IdeaWeave

Elevate your idea from a spark to spectacle

Mini Project Report

Submitted by

Sreerag K U

Reg. No.: AJC22MCA-2089

In Partial Fulfillment for the Award of the Degree of

MASTER OF COMPUTER APPLICATIONS

(MCA TWO YEAR)

[Accredited by NBA]

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

AMAL JYOTHI COLLEGE OF ENGINEERING

KANJIRAPPALLY

[Affiliated to APJ Abdul Kalam Technological University, Kerala. Approved by AICTE, Accredited by NAAC. Koovappally, Kanjirappally, Kottayam, Kerala – 686518]

2023-2024

DEPARTMENT OF COMPUTER APPLICATIONS

AMAL JYOTHI COLLEGE OF ENGINEERING

KANJIRAPPALLY

CERTIFICATE

This is to certify that the Project report, "**IDEAWEAVE"** is the bona fide work of **SREERAG K U (Regno: AJC22MCA-2089)** in partial fulfilment of the requirements for the award of the Degree of Master of Computer Applications under APJ Abdul Kalam Technological University during the year 2023-24.

Mr. Binumon Joseph
Internal Guide

Ms. Meera Rose Mathew
Coordinator

Rev. Fr. Dr. Rubin Thottupurathu Jose
Head of the Department

DECLARATION

I hereby declare that the project report **"IDEAWEAVE"** is a bona fide work done at Amal Jyothi College of Engineering, towards the partial fulfilment of the requirements for the award of the Master of Computer Applications (MCA) from APJ Abdul Kalam Technological University, during the academic year 2023-2024.

Date: 08/12/2023 **SREERAG K U**
KANJIRAPPALLY **Reg: AJC22MCA-2089**

ACKNOWLEDGEMENT

First and foremost, I thank God almighty for his eternal love and protection throughout the project. I take this opportunity to express my gratitude to all who helped me in completing this project successfully. It has been said that gratitude is the memory of the heart. I wish to express my sincere gratitude to our Manager **Rev. Fr. Dr. Mathew Paikatt** and Principal **Dr. Lillykutty Jacob** for providing good faculty for guidance.

I extend my sincere gratitude to **Rev. Fr. Dr. Rubin Thottupurathu Jose**, our Head of the Department, for the significant support and assistance provided. I extend my whole hearted thanks to the project coordinator **Ms. Meera Rose Mathew** for her valuable suggestions and for overwhelming concern and guidance from the beginning to the end of the project. I would also express sincere gratitude to my guide **Mr. Binumon Joseph** for his inspiration and helping hand.

I thank our beloved teachers for their cooperation and suggestions that helped me throughout the project. I express my thanks to all my friends and classmates for their interest, dedication, and encouragement shown towards the project. I convey my hearty thanks to my family for the moral support, suggestions, and encouragement to make this venture a success.

SREERAG K U

ABSTRACT

IdeaWeave is a robust content management system designed to prioritize content distribution and user retention. The platform is committed to providing a user-friendly environment that empowers creators to effectively connect with their target audience.

For all users, IdeaWeave offers essential features such as seamless registration, secure login functionality with reset password options, an intuitive dashboard for easy navigation, and comprehensive profile management tools. Additionally, users can engage in meaningful interactions through a commenting system, while the dynamic recommendation feature enhances content discoverability.

Administrators benefit from a suite of tools for efficient user management, content monitoring, and streamlined communication and support. The platform further allows customization options, enabling administrators to tailor the user experience to meet specific requirements.

Writers on IdeaWeave can leverage a suite of content creation tools, including formatting features, draft and revision management, content preview capabilities, and the ability to schedule publishing. The system also facilitates content categorization, aiding writers in organizing and optimizing their work.

Readers, on the other hand, have access to a robust search and discovery functionality, ensuring they can find content tailored to their interests. The personal library feature allows users to curate their favorite content, and a read history function provides insight into past interactions.

In conclusion, IdeaWeave stands out as a comprehensive content management system that not only addresses the fundamental needs of users such as registration and profile management but also provides advanced features to facilitate content creation, distribution, and consumption.

CONTENT

LIST OF ABBREVIATION

UML	Unified Modelling Language
SEO	Search Engine Optimization
UI / UX	User Interface / User Experience
etc	et cetera
CMS	Content Management System
MERN	MongoDB, Express.js, React, Node.js

CHAPTER 1
INTRODUCTION

1.1 PROJECT OVERVIEW

IdeaWeave is a robust content management system built on the MERN (MongoDB, Express.js, React, Node.js) stack with Next.js integration. Prioritizing content distribution and user retention, it offers a seamless user experience. Users benefit from features like secure registration, intuitive dashboards, and comprehensive profile management. Dynamic recommendations and a commenting system enhance user engagement.

Administrators access efficient tools for user management and content monitoring, with customization options for a tailored experience. Writers leverage content creation tools, formatting features, and scheduling capabilities. Content categorization optimizes organization. Readers enjoy a robust search, personal libraries, and a read history function for a personalized experience.

In conclusion, IdeaWeave is a comprehensive content management system addressing user fundamentals, coupled with advanced features. The integration of the MERN stack, along with Next.js, ensures a powerful and flexible platform for content creators and consumers.

1.2 PROJECT SPECIFICATION

The project specification for IdeaWeave outlines a comprehensive set of features catering to diverse user roles, ensuring a robust and user-friendly experience.

- **All Users**
    - Multi Step Registration with OTP verification
    - Dashboard
    - Comments
    - Recommendation
    - Profile Management
    - Light / Dark Mode
    - Read Latest Releases
- **Admin**
    - Authentication
    - User Management (Enable/Disable)
    - Content Monitoring
    - Customization
    - Comment Monitoring
    - Category Creation
    - Invite New User

    - Statistics
    - Media Library
- **Writers**
    - Content Creation
    - Multiple Volumes and Chapters
    - Formatting Tools
    - Drafts and Revisions
    - Content Preview
    - Schedule Publishing
    - Edit Content and Categorization
    - Media Library
- **Readers**
    - Search and Discovery
    - Personal Library
    - Read History

CHAPTER 2
SYSTEM STUDY

2.1 INTRODUCTION

Data collection and analysis, problem-solving, and system change recommendations are all steps in the process of system analysis. During this problem-solving process, there must be considerable communication between the system users and the system developers. A system analysis or research should be the first step in any system development process. The system analyst acts as an interrogator and examines the operation of the current system in great detail. The system's input is acknowledged, and the system is viewed as a whole. The many processes might be connected to the organizations' outcomes. System analysis involves comprehending the problem, identifying the significant and crucial variables, analyzing and synthesizing the numerous components, and choosing the best or, at the very least, most acceptable course of action.

Preliminary research is the process of gathering and analyzing data in order to use it for upcoming system investigations. Initial research requires strong collaboration between system users and developers since it involves problem-solving. It carries out several feasibility studies. These studies offer a rough idea of the system activities, which can be utilized to choose the methods to employ for effective system research and analysis.

2.2 EXISTING SYSTEM

Current content management systems often exhibit limitations in user-friendliness and collaborative features. Interfaces can be complex, hindering efficient content creation and organization. Customization options are limited, leading to a generic user experience. Discovery mechanisms may lack sophistication, resulting in reduced engagement. Administrative tools for user management and content monitoring may be cumbersome. In response, IdeaWeave addresses these challenges with a focus on intuitive design, advanced content creation tools, and robust customization. The platform aims to enhance collaboration, user engagement, and administrative efficiency, providing a more user-centric and streamlined content management experience.

2.2.1 NATURAL SYSTEM STUDIED

The natural system studied for content management involves analyzing the organic flow of content creation, distribution, and consumption. This includes understanding user interactions, the lifecycle of content, and the dynamics between creators, administrators, and consumers. By delving into this natural system, we gain insights into the intricacies of content management, allowing for a more nuanced and effective approach in the development of the IdeaWeave platform.

2.2.2 DESIGNED SYSTEM STUDIED

The designed system, IdeaWeave, represents a groundbreaking approach to content management. Built on the MERN (MongoDB, Express.js, React, Node.js) stack with Next.js integration, it revolutionizes content creation, distribution, and consumption. IdeaWeave offers a user-centric experience with features tailored for administrators, writers, and readers. Its intuitive interface ensures seamless collaboration, while advanced content creation tools, real-time engagement features, and dynamic recommendation algorithms distinguish it. The platform's robust customization options and efficient user management tools further enhance the overall content management experience.

2.3 DRAWBACKS OF EXISTING SYSTEM

- Limited Customization
- Outdated User Interfaces
- Inefficient Collaboration
- Poor User Engagement
- Complex Administrative Tools

2.4 PROPOSED SYSTEM

The proposed system, IdeaWeave, stands as a revolutionary advancement in the field of content management. Crafted meticulously on the MERN (MongoDB, Express.js, React, Node.js) stack with Next.js integration, IdeaWeave redefines the content creation, distribution, and consumption landscape. The platform focuses on a user-centric approach, offering a seamless experience for administrators, writers, and readers alike. With an intuitive interface and cutting-edge features, IdeaWeave empowers content creators through advanced tools, real-time engagement options, and dynamic recommendation algorithms. The platform's robust customization capabilities and efficient user management tools aim to elevate the overall content management experience, setting new standards in the digital content ecosystem.

2.5 ADVANTAGES OF PROPOSED SYSTEM

- User-Centric Experience
- Advanced Content Creation Tools
- Real-time Engagement Features
- Customization Capabilities

- Efficient User Management
- Dynamic Recommendation Algorithms
- MERN Stack with Next.js Integration
- Innovative Features for Content Discovery

CHAPTER 3
REQUIREMENT ANALYSIS

3.1 FEASIBILITY STUDY

A feasibility study is conducted to determine if the project will, upon completion, fulfil the objectives of the organization in relation to the labor, effort, and time invested in it. A feasibility study enables the developer to predict the project's usefulness and potential future. A system proposal's workability, which includes the influence on the organization, capacity to satisfy user demands, and efficient use of resources, is the basis for a feasibility study. As a result, a feasibility analysis is frequently performed before a new application is approved for development. The paper outlines the project's viability and contains a number of factors that were carefully taken into account throughout this project's feasibility assessment, including its technical, economic, and operational viabilities. The following are its features: -

3.1.1 ECONOMICAL FEASIBILITY

Economically, IdeaWeave's feasibility is anchored in a comprehensive financial analysis. Initial development costs, encompassing coding, marketing, setup, and administration, have been meticulously estimated. Revenue projections are based on models incorporating subscription fees, advertising, and potential merchandise and content sales royalties. Operating expenses, including hosting, maintenance, marketing, and customer support, have been forecasted for sustainable operations. The break-even point, where total revenue equals costs, has been identified, and sensitivity analysis conducted. Potential funding sources, including investment and partnerships, have been explored to secure the required capital.

3.1.2 TECHNICAL FEASIBILITY

IdeaWeave's technical feasibility is solidified by its MERN stack foundation with Next.js integration. The system prioritizes data security through robust encryption, access control, and regular security audits. API integration with third-party services for payments, advertising, and machine translation enhances functionality. Continuous performance optimization ensures a responsive platform during peak usage. The content recommendation algorithm is developed and fine-tuned for personalized and accurate suggestions, providing a technically sound and innovative content management system.

3.1.3 BEHAVIORAL FEASIBILITY

Behaviorally, IdeaWeave is designed to align with user preferences and practices. User acceptance is a priority, emphasizing user-friendliness and intuitiveness for a positive experience. Continuous feedback mechanisms allow for iterative improvements based on user needs and expectations. The

platform actively encourages user engagement through forums, rewards, and community-building initiatives, fostering positive behavioral changes within the content creation and consumption community. User behaviors are continually analyzed to adapt to evolving market dynamics and maintain relevance in the rapidly changing landscape of content management.

3.1.4 FEASIBILITY STUDY QUESTIONNAIRE

1. Project Overview:

IdeaWeave stands as an innovative content management system (CMS) designed to reshape the landscape of digital content creation, distribution, and consumption. Utilizing the powerful MERN (MongoDB, Express.js, React, Node.js) stack with Next.js integration, IdeaWeave offers a dynamic and responsive platform. It prioritizes a seamless user experience for administrators, writers, and readers, fostering collaboration and engagement. The platform's core goal is to empower content creators by providing advanced creation tools, while offering readers personalized content recommendations. IdeaWeave aims to redefine the standards of content management, providing a user-centric ecosystem that facilitates the effective connection between creators and consumers in the digital content realm.

2. To what extent the system is proposed for?

IdeaWeave is proposed as a comprehensive and versatile content management system that extends its functionality across various dimensions. The system caters to content creators, administrators, and readers, providing a seamless and user-friendly environment for each user type. Its proposed features encompass advanced content creation tools, efficient user management for administrators, and personalized content recommendations for readers. With a focus on a user-centric design, IdeaWeave aims to address the diverse needs of individuals involved in the content creation and consumption process. The platform aspires to be a holistic solution that not only facilitates content management but also fosters collaboration, engagement, and efficient communication within the digital content ecosystem.

3. Specify the Viewers/Public which is to be involved in the System:

IdeaWeave's primary audience comprises three key user groups:

- **Admin**: Administrators play a crucial role in overseeing the entire content management system. They are involved in user management, content moderation, customization, and ensuring the overall functionality and security of the platform. Administrators use the system tools to monitor user activities, manage content, and customize the platform to meet specific requirements.

- **Writers/Content Creators:** Writers are actively engaged in content creation using IdeaWeave's suite of tools. They leverage features such as formatting tools, draft and revision management, content preview, and scheduling to efficiently produce and manage their content. The system enhances the writers' experience by providing a user-friendly interface and advanced functionalities for optimal content organization and publication.
- **Readers:** Readers constitute the audience exploring and consuming content on IdeaWeave. They benefit from robust search and discovery functionalities, personal libraries, and a read history feature. The system engages readers through dynamic content recommendations tailored to their interests, fostering a personalized and enjoyable content consumption experience.

