Book Rabbit

Book Store & Book Reviwe Sharing System

BSc (Hons.) in Information & Communication Technology

ICT 3113 – Web Application Development

Lecturer In Charge : Mr. Samitha Nanayakara

Final Project Report – Group Number 05

Department of Information & Communication Tecnology

University Of Sri Jayewardenepura

Acdemic Year 2020/2021

[Submitted Date – 29/08/2024]

Table of Content

Introduction

"A reader lives a thousand lives before he dies. The man who never reads lives only one."

-George R.R. Martin-

In an information-soaked world, books remain a timeless source of knowledge, inspiration, and escape. The amount of literature available, though, can be overwhelming and difficult to get a grip on the books that will really speak to the reader.

That's where **Book Rabbit** our novelty comes in a web-based platform designed to enrich the reading experience through a blend of bookstore and personalized book recommendation system.

Why is Book Rabbit necessary____________________?

The digital age exposes readers to the paradox of choice. Millions of books are published every year, and the chances of getting the right one have grown difficult. Physical libraries and online marketplaces offer enormous collections; most of them, however, lack personalized recommendations that leave users frustrated over searching rather than reading.

Here we were provided answering some of the Important Questions in Society,

- How will readers find a book that reflects their unique interest?
- How could a person select books be intuitive and fun?
- Is there any way to persuade a person to read ?

So everyone Book Rabbit: A Solution

BOOK RABBIT

Apart from just offering a platform to buy books, **Book Rabbit** provides solutions that include personalized recommendations for users reading tastes.

The system will allow users to browse and search through the myriad collections to ensure that they find the exact version of what they are looking for. Through sophisticated algorithms and user-friendly interfaces, **Book Rabbit** automates finding that perfect book down to an easy and pleasurable task.

Target Audience

This is a web system which is designed to cater to a diverse range of readers, each with unique needs and preferences.

Here we mentioned some from that range and the reason to them why we need to access with us.

- Avid readers who constantly seek new literary adventures to casual readers looking for their next great find, the platform offers personalized recommendations that simplify the process of discovering books.
- Students and academics will appreciate the system's ability to quickly identify relevant texts, while book collectors can explore and acquire rare or special editions.
- Parents and gift buyers will find it easy to select the perfect books for their children or loved ones, thanks to tailored suggestions based on age and interests.
- **Book Rabbit** supports book clubs and reading groups by recommending titles that match collective preferences, fostering a sense of community and shared reading experiences.
- Authors and publishers also benefit, as the platform provides a direct channel to reach targeted audiences.

Additionally making **Book Rabbit** an essential tool for anyone passionate about books.

Motivation

The motivation section should clearly convey why our passionate about the project and why it’s worth pursuing. It sets the stage for the rest of your project report by providing a strong rationale for our work.

So these are the driving factors behind our project which is **BOOK RABBIT.**

1. The Overwhelming Volume of Books

Nowadays the number of books available to read is impressively larger. Hundreds of thousands of titles are published annually, and it seems that the search for the right book to read becomes a complicated and frustrating effort not that far from the search of a needle in a haystack. That often brings the burden of choice – one asks oneself what book one should be reading next – or lost opportunities of books that would be perfect if only they could be found.

2. Lack of Personalized Review sharing for Recommendations

Even though many of the media recommendation platforms exist today, they provide readers with general recommendations for books that do not specially cater to readers' tastes. Therefore, most readers go by bestsellers or randomly pick a book off the shelf, most of which are not in sync with their particular interests. There is a clear requirement that the system developed understands personal preferences and helps readers make choices to find books they would really enjoy.

When they can share their personal experience using reviews it become a indirect recommendation to others well.

3. Need for Integrated Bookstore and Recommendation

Numerous online bookstores exist, and there are many recommendation sites, but these two have been operating in isolation. Readers make use of the one to find books of choice and then revert to another one to make a purchase. **Book Rabbit** is an effort in that direction—an integrated platform that assists users in finding as well as, later, obtains books as per recommendations.

4. Promoting Reading and Literacy

Reading is a powerful force for individual development, education, and entertainment. By helping the book lover find the books they love easily, **Book Rabbit** embarks on a mission to increase reading and consequently increase literacy. It will provide an arena—inspiring the love of reading, getting people interested in new genres and authors, and finally enhancing the reading experience.

5. Cultivating a Community of Readers

It really doesn't stop at buying and recommending books. **Book Rabbit** actually cultivates community where readers get to share reviews, recommendations, and their experiences. Adds a bias of value to the website which does not seem to be transactional. It is a platform through which people connect, discuss, and grow.

More often than not, the making of **Book Rabbit** is driven by the need to address the innumerable books published every year and to tap into the hidden gems. With millions of titles published every year, more often than not, readers will have to contend with which pick to read next. Current platforms provide very general recommendations that do not in any specific way represent the readers' preferences, thus losing opportunities for the discovery of books that resonate deep within.

Recognizing this lacuna, **Book Rabbit** was architected to bring the convenience of an online bookstore with an advanced recommendation system to see that one discovers and buys books instantaneously. It additionally strives to simplify and relieve the challenges of locating the perfect book by inspiring reading and literacy through a journey made personal and thrilling. **Book Rabbit** thus has a goal to create a community of readers so that everyone may post their reviews, recommendations, and discuss those that would enrich the reading experience.

Problems & Objectives

This section will be illustrated the specific goals and reason to that why we selected that as our task in our projects. Under the theme here we go,

- *Provide Personalized Recommendations of Books Through the Reviews*

Goal - Design an intelligent, recommendation-based system that would provide books through user preference, reading history, and browsing behavior.

The reason to add this is it helps users to find new books that would suit their interests, hence enhancing their readability experience and read more often.

- *Create an Integrated Bookshop and Recommendation Platform*

Goal: Provide the seamless user experience of discovering, saving, and reviewing all books under one roof. The feature of blog

By bringing all this together into one platform, Book Rabbit makes the user journey much more convenient and effective in terms of simplicity.

Here wish list option give the user the cart experience in a real world shop.

- *Enhanced User Experience through Intuitive User Interface*

Goal: Design and develop an intuitive user interface that shall also be aesthetic and easy to browse, search, and interact with the system.

The reason to use this an interactive and user-friendly interface will guarantee that users move around the platform with much ease, hence higher satisfaction and involvement.

- *Design a Solid, Scalable Database Structure*

Goal: Build a database for efficient storage and handling of a huge catalog of books, user profiles, reviews, and other relevant data.

A well-designed database is crucial to the performance of a system, particularly while the user base and the book catalog are increasing.

- *Ensure the security of user authentication and data protection.*

Goal: Develop secure mechanisms of authentication and protection of user information.

Security is about ensuring that users' private information is safe and retains trust in the platform.

- *Socialize and Engage the Community*

Goal: Include features that users can utilize to share reviews, discuss, and recommend books to one another.

This grows the community feeling between users, which enhances each user's experience, hence their loyalty to come back.

- *Improve Search and Functions*

Objective: powerful competent, search functionality and that permit users to fast-track the finding of certain books or browse through categories, genres, authors, etc.

Efficient search functionality ensures that users readily find books they are looking for, thus improving general satisfaction with the platform.

- *Facilitate To sharing user ideas*

Goal: Provide a forum in which it becomes easy for any author or reader to reach out to them with their books for the targeted audience using their reading experience. They can share their ideas in words as reviews.

The reason to do this it will not only help the readers as well as this is the grate way to persuade others to reading.

Overview of the System

The overview of the system will be covered the all criteria s of our book store and book recommendation web system.

Here we are dividing this under the four main sectors such as,

- Working Scope & User Case Diagram
- System Technical Requirements
- System Architecture
- Database Design

Let us drive into each subsection in this way.

Working Scope & User case Diagrams

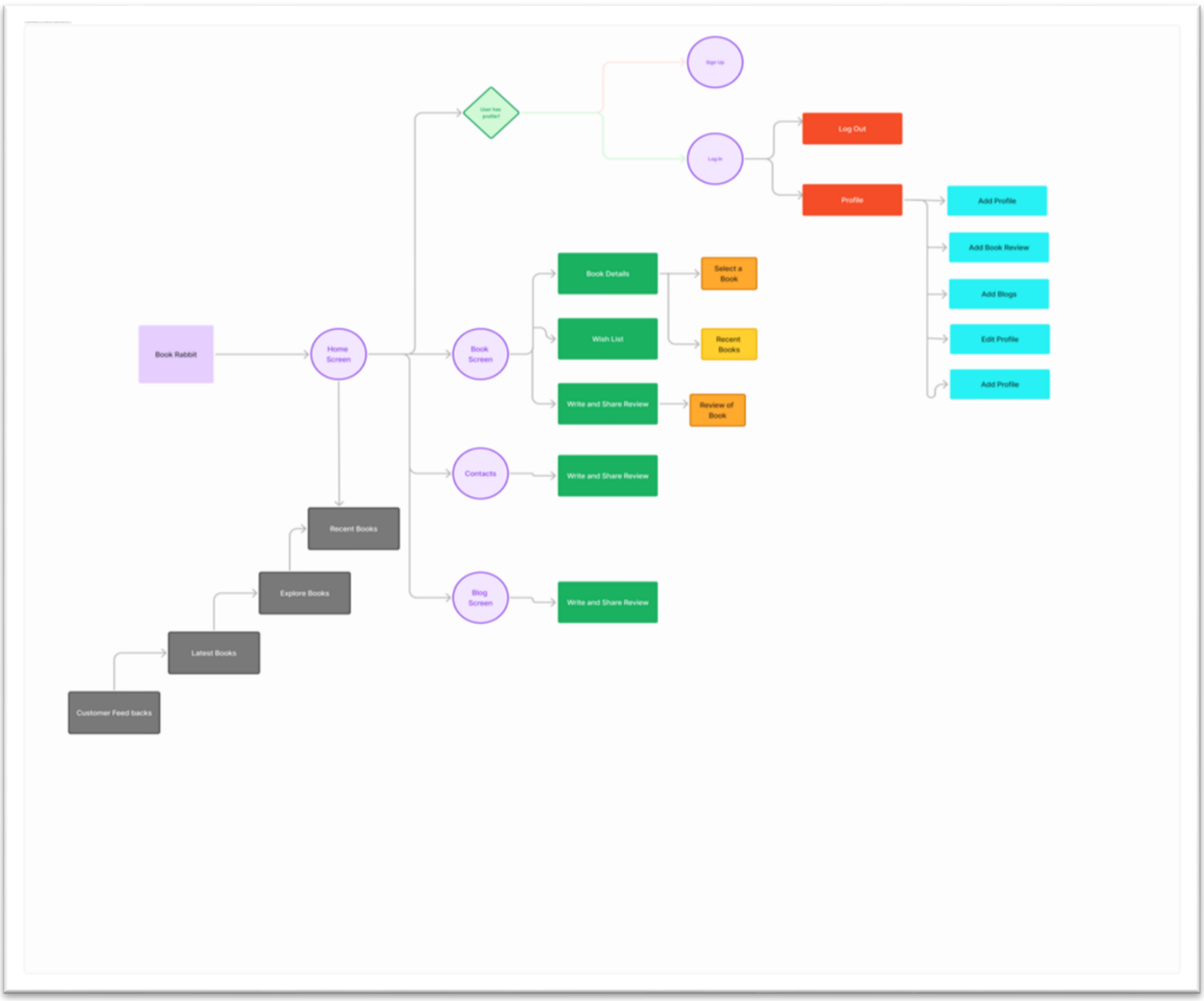
The Book Rabbit project is developed to provide an online comprehensive platform for book lovers. This project integrates a personalized recommendation engine with full-featured bookstore functionality. The major features of the system are user registration and authentication, advanced book searching and browsing, personalized recommendations about books based on user preferences and reading history, and secure online purchasing.

Users will be able to write reviews and rate books, maintain personal lists of books they have read, and socialize with friends and conversation.

The system is, however, oriented to digital interactions, and the case of physical book deliveries is ruled out. Moreover, the system will be optimized for web access; the development of mobile applications falls outside the initial scope of the current project. Running the program should be easy for any heavy, student, or casual reader who is in search of more and exploring exciting books.

After that the we decided to make a user case diagram to represent the all the functionalities entire system as well.

This combination will clearly depicts the all boundaries and flows in our project in standard way.



The above User Flow Diagram Represent the flow direction entire system as well. Here is the Figma Page Link for more information.

Access to figma page –

https://www.figma.com/board/A9STJdqDHd3arpNJ0YG9Sy/User-Flow-Diagram-for-FigJam-(Community)-(Copy)?node-id=202-240&t=YicxGqrgrWBZd36d-1

System Technical Requirements

This subsection outlines all the technical elements needed to develop and run the system. This includes Soft wares which have an additional tools.

- **Frontend**:
    - React
    - Type Script
    - Java Script
    - HTML/CSS
    - Bootstrap
- **Backend**:
    - Node.js
    - Express.js
    - Mongo DB
    - Restful APIs
    - Mongoose
    - React - router
- **Additional tools**
    - GIT / GIT Hub
    - Postman (Testing APIs)
    - Figma
    - AWS

Let us move to next sub section now.

System Architecture

The system architecture is the layout of the web system and architecture can be giving clear and accurate understanding how the system is designed.

Usually here this section focuses about how to design and create our User Interface (UI).

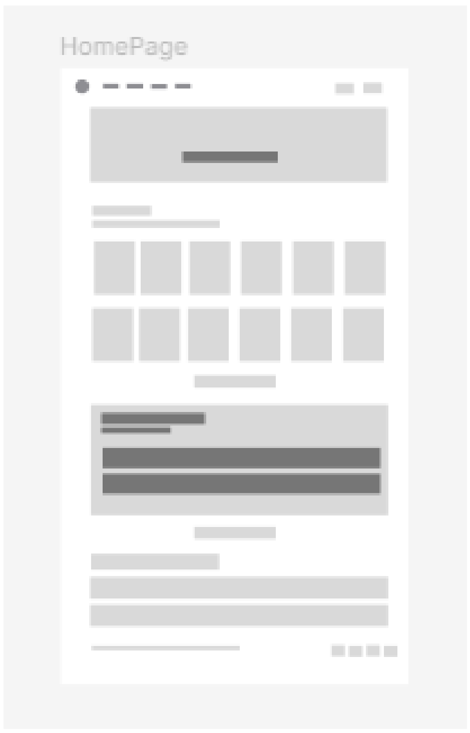
The following visual representation will be provided high level overview about our system which named **Book Rabbit**.

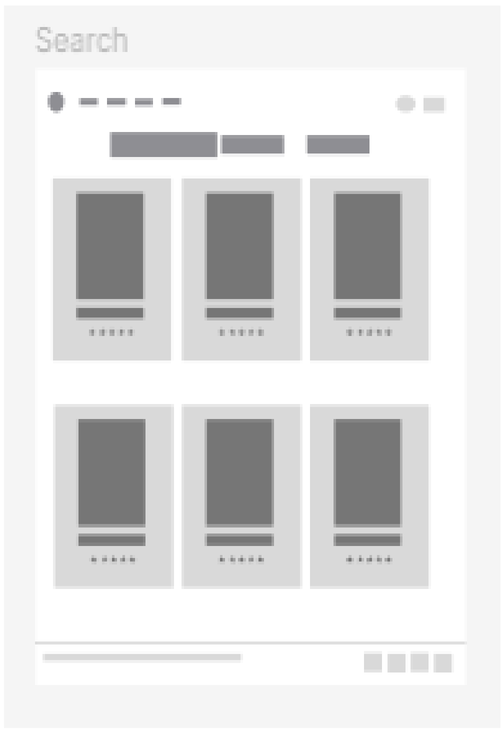
Also in previous stage we illustrate an user case diagram and this section will be give more accusative explanation about that flows too.

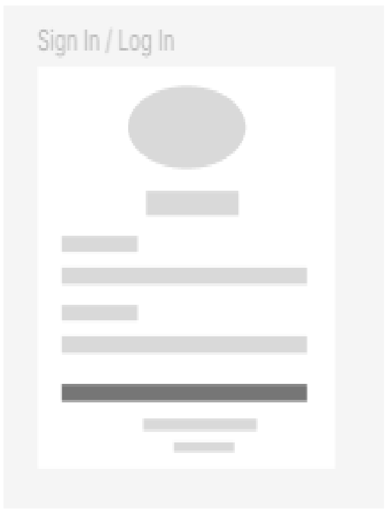
Very first we were sketch our project before the create the final designs of UIs. Here we were using **Figma** wireframes to that activities. The below snapshots convey some ideas how the design come out.

