Ideation Phase
Brainstorm & Idea Prioritization Template

Date	20 Feb 2026
Team ID	LTVIP2026TMIDS79402
Project Name	Booknest: where stories nestle
Maximum Marks	

Objective:

The objective of this project is to develop a full-stack online book nest web application using the **MERN stack (MongoDB, Express.js, React.js, Node.js)** that allows users to browse, search, and purchase books, while providing admin functionality to manage books, users, and orders efficiently.

Step-1: Team Gathering, Collaboration and Select the Problem Statement

The team conducted an initial discussion to identify real-world problems that can be solved using full-stack web development technologies.

We explored various domains such as:

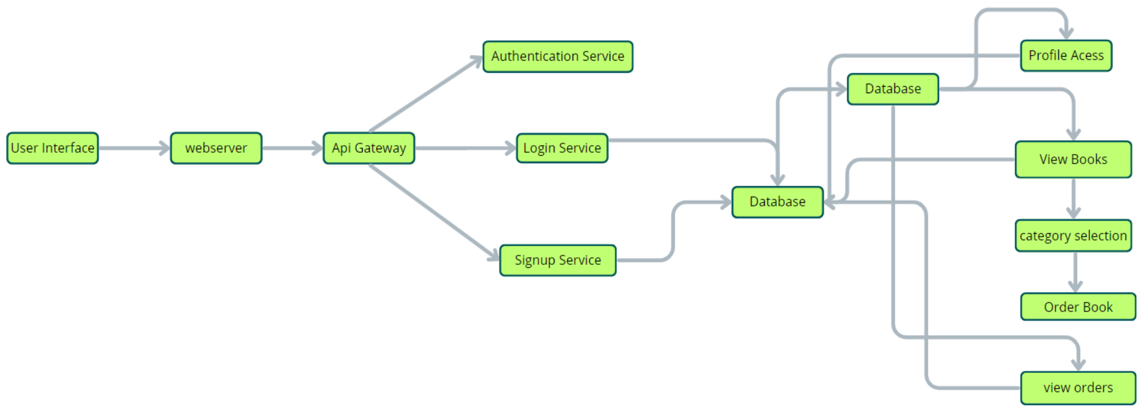
E-commerce platform

Digital libraries

Online learning platforms

Food delivery systems

After discussion and feasibility analysis, we selected the problem statement:"Develop a full-stack online book store platform where users can browse and purchase books, and administrators can manage inventory and orders."



Step-2: Brainstorm, Idea Listing and Grouping

We brainstormed different features and grouped them into functional categories:

 User Features

- User Registration & Login (JWT Authentication)
- Browse books by category
- Search books by title/author
- Add to cart functionality
- Place orders
- View order history

 Admin Features

- Add new books
- Update book details
- Delete books
- Manage users
- View all orders

 Technical Features

REST API using Node.js + Express.js

Database using MongoDB

Frontend built with React.js

Protected routes using JWT

Role-based access (Admin/User)

Environment configuration using .env

Step-3: Idea Prioritization

We used the Impact vs. Feasibility Matrix to score each idea:

Idea	Impact	Feasibility	Priority
User Authentication (JWT)	High	High	Selected
CRUD for Books	High	High	Selected
Cart & Order System	High	Medium	Selected
Online Payment Gateway Integration	High	Low	Postponed

Real-time Chat Support	Medium	Low	Not Selected
Recommendation System	High	Medium	Future Scope

Final Prioritized Idea:

BookNest – A MERN Stack Online Book Store Web Application

Ideation Phase
Define the Problem Statements

Date	20 Feb 2026
Team ID	LTVIP2026TMIDS79402
Project Name	Booknest: where stories nestle
Maximum Marks	

Customer Problem Statement Template:

As a student/book lover/customer, I face difficulties in finding affordable books, checking availability, and purchasing them easily from local stores. Physical bookstores have limited collections, and searching manually is time-consuming. I need a simple, reliable, and user-friendly online platform where I can browse, search, and purchase books conveniently from anywhere.