4. List the Modules included in your System:

IdeaWeave consists of three core modules:

- **Admin Module:** The Admin Module is dedicated to system administrators, providing tools for efficient user management, content moderation, and overall system supervision. Administrators can customize the platform, monitor user activities, and ensure the security and integrity of the content management system.
- **Writer Module:** The Writer Module caters to content creators, offering a suite of advanced content creation tools. Writers can utilize features such as formatting tools, draft and revision management, content preview, and scheduling. This module empowers writers to efficiently produce, organize, and optimize their content for publication.
- **Reader Module:** The Reader Module focuses on enhancing the content consumption experience. Readers benefit from robust search and discovery functionalities, personal libraries for content curation, and a read history feature. Dynamic content recommendations based on user preferences further engage readers, providing a tailored and enjoyable platform for discovering and consuming content.

5. Identify the users in your project:

- **Admins:** System overseers with responsibilities in user management, content moderation, customization, and ensuring overall system functionality and security.
- **Writers:** Individuals responsible for generating and managing content using the platform's advanced tools, including formatting, draft management, content preview, scheduling, and categorization.
- **Readers:** Consumers of content on IdeaWeave who engage with the platform through content discovery, personal libraries, and dynamic recommendations tailored to their preferences.

6. Who owns the system?

The ownership details typically depend on the organization or individuals who initiated and funded the development of the platform.

- Entrepreneur or Founder
- Tech Start-up Company
- Enterprise or Corporation
- Non-Profit Organization
- Government or Public Entity

7. System is related to which firm/industry/organization?

IdeaWeave is directly related to the content management and digital publishing industry. It serves as a transformative platform for content creators, readers, and administrators, revolutionizing the way digital content is created, distributed, and consumed. IdeaWeave aligns with the goals of media organizations, content creators, and entities invested in fostering innovative and efficient solutions for managing and engaging with digital content. It contributes to the growth of the digital content ecosystem by providing a user-centric platform that enhances collaboration, customization, and overall content management processes.

8. Details of the person that you have contacted for data collection:

a. Personal Experience using various CMS

b. The process of data collection for IdeaWeave is multifaceted and involves collaboration with various stakeholders within the content management and digital publishing landscape. It encompasses gathering and curating essential data related to content creation, user interactions, and platform performance. The data collection effort requires engagement with diverse parties including:

- Content Creators and Writers
- Digital Marketers and SEO Specialists
- User Experience (UX/UI) Designers
- Website Administrators and Developers

9. Questionnaire to collect details about the project:

a. What types of content do you create or consume on IdeaWeave?

IdeaWeave caters to diverse content creation, including articles, blogs, and multimedia. Readers consume personalized content through dynamic recommendations.

b. Are there specific features or functionalities you would like to see added to IdeaWeave?

Enhanced collaboration tools, real-time editing, and improved mobile interface would be valuable additions to elevate IdeaWeave's content creation experience.

c. Provide more information about your target user base? Are there any specific demographics or regions you are primarily focusing on?

IdeaWeave targets a global audience, emphasizing content creators, administrators, and readers across diverse demographics, fostering a inclusive digital community.

d. Are there any unique features or functionalities you envision that will set IdeaWeave apart from other content management systems?

To set IdeaWeave apart from other agricultural commerce platforms, we envision unique features such as:

- AI-driven content suggestions for writers.
- Collaborative real-time editing.
- Enhanced reader interaction with content.
- Dynamic content categorization for seamless organization

e. How do you plan to ensure the security of user data and financial transactions on the platform?

To ensure the security of user data and financial transactions, IdeaWeave will implement:

- Robust encryption protocols.
- Regular security audits.
- Compliance with industry best practices.

f. Are there any specific technologies or integrations you require for the advanced content creation and formatting tools on IdeaWeave?

IdeaWeave requires seamless integration with advanced text and image processing technologies for sophisticated content creation and formatting tools.

g. How do you plan to gather and maintain information on user preferences for the recommendation algorithm?

IdeaWeave plans to gather user preferences through feedback, interaction data, and personalized settings, ensuring continuous adaptation for the recommendation algorithm.

h. What specific functionalities should the admin dashboard include?

The admin dashboard in the IdeaWeave platform should provide comprehensive tools and functionalities to enable administrators to effectively manage user accounts, oversee platform operations, and ensure a smooth and secure experience for all users. It should include

- Robust user management tools
- Content monitoring features for platform integrity
- Streamlined communication channels
- Customization options for tailored user experience

i. Are there any legal or regulatory requirements that the platform must adhere to, especially in the agricultural sector?

Yes, IdeaWeave must adhere to legal and regulatory requirements in the content management sector. Key considerations may include:

- Data Privacy and Protection
- Intellectual Property Regulations
- Compliance with Digital Accessibility Standards
- User Agreement and Terms of Service Compliance
- Content Copyright and Licensing Regulations

3.2 SYSTEM SPECIFICATION

3.2.1 HARDWARE SPECIFICATION

Device Processor	-	Intel i3 or more
Device RAM	-	4 GB or more
Device Storage	-	8 GB or more

3.2.2 SOFTWARE SPECIFICATION

Front End	-	React.js, Next.js
Back End	-	Node.js, Express.js
Database	-	MongoDB
Technologies used	-	MERN stack, Next.js, Ant Design

3.3 SOFTWARE DESCRIPTION

3.3.1 NEXT.JS

Next.js, introduced in 2016, is a powerful and open-source React framework for building web applications. It enables server-side rendering and simplifies the creation of dynamic and SEO-friendly websites. Next.js offers a robust development experience with features like automatic code splitting for faster page loads and a straightforward API for server-side rendering and routing. Developers benefit from a smooth transition between static and server-rendered pages, enhancing both performance and user experience. With Next.js, web applications can be efficiently built, offering a seamless and engaging user interface.

3.3.2 MERN STACK

The MERN stack is a full-stack development framework combining MongoDB as the database, Express.js for server-side scripting, React for the front-end, and Node.js for server-side runtime. MongoDB, a NoSQL database, offers flexibility and scalability for data storage. Express.js simplifies server-side scripting, and Node.js provides a runtime environment. React facilitates the creation of dynamic user interfaces. This stack allows developers to build feature-rich web applications efficiently, ensuring smooth communication between the client and server.

3.3.2.1 MongoDB

MongoDB is a NoSQL database platform, designed for scalability and flexibility. It employs a document-oriented data model, storing data in JSON-like BSON format, enabling efficient handling of large volumes of unstructured data. MongoDB's robust features include automatic sharding for horizontal scaling and support for complex queries, making it a preferred choice for dynamic, data-intensive applications.

3.3.2.2 Express JS

Express JS is a minimalistic and flexible Node.js web application framework. It simplifies the creation of robust and scalable web applications by providing a streamlined set of features for building web and mobile applications. Express enables efficient routing, middleware integration, and templating, facilitating the rapid development of server-side applications with Node.js. Its simplicity and extensibility make it a popular choice for developers seeking a lightweight, yet powerful, web framework.

3.3.2.3 React JS

React JS is a declarative JavaScript library for building user interfaces, developed by Facebook. It allows developers to create interactive and dynamic UI components efficiently. React's component-based architecture enables the creation of reusable UI elements, enhancing code modularity and maintainability. With its virtual DOM and one-way data binding, React optimizes performance, providing a seamless user experience. React is widely adopted for building modern, responsive, and scalable web applications.

3.3.2.4 Node JS

Node.js is a server-side JavaScript runtime that facilitates the development of scalable and high-performance network applications. It uses an event-driven, non-blocking I/O model, making it efficient for handling concurrent requests. Node.js leverages the V8 JavaScript engine and a vast ecosystem of packages through npm (Node Package Manager). It's commonly used for building real-time applications, APIs, and microservices, providing a unified language for both client and server-side development.

CHAPTER 4
SYSTEM DESIGN

4.1 INTRODUCTION

System design is a critical phase in the process of application development, playing a pivotal role in shaping the architecture and functionality of software. It encompasses the creation of a structured blueprint that outlines how various components, modules, and services within the application will interact and collaborate to meet specific objectives. This process involves making crucial decisions regarding database architecture, server configuration, and overall system architecture to ensure scalability, efficiency, and robustness. Additionally, system design takes into account factors such as user experience, security, and data management to create a well-rounded and effective application. It requires a deep understanding of the application's requirements, as well as proficiency in various technologies and programming paradigms to construct a robust foundation for the development process.

In essence, system design serves as the architectural backbone of any software project. It involves breaking down the complex functionalities and requirements of the application into manageable components, each with a defined purpose and relationship with other elements. Through meticulous planning and consideration of various technical and user-centric aspects, system design lays the groundwork for developers to implement code efficiently and cohesively. A well-crafted system design not only ensures that the application meets performance and scalability requirements but also provides a framework for future enhancements and maintenance. It is a critical step that bridges the gap between conceptualizing an application and transforming it into a functional, reliable, and user-friendly software solution.

4.2 UML DIAGRAM

Unified Modeling Language (UML) stands as a foundational tool in software engineering, renowned for its role in visually representing complex systems and processes. It provides a standardized set of graphical notations that facilitate the clear depiction of various aspects of a system's structure and behavior. Originating from the collaboration of industry experts, UML has gained widespread acceptance and adoption in both academia and industry. It serves as a powerful communication tool, enabling stakeholders, including developers, designers, and clients, to attain a shared understanding of system architecture, design, and functionality. UML diagrams act as a lingua franca, transcending language barriers and ensuring a consistent means of conveying intricate software concepts, ultimately enhancing the efficiency and effectiveness of the software development process.

Types of UML diagrams

- Class diagram
- Object diagram
- Use case diagram
- Sequence diagram
- Activity diagram
- State chart diagram
- Deployment diagram
- Component diagram

4.2.1 USE CASE DIAGRAM

Use Case Diagrams, a cornerstone in software engineering, serve as a visual representation of the interactions between a system and its external entities. At their core, they provide a structured means of identifying and defining the various functionalities a system offers and how these functionalities are accessed by different actors or entities. Actors, representing users, systems, or external entities, are depicted along with the specific use cases they engage with. Associations between actors and use cases elucidate the nature of these interactions, clarifying the roles and responsibilities of each entity within the system. This detailed visual representation not only enhances communication among stakeholders but also provides a clear blueprint for system functionality, laying the foundation for the subsequent stages of the software development process. Overall, Use Case Diagrams play a pivotal role in aligning development efforts with user expectations, ensuring that the resulting software system fulfills its intended purpose effectively and efficiently.

- **Actor Definition:** Clearly define and label all actors involved in the system. Actors represent external entities interacting with the system.
- **Use Case Naming:** Use descriptive names for use cases to accurately convey the functionality they represent.
- **Association Lines:** Use solid lines to represent associations between actors and use cases. This signifies the interaction between entities.
- **System Boundary:** Draw a box around the system to indicate its scope and boundaries. This defines what is inside the system and what is outside.
- **Include and Extend Relationships:** Use "include" relationships to represent common functionalities shared among multiple use cases. Use "extend" relationships to show optional or extended functionalities.

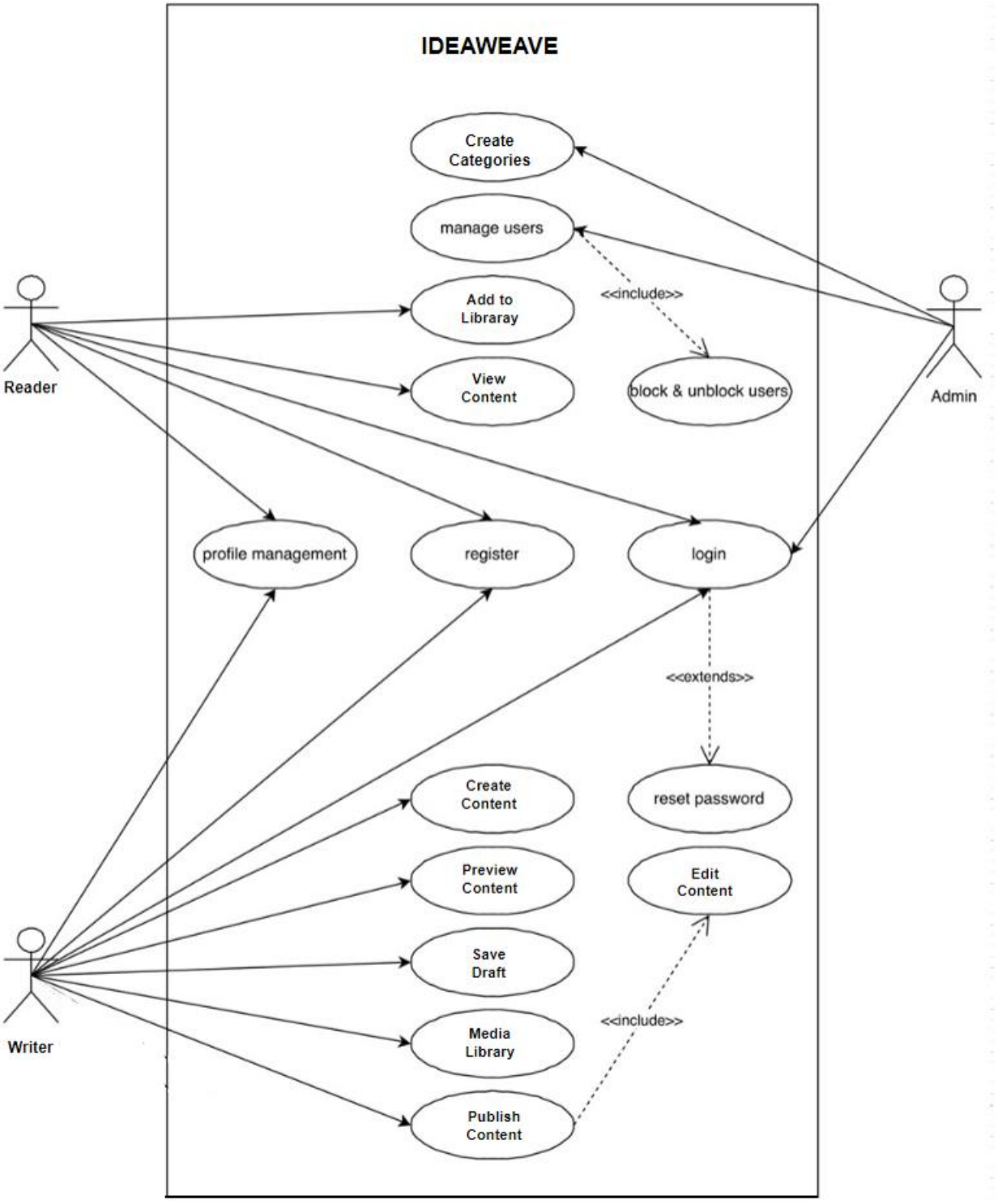


Diagram 4.2.1.1: *Use Case Diagram*

4.2.2 SEQUENCE DIAGRAM

Sequence Diagrams stand as dynamic models in software engineering, portraying the chronological flow of interactions between various objects or components within a system. They spotlight the order in which messages are exchanged, revealing the behaviour of the system over time. Actors and objects are represented along a vertical axis, with arrows indicating the sequence of messages and their direction. Lifelines, extending vertically from actors or objects, illustrate their existence over the duration of the interaction. These diagrams serve as a vital tool for visualizing system behaviour and understanding the temporal aspects of a software process. Through Sequence Diagrams, stakeholders gain valuable insights into how different elements collaborate to achieve specific functionalities, facilitating more effective communication among development teams and stakeholders alike. This detailed representation not only aids in detecting potential bottlenecks or inefficiencies but also provides a foundation for refining system performance in the later stages of software development.