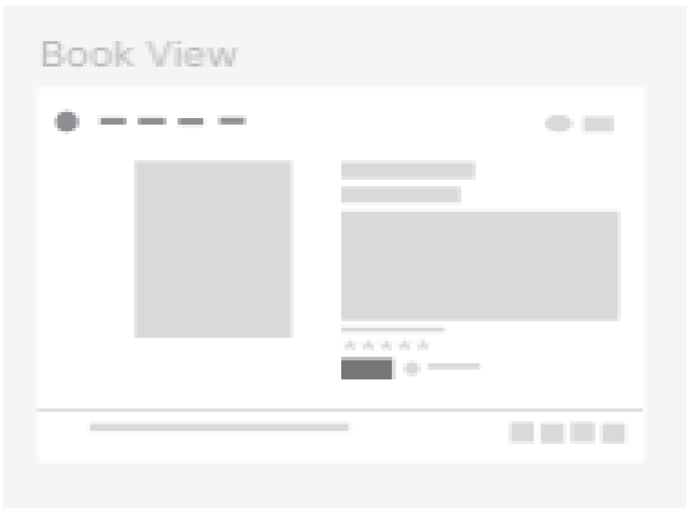
We shared few snapshots of our wireframe images related to some GUIs such as Home screen , Search screen , Book list and Blog here.

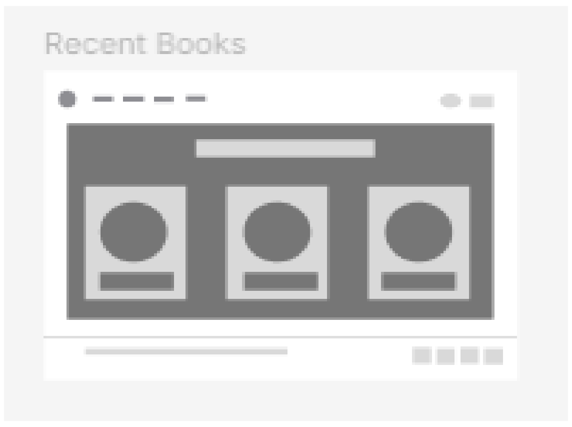
The previous step of design GUIs that we decided to make Wireframes like this.

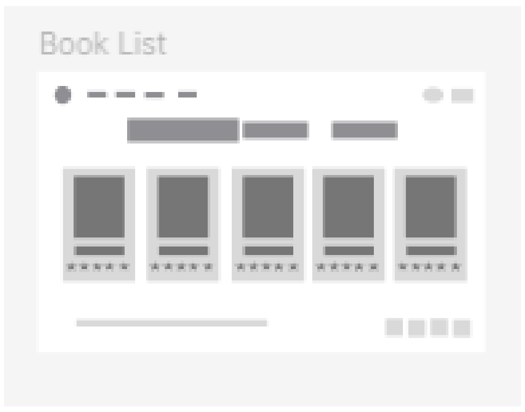


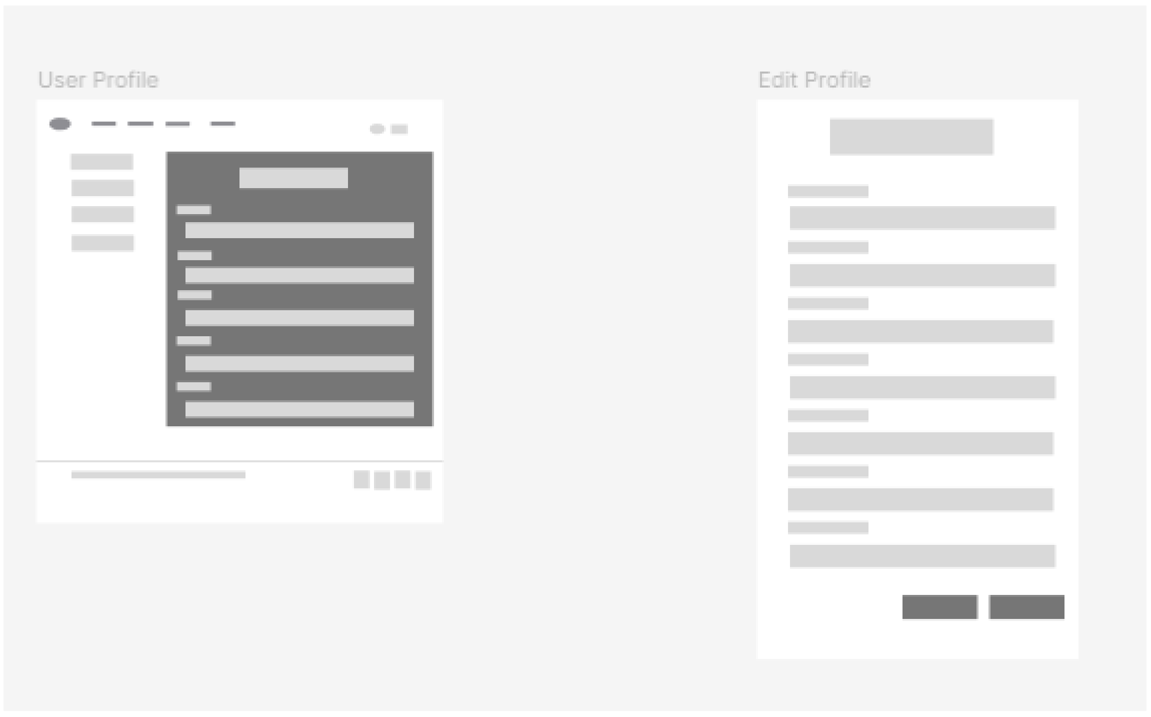










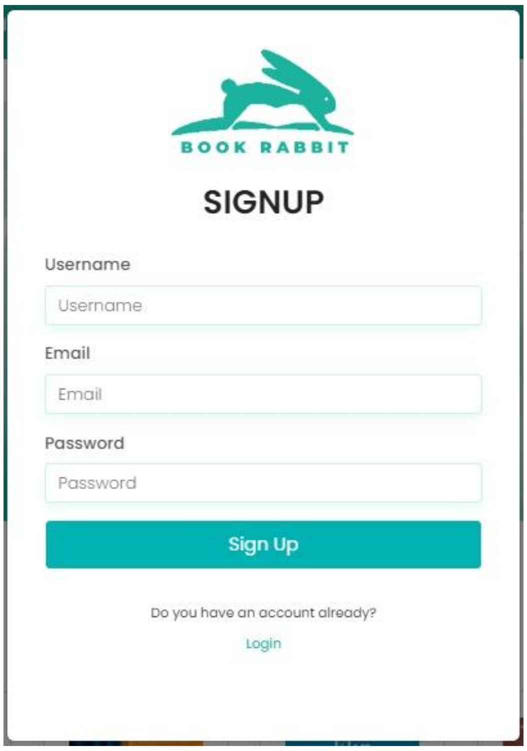


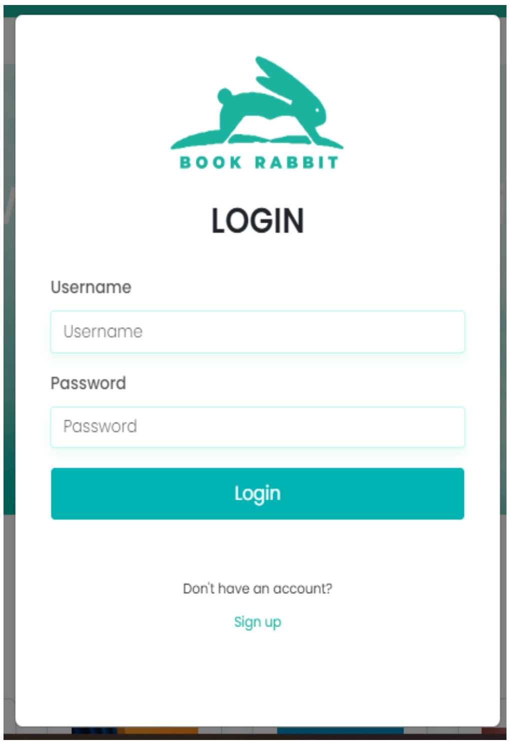
Here we share the Figma wireframes design documentation link here as well for more information.

Access to wireframes here - https://www.figma.com/design/3FnfWFq0xsgODGm9caXVTA/Book-Rabbit?node-id=0-1&t=3poJAEB6We9L5QHP-1

Now we can move into our UI designs well. The following snapshots of UI design scenarios also represent the key features of entire project as well.

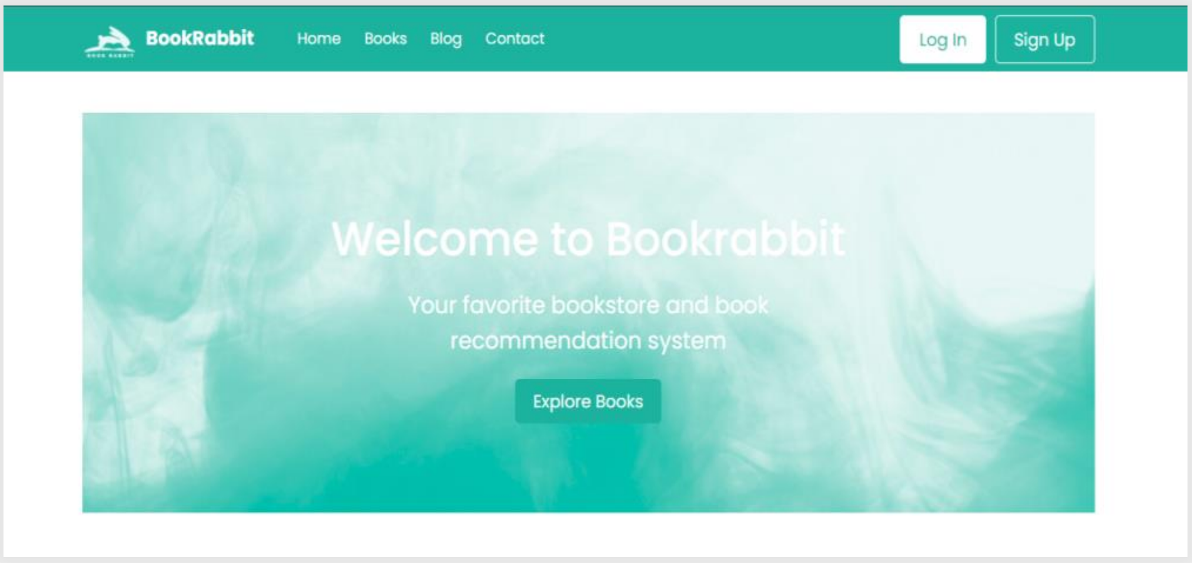
Sign up and Log in



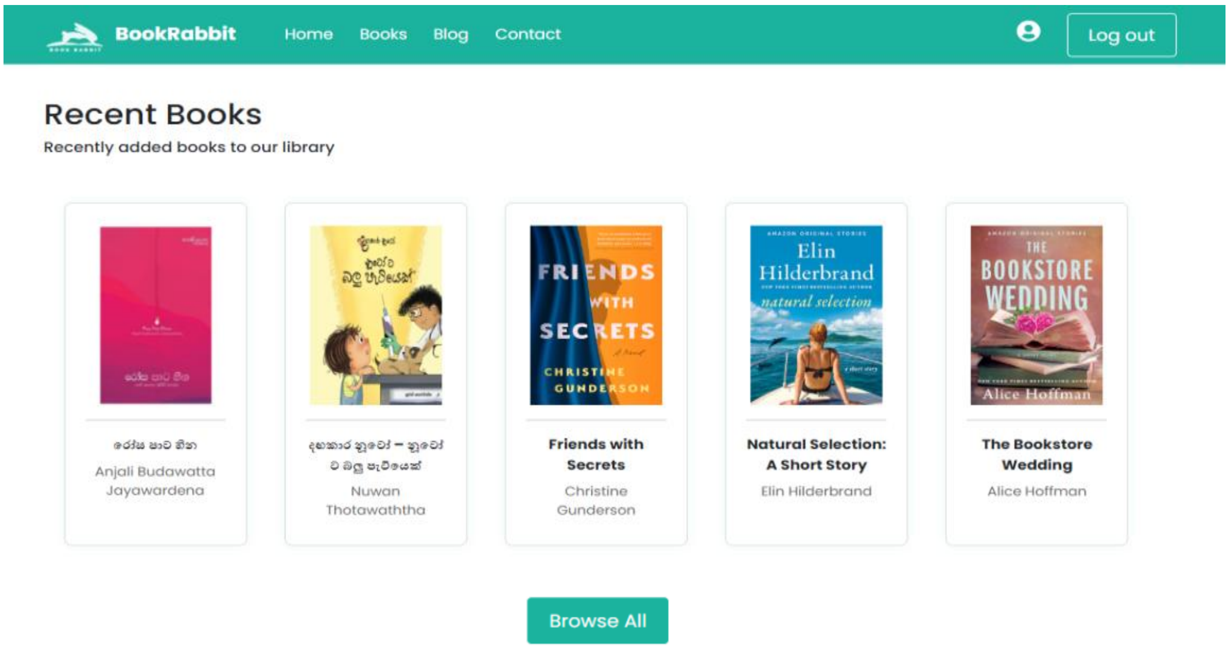


Home screen

Hero Section



Recent book section



Recent Books

Recently added books to our library

ශරෝස පාට හීන
Anjali Budawatta Jayawardena

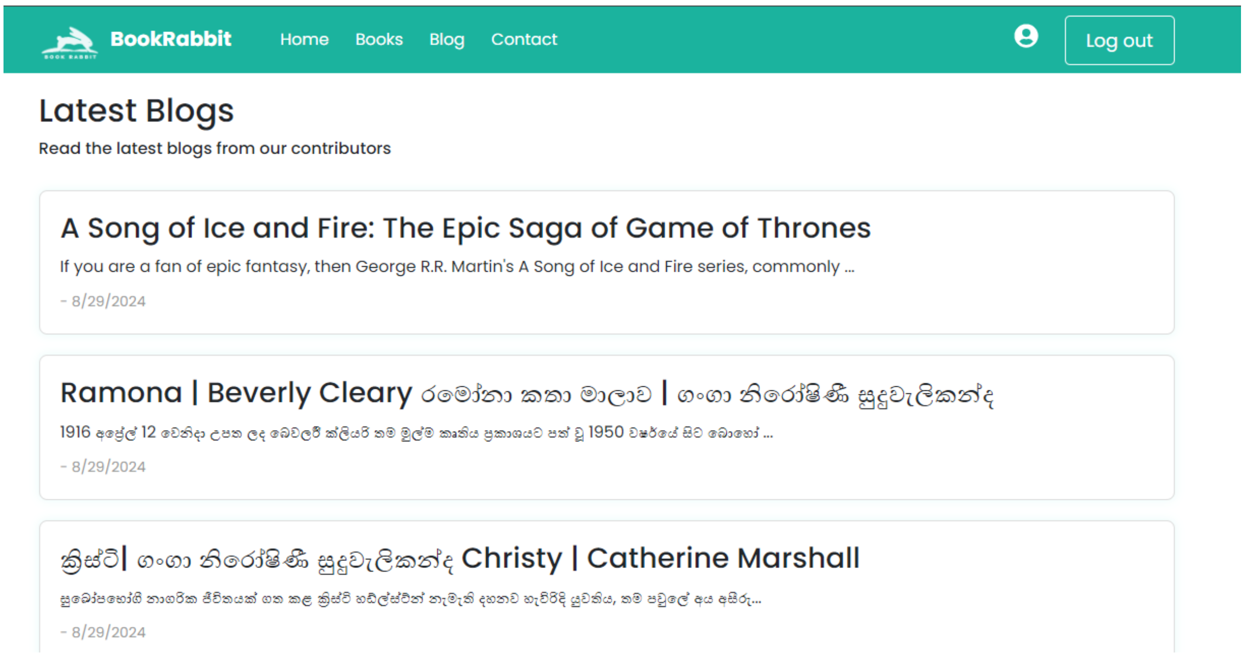
දඟකාර නුවෝ – නුවෝ ට බලු පැටියෙක්
Nuwan Thotawaththa

Friends with Secrets
Christine Gunderson

Natural Selection: A Short Story
Elin Hilderbrand

The Bookstore Wedding
Alice Hoffman

Latest Blogs section



Latest Blogs

Read the latest blogs from our contributors

A Song of Ice and Fire: The Epic Saga of Game of Thrones

If you are a fan of epic fantasy, then George R.R. Martin's A Song of Ice and Fire series, commonly ...

- 8/29/2024

Ramona | Beverly Cleary රමෝනා කතා මාලාව | ගංගා නිරෝෂිණී සුදුවැලිකන්ද

1916 අප්‍රේල් 12 වෙනිදා උපත ලද බෙවලී ක්ලියරි තම මුල්ම කෘතිය ප්‍රකාශයට පත් වූ 1950 වර්ෂයේ සිට බොහෝ ...