I am	I'm trying to	But	Because	Which makes me feel
A college student	Buy affordable academic books	Local stores don't always have required editions	Book availability is limited and prices are high	Stressed and financially pressured
A book enthusias t	Explore and purchase new novels	I can't find all genres in one place	Physical bookstore s have limited collections	Disappointe d and restricted in choices
A working professio nal	Order books convenien tly after work	I don't have time to visit bookstor es	My schedule is busy and stores close early	Frustrated and inconvenien ced
A small bookstor e owner	Expand my business reach	I lack an online platform to sell books	Setting up digital systems is complex	Left behind in digital competition
An admin	Track	Manual	There is	Overwhelme

managing inventory	book stock efficiently	tracking leads to errors	no centralized management system	d and inefficient

Ideation Phase
Empathize & Discover

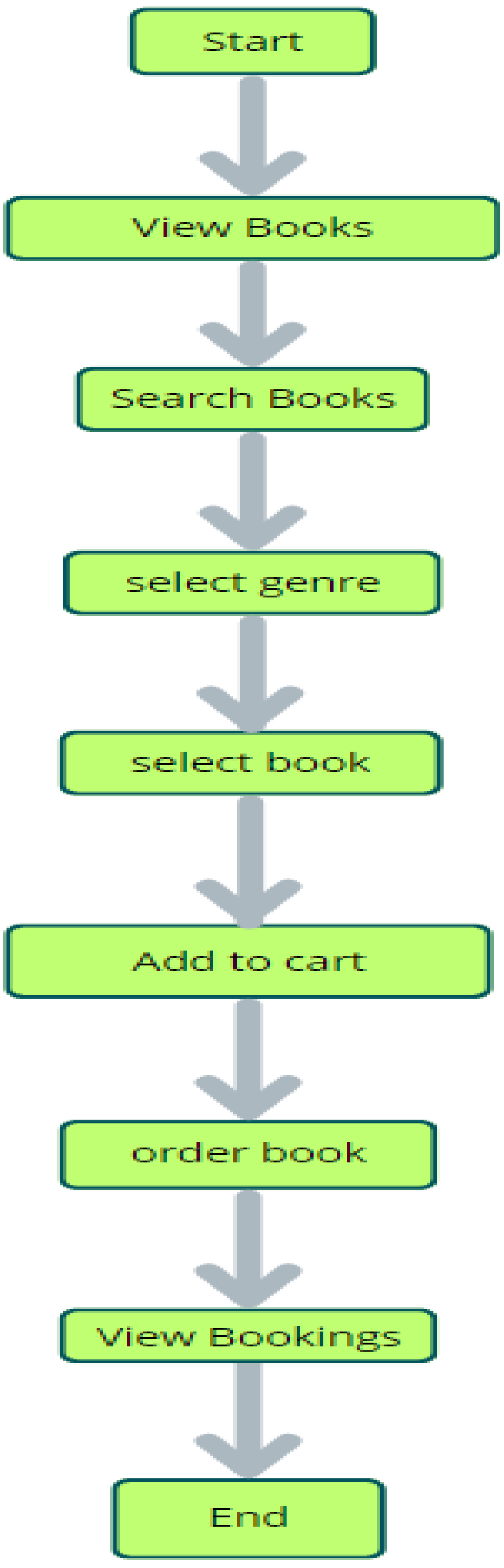
Date	20 Feb 2026
Team ID	LTVIP2026TMIDS79402
Project Name	Booknest: where stories nestle
Maximum Marks	

Empathy Map:
This Empathy Map is created considering the customer or quality inspector as the user**.** The empathy map will include organized text to clearly represent the users' thoughts, feelings, actions, and statements.

Example:

SAYS	**THINKS**
"Is this book available right now?" "Are the prices better than offline stores?"	**"Can I trust this website for secure payment?" "Will my order arrive safely?"**
"I hope the delivery is fast." "Is my payment information secure?"	**"I want to easily find the book I’m looking for." "Is this platform reliable and genuine?"**
DOES	**FEELS**
Compares prices with other websites. Frustrated if a book is out of stock.	**Reads book descriptions and reviews. Worried about payment security.**
Adds books to cart and places orders. Satisfied when delivery is smooth and timely.	**Searches for books by title, author, or category. Excited when finding a desired book.**

DATA FLOW DIAGRAM



Project Design Phase-II
Solution Requirements (Functional & Non-functional)

Date	20Feb 2026
Team ID	LTVIP2026TMIDS79402
Project Name	Booknest : Where Stories nestle
Maximum Marks	

Functional Requirements:

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration & Login	Users can register, login, and logout securely using JWT authentication.
FR-2	Book Browsing & Search	Users can browse books by category and search by title, author, or keyword.
FR-3	Book Details View	Displays book image, description, price, author, category, and availability status.
FR-4	Cart Management	Users can add books to cart, update quantity, and remove items.
FR-5	Order Placement	Users can place orders and receive confirmation after successful checkout.
FR-6	Order History	Users can view previous orders and track order status.
FR-7	Admin Book Management	Admin can add, update, and delete books (CRUD operations).
FR-8	User Management (Admin)	Admin can view registered users and manage user accounts.
FR-9	Order Management (Admin)	Admin can view all orders and update order status.
FR-10	Backend API Integration	REST APIs built using Node.js and Express.js connected to MongoDB database.
FR-11	Role-Based Access Control	System differentiates between Admin and User roles with protected routes.
FR-12	Environment Configuration	Uses .env file for storing secure credentials like database URL and JWT secret.

Non-Functional Requirements:

NFR No.	Non-Functional Requirement	Description
NFR-1	Security	User authentication must use JWT and encrypted passwords (bcrypt).
NFR-2	Performance	Page load time should be under 3 seconds for smooth user experience.
NFR-3	Scalability	System should support adding more categories, books, and users without performance degradation.
NFR-4	Usability	Interface must be clean, responsive, and easy to navigate for all users.
NFR-5	Portability	Application should run on any system with Node.js and MongoDB installed.
NFR-6	Availability	System should be accessible 24/7 when deployed on cloud platform.
NFR-7	Data Integrity	Ensures accurate storage and retrieval of book, user, and order data in MongoDB.
NFR-8	Maintainability	Code should follow modular structure for easy updates and debugging.
NFR-9	Compatibility	Web app must work properly on Chrome, Edge, and other modern browsers.
NFR-10	Reliability	System should handle multiple users simultaneously without crashing.

Technology Stack

Date	20 Feb 2026
Team ID	LTVIP2026TMIDS79402
Project Name	Booknest : Where stories nestle
Maximum Marks	

Technology Stack:

Category	Technology / Tool	Purpose
Frontend Framework	React.js	Building interactive and dynamic user interface
Backend Runtime	Node.js	Server-side JavaScript runtime environment
Backend Framework	Express.js	Building RESTful APIs and handling server routes
Database	MongoDB	Storing book, user, and order data
Database ODM	Mongoose	Connecting and modeling MongoDB data in Node.js
Authentication	JWT (JSON Web Token)	Securing user login and protected routes
Password Encryption	bcrypt	Hashing user passwords securely
State Management	React Hooks / Context API	Managing cart and user state
API Testing Tool	Postman	Testing backend APIs during development
Environment Configuration	.env (dotenv)	Storing sensitive data like DB URL and JWT secret
Deployment Platform (Optional)	Render / Vercel / Localhost	Hosting backend and frontend applications
Version Control	Git + GitHub	Code management and project submission
Package Manager	npm	Managing project dependencies
UI Styling	CSS3 / Bootstrap / Tailwind CSS	Designing responsive and modern interface
Browser Compatibility	Chrome, Edge, Firefox	Ensuring application works across modern browsers
IDE / Code Editor	Visual Studio Code	Developing frontend and backend code

Customer Journey Map

Date	20 Feb 2026
Team ID	LTVIP2026TMIDS79402
Project Name	Booknest : Where stories nestle
Maximum Marks	

Customer Journey Map

Customer Persona: College Student / Book Lover / Working Professional / Admin
Product: BookNest – MERN Stack Online Book Store
Goal: Easily search, purchase, and manage books online with a smooth and secure experience..

Overview Table

Here is the content in a table format:

Stage	Touchpoints	Customer Actions	Experience	Pain Points	Opportunities for Improvement
Awareness	Social media, Word-of-mouth, College groups, Posters	Learns about BookNest platform	Curious and interested	Unsure about trust and reliability	Show testimonials, ratings, and secure payment badges
Consideration	Homepage, Book Listings, Search Bar	Browses categories and searches for books	Hopeful and exploring	Too many options or unclear navigation	Add filters (price, category, author) and clean UI layout
Onboarding	Signup/Login Page	Creates account and logs in	Excited but cautious	Complex signup process	Simplify form fields and add Google login (future scope)
Usage	Book Detail Page, Cart, Checkout	Adds books to cart and places order	Productive and satisfied	Cart issues or slow checkout	Optimize performance and auto-save cart
Payment	Checkout Page, Order Confirmation	Page	Enters details and confirms order	Relieved after successful payment	Concern about payment security
Display SSL badge and secure payment message clearly					