- **Vertical Ordering:** Represent actors and objects along a vertical axis, indicating the order of interactions from top to bottom.
- **Lifelines:** Extend vertical lines from actors or objects to denote their existence and participation in the interaction.
- **Activation Bars:** Use horizontal bars along lifelines to show the period during which an object is active and processing a message.
- **Messages and Arrows:** Use arrows to indicate the flow of messages between objects, specifying the direction of communication.
- **Self-Invocation:** Use a looped arrow to represent self-invocation, when an object sends a message to itself.
- **Return Messages:** Indicate return messages with a dashed arrow, showing the response from the recipient.
- **Focus on Interaction:** Sequence Diagrams focus on the chronological order of interactions, avoiding implementation details.
- **Concise Notation:** Use clear and concise notation to represent messages and interactions, avoiding unnecessary complexity.
- **Consider System Boundaries:** Clearly define the boundaries of the system to indicate what is included in the interaction.
- **Feedback and Validation:** Seek feedback from stakeholders and team members to ensure the diagram accurately represents the system behavior.

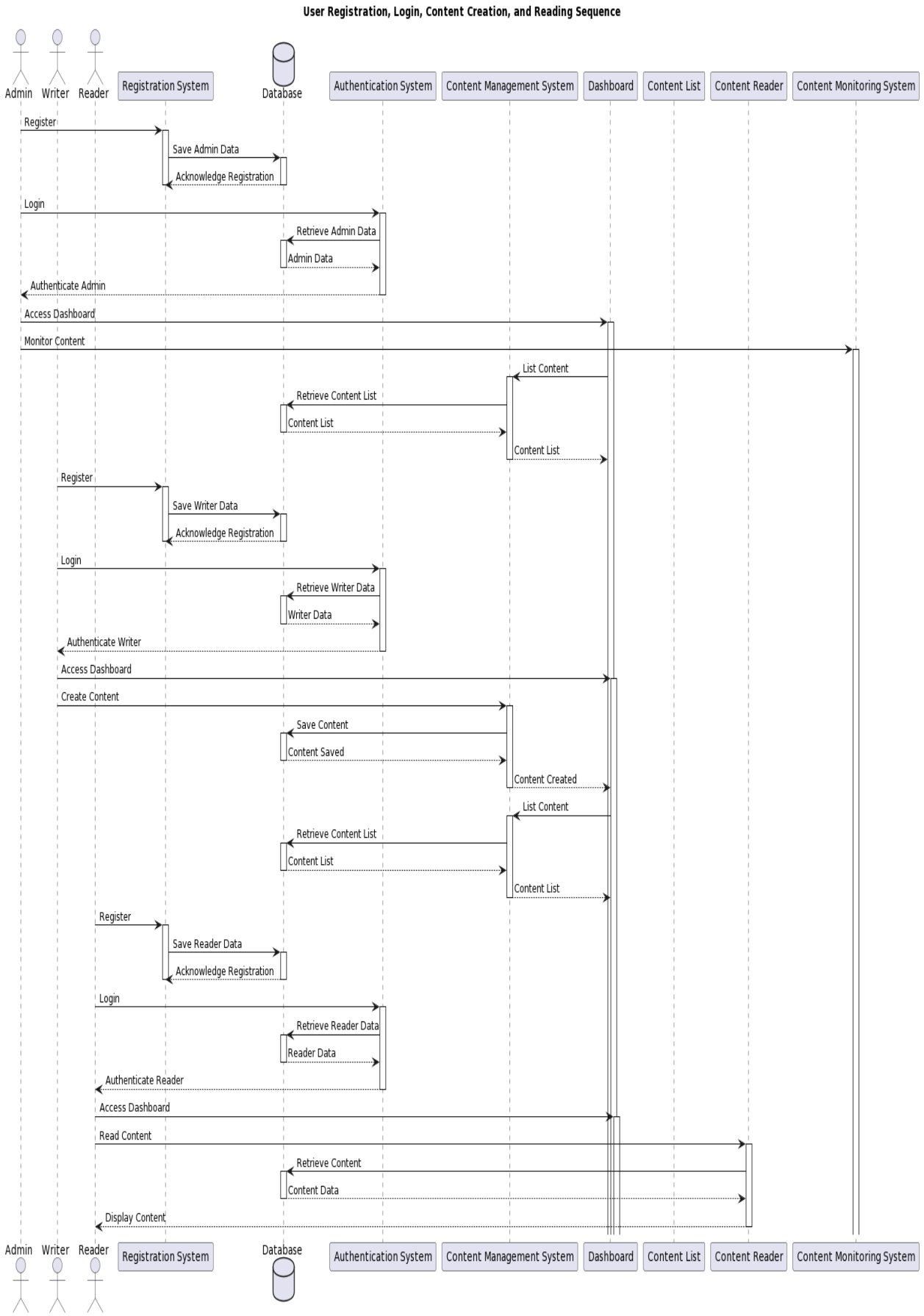


Diagram 4.2.2.1: *Sequence Diagram*

4.2.3 STATE CHART DIAGRAM

A State Chart Diagram, a fundamental component of UML, provides a visual representation of an object's lifecycle states and the transitions between them. It depicts the dynamic behavior of an entity in response to events, showcasing how it transitions from one state to another. Each state represents a distinct phase in the object's existence, while transitions illustrate the conditions triggering state changes. Initial and final states mark the commencement and termination of the object's lifecycle. Orthogonal regions allow for concurrent states, capturing multiple aspects of the object's behavior simultaneously. Hierarchical states enable the representation of complex behaviors in a structured manner. Entry and exit actions depict activities occurring upon entering or leaving a state. Moreover, guard conditions ensure that transitions occur only under specified circumstances. State Chart Diagrams play a crucial role in understanding and designing the dynamic behavior of systems, aiding in the development of robust and responsive software applications.

Key notations for State Chart Diagrams:

- **Initial State:** Represented by a filled circle, it signifies the starting point of the object's lifecycle.
- **State:** Depicted by rounded rectangles, states represent distinct phases in an object's existence.
- **Transition Arrow:** Arrows denote transitions between states, indicating the conditions triggering a change.
- **Event:** Events, triggers for state changes, are labeled on transition arrows.
- **Guard Condition:** Shown in square brackets, guard conditions specify criteria for a transition to occur.
- **Final State:** Represented by a circle within a larger circle, it indicates the end of the object's lifecycle.
- **Concurrent State:** Represented by parallel lines within a state, it signifies concurrent behaviors.
- **Hierarchy:** States can be nested within other states to represent complex behavior.
- **Entry and Exit Actions:** Actions occurring upon entering or leaving a state are labeled within the state.
- **Transition Labels:** Labels on transition arrows may indicate actions or operations that accompany the transition.

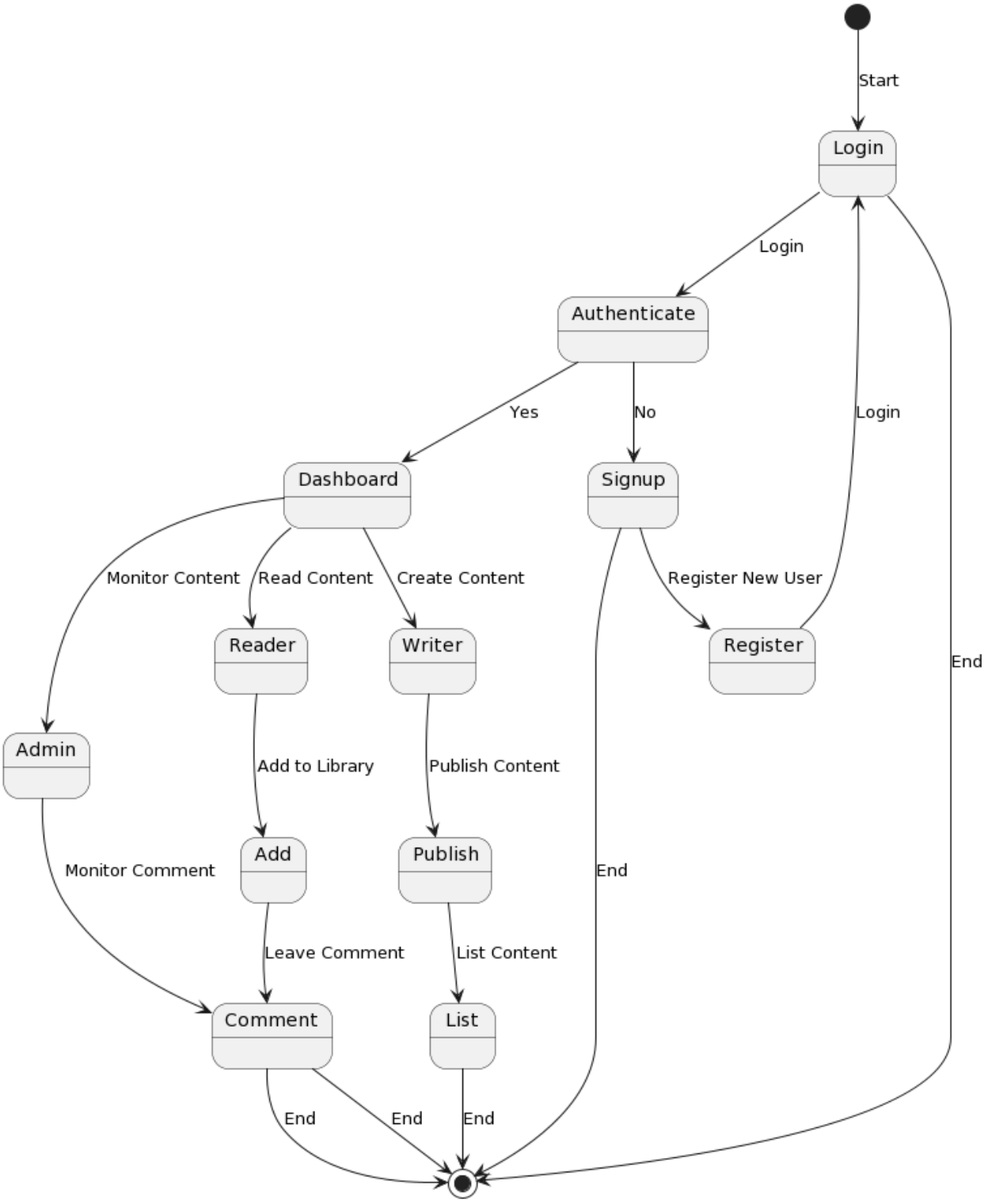


Diagram 4.2.3.1: *State Chart Diagram*

4.2.4 ACTIVITY DIAGRAM

An Activity Diagram is a visual representation within UML that illustrates the flow of activities and actions in a system or process. It employs various symbols to depict tasks, decision points, concurrency, and control flows. Rectangles signify activities or tasks, while diamonds represent decision points, allowing for conditional branching. Arrows indicate the flow of control from one

activity to another. Forks and joins denote concurrency, where multiple activities can occur simultaneously or in parallel. Swimlane segregate activities based on the responsible entity, facilitating clarity in complex processes. Initial and final nodes mark the commencement and completion points of the activity. Decision nodes use guards to determine the path taken based on conditions. Synchronization bars enable the coordination of parallel activities. Control flows direct the sequence of actions, while object flows depict the flow of objects between activities. Activity Diagrams serve as invaluable tools for understanding, modeling, and analyzing complex workflows in systems and processes. They offer a structured visual representation that aids in effective communication and system development.

Key notations for Activity Diagrams:

- **Initial Node:** Represented by a solid circle, it signifies the starting point of the activity.
- **Activity:** Shown as a rounded rectangle, it represents a task or action within the process.
- **Decision Node:** Depicted as a diamond shape, it indicates a point where the process flow can diverge based on a condition.
- **Merge Node:** Represented by a hollow diamond, it signifies a point where multiple flows converge.
- **Fork Node:** Shown as a horizontal bar, it denotes the start of concurrent activities.
- **Join Node:** Depicted as a vertical bar, it marks the point where parallel flows rejoin.
- **Final Node:** Represented by a solid circle with a border, it indicates the end of the activity.
- **Control Flow:** Arrows connecting activities, showing the sequence of actions.
- **Object Flow:** Lines with arrows representing the flow of objects between activities.
- **Swimlane:** A visual container that groups activities based on the responsible entity or system component.
- **Partition:** A horizontal or vertical area within a swimlane, further organizing activities.