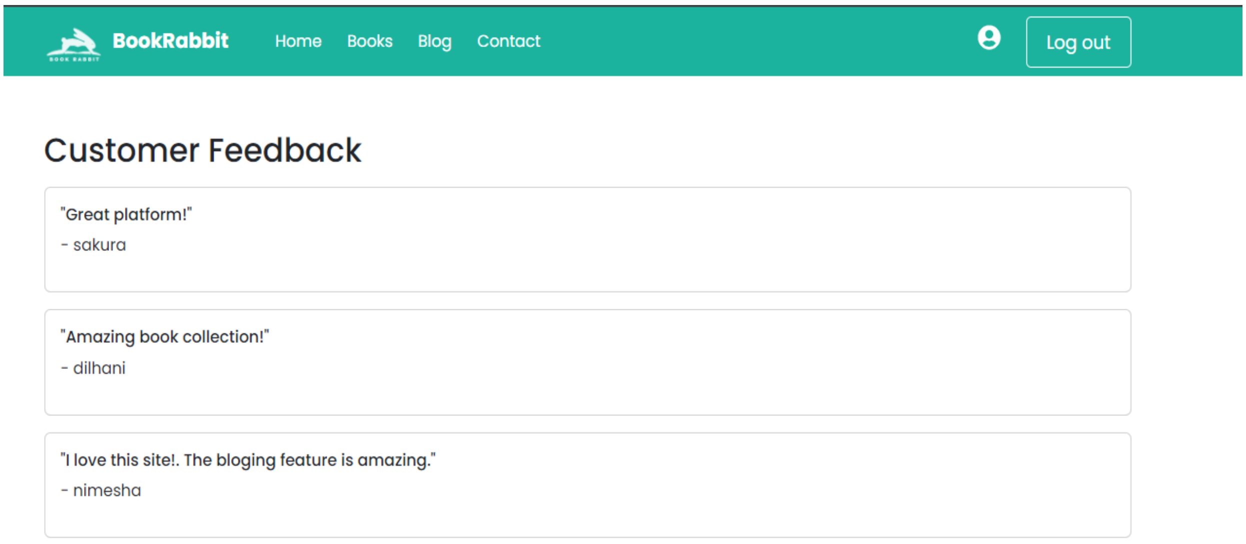
- 8/29/2024

ක්‍රිස්ටි| ගංගා නිරෝෂිණී සුදුවැලිකන්ද Christy | Catherine Marshall

සුඛෝපභෝගී නාගරික ජීවිතයක් ගත කළ ක්‍රිස්ටි හඩ්ල්ස්ටන් නැමැති දහනව හැවිරිදි යුවතිය, තම පවුලේ අය අසිරු...

- 8/29/2024

Customer Feerback section



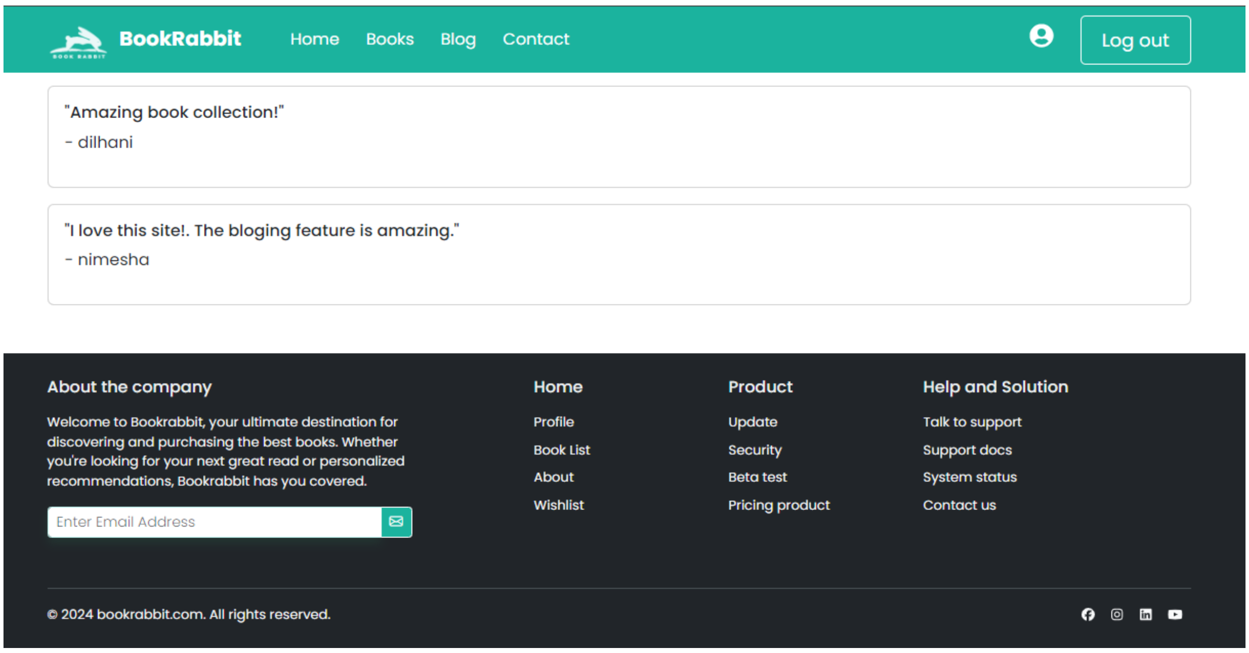
Customer Feedback

"Great platform!"
- sakura

"Amazing book collection!"
- dilhani

"I love this site!. The bloging feature is amazing."
- nimesha

Footer Section



"Amazing book collection!"
- dilhani

"I love this site!. The bloging feature is amazing."
- nimesha

About the company

Welcome to Bookrabbit, your ultimate destination for discovering and purchasing the best books. Whether you're looking for your next great read or personalized recommendations, Bookrabbit has you covered.

Enter Email Address

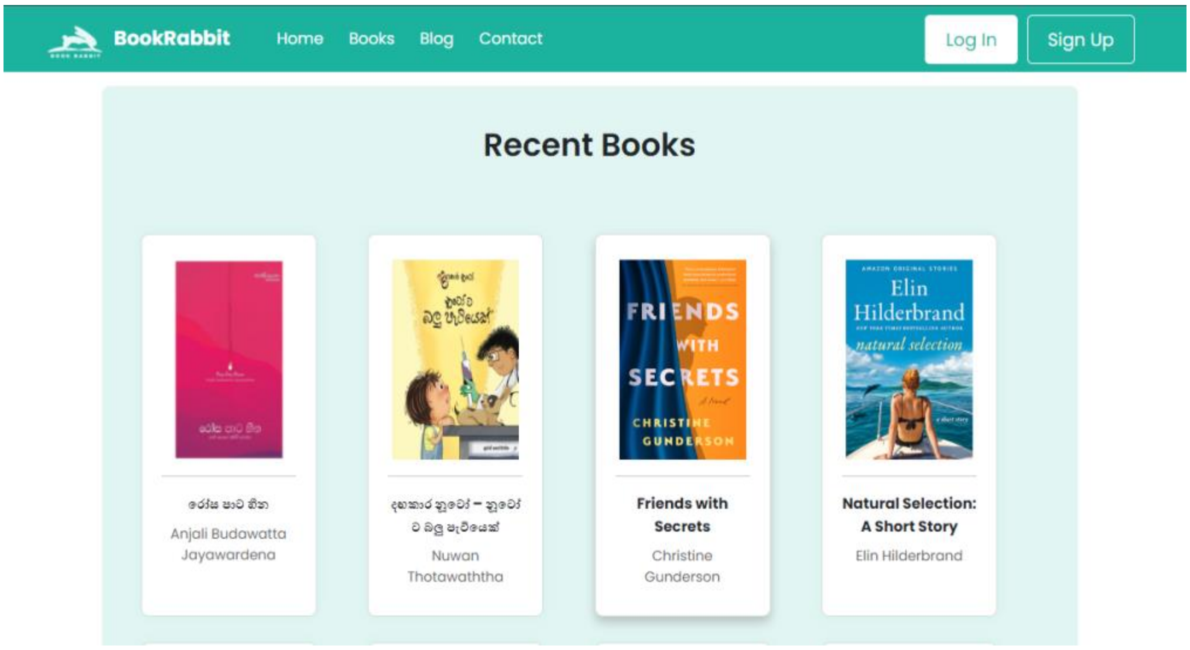
Home
- Profile
- Book List
- About
- Wishlist

Product
- Update
- Security
- Beta test
- Pricing product

Help and Solution
- Talk to support
- Support docs
- System status
- Contact us

© 2024 bookrabbit.com. All rights reserved.

Recent Books


BookRabbit Home Books Blog Contact Log In Sign Up

Recent Books

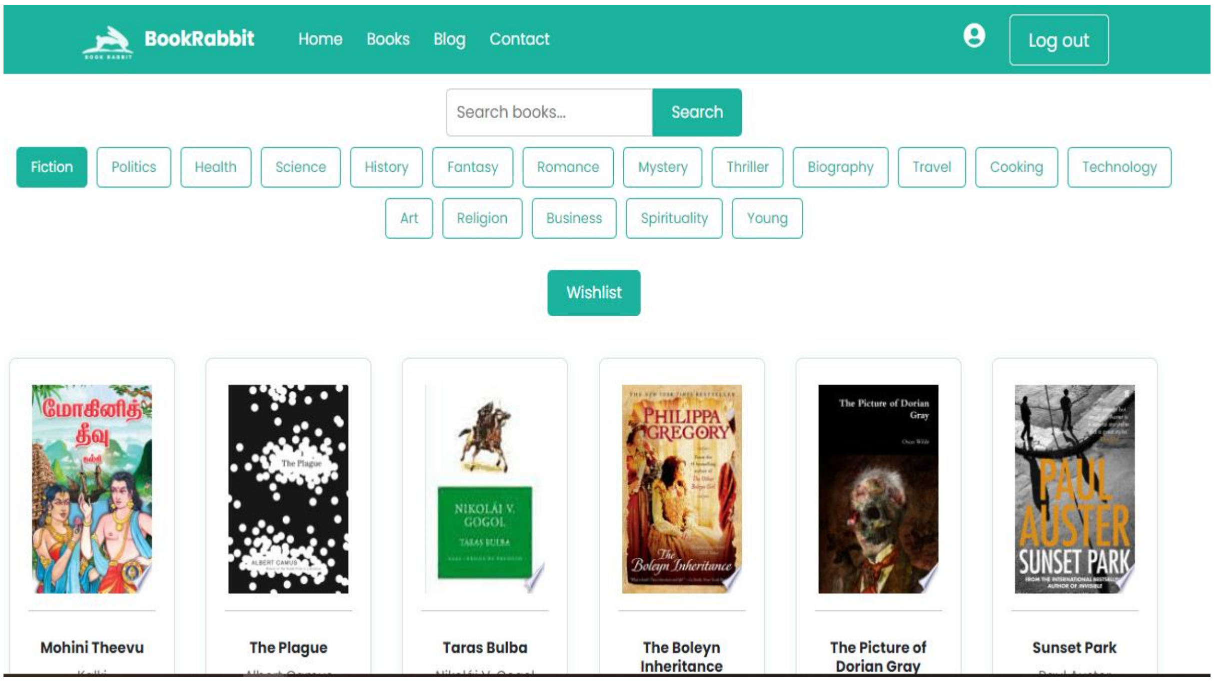
රෝස පාට හීන
Anjali Budawatta Jayawardena

දඟකාර නුවෝ – නුවෝ ව බලු පැටියෙක්
Nuwan Thotawaththa

Friends with Secrets
Christine Gunderson

Natural Selection: A Short Story
Elin Hilderbrand

All Books Page


BookRabbit Home Books Blog Contact Log out

Search books... Search

Fiction Politics Health Science History Fantasy Romance Mystery Thriller Biography Travel Cooking Technology Art Religion Business Spirituality Young