Order Tracking	Order History Page, Email Notification	Tracks order status	Confident and assured	Lack of real-time updates	Add live order status tracking
Monitoring	User Profile Dashboard	Views previous purchases and manages account	Organized	Unable to edit certain details easily	Add profile edit and order cancellation options
Support	Contact Page, FAQ, Email	Asks questions about delivery or refunds	Supported	Delayed response time	Add chatbot or quick-help widget
Feedback	Feedback Form, Ratings & Reviews	Rates books and provides suggestions	Engaged	No follow-up on feedback	Add review approval status and thank-you email
Maintenance	Admin Dashboard	Updates books, stock, and order status	Efficient and in control	Manual stock updates can be time-consuming	Add bulk upload feature for books (future scope)

Project Design Phase
Problem – Solution Fit

Date	20 Feb 2026
Team ID	LTVIP2026TMIDS79402
Project Name	Smart Sorting : Transfer Learning For Identifying Rotten Fruits And Vegetables
Maximum Marks	

Customer Problem Identified

Students, book lovers, and working professionals often struggle to find required books easily, compare prices, and purchase them conveniently from physical stores. Limited availability, time constraints, and lack of centralized platforms lead to inconvenience and missed buying opportunities.

Observed Behavior

People visit multiple bookstores or browse different websites to find a single book. They manually compare prices, check availability through calls, and sometimes delay purchases due to time constraints or uncertainty about stock.

Proposed Solution

A full-stack online bookstore web application built using the MERN stack that allows users to:

- Search books by title, author, or category
- View detailed descriptions and pricing
- Add books to cart
- Place orders securely
- Track order history

The platform also provides an admin dashboard to manage books, users, and orders efficiently.

Why This Solution Works

The system is built using a scalable MERN architecture:

- React.js provides a dynamic and responsive user interface
- Node.js and Express.js handle backend APIs efficiently
- MongoDB stores structured book, user, and order data
- JWT authentication ensures secure access

This ensures fast browsing, secure transactions, and reliable data management.

Unique Value Proposition

- Provides a centralized platform for discovering and purchasing books
- Secure authentication and role-based access (Admin/User)
- Responsive design for mobile and desktop users
- Easy inventory and order management for administrators
- Scalable structure to add future features like recommendations and online payments

Impact of the Solution

- Saves time for users
- Expands digital reach for bookstores
- Improves purchasing convenience
- Enhances inventory management
- Encourages digital transformation of small book businesses
- Suitable for integration with payment gateways and recommendation systems in future

Project Design Phase
Proposed Solution

Date	20 Feb 2026
Team ID	LTVIP2026TMIDS79402
Project Name	Booknest : Where stories nestle
Maximum Marks	

Proposed Solution:

S.No.	Parameter	Description
1	Problem Statement (Problem to be solved)	Students, book lovers, and working professionals face difficulty in finding, comparing, and purchasing books easily due to limited availability in local stores and lack of a centralized digital platform.
2	Idea / Solution Description	A full-stack MERN web application that allows users to browse, search, and purchase books online, while enabling administrators to manage books, users, and orders efficiently through a secure dashboard.
3	Novelty / Uniqueness	Combines a responsive React frontend with secure JWT-based authentication and a scalable Node.js + Express backend connected to MongoDB, offering role-based access (Admin/User) and seamless cart-to-checkout functionality.
4	Social Impact / Customer Satisfaction	Provides convenient access to books anytime, reduces time spent searching across multiple stores, supports small bookstores in digital transformation, and enhances customer satisfaction through smooth online purchasing experience.
5	Business Model (Revenue Model)	Revenue through book sales margin, premium seller listings, sponsored books, and potential integration of online payment gateways. Future scope includes

		subscription-based membership for discounts and early access.
6	Scalability of the Solution	System can be scaled to include e-books, recommendation systems, online payment integration, multi-vendor marketplace model, and mobile application support. Cloud deployment enables handling large user traffic.