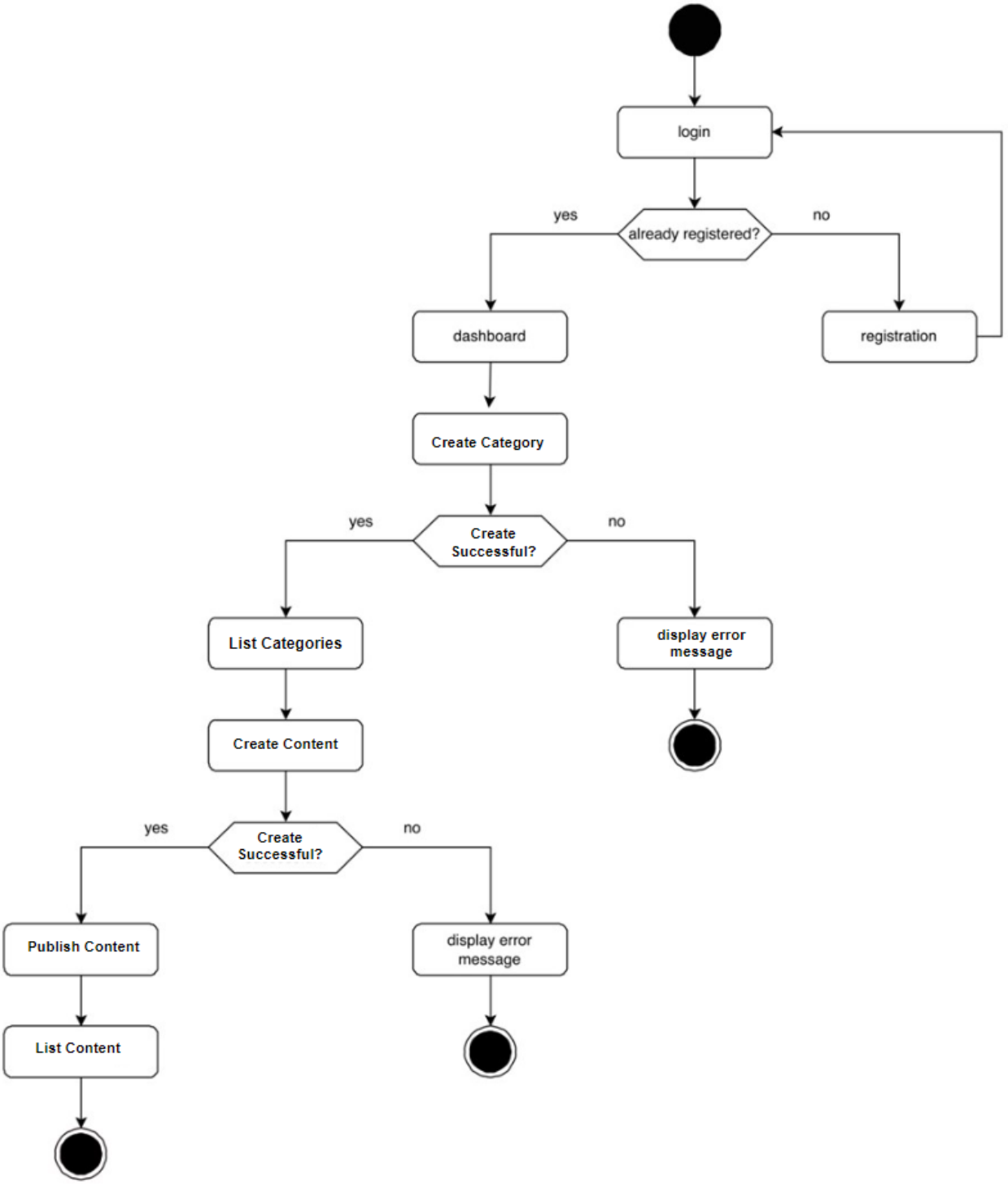


Diagram 4.2.4.1: *Activity Diagram*

4.2.5 CLASS DIAGRAM

A Class Diagram, a fundamental tool in UML, visually represents the structure of a system by illustrating classes, their attributes, methods, and relationships. Classes, depicted as rectangles, encapsulate data and behavior within a system. Associations between classes indicate relationships, showcasing how they interact. Multiplicity notations specify the cardinality of associations. Inheritance is denoted by an arrow indicating the subclass inheriting from a super-class. Aggregation and composition illustrate whole-part relationships between classes. Interfaces, depicted as a circle, outline the contract of behavior a class must implement. Stereotypes provide additional information about a class's role or purpose. Dependencies highlight the reliance of one class on another. Association classes facilitate additional information about associations. Packages group related classes together, aiding in system organization. Class Diagrams play a pivotal role in system design, aiding in conceptualizing and planning software architectures. They serve as a blueprint for the development process, ensuring a clear and structured approach to building robust software systems.

Key notations for Class Diagrams:

- **Class:** Represented as a rectangle, it contains the class name, attributes, and methods.
- **Attributes:** Displayed as a list within the class, they describe the properties or characteristics of the class.
- **Methods:** Also listed within the class, they define the behaviors or operations of the class
- **Associations:** Lines connecting classes, indicating relationships and connections between them.
- **Multiplicity Notation:** Indicates the number of instances one class relates to another.
- **Inheritance:** Shown as an arrow, it signifies that one class inherits properties and behaviors from another.
- **Interfaces:** Represented by a dashed circle, they define a contract of behavior that implementing classes must follow.
- **Stereotypes:** Additional labels or annotations applied to classes to provide more information about their role or purpose.
- **Dependencies:** Shown as a dashed line with an arrow, they indicate that one class relies on another in some way.
- **Association Classes:** Represented as a class connected to an association, they provide additional information about the relationship.

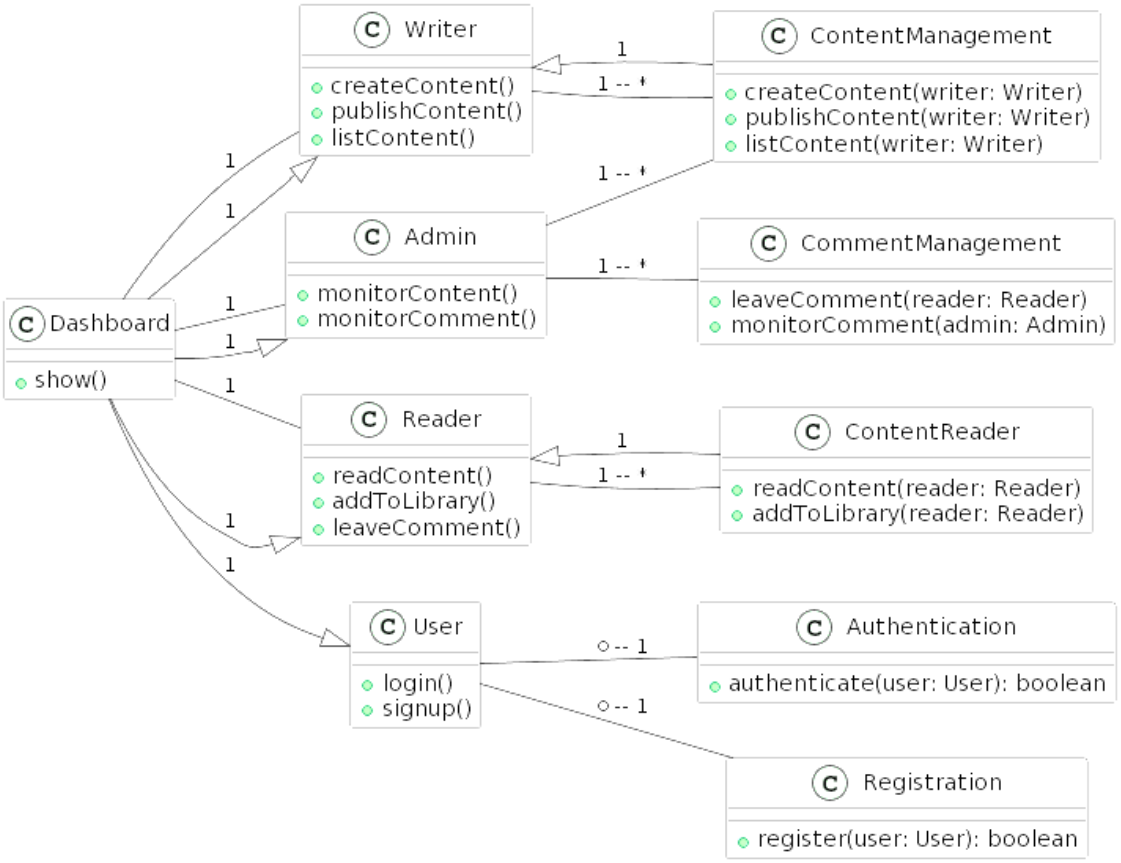


Diagram 4.2.5.1: *Class Diagram*

4.2.6 OBJECT DIAGRAM

An Object Diagram in UML provides a snapshot of a system at a specific point in time, displaying the instances of classes and their relationships. Objects, represented as rectangles, showcase the state and behavior of specific instances. Links between objects depict associations, highlighting how they interact. Multiplicity notations indicate the number of instances involved in associations. The object's state is displayed through attributes and their corresponding values. Object Diagrams offer a detailed view of runtime interactions, aiding in system understanding and testing. They focus on real-world instances, providing a tangible representation of class relationships. While similar to Class Diagrams, Object Diagrams emphasize concrete instances rather than class definitions. They serve as valuable tools for validating system design and verifying that classes and associations work as intended in practice. Object Diagrams play a crucial role in system validation, ensuring that the system's components and their interactions align with the intended design and requirements.

Key notations for Object Diagrams:

- **Object**: Represented as a rectangle, it contains the object's name and attributes with their values.

- **Links**: Lines connecting objects, indicating associations or relationships between them.
- **Multiplicity Notation**: Indicates the number of instances involved in associations.
- **Attributes with Values**: Displayed within the object, they represent the state of the object at a specific point in time.
- **Role Names**: Labels applied to associations, providing additional information about the nature of the relationship.
- **Object Name**: Represents the name of the specific instance.
- **Association End**: Indicates the end of an association, often with a role name and multiplicity.
- **Dependency Arrow**: Indicates a dependency relationship, where one object relies on another.
- **Composition Diamond**: Represents a stronger form of ownership, where one object encapsulates another.
- **Aggregation Diamond**: Signifies a whole-part relationship between objects.

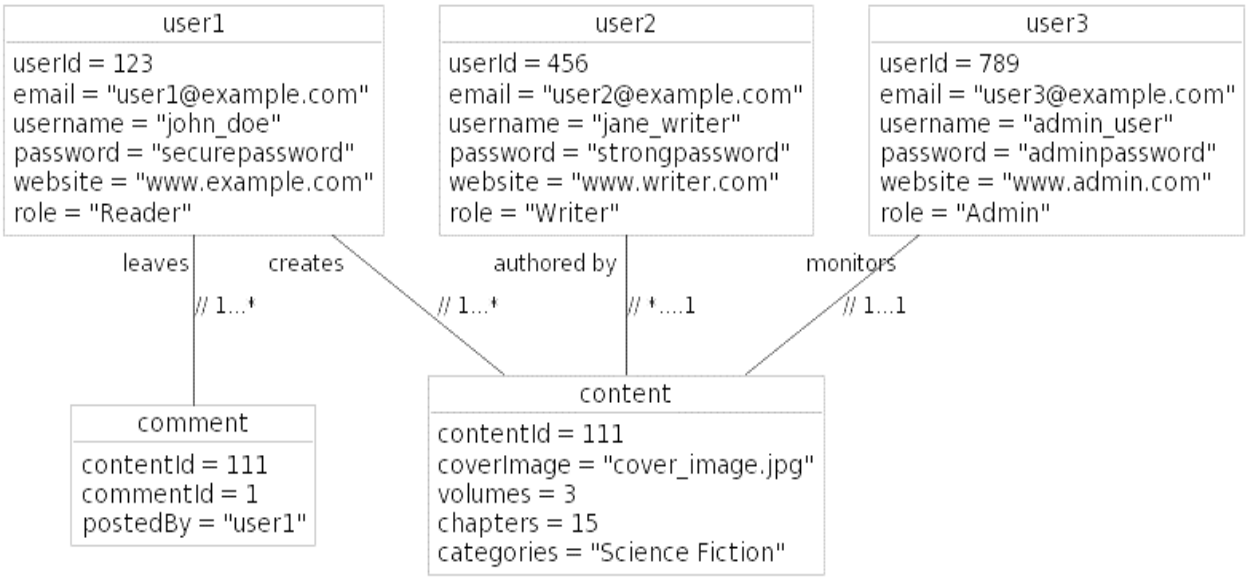


Diagram 4.2.6.1: *Object Diagram*

4.2.7 COMPONENT DIAGRAM

A Component Diagram, a vital aspect of UML, offers a visual representation of a system's architecture by showcasing the high-level components and their connections. Components, depicted as rectangles, encapsulate modules, classes, or even entire systems. Dependencies between components are displayed through arrows, signifying the reliance of one component on another. Interfaces, represented by a small circle, outline the services a component offers or requires. Connectors link interfaces to denote the required or provided services. Ports, depicted as small squares, serve as connection points between a component and its interfaces. Stereotypes provide additional information about the role or purpose of a component. Deployment nodes indicate the physical location or environment in which components are deployed. Component Diagrams are instrumental in system design, aiding in the organization and visualization of system architecture. They emphasize the modular structure, facilitating ease of development, maintenance, and scalability of complex software systems. Overall, Component Diagrams play a pivotal role in planning and orchestrating the architecture of sophisticated software applications.