Wishlist

Mohini Theevu

The Plague

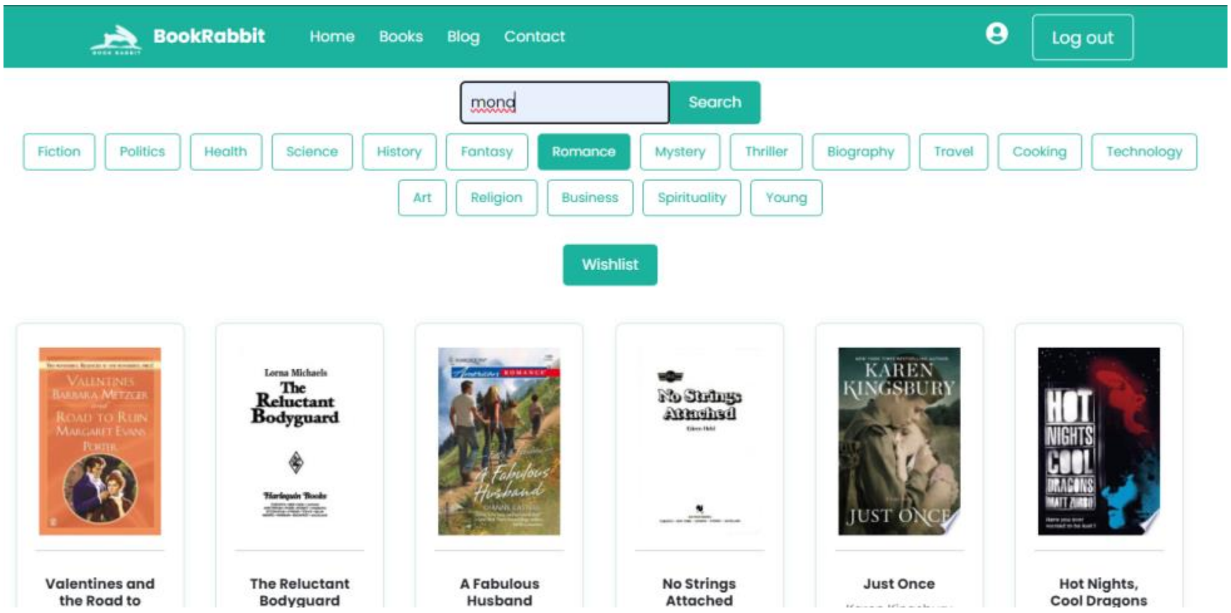
Taras Bulba

The Boleyn Inheritance

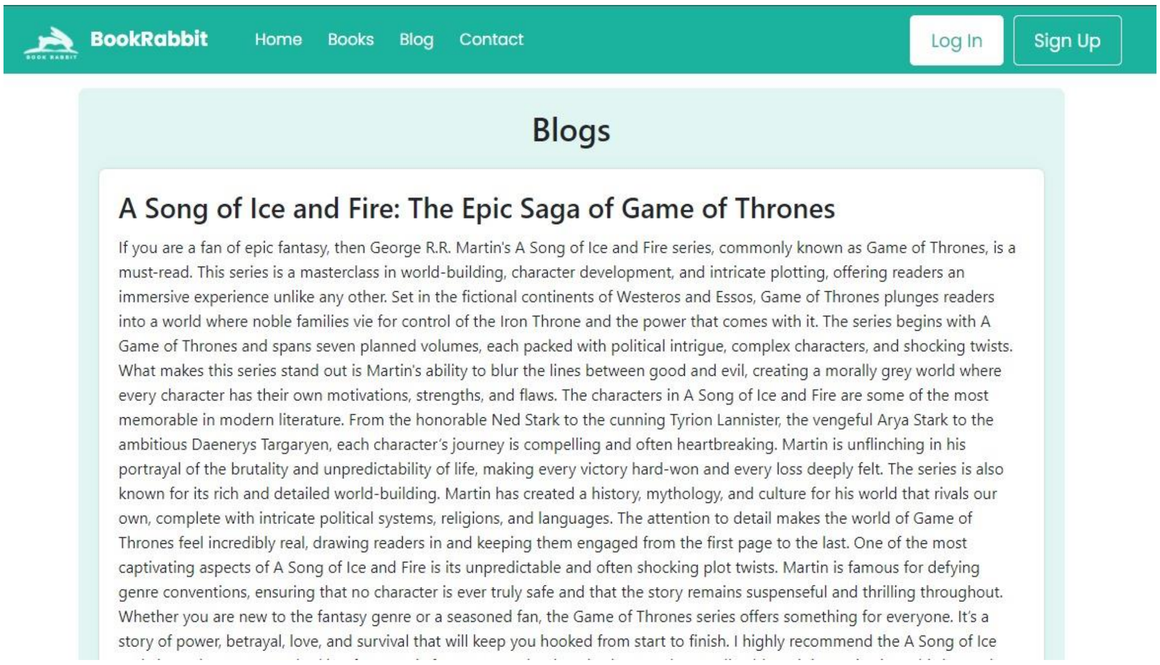
The Picture of Dorian Gray

Sunset Park

Search



Blog Screen

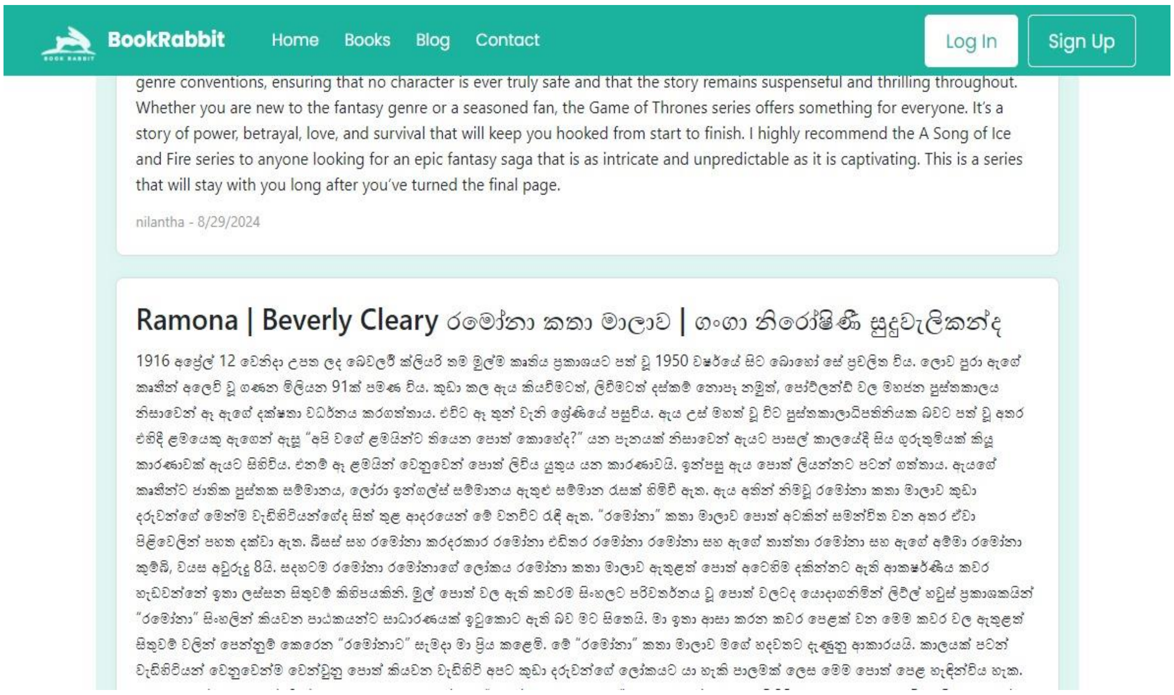


Blogs

A Song of Ice and Fire: The Epic Saga of Game of Thrones

If you are a fan of epic fantasy, then George R.R. Martin's A Song of Ice and Fire series, commonly known as Game of Thrones, is a must-read. This series is a masterclass in world-building, character development, and intricate plotting, offering readers an immersive experience unlike any other. Set in the fictional continents of Westeros and Essos, Game of Thrones plunges readers into a world where noble families vie for control of the Iron Throne and the power that comes with it. The series begins with A Game of Thrones and spans seven planned volumes, each packed with political intrigue, complex characters, and shocking twists. What makes this series stand out is Martin's ability to blur the lines between good and evil, creating a morally grey world where every character has their own motivations, strengths, and flaws. The characters in A Song of Ice and Fire are some of the most memorable in modern literature. From the honorable Ned Stark to the cunning Tyrion Lannister, the vengeful Arya Stark to the ambitious Daenerys Targaryen, each character's journey is compelling and often heartbreaking. Martin is unflinching in his portrayal of the brutality and unpredictability of life, making every victory hard-won and every loss deeply felt. The series is also known for its rich and detailed world-building. Martin has created a history, mythology, and culture for his world that rivals our own, complete with intricate political systems, religions, and languages. The attention to detail makes the world of Game of Thrones feel incredibly real, drawing readers in and keeping them engaged from the first page to the last. One of the most captivating aspects of A Song of Ice and Fire is its unpredictable and often shocking plot twists. Martin is famous for defying genre conventions, ensuring that no character is ever truly safe and that the story remains suspenseful and thrilling throughout. Whether you are new to the fantasy genre or a seasoned fan, the Game of Thrones series offers something for everyone. It's a story of power, betrayal, love, and survival that will keep you hooked from start to finish. I highly recommend the A Song of Ice

Blog Screen



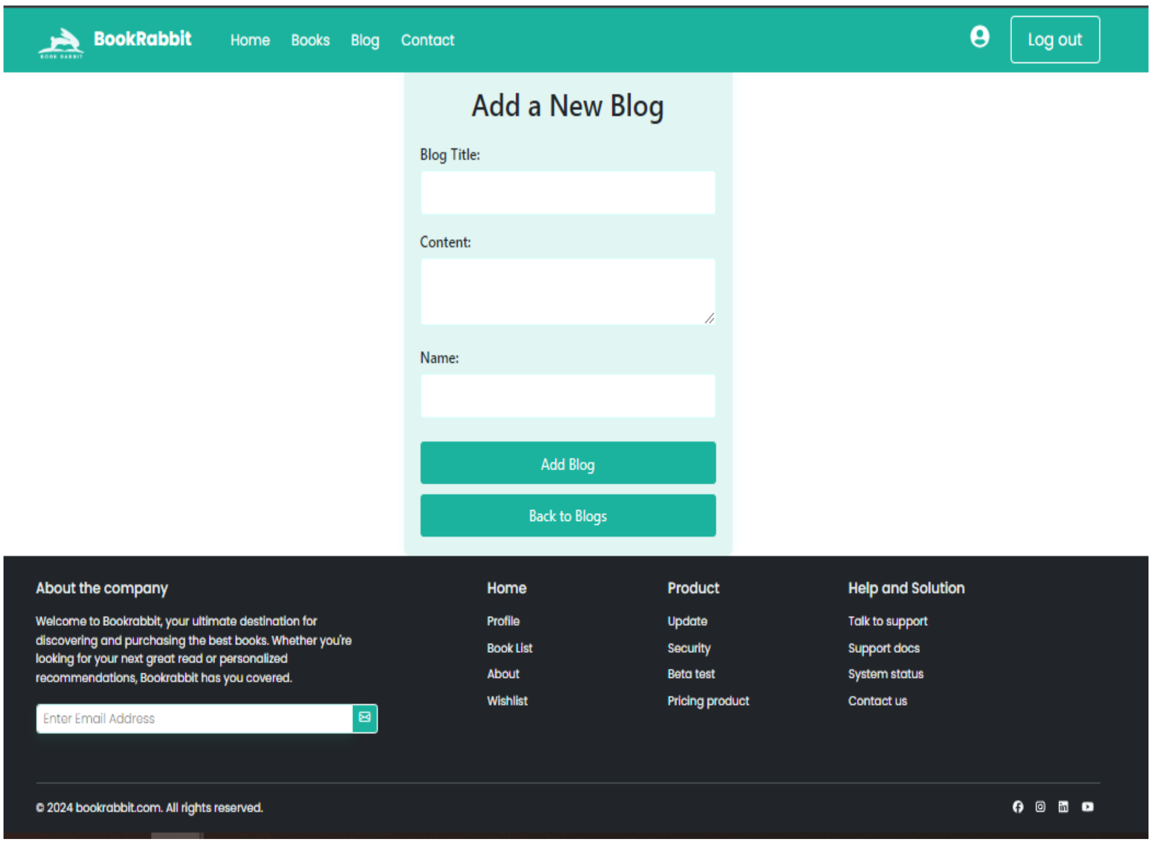
genre conventions, ensuring that no character is ever truly safe and that the story remains suspenseful and thrilling throughout. Whether you are new to the fantasy genre or a seasoned fan, the Game of Thrones series offers something for everyone. It's a story of power, betrayal, love, and survival that will keep you hooked from start to finish. I highly recommend the A Song of Ice and Fire series to anyone looking for an epic fantasy saga that is as intricate and unpredictable as it is captivating. This is a series that will stay with you long after you've turned the final page.

nilantha - 8/29/2024

Ramona | Beverly Cleary රමෝනා කතා මාලාව | ගංගා නිරෝෂිණී සුදුවැලිකන්ද

1916 අප්‍රේල් 12 වෙනිදා උපත ලද බෙවලී ක්ලියරි තම මුල්ම කෘතිය ප්‍රකාශයට පත් වූ 1950 වර්ෂයේ සිට බොහෝ සේ ප්‍රචලිත විය. ලොව පුරා ඇගේ කෘතීන් අලෙවි වූ ගණන මිලියන 91ක් පමණ විය. කුඩා කල ඇය කියවීමටත්, ලිවීමටත් දක්ෂ නොවූ නමුත්, පෝට්ලන්ඩ් වල මහජන පුස්තකාලය නිසාවෙන් ඇ ඇගේ දක්ෂතා වර්ධනය කරගත්තාය. එවිට ඇ තුන් වැනි ශ්‍රේණියේ පසුවිය. ඇය උස් මහත් වූ විට පුස්තකාලාධිපතිනියක බවට පත් වූ අතර එහිදී ළමයෙකු ඇගෙන් ඇසූ "අපි වගේ ළමයින්ට තියෙන පොත් කොහේද?" යන පැනයක් නිසාවෙන් ඇයට පාසල් කාලයේදී සිය ගුරුතුමියක් කියූ කාරණාවක් ඇයට සිහිවිය. එනම් ඇ ළමයින් වෙනුවෙන් පොත් ලිවිය යුතුය යන කාරණාවයි. ඉන්පසු ඇය පොත් ලියන්නට පටන් ගත්තාය. ඇයගේ කෘතීන්ට ජාතික පුස්තක සම්මානය, ලෝරා ඉන්ගල්ස් සම්මානය ඇතුළු සම්මාන රැසක් හිමිවී ඇත. ඇය අතින් නිමවූ රමෝනා කතා මාලාව කුඩා දරුවන්ගේ මෙන්ම වැඩිහිටියන්ගේද සිත් තුළ ආදරයෙන් මේ වනවිට රැඳී ඇත. "රමෝනා" කතා මාලාව පොත් අටකින් සමන්විත වන අතර ඒවා පිළිවෙලින් පහත දක්වා ඇත. බීසස් සහ රමෝනා කරදරකාර රමෝනා එඩිතර රමෝනා රමෝනා සහ ඇගේ තාත්තා රමෝනා සහ ඇගේ අම්මා රමෝනා කුම්බි, වයස අවුරුදු 8යි. සදහටම රමෝනා රමෝනාගේ ලෝකය රමෝනා කතා මාලාව ඇතුළත් පොත් අටෙහිම දකින්නට ඇති ආකර්ෂණීය කවර හැඩවන්නේ ඉතා ලස්සන සිතුවම් කිහිපයකිනි. මුල් පොත් වල ඇති කවරම සිංහලට පරිවර්තනය වූ පොත් වලටද යොදාගනිමින් ලිට්ල් හවුස් ප්‍රකාශකයින් "රමෝනා" සිංහලින් කියවන පාඨකයන්ට සාධාරණයක් ඉටුකොට ඇති බව මට සිතෙයි. මා ඉතා ආසා කරන කවර පෙළක් වන මෙම කවර වල ඇතුළත් සිතුවම් වලින් පෙන්නුම් කෙරෙන "රමෝනාට" සැමදා මා ප්‍රිය කළෙමි. මේ "රමෝනා" කතා මාලාව මගේ හදවතට දැණුනු ආකාරයයි. කාලයක් පටන් වැඩිහිටියන් වෙනුවෙන්ම වෙන්වුනු පොත් කියවන වැඩිහිටි අපට කුඩා දරුවන්ගේ ලෝකයට යා හැකි පාලමක් ලෙස මෙම පොත් පෙළ හැඳින්විය හැක.

Add Blog Page



Add a New Blog

Blog Title:

Content:

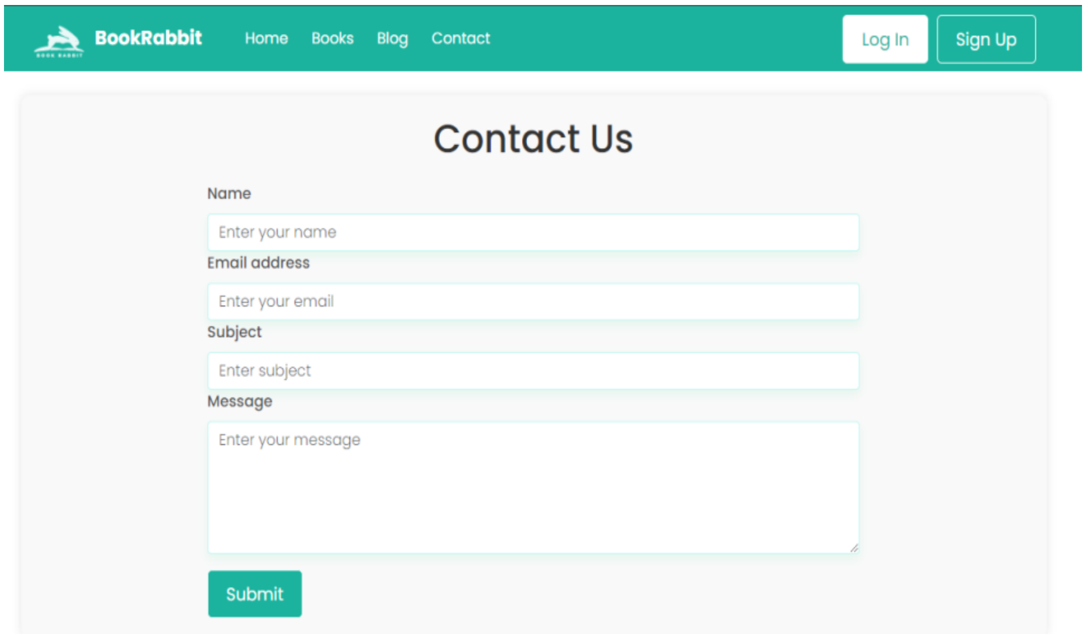
Name:

About the company

Welcome to Bookrabbit, your ultimate destination for discovering and purchasing the best books. Whether you're looking for your next great read or personalized recommendations, Bookrabbit has you covered.

© 2024 bookrabbit.com. All rights reserved.