Solution Architecture

Date	20 Feb 2026
Team ID	LTVIP2026TMIDS79402
Project Name	Booknest : Where stories nestle
Maximum Marks	

BookNest Solution Architecture

1. User Interface (Frontend)

Technology: React.js, HTML5, CSS3 (Bootstrap / Tailwind CSS)

Functionality:

- Users can register and login securely
- Browse books by category
- Search books by title, author, or keyword
- View detailed book information (price, description, availability)
- Add books to cart
- Place orders through checkout
- Admin dashboard to manage books and orders

2. Backend Server (API Layer)

Technology: Node.js + Express.js

Functionality:

- Handles REST API routes:
  - `/api/users` → Registration & Login
  - `/api/books` → Book CRUD operations
  - `/api/orders` → Order placement & tracking
- Implements JWT authentication
- Encrypts passwords using bcrypt
- Validates user roles (Admin/User)
- Connects frontend with database
- Handles error responses and status codes

3. Database Layer

Technology: MongoDB + Mongoose

Collections:

- Users Collection
- Books Collection
- Orders Collection

Functionality:

- Stores user credentials securely
- Stores book details (title, author, category, price, stock)
- Stores order history and status
- Ensures data consistency and integrity

4. Application Flow (Data Pipeline)

1. User interacts with React frontend
2. Frontend sends API request to Express backend
3. Backend validates request and authenticates user
4. Backend interacts with MongoDB
5. Database sends response back to backend
6. Backend returns JSON response to frontend
7. Frontend updates UI dynamically

5. Deployment (Local / Cloud)

Option 1: Local Deployment

- Install Node.js and MongoDB
- Run backend using:

  ```
  npm run server
  ```

- Run frontend using:

  ```
  npm start
  ```

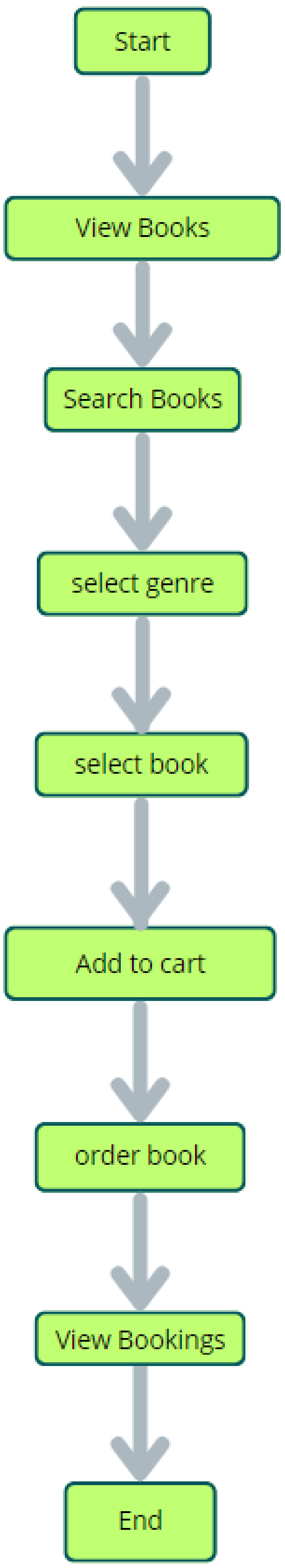
- Access application at:

  ```
  http://localhost:3000/
  ```

Option 2: Cloud Deployment

- Backend hosted on:
  - Render
  - Railway
  - Heroku
- Frontend hosted on:
  - Vercel
  - Netlify
- MongoDB Atlas for cloud database

Architecture Diagram:



Project Planning Phase

Date	20 Feb 2026
Team ID	LTVIP2026TMIDS79402
Project Name	Booknest : Where stories nestle
Maximum Marks	

1. Project Description

BookNest is a full-stack online bookstore web application developed using the MERN stack. The platform enables users to browse, search, and purchase books online while providing administrators with tools to manage books, users, and orders efficiently.

The system supports secure authentication, cart functionality, order placement, and role-based access control. It is designed to be scalable, responsive, and suitable for real-world deployment.

2. Project Objectives

- Develop a fully functional MERN stack web application.
- Implement secure user authentication using JWT.
- Enable users to browse, search, and purchase books online.
- Create an admin dashboard for managing books and orders.
- Design a responsive UI for both desktop and mobile users.
- Deploy the application locally and optionally on cloud platforms.