Key notations for Component Diagrams:

- **Component**: Represented as a rectangle, it encapsulates a module, class, or system.
- **Dependency Arrow**: Indicates that one component relies on or uses another.
- **Interface**: Depicted as a small circle, it outlines the services a component offers or requires.
- **Provided and Required Interfaces**: Connectors link provided interfaces to required interfaces.
- **Port**: Shown as a small square, it serves as a connection point between a component and its interfaces.
- **Stereotypes**: Additional labels or annotations applied to components to provide more information about their role or purpose.
- **Assembly Connector**: Represents the physical connection between two components.
- **Artifact**: A physical piece of information that is used or produced by a software development process.
- **Deployment Node**: Indicates the physical location or environment in which components are deployed.
- **Manifestation Arrow**: Indicates the implementation of an interface by a component

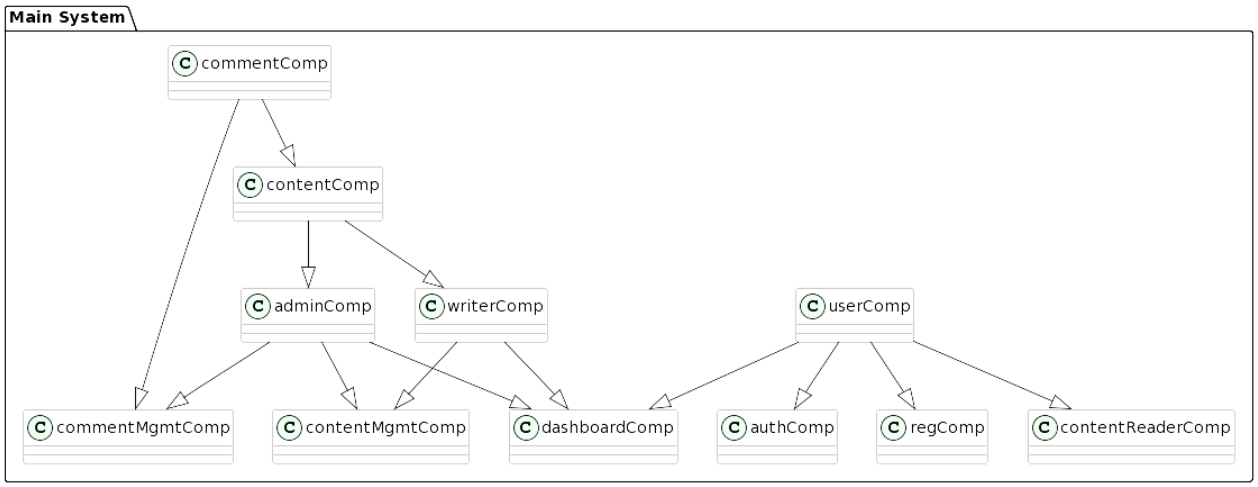


Diagram 4.2.7.1: *Component Diagram*

4.2.8 DEPLOYMENT DIAGRAM

A Deployment Diagram, a crucial facet of UML, provides a visual representation of the physical architecture of a system, showcasing the hardware nodes and software components. Nodes, representing hardware entities like servers or devices, are depicted as rectangles. Artifacts, denoted by rectangles with a folded corner, represent software components or files deployed on nodes. Associations between nodes and artifacts indicate the deployment of software on specific hardware. Dependencies illustrate the reliance of one node on another. Communication paths, shown as dashed lines, represent network connections between nodes. Stereotypes provide additional information about the role or purpose of nodes and artifacts. Deployment Diagrams are instrumental in system planning, aiding in the visualization and organization of hardware and software components. They emphasize the allocation of software modules to specific hardware nodes, ensuring efficient utilization of resources. Overall, Deployment Diagrams play a pivotal role in orchestrating the physical infrastructure of complex software applications.

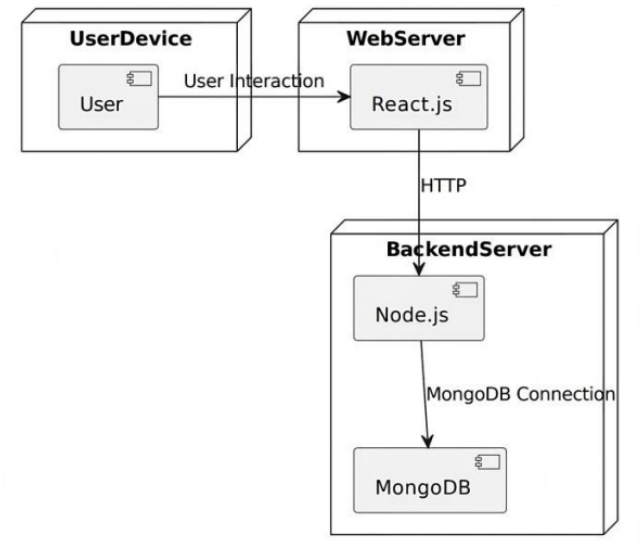
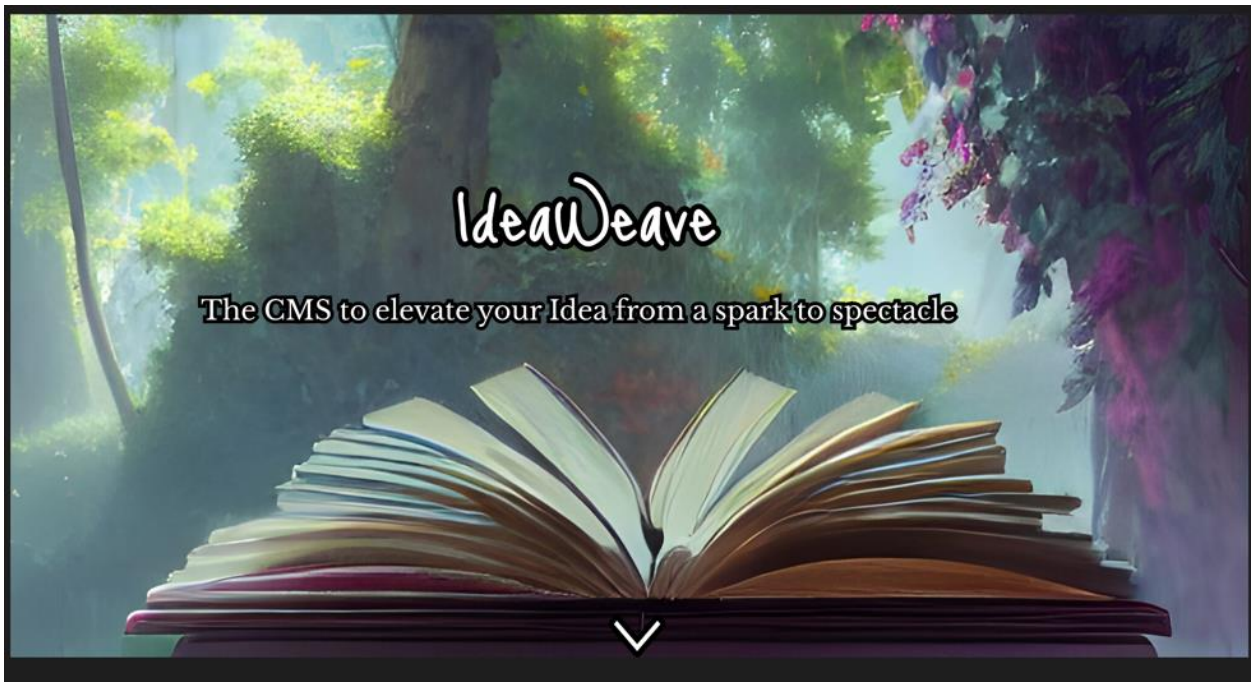
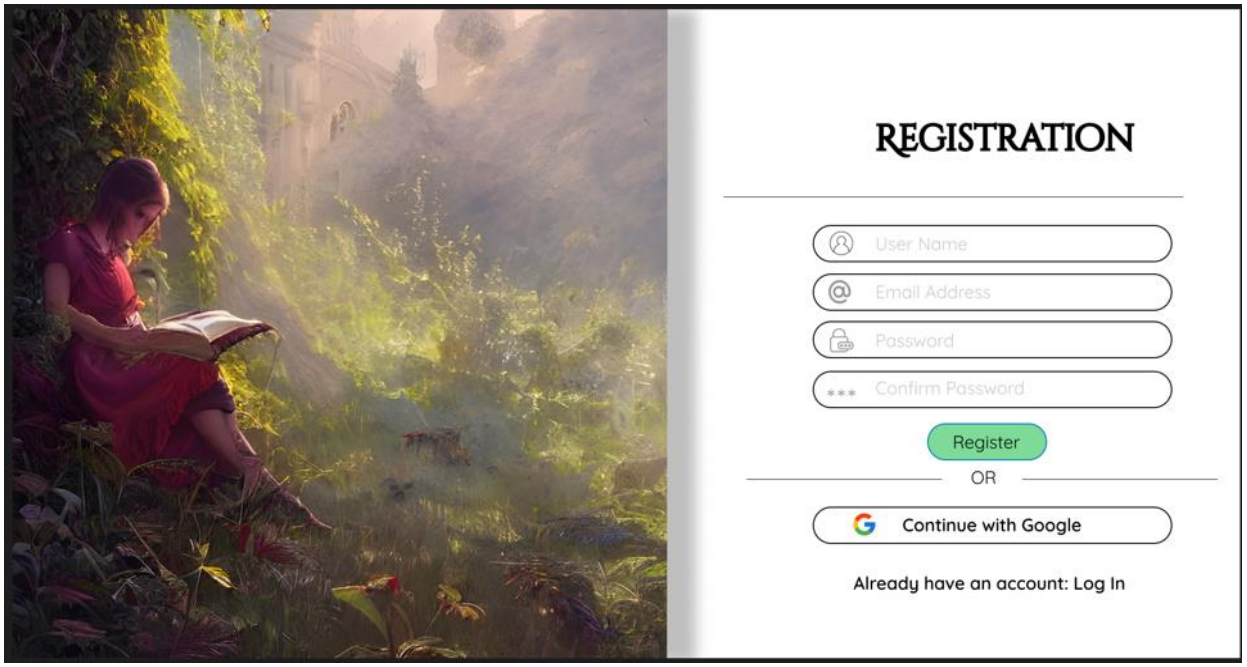


