BookNest: Where Stories Nestle

1. INTRODUCTION

1.1 Project Overview

BookNest – Where Stories Nestle is a full-stack web application developed using the MERN stack (MongoDB, Express.js, React.js, Node.js). The platform provides a centralized digital bookstore where customers can browse, search, purchase, and review books online.

The application supports three main roles:

- Customer (Buy Books)
- Seller (Manage Books & Orders)
- Admin (Monitor Platform Activities)

BookNest aims to bridge the gap between traditional bookstores and modern digital commerce by offering a secure, scalable, and user-friendly online book marketplace.

1.2 Purpose

The purpose of BookNest is to:

- Provide a convenient online platform for book purchasing
- Enable sellers to manage inventory digitally
- Offer secure authentication and role-based access
- Reduce dependency on physical bookstores
- Deliver seamless user experience with secure payments

2. IDEATION PHASE

2.1 Problem Statement

Book lovers face several challenges such as:

- Limited time to visit physical bookstores
- Limited stock availability
- Difficulty comparing books and prices
- Manual order tracking

Sellers face:

- Limited reach to customers
- Manual inventory handling
- Lack of analytics and reporting

2.2 Empathy Map Canvas

Think & Feel

- Loves reading
- Wants convenience
- Concerned about secure payments

Hear

- Friends recommend books
- Social media promotions
- Online reviews

See

- Bestseller lists
- Discount offers
- Book categories

Say & Do

- Search books
- Add to cart
- Compare prices
- Leave reviews

Pain Points

- No time for physical stores
- Out-of-stock books
- Manual tracking

Gains

- Easy browsing
- Secure checkout
- Quick delivery

2.3 Brainstorming

Ideas generated:

- Online bookstore with role-based login

- Real-time inventory updates
- Cart & secure payment system
- Seller dashboard
- Admin analytics panel
- Future AI-based recommendation system

3. REQUIREMENT ANALYSIS

3.1 Customer Journey Map

1. User visits website
2. Registers / Logs in
3. Browses books
4. Searches / Filters
5. Adds to cart
6. Proceeds to checkout
7. Makes payment
8. Receives confirmation
9. Tracks order

3.2 Solution Requirement

Functional Requirements:

- User registration & login
- Book browsing & search
- Cart management
- Secure payment
- Order history
- Seller book management
- Admin user management

Non-Functional Requirements:

- Fast response time
- Secure authentication (JWT)

- Scalability
- Data security
- Responsive design

3.3 Data Flow Diagram

User → Frontend → Backend → Database
Backend → Payment Gateway
Backend → Email Service
Response → User

3.4 Technology Stack

Frontend : React.js , Bootstrap

Backend : Node.js, Express.js

Database : MongoDB

4. PROJECT DESIGN

4.1 Problem Solution Fit

BookNest solves the inconvenience of traditional bookstores by offering:

- 24/7 access
- Organized categories
- Secure checkout
- Real-time updates

4.2 Proposed Solution

BookNest provides:

- Customer Panel
- Seller Dashboard
- Admin Dashboard
- Secure Authentication
- Payment Integration
- Cloud Deployment

4.3 Solution Architecture

Presentation Layer → React
Business Logic Layer → Node.js + Express
Data Layer → MongoDB

5. PROJECT PLANNING & SCHEDULING

5.1 Project Planning

Phase	Duration
Requirement Analysis	3 Days
UI Design	3 Days
Backend Development	5 Days
Frontend Integration	4 Days
Testing	3 Days
Deployment	2 Days

6. FUNCTIONAL AND PERFORMANCE TESTING

6.1 Performance Testing

□ Average API response time: < 2 seconds

□ Load handling: 50+ concurrent users (local testing)

□ Database query optimization applied

□ No critical security vulnerabilities detected

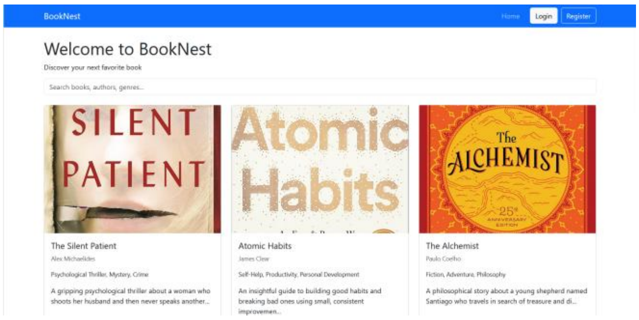
7. RESULTS

The BookNest platform successfully:

- Implements role-based authentication
- Manages inventory efficiently
- Stores user and order data correctly
- Provides seamless user experience

7.1 Output Screenshots

Home Page



Welcome to BookNest

Discover your next favorite book

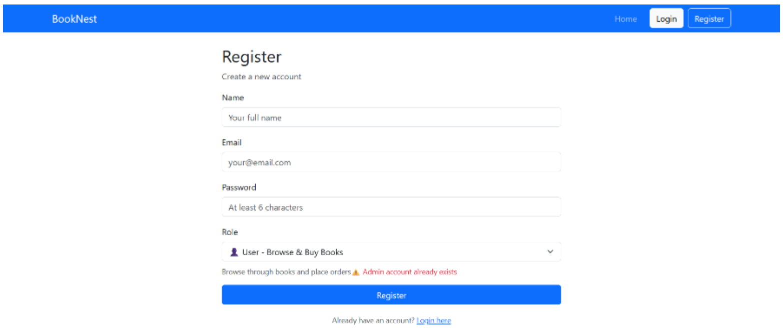
Search books, authors, genres...

The Silent Patient
Alex Michaelides
Psychological Thriller, Mystery, Crime
A gripping psychological thriller about a woman who shoots her husband and then never speaks another...

Atomic Habits
James Clear
Self-Help, Productivity, Personal Development
An insightful guide to building good habits and breaking bad ones using small, consistent improvemen...

The Alchemist
Paulo Coelho
Fiction, Adventure, Philosophy
A philosophical story about a young shepherd named Santiago who travels in search of treasure and di...

Signup Page



Register

Create a new account

Name: Your full name

Email: your@email.com

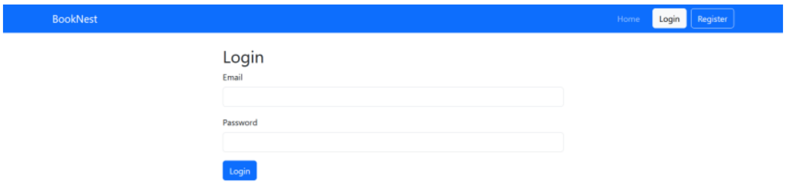
Password: At least 6 characters

Role: 👤 User - Browse & Buy Books

Browse through books and place orders ⚠ Admin account already exists

Already have an account? Login here

Login Page

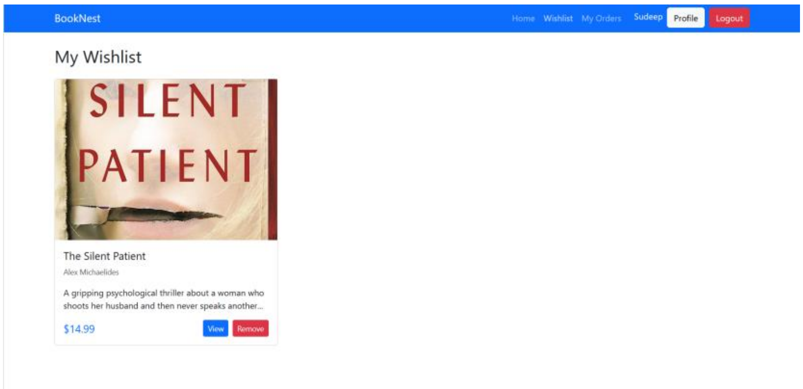


Login

Email

Password

User Whishlist Page



My Wishlist

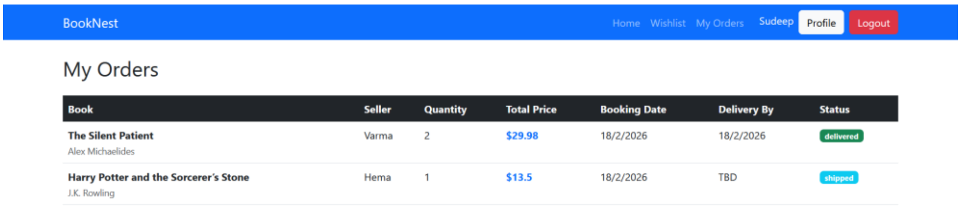
The Silent Patient

Alex Michaelides

A gripping psychological thriller about a woman who shoots her husband and then never speaks another...

