserRepo userDetailsRepo;

   //post and create
   public User createUserDetails(User userDetails)
```

```
{
   return userDetailsRepo.save(userDetails);
}

//get
public List<User> getUserDetails()
{
   return userDetailsRepo.findAll();
}

//get by id
public User getUserById(int id)
{
   return userDetailsRepo.findById(id).orElse(null);
}

  //update or put
  public boolean updateUserDetails(int id,User userDetails)
  {
     if(this.getUserById(id)==null)
     {
        return false;
     }
     try{
        userDetailsRepo.save(userDetails);
     }
     catch(Exception e)
     {
        return false;
     }
     return true;
     }

   //delete
   public boolean deleteUserDetails(int id)
   {
      if(this.getUserById(id)==null)
      {
         return false;
      }
      userDetailsRepo.deleteById(id);
      return true;
   }}
}
```

6.9 SECURITY AND AUTHENTICATION

Security and authentication lie at the heart of the application's robust infrastructure, ensuring the protection of sensitive data and upholding the integrity of the system.

6.9.1 USER AUTHENTICATION:

User authentication is a fundamental pillar, allowing users, including administrators and employees, to register securely by leveraging email and strong, hashed passwords. A robust login system verifies user credentials and controls access to the application. This ensures that only authorized users can access and manage sensitive tax information.

6.9.2 DATA ENCRYPTION:

Data encryption, both in transit and at rest, is a core component of the security framework. Secure communication channels protect data during interactions between the frontend and backend, while encryption of sensitive data in the database safeguards information in the event of a breach. This dual-layer encryption approach ensures that user data, including personal and financial details, remains confidential and protected.

6.9.3 SESSION MANAGEMENT:

Session management maintains secure user sessions, preventing unauthorized access or data exposure. By implementing secure session tokens, the application ensures that users' sessions are both authenticated and securely managed throughout their interaction with the platform.

6.9.4 SECURITY FRAMEWORKS:

Utilizing security libraries and frameworks, such as Spring Security, enhances the efficiency of authentication and access control. Spring Security provides comprehensive features for securing applications, including user registration, login, password hashing, and role-based permissions, which are essential for protecting sensitive user data and controlling access within the system.

6.10 CODING

6.10.1 AUTHCONTROLLER:

```
package com.example.storybookbackend.Controller;

import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.example.storybookbackend.Service.AuthService;
import com.example.storybookbackend.dto.request.LoginRequest;
import com.example.storybookbackend.dto.request.RegisterRequest;

import io.swagger.v3.oas.annotations.Operation;
import io.swagger.v3.oas.annotations.Parameter;
import io.swagger.v3.oas.annotations.tags.Tag;
import lombok.RequiredArgsConstructor;

@RestController
@RequestMapping("/api/auth")
@CrossOrigin(origins = "http://localhost:3000")
@RequiredArgsConstructor
@Tag(name = "Authentication", description = "Endpoints for user authentication")
public class AuthController {

    private final AuthService authService;
    @PostMapping("/register")
    @Operation(summary = "Register a new user", description = "Allows users to register
by providing necessary registration details.")
    public ResponseEntity<?> register(@Parameter(description = "Registration details of the
user") @RequestBody RegisterRequest registerRequest) {
        return new ResponseEntity<>(authService.register(registerRequest), HttpStatus.OK);
    }
    @PostMapping("/login")
    @Operation(summary = "Authenticate user", description = "Allows users to authenticate
by providing valid login credentials.")
    public ResponseEntity<?> login(@Parameter(description = "Login credentials of the
user") @RequestBody LoginRequest loginRequest) {
        return new ResponseEntity<>(authService.login(loginRequest), HttpStatus.OK);
    }}
```

6.10.2 AUTHSERVICE:

```
package com.example.storybookbackend.Service.impl;
import java.util.HashMap;
import java.util.Map;
import java.util.Optional;
import org.springframework.security.authentication.AuthenticationManager;
import
org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.stereotype.Service;
import com.example.storybookbackend.Model.Token;
import com.example.storybookbackend.Model.User;
import com.example.storybookbackend.Repository.JwtRepo;
import com.example.storybookbackend.Repository.UserRepo;
import com.example.storybookbackend.Service.AuthService;
import com.example.storybookbackend.dto.request.LoginRequest;
import com.example.storybookbackend.dto.request.RegisterRequest;
import com.example.storybookbackend.dto.response.LoginResponse;
import com.example.storybookbackend.enums.Role;
import com.example.storybookbackend.utils.JwtToken;

import lombok.RequiredArgsConstructor;