Diagram 4.2.8.1: *Deployment Diagram*

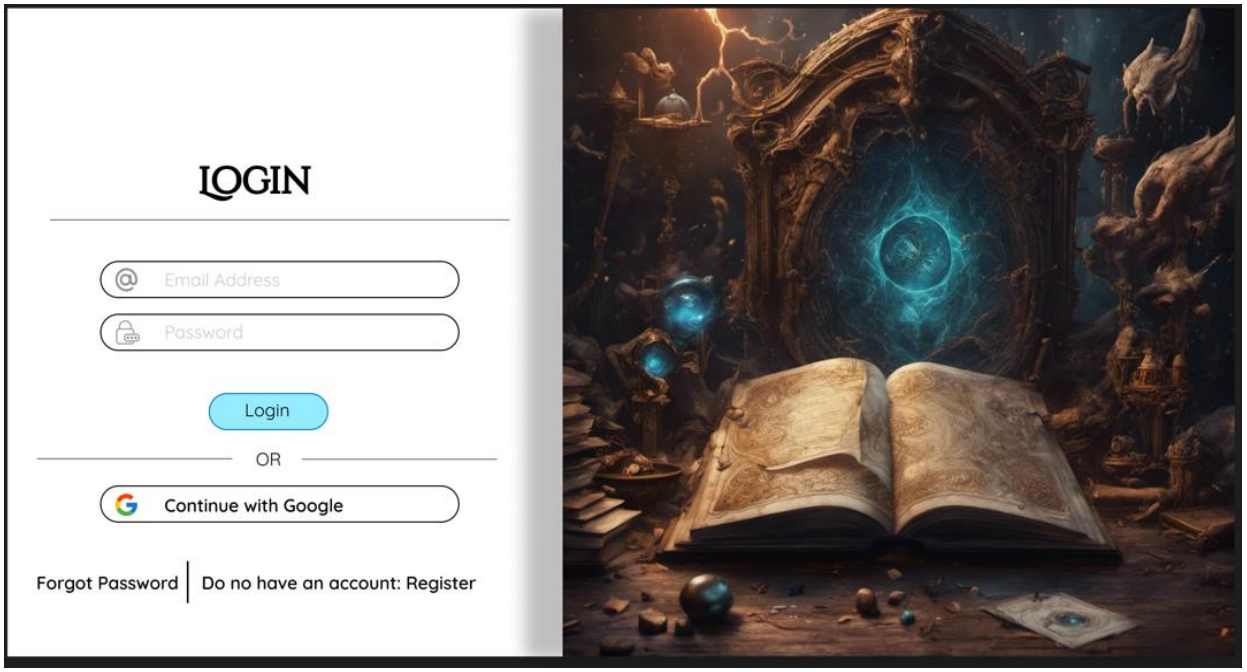
4.3 USER INTERFACE DESIGN USING FIGMA



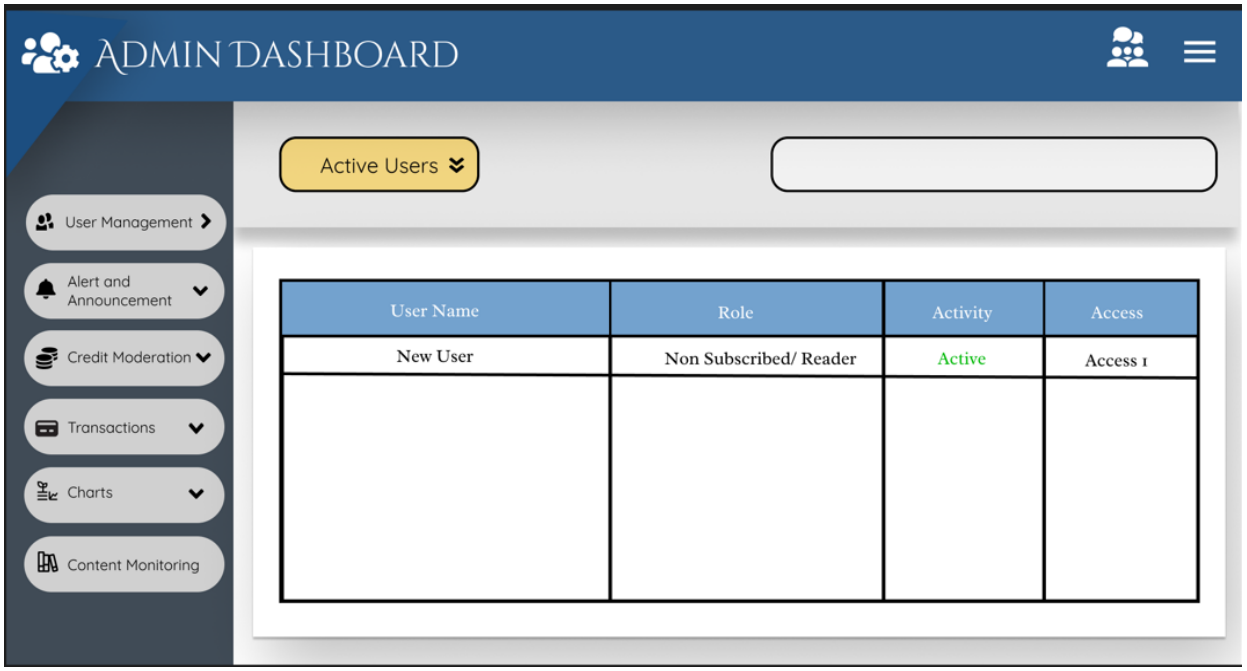
Figma Design 4.3.1: *Home Page*



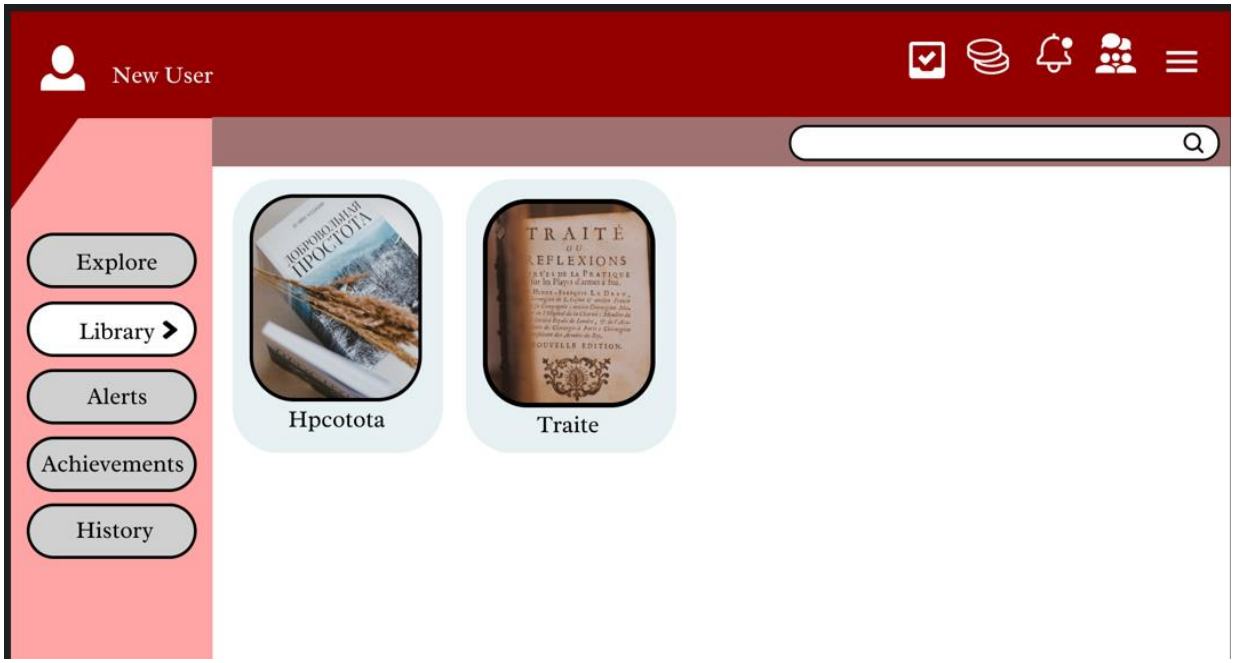
Figma Design 4.3.2: *Registration Page*



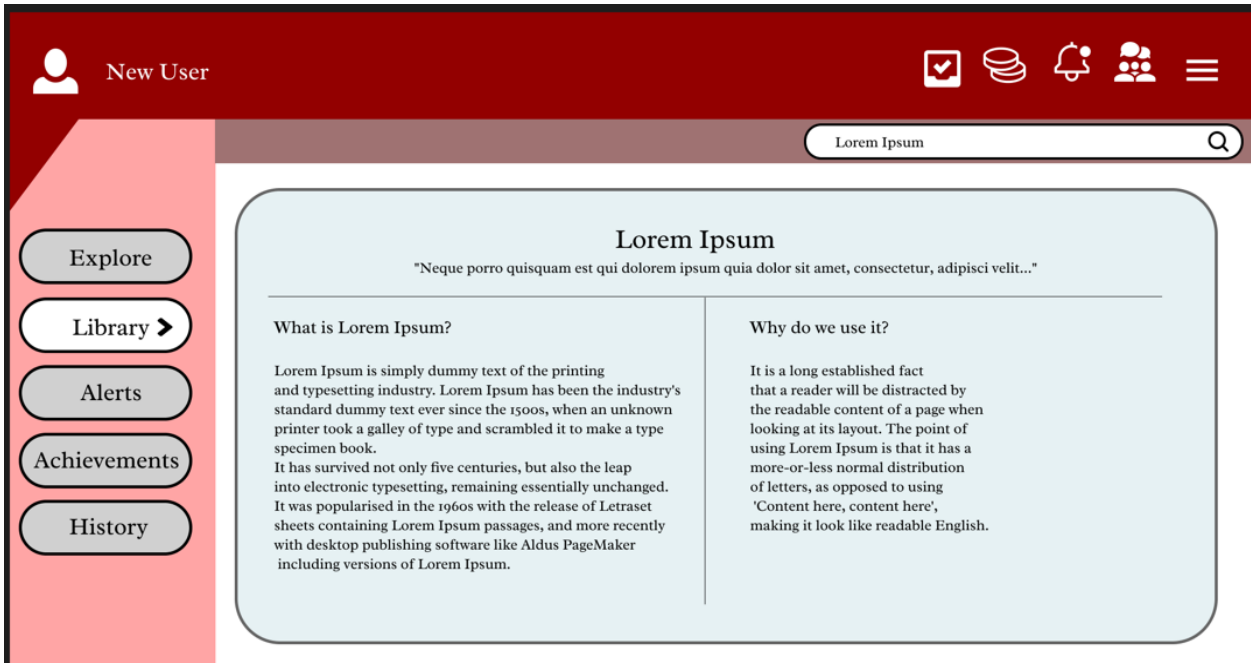
Figma Design 4.3.3: *Login Screen*



Figma Design 4.3.4: *Dashboard*



Figma Design 4.3.5: *Library*



Figma Design 4.3.6: *Selected Book*

4.4 DATABASE DESIGN

Database Design is a critical component in the realm of information management and software development. It involves the thoughtful and systematic organization of data to ensure efficient storage, retrieval, and manipulation. A well-designed database serves as the backbone of applications, enabling them to handle large volumes of information with speed and accuracy. This process encompasses defining the structure, relationships, and constraints of data entities, optimizing for performance and scalability. Effective database design is pivotal in minimizing redundancy, ensuring data integrity, and providing a foundation for robust data analytics. It involves a deep understanding of business requirements and user needs, translating them into a coherent and logical data model. The goal of a sound database design is to create a reliable, scalable, and maintainable system that supports the organization's objectives and facilitates seamless information flow.

4.4.1 NOSQL DATABASES (NOSQL)

NoSQL databases, or "Not Only SQL," represent a transformative approach to database management. They diverge from traditional relational databases, excelling in handling large volumes of unstructured or semi-structured data. NoSQL databases are highly scalable, allowing for horizontal scaling across multiple servers, making them ideal for rapidly growing data volumes in web applications and Big Data environments. These databases come in various types, such as document-oriented, key-value stores, wide-column stores, and graph databases. Each type caters to specific use cases. Document-oriented databases, like MongoDB, are adept at storing JSON-like documents, making them popular for content management systems and real-time analytics.

One key advantage of NoSQL databases is their schema flexibility. They allow for dynamic or semi-structured schemas, enabling data to be added or modified on the fly. This characteristic is invaluable in projects where data structures are likely to evolve over time. NoSQL databases have found wide adoption in domains like social media, IoT applications, gaming, and real-time analytics. However, it's crucial to choose between NoSQL and traditional relational databases based on the unique requirements of the application. In essence, NoSQL databases offer a potent alternative, providing scalability, flexibility, and high performance in scenarios demanding the handling of large volumes of diverse data. Their diverse types make them indispensable tools in modern data management.

4.4.2 INDEXING

Indexing in NoSQL databases is a crucial technique to enhance query performance and retrieval speed. Unlike relational databases, NoSQL databases utilize a variety of indexing methods tailored to different data models. In document-oriented databases like MongoDB, B-tree indexes are commonly used to accelerate search operations based on keys or fields within documents. Hash indexes, on the other hand, are prevalent in key-value stores like Redis, enabling swift retrieval of values associated with specific keys. Wide-column stores like Cassandra utilize techniques like row-level indexing to swiftly locate specific columns within a wide row. Graph databases like Neo4j employ specialized index structures optimized for graph traversal operations, allowing for rapid traversal of connected nodes. While indexing significantly improves read performance, it's important to weigh the trade-offs, as indexes can increase storage requirements and potentially slow down write operations. In summary, indexing plays a vital role in optimizing query performance and is a key aspect of designing efficient data retrieval systems in NoSQL databases.

4.4.3 TABLE/COLLECTION DESIGN

4.4.3.1 COLLECTION NAME: USERS

Field Name	Data Type	Description of the field
_id	ObjectId	Unique identifier for the user
name	String	Full name of the user
email	String	Email address of the user
password	String	Password of the user
role	String	Role of the user (default: "Subscriber")
image	ObjectId	Reference to the Media used as the user's image
website	String	User's website
resetCode	String	Code for password reset
isActive	Boolean	Indicates whether the user account is active
posts	[ObjectId]	Array of references to Post Schema
createdAt	Date	Timestamp of creation
updatedAt	Date	Timestamp of last update

Collection 4.4.3.1: *Users*

4.4.3.2 COLLECTION NAME: POST

Field Name	Data Type	Description of the field
_id	ObjectId	Unique identifier for the post
title	String	Title of the post
content	String	Content of the post
volumes	[Volume]	Array of Volume Schema
categories	[ObjectId]	Array of references to Category Schema
published	Boolean	Indicates whether the post is published
postedBy	ObjectId	Reference to the User who posted the post
coverImage	ObjectId	Reference to the Media used as the cover image
slug	String	Unique lowercase identifier for the post
commentCount	Number	Number of comments on the post
createdAt	Date	Timestamp of creation

Collection 4.4.3.2: *Post*

4.4.3.3 COLLECTION NAME: MEDIA

Field Name	Data Type	Description of the field
_id	ObjectId	Unique identifier for the media
url	String	URL of the media
public_id	String	Public identifier for the media
postedBy	ObjectId	Reference to the User who posted the media
createdAt	Date	Timestamp of creation
updatedAt	Date	Timestamp of last update

Collection 4.4.3.3: *Media*

4.4.3.4 COLLECTION NAME: COMMENT

Field Name	Data Type	Description of the field
_id	ObjectId	Unique identifier for the comment
content	String	Content of the comment
postedBy	ObjectId	Reference to the User who posted the comment
postId	ObjectId	Reference to the Post the comment belongs to
createdAt	Date	Timestamp of creation
updatedAt	Date	Timestamp of last update

Collection 4.4.3.4: *Comment*

4.4.3.5 COLLECTION NAME: CATEGORY

Field Name	Data Type	Description of the field
_id	ObjectId	Unique identifier for the category
name	String	Name of the category
slug	String	Unique lowercase identifier
createdAt	Date	Timestamp of creation
updatedAt	Date	Timestamp of last update

Collection 4.4.3.5: *Category*

4.4.3.6 COLLECTION NAME: WEBSITE

Field Name	Data Type	Description of the field
_id	ObjectId	Unique identifier for the website
page	String	Page identifier
title	String	Title of the website
subtitle	String	Subtitle of the website
fullWidthImage	ObjectId	Reference to the Media used as full-width image
createdAt	Date	Timestamp of creation
updatedAt	Date	Timestamp of last update

Collection 4.4.3.6: *Website*

CHAPTER 5
SYSTEM TESTING

5.1 INTRODUCTION

System Testing is a crucial phase in the software development life cycle, where the entire system is evaluated against specified requirements and functionalities. It is a comprehensive and structured approach to validate that the software meets its intended objectives. This phase involves testing the integrated system as a whole to ensure that all components work together seamlessly. System Testing verifies the system's compliance with both functional and non-functional requirements, including performance, security, and usability. It is conducted in an environment that closely simulates the production environment, providing a real-world scenario for testing. The primary goal of System Testing is to identify and rectify any discrepancies or defects before the software is deployed to end-users. Through rigorous testing processes and thorough documentation, System Testing helps in delivering a reliable and high-quality software product.