3. Technology Stack

Layer	Tools / Frameworks Used
Frontend	React.js, HTML5, CSS3, Bootstrap / Tailwind CSS
Backend	Node.js, Express.js
Database	MongoDB (Local / MongoDB Atlas)
Authentication	JWT (JSON Web Token), bcrypt
State Management	React Hooks / Context API

API Testing	Postman
Version Control	Git, GitHub
Hosting (Optional)	Render (Backend), Vercel / Netlify (Frontend)
Development IDE	Visual Studio Code

4. Resources Required

Resource Type	Description
Hardware	Local PC/Laptop with minimum 8GB RAM
Software	Node.js, MongoDB, VS Code, Web Browser
Database	MongoDB Local or MongoDB Atlas (Cloud)
Libraries & Packages	Express, Mongoose, bcrypt, jsonwebtoken, cors, dotenv
Frontend Packages	React Router, Axios
Collaboration Tools	GitHub, Google Drive, Documentation tools

5. Risk Analysis & Mitigation

Risk	Mitigation Strategy
Authentication vulnerabilities	Use JWT securely and hash passwords with bcrypt
Database connection errors	Validate .env configuration and test locally
Slow performance with many users	Optimize queries and use indexing in MongoDB
Cart or order logic bugs	Thorough testing of API endpoints using Postman
Deployment issues	Test locally before deploying to cloud

Data inconsistency	Use proper schema validation in Mongoose

6. Success Criteria

- Fully functional user registration and login system.
- Users can browse, search, and add books to cart.
- Successful order placement and order history tracking.
- Admin can manage books and orders via dashboard.
- Responsive UI across devices.
- Clean, well-documented GitHub repository.
- Successful local deployment and optional cloud hosting.

Project Development Phase
Model Performance Testing

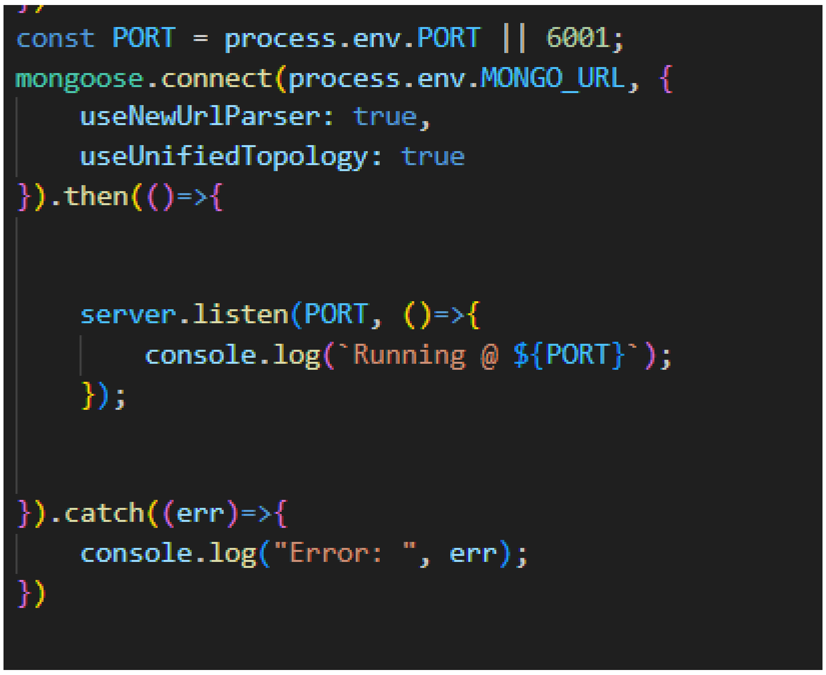
Date	20 Feb 2026
Team ID	LTVIP2026TMIDS79402
Project Name	Booknest : Where stories nestle

Model Performance Testing:

S.NO.	**Parameter**	**Values**
1.	**System Architecture**	**Frontend: React.js**
	Backend: Node.js + Express.js	
	Database: MongoDB	
	Authentication: JWT + bcrypt	
2.	**API Response Time**	**Average Response Time: < 300 ms (local testing)**
	Page Load Time: < 3 seconds	
3.	**Authentication Performance**	**Login & Token Generation: < 500 ms**
	Password Encryption: bcrypt with secure hashing	
4.	**Database Performance**	**Efficient CRUD operations using Mongoose**
	Indexed queries for faster book search	
5.	**Load Handling**	**Tested with multiple concurrent requests using Postman**
	System handles simultaneous users without crashing	
6.	**Frontend Performance**	**Fast rendering using React virtual DOM**
	Dynamic state updates without full page reload	

7.	**Optimization Applied**	**Environment variables for secure config**
	Proper error handling & validation	
	Modular folder structure for maintainability	