Contact



Contact Us

Name

Enter your name

Email address

Enter your email

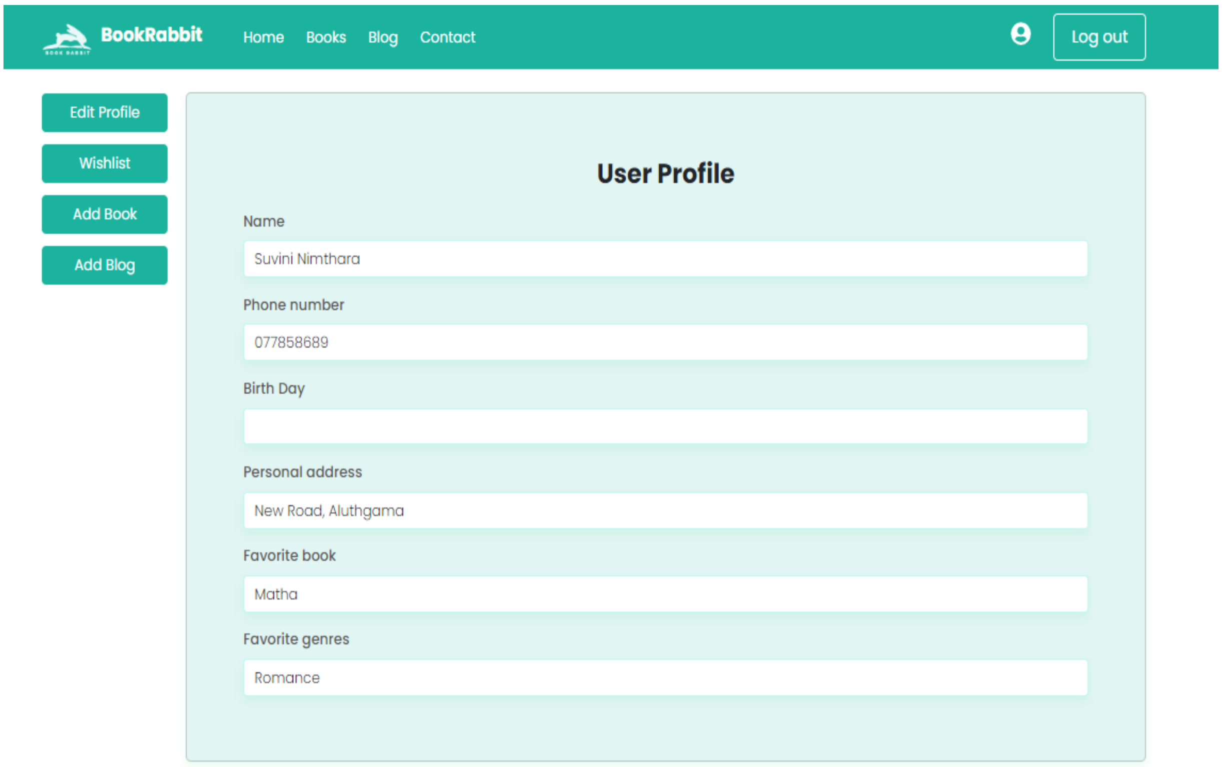
Subject

Enter subject

Message

Enter your message

User Profile



User Profile

Name

Suvini Nimthara

Phone number

077858689

Birth Day

Personal address

New Road, Aluthgama

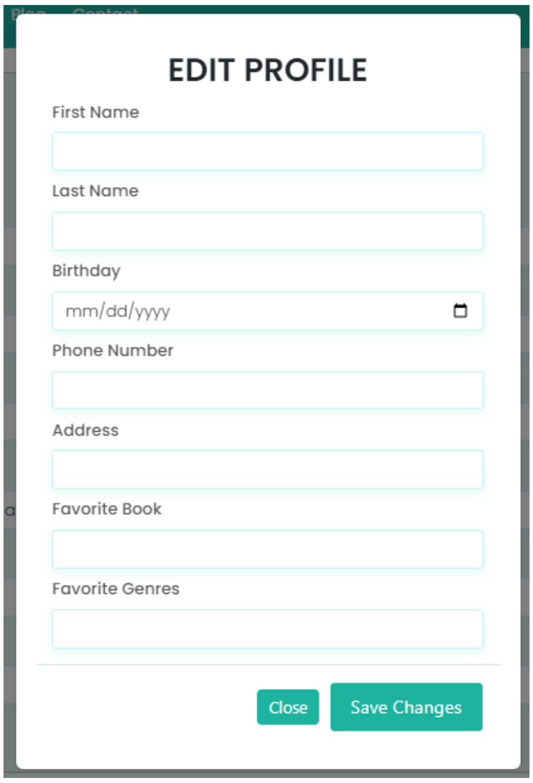
Favorite book

Matha

Favorite genres

Romance

Edit Profile


EDIT PROFILE

First Name

Last Name

Birthday

mm/dd/yyyy

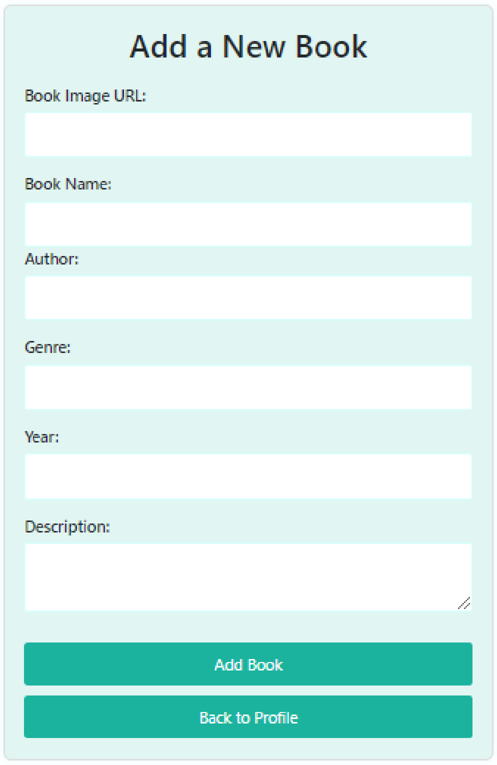
Phone Number

Address

Favorite Book

Favorite Genres

Add Book


Add a New Book

Book Image URL:

Book Name:

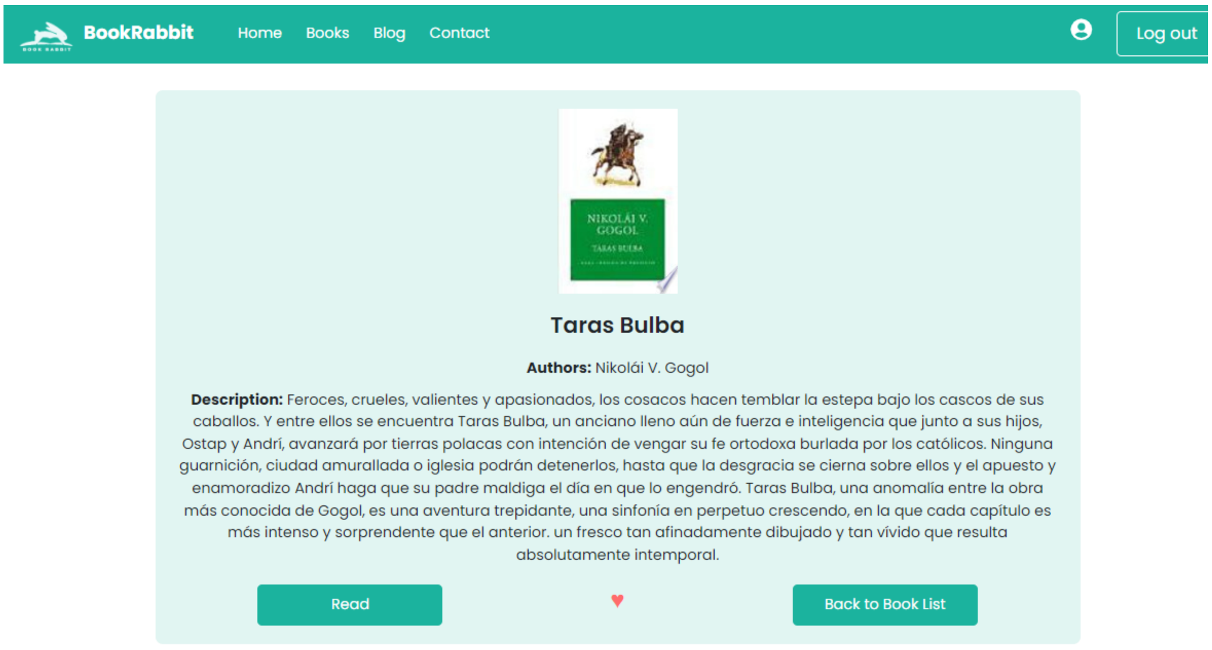
Author:

Genre:

Year:

Description:

Book Details



Taras Bulba

Authors: Nikolái V. Gogol

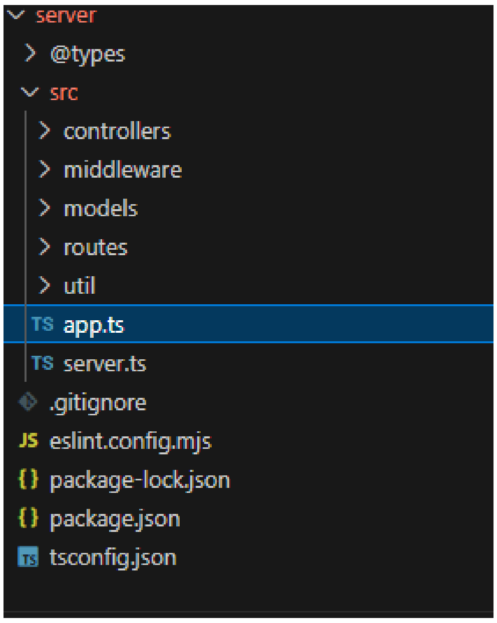
Description: Feroces, crueles, valientes y apasionados, los cosacos hacen temblar la estepa bajo los cascos de sus caballos. Y entre ellos se encuentra Taras Bulba, un anciano lleno aún de fuerza e inteligencia que junto a sus hijos, Ostap y Andrí, avanzará por tierras polacas con intención de vengar su fe ortodoxa burlada por los católicos. Ninguna guarnición, ciudad amurallada o iglesia podrán detenerlos, hasta que la desgracia se cierna sobre ellos y el apuesto y enamoradizo Andrí haga que su padre maldiga el día en que lo engendró. Taras Bulba, una anomalía entre la obra más conocida de Gogol, es una aventura trepidante, una sinfonía en perpetuo crescendo, en la que cada capítulo es más intenso y sorprendente que el anterior. un fresco tan afinadamente dibujado y tan vívido que resulta absolutamente intemporal.

Database Structure

This is the last section in overview of the system. In here we also explain API implementation and Database connections.

Database Connection

The data base connection in our project **Book Rabbit** manage through the “ app.ts “ file which is located in “src” directory in server folder. The connection Mongo DB settled using Mongoose library, a popular ODM (Object Data Modeling) library for Mongo DB and Node.js.



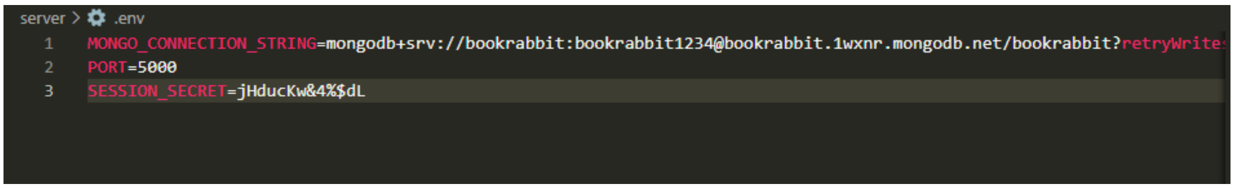
```
server
  @types
  src
    controllers
    middleware
    models
    routes
    util
    app.ts
    server.ts
  .gitignore
  eslint.config.mjs
  package-lock.json
  package.json
  tsconfig.json
```

“Sever.ts “ aslo connect Mongo DB related to project.

Using this code connects the application to the Mongo DB database using a connection string stored in an environment variable (MONGO_URI).

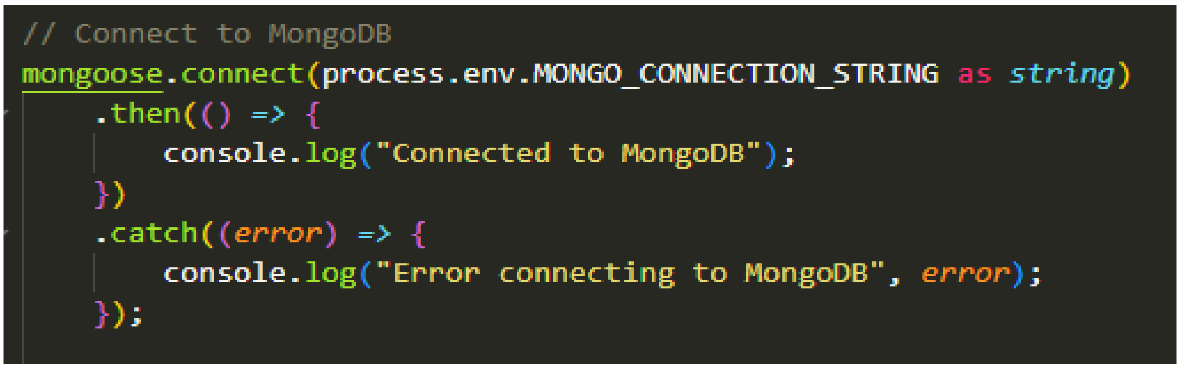
Error handling is implemented to ensure that if the connection fails, the application exits with an error message.

Here is the relevant code snippet.

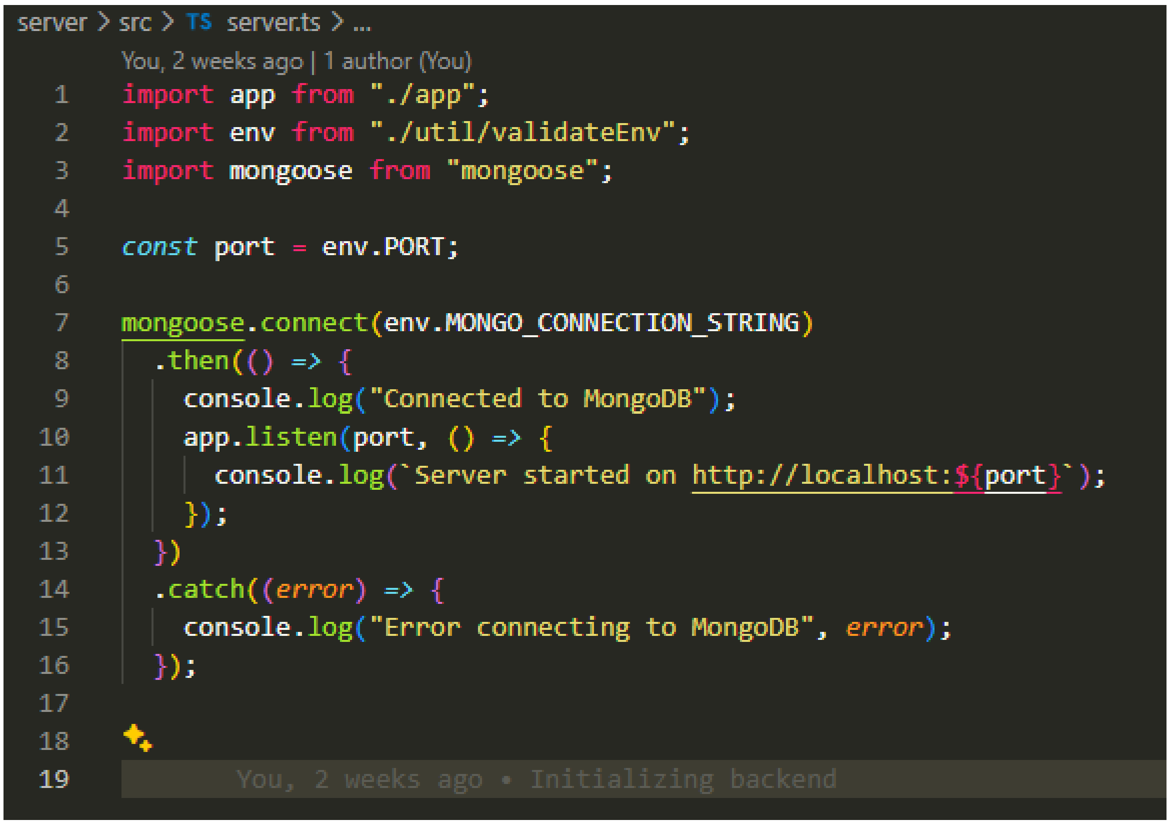


server > .env

```
MONGO_CONNECTION_STRING=mongodb+srv://bookrabbit:bookrabbit1234@bookrabbit.1wxnr.mongodb.net/bookrabbit?retryWrite
PORT=5000
SESSION_SECRET=jHducKw&4%$dL
```



```
// Connect to MongoDB
mongoose.connect(process.env.MONGO_CONNECTION_STRING as string)
    .then(() => {
        console.log("Connected to MongoDB");
    })
    .catch((error) => {
        console.log("Error connecting to MongoDB", error);
    });
```


server > src > TS server.ts > ...

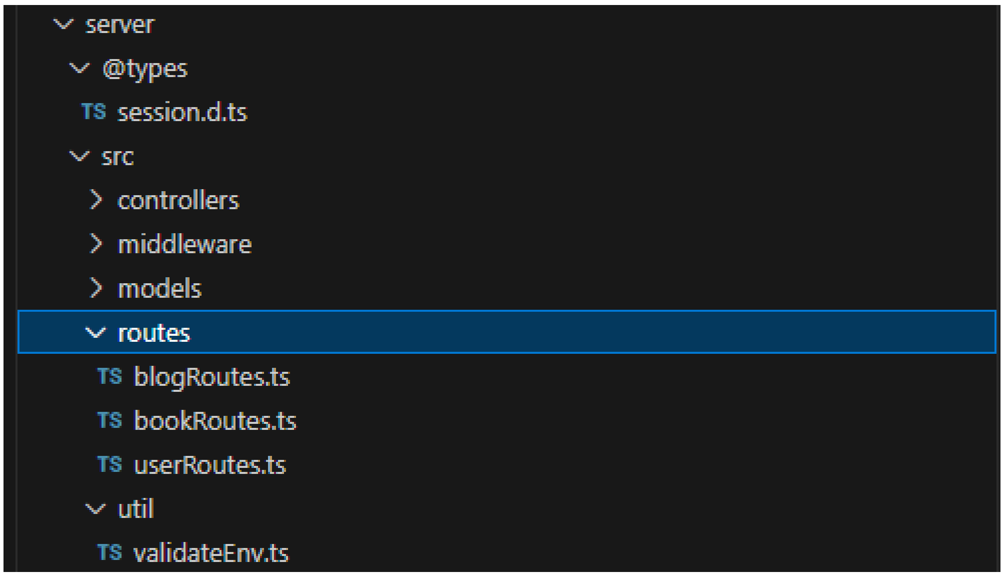
You, 2 weeks ago | 1 author (You)

```ts
import app from "./app";
import env from "./util/validateEnv";
import mongoose from "mongoose";

const port = env.PORT;

mongoose.connect(env.MONGO_CONNECTION_STRING)
  .then(() => {
    console.log("Connected to MongoDB");
    app.listen(port, () => {
      console.log(`Server started on http://localhost:${port}`);
    });
  })
  .catch((error) => {
    console.log("Error connecting to MongoDB", error);
  });


        You, 2 weeks ago • Initializing backend
```

<u>Routers</u>

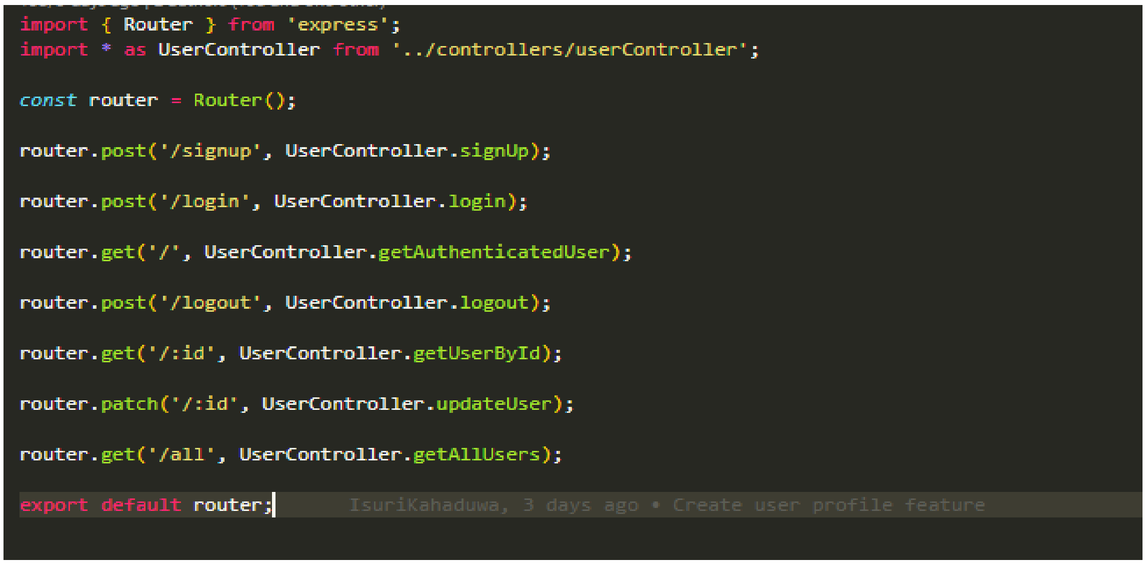
The routers in our project are organized under the routes directory within the “src” folder. These routers define the **endpoints** that the client-side application can interact with, directing HTTP requests to the appropriate controllers.