@Service
@RequiredArgsConstructor

public class AuthServiceImpl implements AuthService {

    private final UserRepo userRepository;
    private final JwtRepo tokenRepository;
    private final PasswordEncoder passwordEncoder;
    private final AuthenticationManager authenticationManager;
    private final JwtToken jwtUtil;

    @Override
    public String register(RegisterRequest registerRequest) {
        Optional<User> userExist =
userRepository.findByEmail(registerRequest.getEmail());
        if (userExist.isPresent()) {
            return "User already exists with email id " + registerRequest.getEmail();
        }
        var user = User.builder()
                .name(registerRequest.getName())
                .email(registerRequest.getEmail())
                .password(passwordEncoder.encode(registerRequest.getPassword()))
                .role(Role.User)
                .build();
        userRepository.save(user);
        return "User registered successfully.";
    }

    @Override
    public LoginResponse login(LoginRequest loginRequest) {
        authenticationManager.authenticate(
                new UsernamePasswordAuthenticationToken(loginRequest.getEmail(),
loginRequest.getPassword()));
        var user = userRepository.findByEmail(loginRequest.getEmail()).orElseThrow();
        Map<String, Object> extraClaims = new HashMap<>();
        extraClaims.put("role", user.getRole().toString());
        var accessToken = jwtUtil.generateToken(extraClaims, user);
        revokeAllUserTokens(user);
        saveUserToken(user, accessToken);
        return LoginResponse.builder().accessToken(accessToken).build();
    }

    private void saveUserToken(User user, String accessToken) {
        var token = 
Token.builder().user(user).token(accessToken).expired(false).revoked(false).build();
        tokenRepository.save(token);
    }

    private void revokeAllUserTokens(User user) {
        var validUserTokens = 
tokenRepository.findAllByUser_UserIdAndExpiredFalseAndRevokedFalse(user.getUserI
d());
        if (validUserTokens.isEmpty())
            return;
        validUserTokens.forEach(token -> {
            token.setExpired(true);
            token.setRevoked(true);
        });
        tokenRepository.saveAll(validUserTokens);
    }

    @Override
    public String createAdmin() {
        Optional<User> userExist = userRepository.findByEmail("admin@gmail.com");
        if (userExist.isPresent()) {
            return "User already exists with email id - admin@gmail.com";
        var user = User.builder()
                .name("Admin")
                .email("admin@gmail.com")
                .password(passwordEncoder.encode("Admin@123"))
                .role(Role.Admin)
                .build();
        userRepository.save(user);
        return "Admin registered successfully.";}}
```

CHAPTER 7

CONCLUSION

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

7.1 CONCLUSION

In conclusion, the proposed interactive children's storybook and education app is designed to benefit young readers, parents, and educators by combining entertainment with learning. This app is scalable to accommodate a wide range of books and educational content, making it suitable for children of various age groups and learning levels. It aims to enhance the reading experience through features like read-aloud options, interactive quizzes, and the ability to purchase books directly within the app. Additionally, the app fosters a love for reading while promoting cognitive development through interactive content. With its user-friendly interface and engaging functionalities, this app empowers children to explore stories independently or with guidance, making learning a fun and rewarding experience for both children and parents alike.

7.2 FUTURE SCOPE

Future updates could introduce advanced interactivity, such as gamified learning, voice recognition for quizzes, and augmented reality (AR) elements that bring characters to life, making reading more engaging. Expanding the app to include multilingual support would promote language learning and cultural awareness, while partnerships with publishers and content creators could continuously enrich the library with new storybooks and educational materials. Enhancing parental controls and progress tracking tools would enable parents to monitor their child's educational development more effectively. Furthermore, developing a mobile version and enabling offline access would increase accessibility, allowing children to learn anytime, anywhere.

CHAPTER 8

REFERENCES

[1] **Maven Dependency Reference**:

- Maven Central Repository
- Maven Repository Guide

[2] **React Medium Articles**:

- React Development on Medium
- Popular React Articles on Medium

[3] **JWT Token References**:

- JWT Official Documentation
- Auth0 JWT Guide
- JWT Handbook

[4] **React Vite Documentation**:

- Vite Official Documentation

[5] **Speech Analysis and Recognition Tools**:

- React Speech Recognition, available from https://www.npmjs.com/package/react-speech-recognition

[6] **Additional References**:

- Spring Boot Official Documentation, available from https://spring.io/projects/spring-boot
- Material-UI Documentation, available from https://mui.com/getting-started/installation/
- React Official Documentation, available from https://reactjs.org/docs/getting-started.html
- GitHub Repository for React Projects, available from https://github.com/deva0506/Children-s-Story-Book-Web