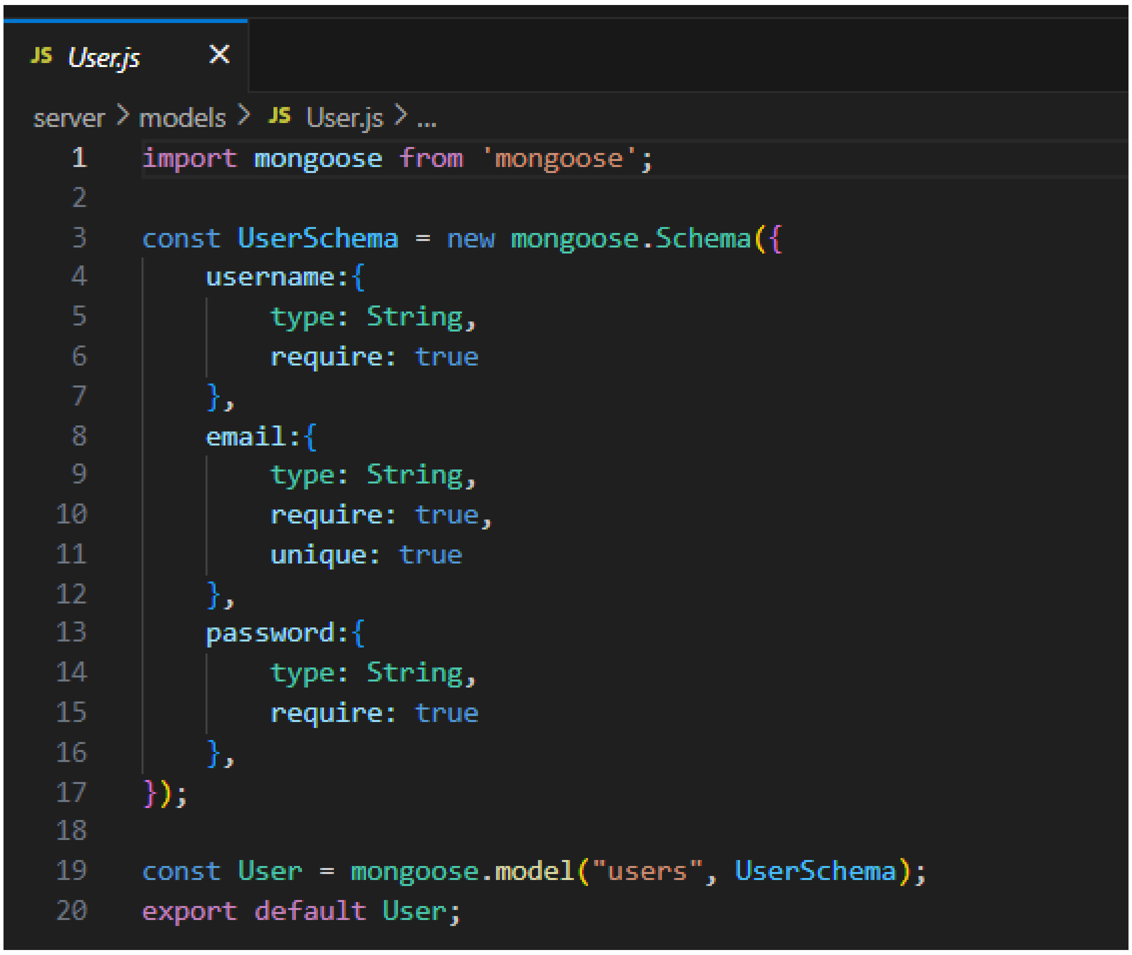
Screenshots:
The screenshots are given below



```
const PORT = process.env.PORT || 6001;
mongoose.connect(process.env.MONGO_URL, {
    useNewUrlParser: true,
    useUnifiedTopology: true
}).then(()=>{


    server.listen(PORT, ()=>{
        console.log(`Running @ ${PORT}`);
    });


}).catch((err)=>{
    console.log("Error: ", err);
})
```



```
import mongoose from 'mongoose';

const UserSchema = new mongoose.Schema({
    username:{
        type: String,
        require: true
    },
    email:{
        type: String,
        require: true,
        unique: true
    },
    password:{
        type: String,
        require: true
    },
});

const User = mongoose.model("users", UserSchema);
export default User;
```

The above are some screenshots of my project.