Here is the snapshot of the relevant directory.



```
server
  @types
    session.d.ts
  src
    controllers
    middleware
    models
    routes
      blogRoutes.ts
      bookRoutes.ts
      userRoutes.ts
    util
      validateEnv.ts
```

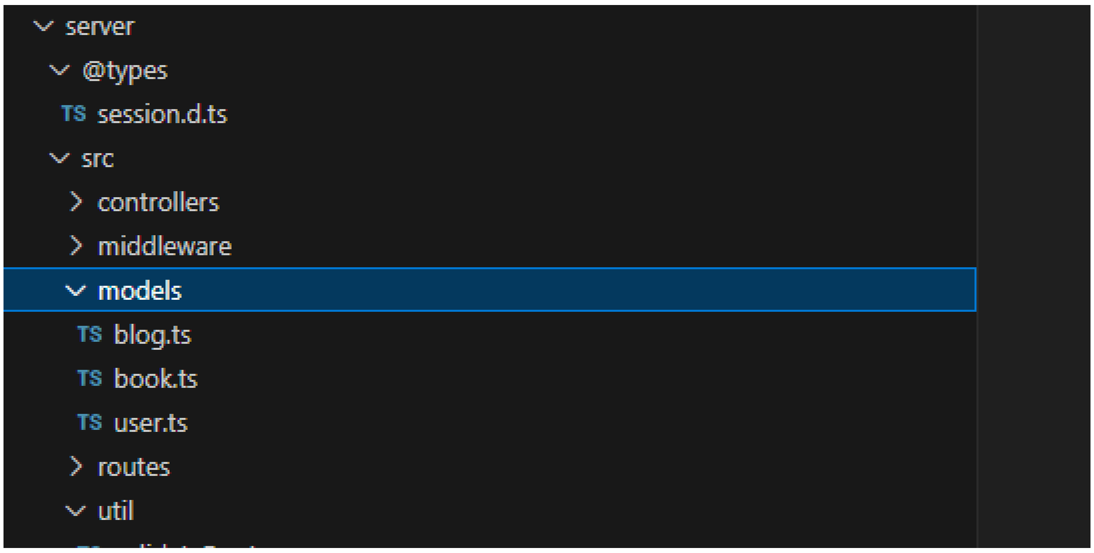
The route file typically corresponds to a specific resource, such as users, books, or reviews. The routes are connected to controller functions. The following code snippet prove it.



```
import { Router } from 'express';
import * as UserController from '../controllers/userController';

const router = Router();

router.post('/signup', UserController.signUp);

router.post('/login', UserController.login);

router.get('/', UserController.getAuthenticatedUser);

router.post('/logout', UserController.logout);

router.get('/:id', UserController.getUserById);

router.patch('/:id', UserController.updateUser);

router.get('/all', UserController.getAllUsers);

export default router;        IsuriKahaduwa, 3 days ago • Create user profile feature
```

Models

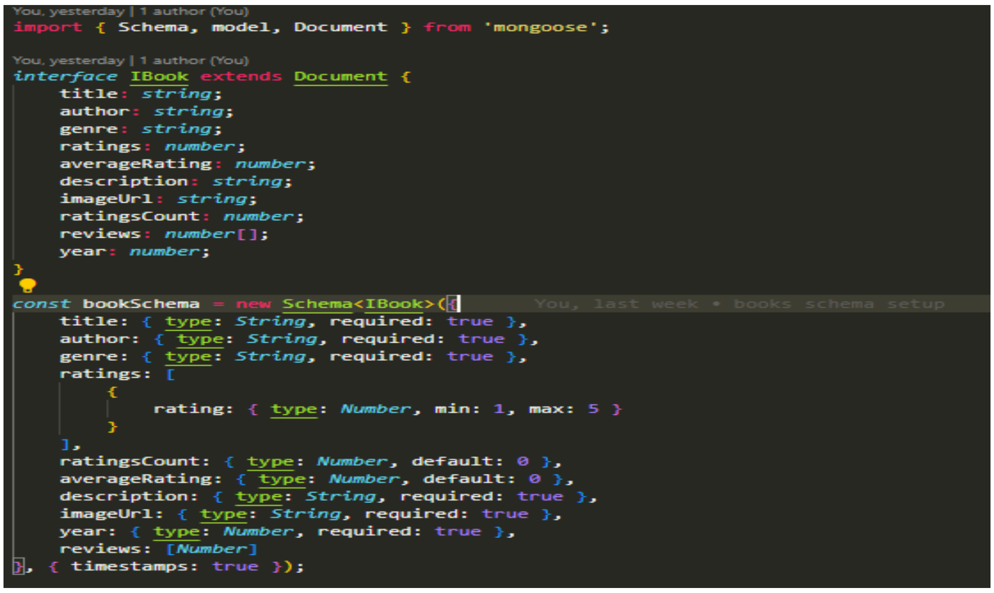
The model files inside the server directory under the “src “contains the models entire project. This defines the structure of the data within your Mongo DB collections.



```
server
  @types
    session.d.ts
  src
    controllers
    middleware
    models
      blog.ts
      book.ts
      user.ts
    routes
    util
```

Here we creating three files which are named as **Blog .ts /Book .ts / User .ts** through these files we can get user information, information about books and blog details like reviews as well.

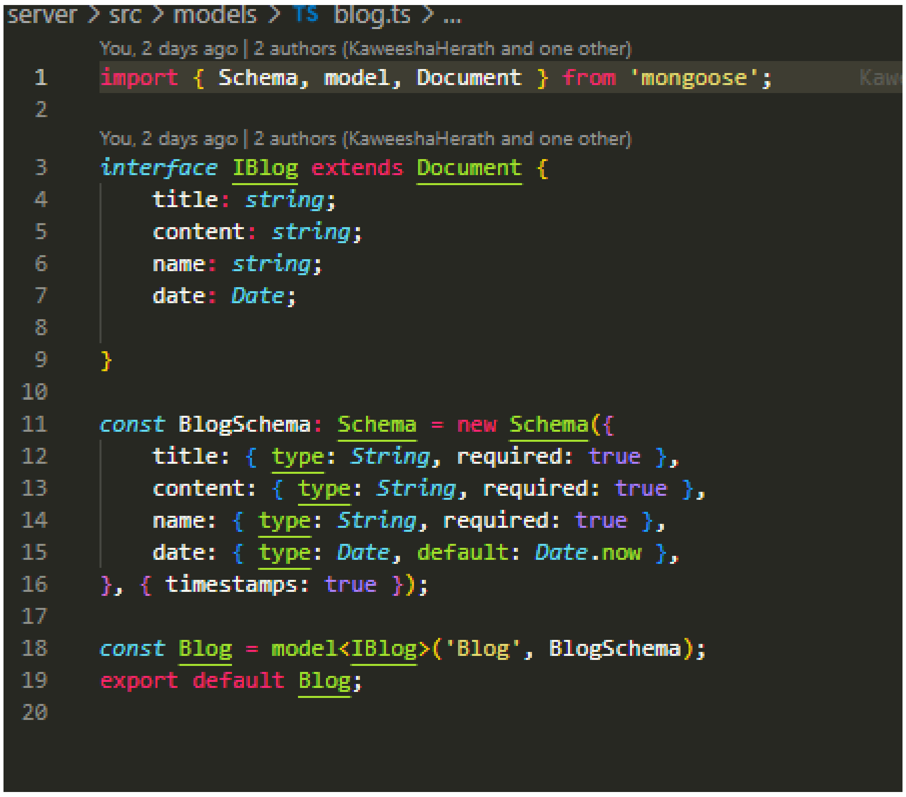
Here we provide the relevant code snippet about Book , Blog and User models respectively.



```
You, yesterday | 1 author (You)
import { Schema, model, Document } from 'mongoose';

You, yesterday | 1 author (You)
interface IBook extends Document {
    title: string;
    author: string;
    genre: string;
    ratings: number;
    averageRating: number;
    description: string;
    imageUrl: string;
    ratingsCount: number;
    reviews: number[];
    year: number;
}

const bookSchema = new Schema<IBook>({        You, last week • books schema setup
    title: { type: String, required: true },
    author: { type: String, required: true },
    genre: { type: String, required: true },
    ratings: [
        {
            rating: { type: Number, min: 1, max: 5 }
        }
    ],
    ratingsCount: { type: Number, default: 0 },
    averageRating: { type: Number, default: 0 },
    description: { type: String, required: true },
    imageUrl: { type: String, required: true },
    year: { type: Number, required: true },
    reviews: [Number]
}, { timestamps: true });
```

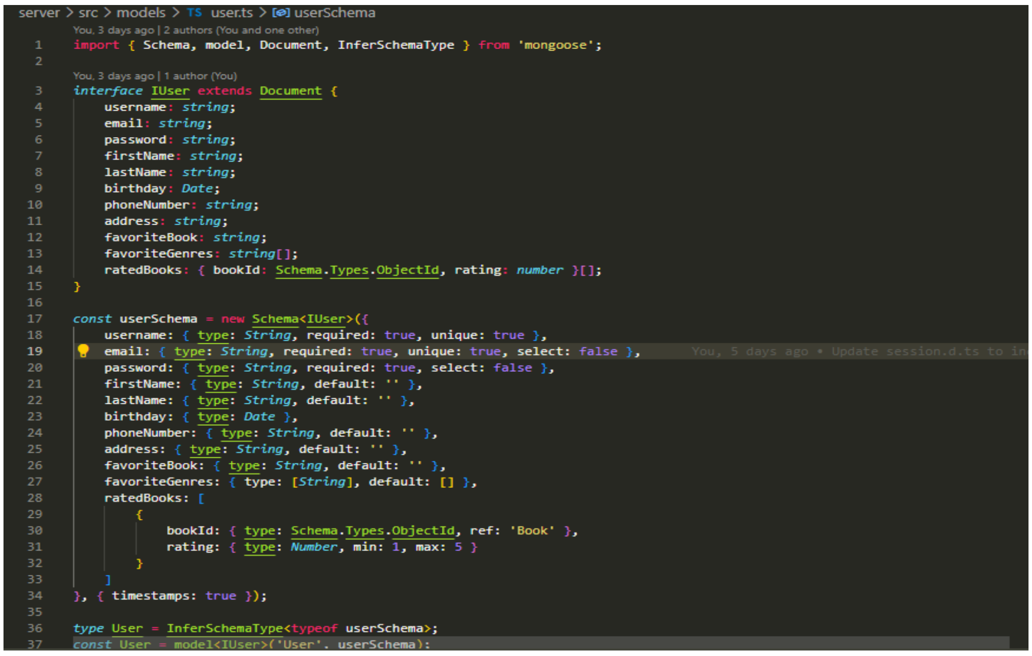
Book.ts

Blog.ts



```
server > src > models > TS blog.ts > ...
You, 2 days ago | 2 authors (KaweeshaHerath and one other)
import { Schema, model, Document } from 'mongoose';

You, 2 days ago | 2 authors (KaweeshaHerath and one other)
interface IBlog extends Document {
    title: string;
    content: string;
    name: string;
    date: Date;

}

const BlogSchema: Schema = new Schema({
    title: { type: String, required: true },
    content: { type: String, required: true },
    name: { type: String, required: true },
    date: { type: Date, default: Date.now },
}, { timestamps: true });

const Blog = model<IBlog>('Blog', BlogSchema);
export default Blog;
```

User .ts

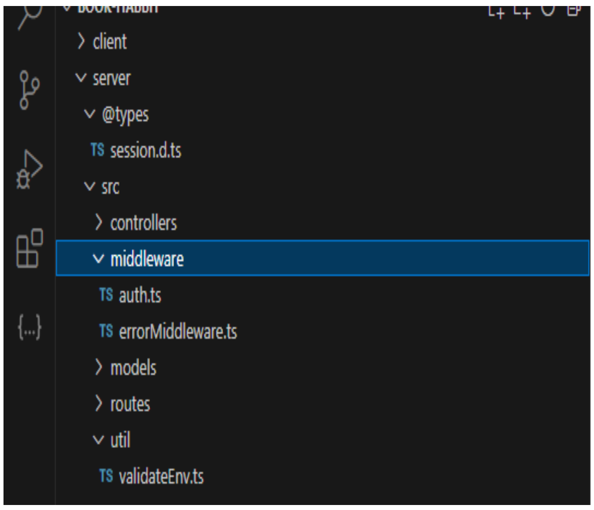


```
server > src > models > TS user.ts > userSchema
You, 3 days ago | 2 authors (You and one other)
import { Schema, model, Document, InferSchemaType } from 'mongoose';

You, 3 days ago | 1 author (You)
interface IUser extends Document {
    username: string;
    email: string;
    password: string;
    firstName: string;
    lastName: string;
    birthday: Date;
    phoneNumber: string;
    address: string;
    favoriteBook: string;
    favoriteGenres: string[];
    ratedBooks: { bookId: Schema.Types.ObjectId, rating: number }[];
}

const userSchema = new Schema<IUser>({
    username: { type: String, required: true, unique: true },
    email: { type: String, required: true, unique: true, select: false },      You, 5 days ago • Update session.d.ts to in
    password: { type: String, required: true, select: false },
    firstName: { type: String, default: '' },
    lastName: { type: String, default: '' },
    birthday: { type: Date },
    phoneNumber: { type: String, default: '' },
    address: { type: String, default: '' },
    favoriteBook: { type: String, default: '' },
    favoriteGenres: { type: [String], default: [] },
    ratedBooks: [
        {
            bookId: { type: Schema.Types.ObjectId, ref: 'Book' },
            rating: { type: Number, min: 1, max: 5 }
        }
    ]
}, { timestamps: true });

type User = InferSchemaType<typeof userSchema>;
const User = model<IUser>('User', userSchema);
```

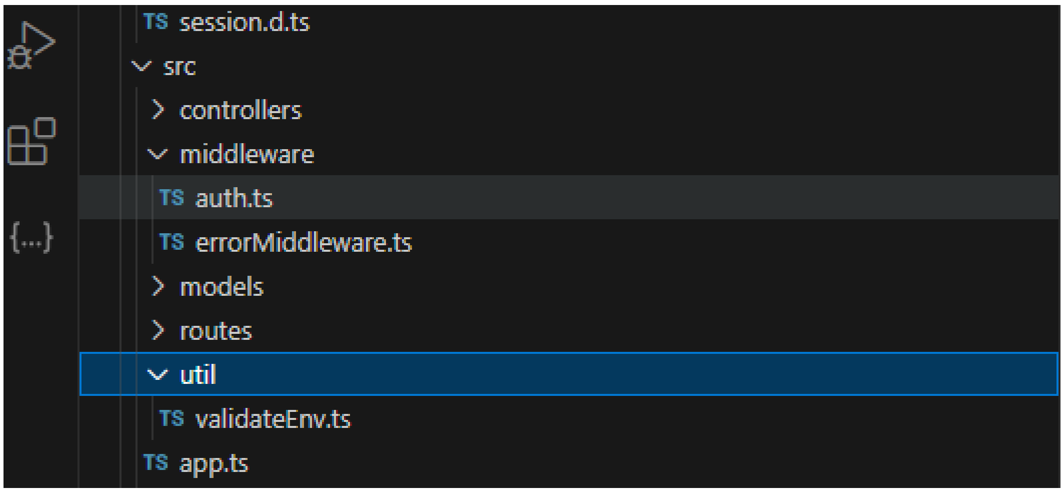
Middleware and Util

Our project structure indicates the presence of middleware and utility functions, which are located under middleware and util folders in server directory, respectively.