$14.99

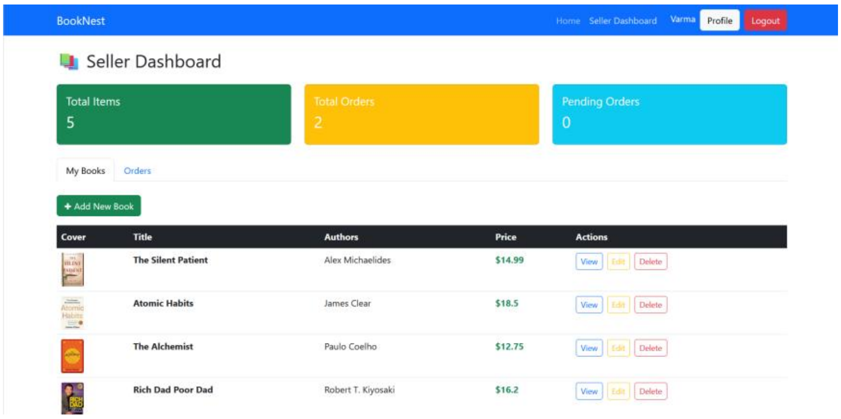
My Orders Page



My Orders

Book	Seller	Quantity	Total Price	Booking Date	Delivery By	Status
The Silent Patient Alex Michaelides	Varma	2	$29.98	18/2/2026	18/2/2026	delivered
Harry Potter and the Sorcerer's Stone J.K. Rowling	Hema	1	$13.5	18/2/2026	TBD	shipped

Seller Dashboard



Seller Dashboard

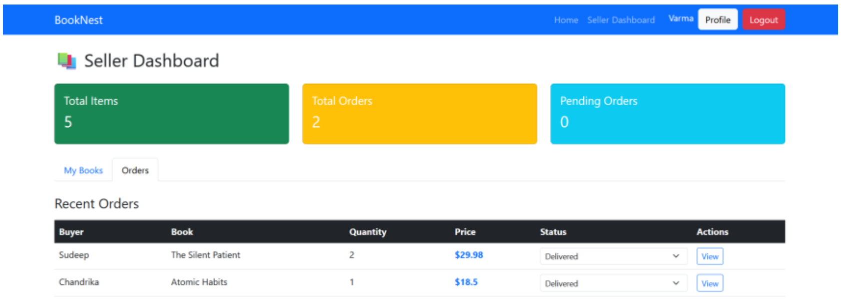
Total Items
5

Total Orders
2

Pending Orders
0

Cover	Title	Authors	Price	Actions
	The Silent Patient	Alex Michaelides	$14.99	View Edit Delete
	Atomic Habits	James Clear	$18.5	View Edit Delete
	The Alchemist	Paulo Coelho	$12.75	View Edit Delete
	Rich Dad Poor Dad	Robert T. Kiyosaki	$16.2	View Edit Delete

Seller Orders Page



Seller Dashboard

Total Items
5

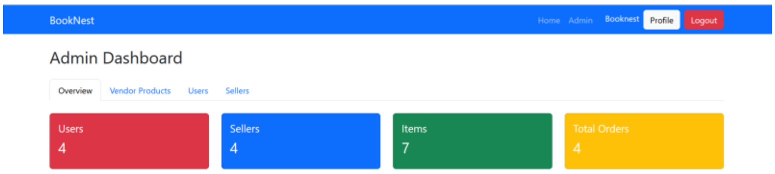
Total Orders
2

Pending Orders
0

Recent Orders

Buyer	Book	Quantity	Price	Status	Actions
Sudeep	The Silent Patient	2	$29.98	Delivered	View
Chandrika	Atomic Habits	1	$18.5	Delivered	View