Testing is the systematic process of running a program to uncover potential errors or flaws. An effective test case possesses a high likelihood of revealing previously unnoticed issues. A test is considered successful when it reveals a previously unidentified error. If a test functions as intended and aligns with its objectives, it can detect flaws in the software. The test demonstrates that the computer program is operating in accordance with its intended functionality and performing optimally. There are three primary approaches to assessing a computer program: evaluating for accuracy, assessing implementation efficiency, and analyzing computational complexity.

5.2 TEST PLAN

A test plan is a thorough document that delineates the strategy, scope, objectives, resources, schedule, and expected outcomes for a specific testing endeavor. It functions as a guiding framework for carrying out testing activities, guaranteeing that every facet of the testing process is methodically organized and executed. Additionally, the test plan establishes the roles and responsibilities of team members, outlines the required testing environment, and sets forth the criteria for the successful completion of testing activities. This document plays a pivotal role in ensuring that the testing phase is conducted in a structured and effective manner, ultimately contributing to the overall success of the project.

The levels of testing include:

- Integration Testing
- Unit testing
- Validation Testing or System Testing

- Output Testing or User Acceptance Testing
- Automation Testing

5.2.1 UNIT TESTING

Unit Testing is not only a meticulous examination of discrete units or components within a software system but also an indispensable quality assurance measure. This phase serves as a crucial foundation for the entire software testing process, where the focus lies on isolating and scrutinizing individual units of code. The objective remains unwavering: to verify that each unit performs its designated function accurately, yielding precise outputs for predefined inputs.

Moreover, Unit Testing operates independently, detached from other components, and any external dependencies are either emulated or replaced by "mock" objects, ensuring controlled evaluation. This meticulous process establishes a robust foundation for the software, confirming that each unit functions reliably and adheres meticulously to its predefined behavior.

The significance of Unit Testing cannot be overstated, as it acts as a vanguard against potential discrepancies or errors early in the development cycle. This proactive approach not only fortifies the integrity and reliability of the software but also lays the groundwork for subsequent testing phases, thereby fostering a robust and dependable software solution. This meticulous process ensures that each unit functions reliably and adheres precisely to its defined behavior. By subjecting individual code units to rigorous scrutiny, any discrepancies or errors are identified and rectified early in the development cycle, bolstering the overall integrity and reliability of the software.

5.2.2 INTEGRATION TESTING

Integration Testing stands as a pivotal phase in the software testing process, dedicated to scrutinizing the interactions and interfaces among diverse modules or components within a software system. Its primary objective is to ascertain that individual units of code seamlessly converge to create a unified and functional system. In stark contrast to unit testing, which assesses individual units in isolation, integration testing delves into the interplay between these units, with a keen eye for any disparities, communication glitches, or integration hurdles. By subjecting the integrated components to rigorous testing, development teams aim to affirm that these elements function cohesively, addressing any potential issues before deployment. This systematic evaluation is instrumental in ensuring that the software operates as an integrated whole, free from any unforeseen conflicts or errors that may arise from the convergence of individual modules.

5.2.3 VALIDATION TESTING OR SYSTEM TESTING

Validation Testing places the end-users at the forefront of evaluation, ensuring that the software aligns precisely with their anticipated needs and expectations. This phase stands distinct from other testing methodologies, as its primary objective is to authenticate that the software, in its final form, serves its intended purpose seamlessly within the real-world scenarios it was designed for. As a culmination of the testing process, Validation Testing carries the responsibility of confirming that the software not only meets the defined technical specifications but also delivers genuine value to its users. It does so by scrutinizing the software against the backdrop of actual usage, thereby fortifying its readiness for deployment. Moreover, in Validation Testing, user stories and acceptance criteria form the cornerstone of assessment. Stakeholders' expectations are meticulously validated, ensuring that every specified requirement is met. Additionally, beta testing, a common practice in this phase, involves a select group of end-users testing the software in a live environment, providing invaluable feedback that can inform potential refinements.

5.2.4 OUTPUT TESTING OR USER ACCEPTANCE TESTING.

Output Testing, also known as Results Validation, is a critical phase in the software testing process. Its primary focus is to verify the correctness and accuracy of the output generated by a software application. The goal is to ensure that the system produces the expected results for a given set of inputs and conditions.

Key aspects of Output Testing include:

- **Comparison with Expected Results**: This phase involves comparing the actual output of the software with the expected or predefined results.
- **Test Case Design:** Test cases are designed to cover various scenarios and conditions to thoroughly evaluate the accuracy of the output.
- **Validation Criteria**: The criteria for validating the output are typically defined during the requirements and design phase of the software development process.
- **Regression Testing**: Output Testing often includes regression testing to ensure that changes or updates to the software do not affect the correctness of the output.
- **Data Integrity**: It verifies that data is processed and displayed correctly, without any corruption or loss.
- **Precision and Completeness**: Output Testing assesses not only the precision of the results but also their completeness in addressing the requirements.

- **Error Handling**: It evaluates how the system handles errors or exceptions and ensures that appropriate error messages are displayed.

5.2.5 AUTOMATION TESTING

Automation Testing stands as a cornerstone in the software testing process, harnessing the power of automated tools and scripts to meticulously execute test cases. In stark contrast to manual testing, which hinges on human intervention, automation testing brings forth a streamlined approach, employing software to conduct repetitive, intricate, and time-consuming tests. This methodology not only heightens operational efficiency but also significantly diminishes the likelihood of human error, ensuring precise and reliable results. Moreover, it empowers thorough testing across a diverse array of scenarios and configurations, from browser compatibility to load and performance assessments.

By automating the testing process, organizations can realize a myriad of benefits. It enables the seamless execution of regression tests, providing confidence that existing functionalities remain intact after each round of enhancements or modifications. Furthermore, automation facilitates the concurrent execution of multiple tests, thereby expediting the overall testing cycle. This approach is particularly invaluable in environments characterized by rapid development and frequent software updates, such as Agile and DevOps setups.

TEST CASE 1

```
> Feature: Login Feature
  > As a user
  > I want to login to the application
  > So that I can access my account

  > Scenario: Successful Login
    > Given I am on the login page
    > When I enter valid credentials
    > And click on the login button
    > Then I should be redirected to the home page
```

Code

```
const { Given, When, Then } = require("cucumber");
const { Builder, By, until } = require("selenium-webdriver");

let driver;

Given("I am on the login page", async function () {
   driver = await new Builder().forBrowser("chrome").build();
   await driver.get("http://localhost:3000/signin");
});
When("I enter valid credentials", async function () {
   await driver.findElement(By.id("em")).sendKeys("sreeragkuofficial@gmail.com");
   await driver.findElement(By.id("ps")).sendKeys("sku@MCA20");
   console.log("Found the Email and Password fields and Valid credentials entered successfully");
});
When("click on the login button", async function () {
   try {
      const button = await driver.wait(
         until.elementLocated(By.id("testid")),
         10000
      );
      console.log("Login Button Found and Waiting to be Clicked")
      await button.click();
      console.log("Login successful...!");
   } catch (error) {
      console.error("Error clicking on the login button:", error);
   }
});
Then("I should be redirected to the home page", async function () {
   await driver.wait(until.urlIs("http://localhost:3000/"));
   console.log("Successfully redirected to the home page");
   await driver.quit();
   console.log("Test Passed at Scenario 1: Login");
});
```

Test Report

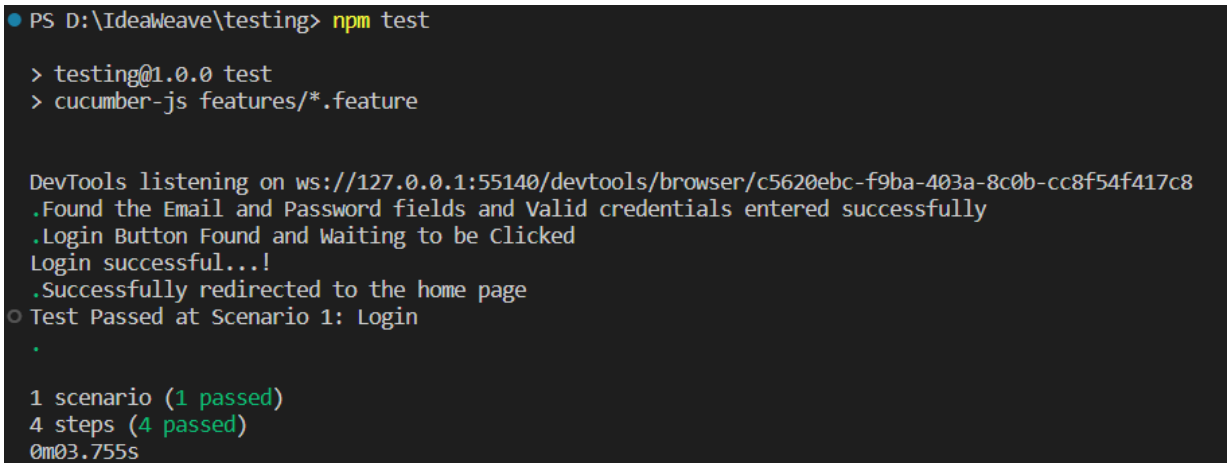
Test Case 1

Project Name: IdeaWeave	
Login Test Case	
Test Case ID: Test_1	**Test Designed By: Sreerag K U**
Test Priority (Low/Medium/High): High	**Test Designed Date: 23/11/2023**
Module Name: All Users	**Test Executed By: Mr. Binumon Joseph**
Test Title : Login	**Test Execution Date: 27/11/2023**
Description: Test the Login functionality	
Pre-Condition: Must be Registered User	

Step	Test Step	Test Data	Expected Result	Actual Result	Status (Pass/Fail)
1	Chrome Driver Opened		Browser Opened	Browser Opened	Pass
2	Navigation To sign in page		Navigate to Sign in page	Navigated to Sign in page	Pass
3	Entered valid email and password	sreeragkuofficial@gmail.com sku@MCA20	Error message should not be displayed	Error message not displayed	Pass

Post-Condition: Can access dashboard and use the functionalities the role provides

Screenshot



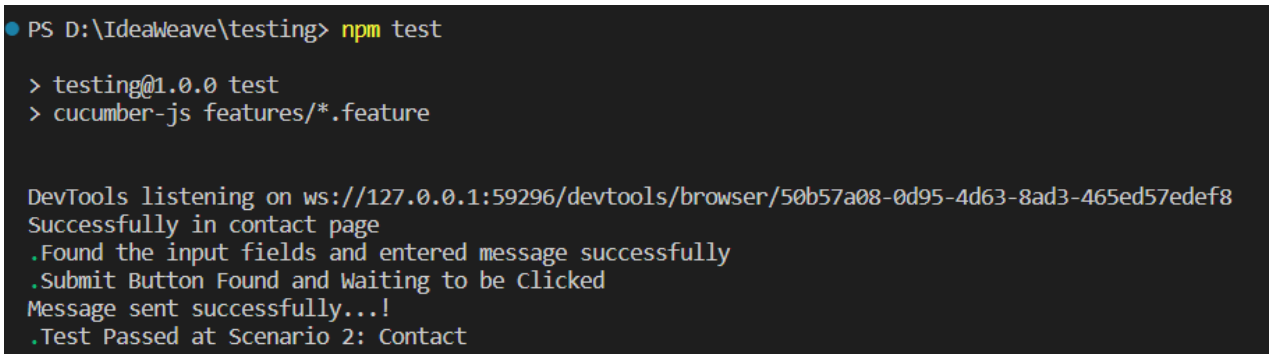
TEST CASE 2

```
> Feature: Contact Feature
  > As a user
  > I want to Contact the Administrator
  > So that I can communicate my queries
  > Scenario: Successful Login
    > Given I am on contact page
    > And I enter the contact query
    > And click on submit
    > Then contact is accomplished
```

Code

```
const { Given, When, Then } = require("cucumber");
const { Builder, By, until } = require("selenium-webdriver");

let driver;

Given("I am on contact page", async function () {
    driver = await new Builder().forBrowser("chrome").build();
    await driver.get("http://localhost:3000/contact");
    console.log("Successfully in contact page");
});
When("I enter the contact query", async function (){
    await driver.findElement(By.id("name")).sendKeys("Test");
    await driver.findElement(By.id("email")).sendKeys("sreeragkuofficial@gmail.com");
    await driver.findElement(By.id("message")).sendKeys("Did you get this test mail?");
    console.log("Found the input fields and entered message successfully");
})
When("click on submit", async function () {
    try {
        const button = await driver.wait(
            until.elementLocated(By.id("submit")),
            10000
        );
        console.log("Submit Button Found and Waiting to be Clicked")
        await button.click();
        console.log("Message sent successfully...!");
    } catch (error) {
        console.error("Error clicking on the login button:", error);
    }
});
Then("contact is accomplished", async function () {
    await driver.quit();
    console.log("Test Passed at Scenario 2: Contact");
});
```

Screenshot



Test Report

Test Case 2

Project Name: IdeaWeave					
Contact Test Case					
Test Case ID: Test_2			**Test Designed By: Sreerag K U**		
Test Priority (Low/Medium/High): Low			**Test Designed Date: 23/11/2023**		
Module Name: All Users			**Test Executed By: Mr. Binumon Joseph**		
Test Title : Contact			**Test Execution Date: 27/11/2023**		
Description: Test the Contact functionality					
Pre-Condition: None					
Step	**Test Step**	**Test Data**	**Expected Result**	**Actual Result**	**Status (Pass/Fail)**
1	Chrome Driver Opened		Browser Opened	Browser Opened	Pass
2	Navigation To contact page		Navigate to Contact page	Navigated to Contact page	Pass
3	Fill the form and submit	Sreerag sreeragkuofficial@gmail.com This is a great website	Error message should not be displayed	Error message not displayed	Pass
Post-Condition: Mail send to the admin mail account					