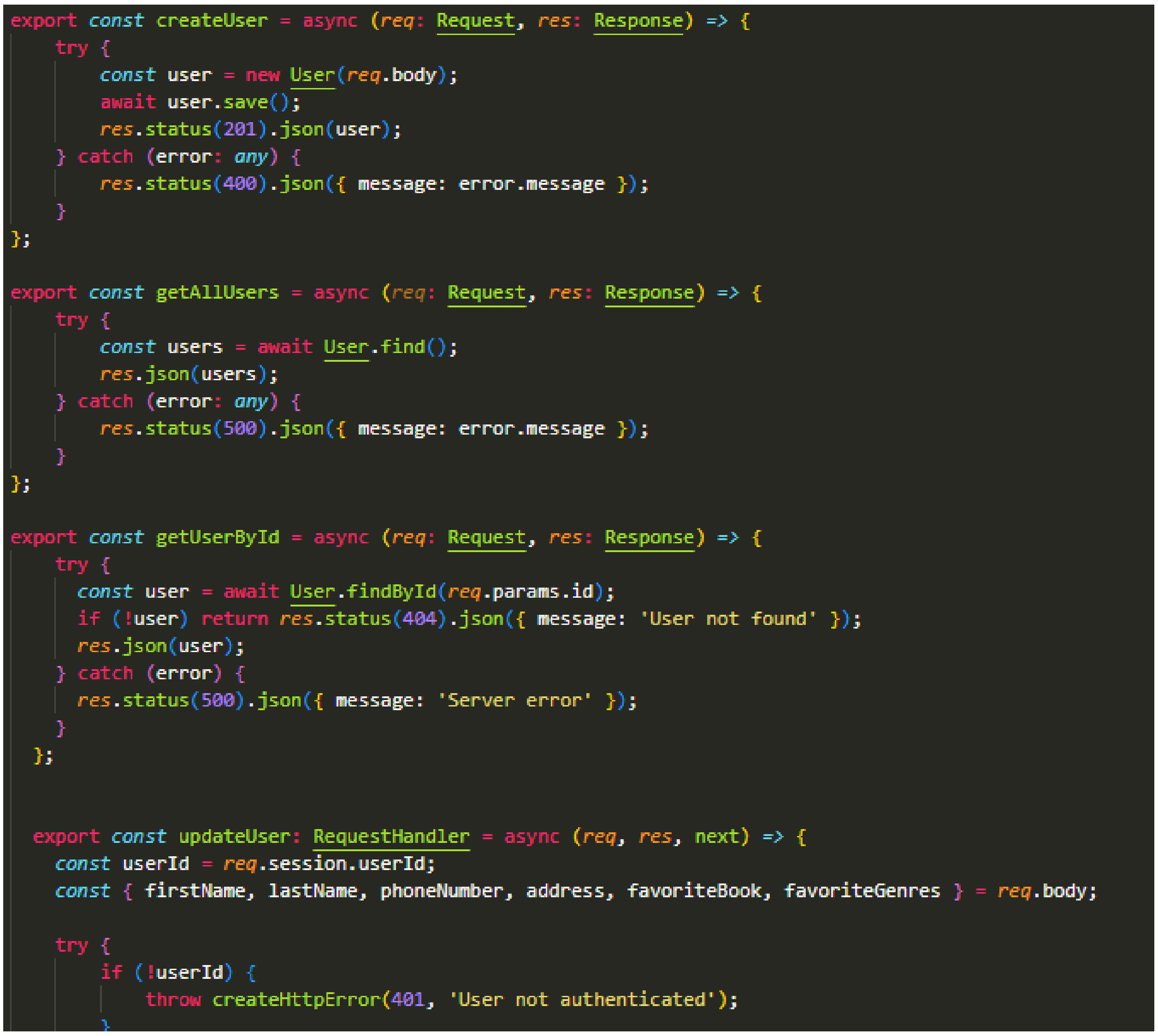
These help in handling authentication, error handling, and other reusable functions that support the main application logic.



```
> client
∨ server
  ∨ @types
    TS session.d.ts
  ∨ src
    > controllers
    ∨ middleware
      TS auth.ts
      TS errorMiddleware.ts
    > models
    > routes
    ∨ util
      TS validateEnv.ts
```



Here is few code snippet extracted from the entire code process.



```
export const createUser = async (req: Request, res: Response) => {
    try {
        const user = new User(req.body);
        await user.save();
        res.status(201).json(user);
    } catch (error: any) {
        res.status(400).json({ message: error.message });
    }
};

export const getAllUsers = async (req: Request, res: Response) => {
    try {
        const users = await User.find();
        res.json(users);
    } catch (error: any) {
        res.status(500).json({ message: error.message });
    }
};

export const getUserById = async (req: Request, res: Response) => {
    try {
      const user = await User.findById(req.params.id);
      if (!user) return res.status(404).json({ message: 'User not found' });
      res.json(user);
    } catch (error) {
      res.status(500).json({ message: 'Server error' });
    }
  };


  export const updateUser: RequestHandler = async (req, res, next) => {
    const userId = req.session.userId;
    const { firstName, lastName, phoneNumber, address, favoriteBook, favoriteGenres } = req.body;

    try {
        if (!userId) {
            throw createHttpError(401, 'User not authenticated');
        }
```

API Implementation

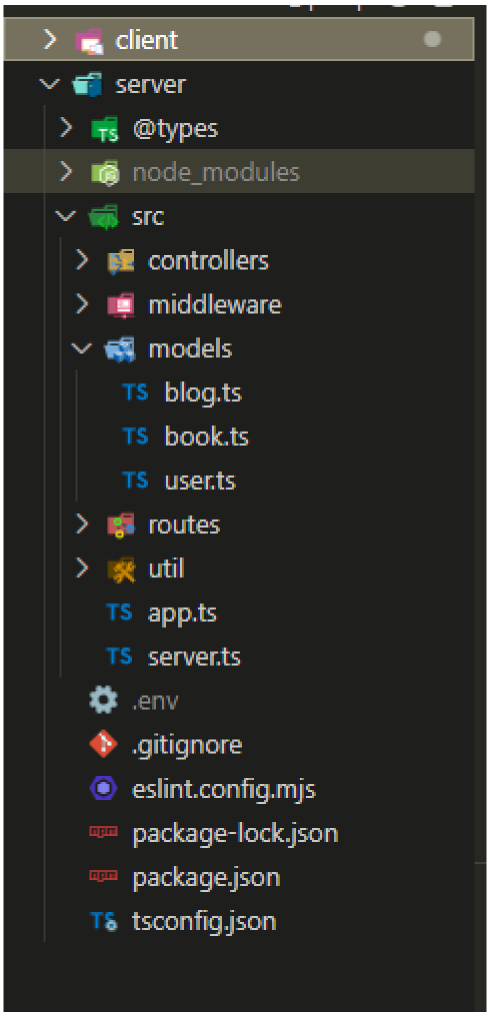
The Book Rabbit API is developed to manage a great variety of users' interactions, including books and blogs management, and data about the users.

Here we used these interactions through three main controllers **BookController, BlogController, and UserController.**

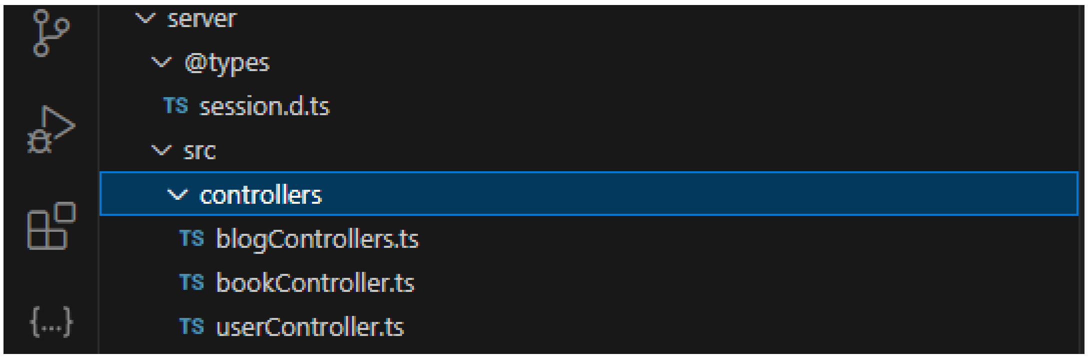
Each controller is responsible for handling specific types of requests and interacting with the database accordingly.

<u>Controllers</u>

There are three controllers used here in the project. The folder is separately created in “src” folder under the “server” directory.

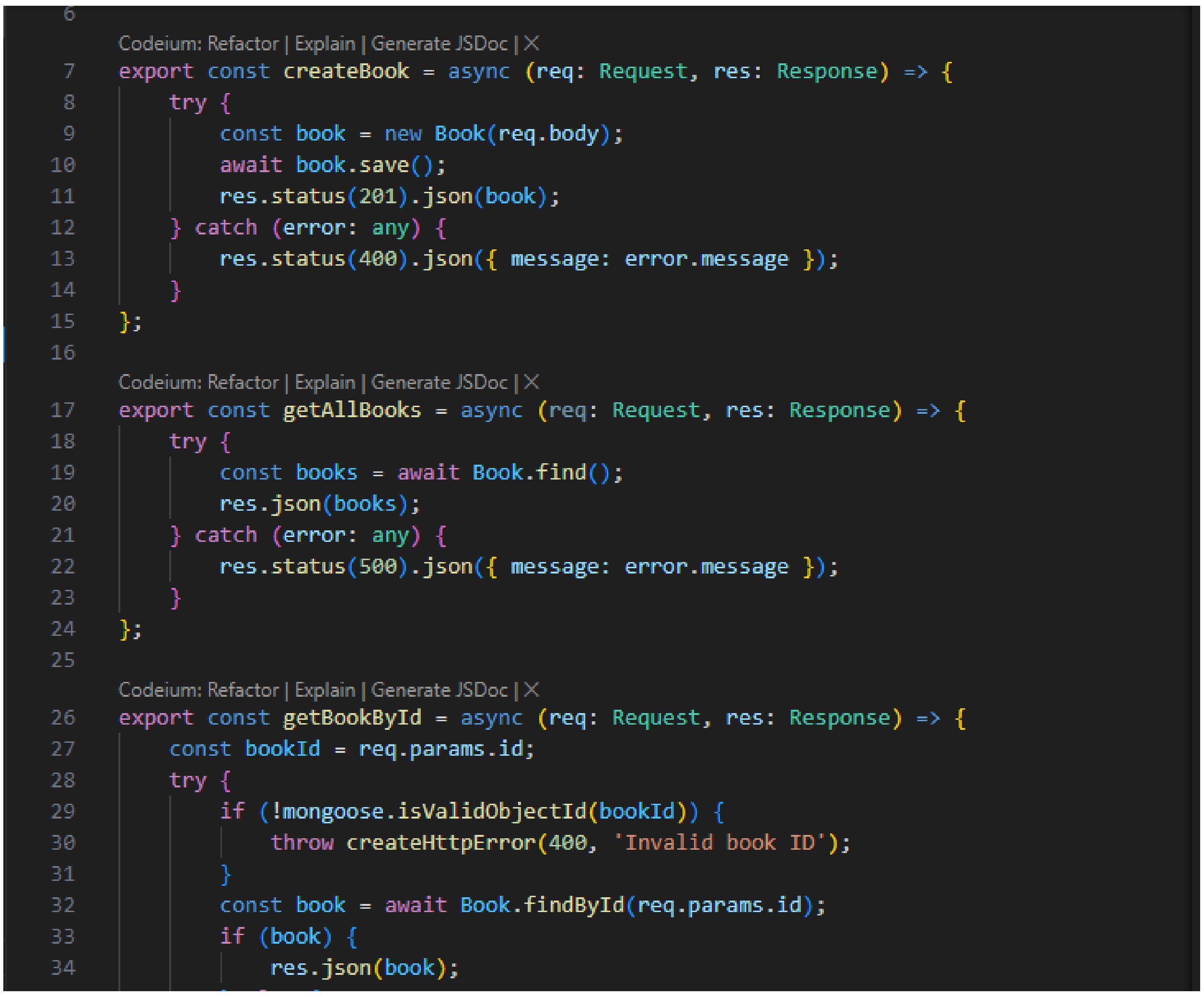


```
> client
v server
  > @types
  > node_modules
  v src
    > controllers
    > middleware
    v models
        blog.ts
        book.ts
        user.ts
    > routes
    > util
      app.ts
      server.ts
    .env
    .gitignore
    eslint.config.mjs
    package-lock.json
    package.json
    tsconfig.json
```



- Book Controller – this manages all the operations related to books, such as adding new books, fetching book details, updating book information, and deleting books. (CURD)

Here we provide some code extraction from our project.



```
Codeium: Refactor | Explain | Generate JSDoc | X
export const createBook = async (req: Request, res: Response) => {
    try {
        const book = new Book(req.body);
        await book.save();
        res.status(201).json(book);
    } catch (error: any) {
        res.status(400).json({ message: error.message });
    }
};

Codeium: Refactor | Explain | Generate JSDoc | X
export const getAllBooks = async (req: Request, res: Response) => {
    try {
        const books = await Book.find();
        res.json(books);
    } catch (error: any) {
        res.status(500).json({ message: error.message });
    }
};

Codeium: Refactor | Explain | Generate JSDoc | X
export const getBookById = async (req: Request, res: Response) => {
    const bookId = req.params.id;
    try {
        if (!mongoose.isValidObjectId(bookId)) {
            throw createHttpError(400, 'Invalid book ID');
        }
        const book = await Book.findById(req.params.id);
        if (book) {
            res.json(book);
```

In here GET/api/books, PUT/api/books/ ,POST/api/books behave as Key end points as well.

Using key endpoints we can

Fetch list of books

Update the information in specific book

Add new book to the data base

Delete the books from the entire process respectively

- Blog Controller - this manages blog posts on the platform. Users can create, read, update, and delete blog posts related to books

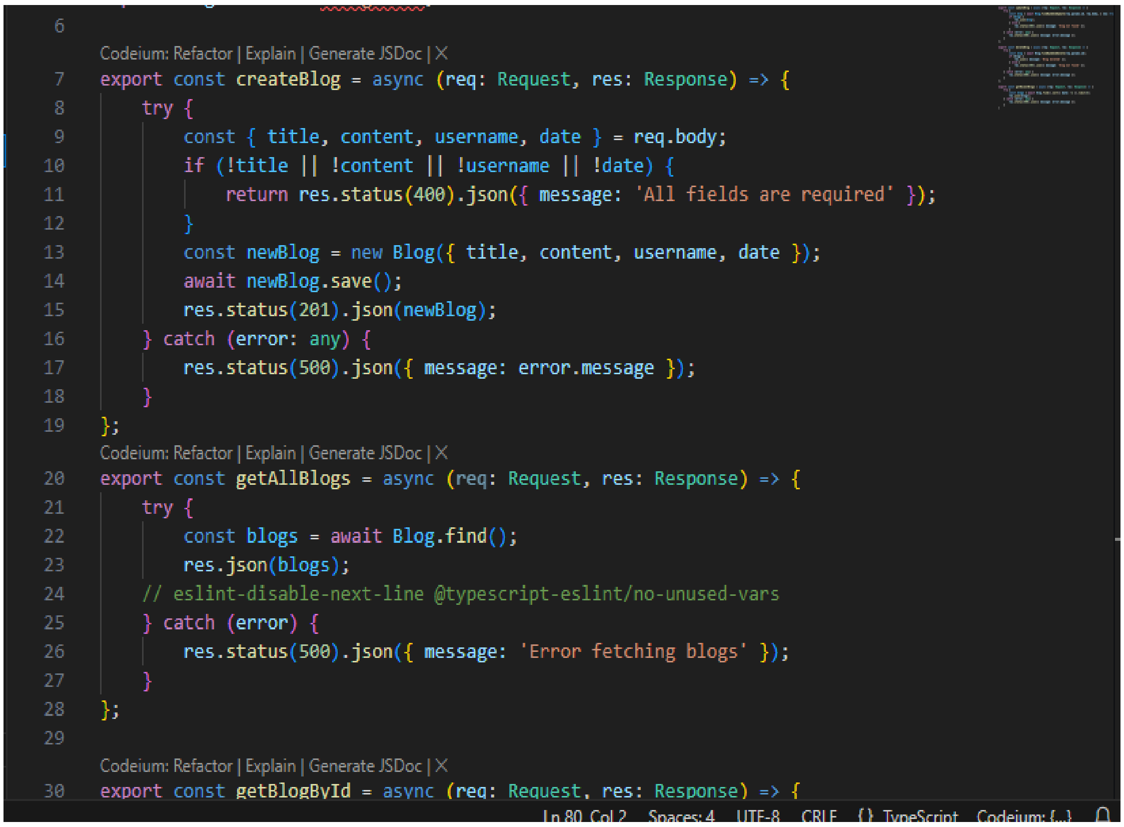
Here

GET/api/Blog – retrieve all blog post / review

PUT/api/blogs – update a specific blog post / review

POST/api/blogs – add a new blog post /review

DELETE/api/blogs – delete the blog post / review are the end key points.



```
Codeium: Refactor | Explain | Generate JSDoc | X
export const createBlog = async (req: Request, res: Response) => {
    try {
        const { title, content, username, date } = req.body;
        if (!title || !content || !username || !date) {
            return res.status(400).json({ message: 'All fields are required' });
        }
        const newBlog = new Blog({ title, content, username, date });
        await newBlog.save();
        res.status(201).json(newBlog);
    } catch (error: any) {
        res.status(500).json({ message: error.message });
    }
};
Codeium: Refactor | Explain | Generate JSDoc | X
export const getAllBlogs = async (req: Request, res: Response) => {
    try {
        const blogs = await Blog.find();
        res.json(blogs);
    // eslint-disable-next-line @typescript-eslint/no-unused-vars
    } catch (error) {
        res.status(500).json({ message: 'Error fetching blogs' });
    }
};

Codeium: Refactor | Explain | Generate JSDoc | X
export const getBlogById = async (req: Request, res: Response) => {
```

Ln 80, Col 2 Spaces: 4 UTF-8 CRLF {} TypeScript Codeium: {...}

- User Controller – this handles user-related operations, such as user registration, authentication, profile management, and other user-specific activities.