Admin Dashboard Page



Admin Dashboard

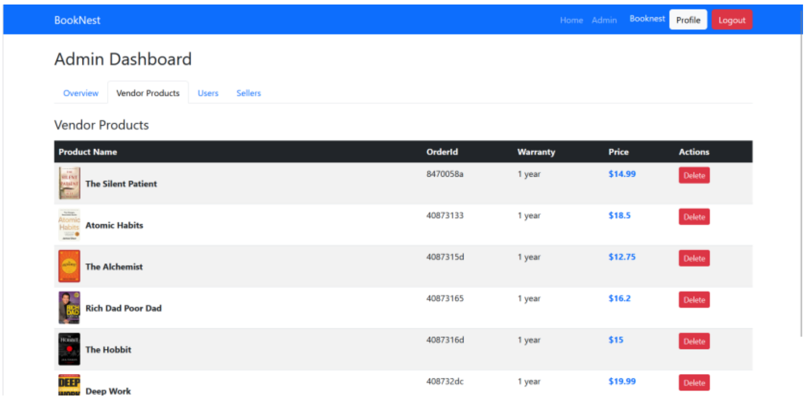
Users
4

Sellers
4

Items
7

Total Orders
4

Admin Dashboard Page

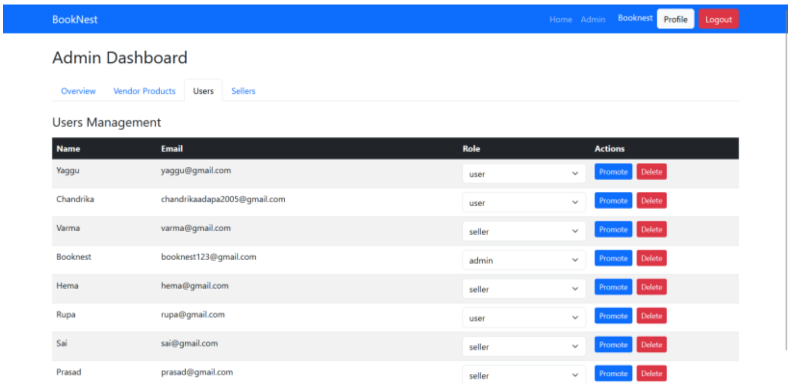


Admin Dashboard

Vendor Products

Product Name	OrderId	Warranty	Price	Actions
The Silent Patient	8470058a	1 year	$14.99	Delete
Atomic Habits	40873133	1 year	$18.5	Delete
The Alchemist	4087315d	1 year	$12.75	Delete
Rich Dad Poor Dad	40873165	1 year	$16.2	Delete
The Hobbit	4087316d	1 year	$15	Delete
Deep Work	408732dc	1 year	$19.99	Delete

Admin Dashboard Page

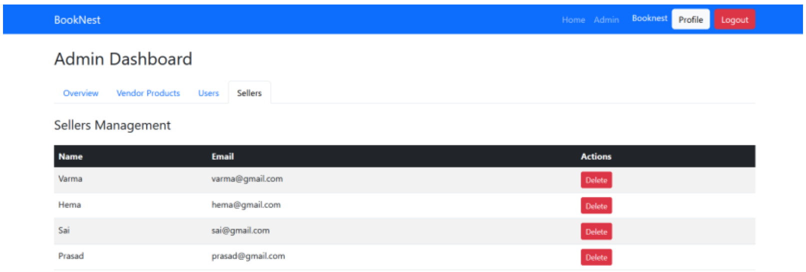


Admin Dashboard

Users Management

Name	Email	Role	Actions
Yaggu	yaggu@gmail.com	user	Promote Delete
Chandrika	chandrikaadapa2005@gmail.com	user	Promote Delete
Varma	varma@gmail.com	seller	Promote Delete
Booknest	booknest123@gmail.com	admin	Promote Delete
Hema	hema@gmail.com	seller	Promote Delete
Rupa	rupa@gmail.com	user	Promote Delete
Sai	sai@gmail.com	seller	Promote Delete
Prasad	prasad@gmail.com	seller	Promote Delete

Admin Dashboard Page



Admin Dashboard

Sellers Management

Name	Email	Actions
Varma	varma@gmail.com	Delete
Hema	hema@gmail.com	Delete
Sai	sai@gmail.com	Delete
Prasad	prasad@gmail.com	Delete

8. ADVANTAGES & DISADVANTAGES

Advantages

✓ User-friendly interface
✓ Secure authentication
✓ Real-time order tracking
✓ Scalable architecture
✓ Role-based system

Disadvantages

✕ Requires internet connection
✕ Initial server setup complexity

9. CONCLUSION

BookNest successfully demonstrates the implementation of a full-stack e-commerce application using the MERN stack. The platform provides a scalable, secure, and efficient online bookstore solution.