Basically these are the key api endpoints we used in this section.

POST /api/users/register - registers a new user and stores their information in the db.

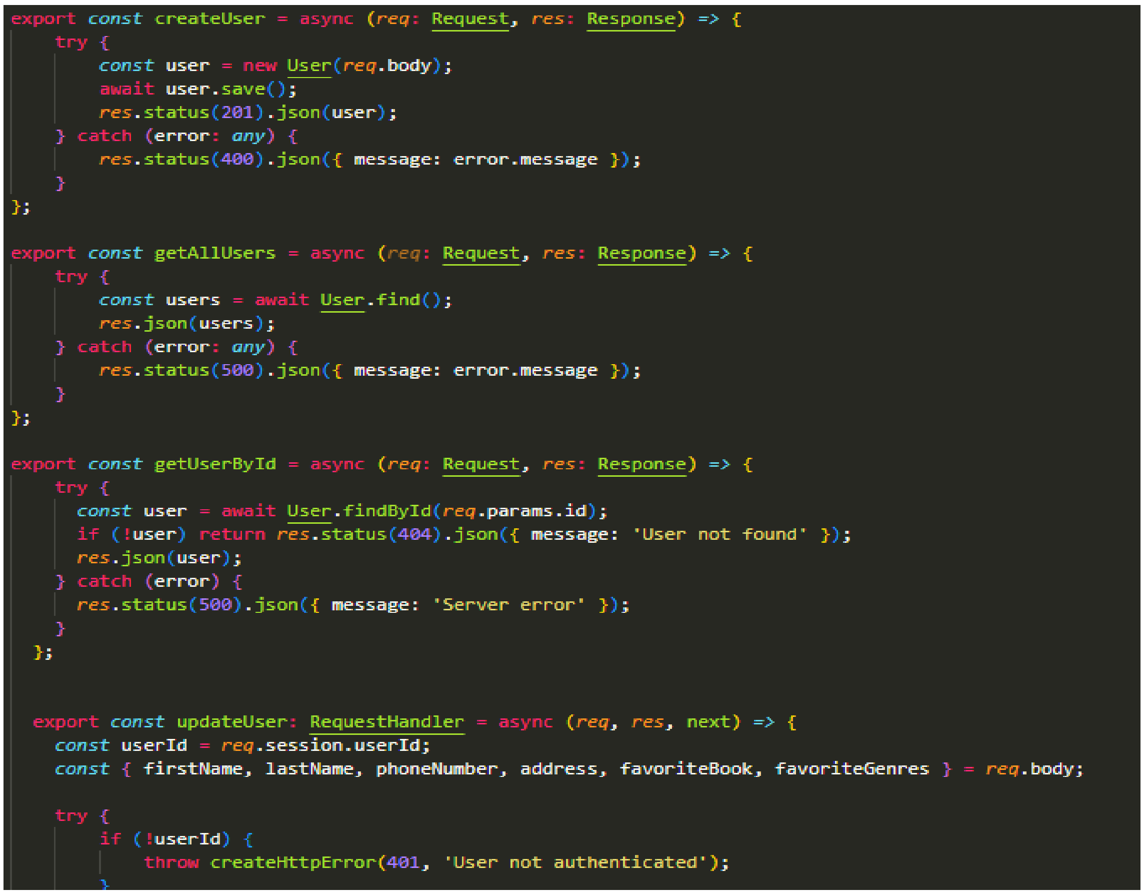
POST /api/users/login - authenticates a user and returns a token for session management.

GET /api/users/ - Retrieves user profile information.

PUT /api/users/ - updates user profile details.

DELETE /api/users/ - deletes a user account.

Here we provided some extraction in our main code suitable to this.



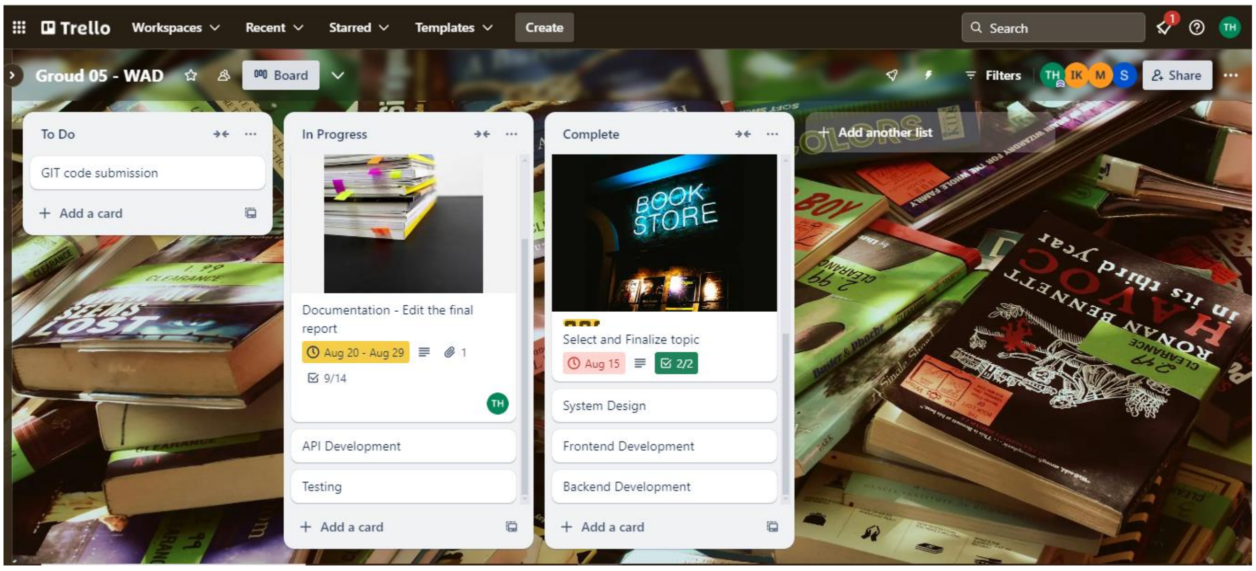
```
export const createUser = async (req: Request, res: Response) => {
    try {
        const user = new User(req.body);
        await user.save();
        res.status(201).json(user);
    } catch (error: any) {
        res.status(400).json({ message: error.message });
    }
};

export const getAllUsers = async (req: Request, res: Response) => {
    try {
        const users = await User.find();
        res.json(users);
    } catch (error: any) {
        res.status(500).json({ message: error.message });
    }
};

export const getUserById = async (req: Request, res: Response) => {
    try {
      const user = await User.findById(req.params.id);
      if (!user) return res.status(404).json({ message: 'User not found' });
      res.json(user);
    } catch (error) {
      res.status(500).json({ message: 'Server error' });
    }
  };


  export const updateUser: RequestHandler = async (req, res, next) => {
    const userId = req.session.userId;
    const { firstName, lastName, phoneNumber, address, favoriteBook, favoriteGenres } = req.body;

    try {
        if (!userId) {
            throw createHttpError(401, 'User not authenticated');
        }
```

This is a comprehensive analysis about database structure in our **BOOK RABBIT** project.

Time Frame

In the project scenario we create a Time Frame using project management software which name as Trello to manage our works among our team members in perfect way.

Here is the a snap How we engage with our works and you can access to our board and see how we engage with our works as well.



To Do
- GIT code submission
- Add a card

In Progress
- Documentation - Edit the final report
  Aug 20 - Aug 29 · 1 · 9/14 · TH
- API Development
- Testing
- Add a card

Complete
- Select and Finalize topic
  Aug 15 · 2/2
- System Design
- Frontend Development
- Backend Development
- Add a card

Board Link - https://trello.com/invite/b/66c34dae5254ffaaed6d73a8/ATTIa52b79dce5aba5ec6292b532595a503174E23903/groud-05-wad

Risk Mitigation

This section discusses all the risks having to do with the **Book Rabbit** project and the means our team put in place to alleviate those risks. Everything from problems dealing with Mongo DB, to API selections and code integrations, puts into perspective how this project will actually feel like it is in development. Adding a SWOT analysis gives an overview of the potential of the whole project.

A great technical risk was with the connection to the Mongo DB database. Since it did not go very well in the beginning, it had threatened to delay progress in this project. Given that stable data handling and storage are critical to the operation of the platform, this may constitute a high risk. To hedge against this risk, the team undertook deep troubleshooting and managed to get the connection stabilized by optimizing the configurations of the servers, doing regular monitoring of the database performance.

Another technical challenge arose around the choice of API methods—crucial for the system to return and manipulate information retrieved from external sources. The team decided to work with Amazon's API after evaluating several options, since it is a reliable API with good documentation. Since the exposure was medium risk because of the choice of the wrong API, this decision was substantiated by more than one test to assure the solution would meet needs. Continual reviews and updates will be performed to make it compatible and efficient.

This distributed nature of the development team introduced serious risks to integration and collaboration. Working from different locations, integrating code blocks presented an increased risk of inconsistency within the project. The risk here turned very high for the project, knowing how version control issues would lead to delays of considerable proportions. In light of this, the team created one central Git Hub repository where code is integrated, easily merged, and managed collaboratively. And of course, regular synchronization meetings and code reviews were put into place to keep all aligned by resolving any conflicts as soon as possible.

The third level of risk is external, represented by the regulation changes with an impact on digital book sales. While these would be unlikely to happen, their impact would be relatively high since there would be legal implications to them as well. The team remains updated on the status of regulations in the industry through consultation and working relations with legal consultants towards ensuring their continuous compliance. Their design is such that the system is versatile enough to accommodate changing conditions, should it become necessary on account of revised regulation.

A SWOT analysis was performed in the process of risk management to assess the strengths, weaknesses, opportunities, and threats of the Book Rabbit project. In a nutshell, it points out that strong expertise within the team, a well-defined technological stack, and a robust database structure are the main influencers in the group of strengths. On the other hand, initial problems with Mongo DB connectivity, API selection, and the distributed nature of the team were outlined as key items in the line of weaknesses. These are increasing demand for online book platforms and prospects of collaboration with publishers and authors. While competitions due to established players and future regulatory changes have been some gray areas that the company needed to keep a close eye on.

Summery & Conclusion

The **Book Rabbit** represents a quantum leap in the way in which people find, purchase, and read books. As a web application, it aims at achieving an integrated bookstore and book recommender system, fusing the efficiency of online book purchases with the personalized touch of discovering books. While there has been the facility for purely algorithm-derived kind of recommendations in conventional recommendation systems, **Book Rabbit** will add user reviews and community interactions as input to the recommendation capability. Users can view a large catalog of books, read e-versions of select titles for free, and leave reviews to help others decide what to read. This system architecture is built on the MERN Stack using Type Script, hence robust, scalable, and capable of handling increasing numbers of users.

Here, Amazon's API is to be used for fetching data, while Mongo DB is the database that was used for storage. These will provide a robust yet efficient backbone for operating the platform. Some of the key features this application will provide to the user include storing and buying books, creating their personal libraries, and even sharing their reviews and recommendations of books with other readers, thus making **Book Rabbit** an all-inclusive solution for the modern-day reader. While integrating the project on Code Igniter, there had been some problems at first regarding the database connection and the selection of APIs. They were taken care of by implementing the project working plan easily. The care taken by the team concerning security, user experience, and the will for improvement was easily noticeable as the system was designed and worked appropriately. Through the use of a SWOT analysis, it was clearer how the project was done in a way that capitalized on strengths, presented opportunities to grow, and covered possible dangers.

Finally, **Book Rabbit** is an amazing user-based platform that not only enables its users to access the e-versions of books to read freely but share their various insights through reviews, thereby creating a community-driven environment thus enhancing the experience of reading books. This will not only help the user find the books relevant to their taste but also foster a collaborative culture by getting influenced by each other.

The fact that this works, making the platform have replication, drives home the point of strength in the development team as a whole when facing technical challenges. Deciding to use Amazon's API and Mongo DB, applying the project in a collaborative development environment, portrays the project as technically savvy and adaptable.

Looking ahead, **Book Rabbit** is well-positioned for further development and growth by adding features and functionality that scales with an increasing user base. The project also brings proof of some model of development in the digital book space, as these combined technologies and knowledge of user needs again intersect in the creation of an innovative and influential platform.

Citation & Appendices

1. "Resolving a merge conflict on github," GitHub Docs, https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/addressing-merge-conflicts/resolving-a-merge-conflict-on-github (accessed Aug. 20, 2024).
2. J. D. C. Marin et al., "Network request failed on react native upload image," Stack Overflow, https://stackoverflow.com/questions/62450971/network-request-failed-on-react-native-upload-image#:~:text=The%20solution%20is%20to%20use,file%20would%20be%20image%2Fpng (accessed Aug. 25, 2024)
3. Bawane, M., Gawande, I., Joshi, V., Nikam, R., & Bachwani, S. A. (2022). A Review on Technologies used in MERN stack. *Int. J. Res. Appl. Sci. Eng. Technol*, *10*(1), 479-488.
4. Subramanian, V. (2019). *Pro Mern Stack: Full Stack Web App Development with Mongo, Express, React and Node*. Apress.
5. Abdala, M. A., & Khider, N. A. (2011). Online E-book Store Website Design. *i-Manager's Journal on Software Engineering*, *5*(4), 41..
6. Wittig, A., & Wittig, M. (2023). *Amazon Web Services in Action: An in-depth guide to AWS*. Simon and Schuster.
7. Siriarcharungroj, S. (2002). Practical development of information system in business context: electronic commerce of web hosting for Business Solution Network Co., Ltd.
8. Prodan, R., & Ostermann, S. (2009, October). A survey and taxonomy of infrastructure as a service and web hosting cloud providers. In *2009 10th IEEE/ACM International Conference on Grid Computing* (pp. 17-25). IEEE.
9. Zagalsky, A., Feliciano, J., Storey, M. A., Zhao, Y., & Wang, W. (2015, February). The emergence of github as a collaborative platform for education. In *Proceedings of the 18th ACM conference on computer supported cooperative work & social computing* (pp. 1906-1917).
10. Effendy, F., & Adhilaksono, B. (2021). Performance comparison of web backend and database: A case study of node. js, Golang and MySQL, Mongo DB. *Recent Advances in Computer Science and Communications (Formerly: Recent Patents on Computer Science)*, *14*(6), 1955-1961.

11. Botoeva, E., Calvanese, D., Cogrel, B., Rezk, M., & Xiao, G. (2016, April). OBDA beyond relational DBs: A study for MongoDB. In *CEUR Workshop Proceedings* (Vol. 1577). CEUR-WS. org.

Members Of Group 05

AR-102269

AF/20/16438

K.K.I.Uthpala

AR101874

AF/20/16043

O. S. Nimthara

AR 102224

AF/20/16393

E.G.T.K.S. Herath

AR 102118

AF/20/16287

O.M.Madhusha Chinthani

_____End of the Report_____