The project integrates authentication, REST APIs, database management, and cloud deployment, showcasing practical full-stack development skills.

10. FUTURE SCOPE

- AI-based book recommendation system
- Multi-language support
- Mobile application version
- Advanced analytics dashboard
- Real-time chat support
- Subscription-based book services

11. APPENDIX

Source Code:

```
const express = require('express');
const cors = require('cors');
const dotenv = require('dotenv');
const mongoose = require('mongoose');
const path = require('path');
const fs = require('fs');
dotenv.config();
const app = express();
// Setup uploads directory - use absolute path
const uploadsDir = path.resolve(__dirname, '../../uploads');
console.log('Setting up uploads directory:', uploadsDir);
if (!fs.existsSync(uploadsDir)) {
  fs.mkdirSync(uploadsDir, { recursive: true });
  console.log('Created uploads directory');
}
// CORS setup
app.use(cors({
  origin: ['http://localhost:5173', 'http://localhost:3000', 'http://localhost:5000'],
  credentials: true
}));
// Body parser - JSON
app.use(express.json());
app.use(express.urlencoded({ extended: true }));
```

```
// Custom image serving route with proper headers
app.get('/uploads/:filename', (req, res) => {
  const filename = req.params.filename;
  const filepath = path.join(uploadsDir, filename);
  // Security: prevent directory traversal
  if (!path.resolve(filepath).startsWith(path.resolve(uploadsDir))) {
    return res.status(403).json({ error: 'Forbidden' });
  }
  // Check if file exists
  if (!fs.existsSync(filepath)) {
    console.log('[Image] File not found:', filepath);
    return res.status(404).json({ error: 'Image not found' });
  }
  // Set proper headers
  res.setHeader('Cache-Control', 'public, max-age=3600');
  res.setHeader('Access-Control-Allow-Origin', '*');
  // Send the file
  res.sendFile(filepath);
});
console.log('Image serving route configured for:', uploadsDir);
// Test endpoint to check uploads folder
app.get('/api/test/uploads', (req, res) => {
  try {
    const files = fs.readdirSync(uploadsDir);
    res.json({
      path: uploadsDir,
      exists: true,
      fileCount: files.length,
      files: files.slice(0, 10) // first 10 files
    });
  } catch (err) {
    res.json({
      path: uploadsDir,
      exists: false,
      error: err.message
    });
  }
});
// connect to mongo
const mongoUri = process.env.MONGO_URI || 'mongodb://localhost:27017/booknest';
mongoose.connect(mongoUri, { useNewUrlParser: true, useUnifiedTopology: true })
  .then(() => console.log('MongoDB connected'))
  .catch(err => console.error('MongoDB connection error:', err));
// routes (will create files if they don't exist)
try {
  app.use('/api/auth', require('./routes/auth'));
} catch (e) {
  console.warn('Auth routes missing:', e.message);
}
try {
  app.use('/api/books', require('./routes/books'));
} catch (e) {
  console.warn('Books routes missing:', e.message);
}
try {
  app.use('/api/users', require('./routes/users'));
} catch (e) {
  console.warn('Users routes missing:', e.message);
}
try {
  app.use('/api/orders', require('./routes/orders'));
} catch (e) {
  console.warn('Orders routes missing:', e.message);
}
try {
  app.use('/api/reviews', require('./routes/reviews'));
} catch (e) {
  console.warn('Reviews routes missing:', e.message);
}
try {
  app.use('/api/wishlist', require('./routes/wishlist'));
} catch (e) {
  console.warn('Wishlist routes missing:', e.message);
}
const PORT = process.env.PORT || 5000;
// Log upload folder info
console.log('Uploads folder path:', path.join(__dirname, '../../uploads'));
console.log('Uploads folder exists:', require('fs').existsSync(path.join(__dirname, '../../uploads')));
app.listen(PORT, () => console.log(`Server started on port ${PORT}`));
```

GitHub Link : https://github.com/JnanaChandrikaAdapa/Booknest-Where-Stories-Nestle

Project Demo Link : https://github.com/JnanaChandrikaAdapa/Booknest-Where-Stories-Nestle