TEST CASE 3

```
> Feature: Profile Update Feature
  > As a user
  > I want to update profile
  > So that I can customize my account
  > Scenario: Successful Login
    > Given I am on the login page
    > When I enter valid credentials
    > And click on the login button
    > Then I should be redirected to the home page
    > When I am on the home page
    > And I click on the explore button
    > Then I should be redirected to the author dashboard
    > When I am on the profile page
    > And I enter a new username and password
    > And I click on the save changes button
    > Then I should remain in profile page
```

Code

```
const { Given, When, Then } = require("cucumber");
const { Builder, By, until } = require("selenium-webdriver");

let driver;

Given("I am on the login page", async function () {
   driver = await new Builder().forBrowser("chrome").build();
   await driver.get("http://localhost:3000/signin");
});
When("I enter valid credentials", async function () {
   await driver.findElement(By.id("em")).sendKeys("sreeragkuofficial@gmail.com");
   await driver.findElement(By.id("ps")).sendKeys("sku@MCA20");
   console.log("Found the Email and Password fields and Valid credentials entered successfully");
});
When("click on the login button", async function (){
       try {
       const button = await driver.wait(
          until.elementLocated(By.id("testid")),
          10000
       );
       console.log("Login Button Found and Waiting to be Clicked")
       await button.click();
       console.log("Login successful...!");
       } catch (error) {
       console.error("Error clicking on the login button:", error);
      }
});
Then("I should be redirected to the home page", async function () {
```

```
    await driver.wait(until.urlIs("http://localhost:3000/"));
    console.log("Successfully redirected to the home page");
});
When("I am on the home page", async function () {
    await driver.wait(until.urlIs("http://localhost:3000/"));
});
When("I click on the explore button", async function () {
    const exploreButton = await driver.findElement(By.id("explore"));
    await exploreButton.click();
    console.log("Clicked on the explore button");
});
Then("I should be redirected to the author dashboard", async function () {
    await driver.wait(until.urlIs("http://localhost:3000/author"));
    console.log("Successfully redirected to the author dashboard");
});
When("I am on the profile page", async function () {
    await driver.wait(until.urlIs("http://localhost:3000/author/654b5c2301edcd4fcedc9af9"),
10000);
    console.log("Successfully on the profile page");
});
When("I enter a new username and password", async function () {
    await driver.findElement(By.id("username")).sendKeys("NewSreerag");
    await driver.findElement(By.id("password")).sendKeys("20SKU@mca");
    console.log("Entered new username and password");
});
Then("I click on the save changes button", async function () {
    const saveChangesButton = await driver.findElement(By.id("submit-btn"));
    await saveChangesButton.click();
    console.log("Clicked on the submit button");
});
Then("I should remain in profile page", async function () {
    await driver.wait(until.urlIs("http://localhost:3000/author/654b5c2301edcd4fcedc9af9"));
    await driver.quit();
    console.log("Test Passed at Scenario 3: Profile Update")
});
```

Screenshot

```
PS D:\IdeaWeave\testing> npm test

> testing@1.0.0 test
> cucumber-js features/*.feature

.Found the Email and Password fields and Valid credentials entered successfully
.Login Button Found and Waiting to be Clicked
Login successful...!
.Successfully redirected to the home page
..Clicked on the explore button
.Successfully redirected to the author dashboard
.Successfully on the profile page
.Entered new username and password
.Clicked on the submit button
.Test Passed at Scenario 3: Profile Update
.

1 scenario (1 passed)
11 steps (11 passed)
```

Test Report

Test Case 3

Project Name: IdeaWeave					
Profile Update Test Case					
Test Case ID: Test_3			**Test Designed By: Sreerag K U**		
Test Priority (Low/Medium/High): High			**Test Designed Date: 23/11/2023**		
Module Name: All Users			**Test Executed By: Mr. Binumon Joseph**		
Test Title : Profile Update			**Test Execution Date: 27/11/2023**		
Description: Test the Profile Update functionality					
Pre-Condition: None					
Step	**Test Step**	**Test Data**	**Expected Result**	**Actual Result**	**Status (Pass/Fail)**
1	Chrome Driver Opened		Browser Opened	Browser Opened	Pass
2	Navigation To sign in page		Navigate to Sign in page	Navigated to Sign in page	Pass
3	Enter Valid Credentitals	sreeragkuofficial@gmail.com sku@MCA20	Error message should not be displayed	Error message not displayed	Pass
4	Click Explore Button		Navigate to Author dashboard	Navigated to Author dashboard	Pass
5	Go to Profile Page		Navigate to profile page	Navigated to profile page	Pass
6	Enter new username and password	NewSreerag 20@MCAsku	Error message should not be displayed	Error message not displayed	Pass
Post-Condition: User Credentials changed and should reflect in their account					

TEST CASE 4

```
> Feature: Create Category Feature
  > As an admin
  > I want to create new category
  > So that users can add diverse content
  > Scenario: Successful Login
    > Given I am on the login page
    > When I enter valid credentials
    > And click on the login button
    > Then I should be redirected to the home page
    > When I am on the home page
    > And I click on the explore button
    > Then I should be redirected to the admin dashboard
    > When I am on the category page
    > And I enter a new category
    > And I click on the save changes button
    > Then I should remain in category page
```

Code

```
const { Given, When, Then } = require("cucumber");
const { Builder, By, until } = require("selenium-webdriver");

let driver;

Given("I am on the login page", async function () {
   driver = await new Builder().forBrowser("chrome").build();
   await driver.get("http://localhost:3000/signin");
});
When("I enter valid credentials", async function () {
   await driver.findElement(By.id("em")).sendKeys("ideaweavep@gmail.com");
   await driver.findElement(By.id("ps")).sendKeys("sku@ADMIN20");
   console.log("Found the Email and Password fields and Valid credentials entered successfully");
});
When("click on the login button", async function (){
   try {
      const button = await driver.wait(
         until.elementLocated(By.id("testid")),
         10000
      );
      console.log("Login Button Found and Waiting to be Clicked")
      await button.click();
      console.log("Login successful...!");
   } catch (error) {
      console.error("Error clicking on the login button:", error);
   }
});
Then("I should be redirected to the home page", async function () {
   await driver.wait(until.urlIs("http://localhost:3000/"));
```

```
    console.log("Successfully redirected to the home page");
});
When("I am on the home page", async function () {
    await driver.wait(until.urlIs("http://localhost:3000/"));
});
When("I click on the explore button", async function () {
    const exploreButton = await driver.findElement(By.id("explore"));
    await exploreButton.click();
    console.log("Clicked on the explore button");
});
Then("I should be redirected to the admin dashboard", async function () {
    await driver.wait(until.urlIs("http://localhost:3000/admin"));
    console.log("Successfully redirected to the admin dashboard");
});
When("I am on the category page", async function () {
    await driver.wait(until.urlIs("http://localhost:3000/admin/categories"), 10000);
    console.log("Successfully on the category page");
});
When("I enter a new category", async function () {
    await driver.findElement(By.id("category")).sendKeys("Drama");
    console.log("Entered new category");
});
Then("I click on the save changes button", async function () {
    const saveChangesButton = await driver.findElement(By.id("submit-btn"));
    await saveChangesButton.click();
    console.log("Clicked on the submit button");
});
Then("I should remain in category page", async function () {
    await driver.wait(until.urlIs("http://localhost:3000/admin/categories"));
    await driver.quit();
    console.log("Test Passed at Scenario 4: Category created")
});
```

Screenshot

```
PS D:\IdeaWeave\testing> npm test

> testing@1.0.0 test
> cucumber-js features/*.feature

.Found the Email and Password fields and Valid credentials entered successfully
.Login Button Found and Waiting to be Clicked
Login successful...!
.Successfully redirected to the home page
..Clicked on the explore button
.Successfully redirected to the admin dashboard
.Successfully on the category page
.Entered new category
.Clicked on the submit button
.Test Passed at Scenario 4: Category created
.

1 scenario (1 passed)
11 steps (11 passed)
```

Test Report

Test Case 4

Project Name: IdeaWeave					
Create Category Test Case					
Test Case ID: Test_4			**Test Designed By: Sreerag K U**		
Test Priority (Low/Medium/High): Medium			**Test Designed Date: 23/11/2023**		
Module Name: Admin			**Test Executed By: Mr. Binumon Joseph**		
Test Title : Create Category			**Test Execution Date: 27/11/2023**		
Description: Test the create category functionality					
Pre-Condition: None					
Step	**Test Step**	**Test Data**	**Expected Result**	**Actual Result**	**Status (Pass/Fail)**
1	Chrome Driver Opened		Browser Opened	Browser Opened	Pass
2	Navigation To sign in page		Navigate to Sign in page	Navigated to Sign in page	Pass
3	Enter Valid Credentials	ideaweavep@gmail.com sku@ADMIN20	Error message should not be displayed	Error message not displayed	Pass
4	Click Explore Button		Navigate to Author dashboard	Navigated to Author dashboard	Pass
5	Go to Category Page		Navigate to categories page	Navigated to categories page	Pass
6	Enter new category	Drama	Error message should not be displayed	Error message not displayed	Pass
Post-Condition: New Category added for all users					

CHAPTER 6
IMPLEMENTATION

6.1 INTRODUCTION

Project implementation is the phase where plans and strategies transform into tangible actions and outcomes. It marks the transition from theoretical concepts to practical application. This pivotal stage requires meticulous planning, resource allocation, and a dedicated team to execute tasks according to the established timeline and objectives. In this phase, the project team translates the project's blueprints into real-world activities, ensuring that each step aligns with the overarching goals. Effective project implementation demands clear communication, robust leadership, and a keen eye for detail. This introduction sets the stage for a comprehensive understanding of the project implementation process, emphasizing its significance in achieving the envisioned goals. As we delve deeper, we will explore key components, strategies, and best practices that contribute to successful project implementation.

The crux of successful project implementation lies not just in technical proficiency, but also in the art of effective communication. Clear channels of dialogue serve as the lifeblood that sustains the project's momentum, fostering synergy among team members and stakeholders alike. A bedrock of robust leadership provides the necessary guidance and inspiration, steering the ship through uncharted waters with confidence and purpose. Additionally, an unyielding commitment to detail acts as the linchpin that secures the integrity of each executed task

This introduction sets the stage for a profound comprehension of the project implementation process, underlining its indomitable significance in realizing the envisioned goals. As we embark on this journey of exploration, we will unfurl the tapestry of key components, unveil strategies, and illuminate best practices that form the crucible of triumphant project implementation.

The implementation state involves the following tasks:

- Careful planning.
- Investigation of system and constraints.
- Design of methods to achieve the changeover.

6.2 IMPLEMENTATION PROCEDURES

6.2.1 USER TRAINING

User training is a critical component of ensuring the effective utilization of any website. It involves imparting the necessary knowledge and skills to end-users, enabling them to navigate and utilize the website efficiently. This training equips users with a comprehensive understanding of the

website's features, functions, and capabilities. Through hands-on sessions and guided tutorials, users learn how to perform tasks, customize settings, and troubleshoot common issues. Moreover, user training fosters confidence and proficiency, empowering individuals to maximize their productivity while using the application. Regular updates and refresher sessions further enhance user competence, ensuring they stay abreast of new features and functionalities.

6.2.2 TRAINING ON THE WEBSITE

Training on the website is a structured program designed to familiarize individuals with the intricacies and functionalities of a specific website. It encompasses a range of topics, from basic navigation to advanced features, tailored to meet the diverse needs of users. This training often includes interactive demonstrations, hands-on exercises, and Q&A sessions to facilitate effective learning. Trainers may also provide supplemental resources such as user manuals or online guides for reference. By the end of the training, participants are equipped skills and knowledge required to proficiently utilize the website in their respective contexts.

6.2.3 SYSTEM MAINTENANCE

System maintenance is a crucial aspect of ensuring the seamless operation and longevity of any website application. It encompasses a series of tasks aimed at monitoring, optimizing, and troubleshooting the underlying infrastructure on which the application runs. This includes activities such as regular performance monitoring, and data backups. Additionally, system maintenance involves identifying and rectifying any potential vulnerabilities or inefficiencies that may impede the website’s performance. Proactive maintenance measures contribute to a stable and secure environment, minimizing the risk of unexpected downtime or data loss.

CHAPTER 7
CONCLUSION AND FUTURE SCOPE

7.1 CONCLUSION

IdeaWeave stands as a transformative force in the digital content creation realm, connecting creators and consumers seamlessly. Powered by the dynamic MERN stack with Next.js integration, it redefines content management by offering intuitive features for writers, readers, and administrators. The platform's modules, spanning robust admin tools to tailored content creation features, exemplify its dedication to user-centricity, collaboration, and innovation.

Administrators wield powerful tools for user management and content oversight, ensuring a secure and efficient platform. Writer’s benefit from a suite of advanced content creation tools, fostering creativity and organization. Readers enjoy personalized content discovery and engaging features for a fulfilling reading experience. IdeaWeave is not just a content distribution platform; it's a vibrant community where ideas flourish.

The commitment to innovation is evident in features such as AI-driven content suggestions, collaborative editing, and blockchain-powered authenticity verification. IdeaWeave goes beyond transactional interactions, creating an ecosystem where content creators and consumers connect, collaborate, and thrive.

7.2 FUTURE SCOPE

As IdeaWeave advances into future development stages, the integration of additional functionalities promises to elevate the platform's capabilities:

- **Advanced AI and ML Integration:** Implementation of AI and ML algorithms to enhance content discovery, content translation and offering personalized recommendations based on user engagement and preferences.
- **Diverse Content Categories:** Expansion of content categories based on user feedback and emerging trends, catering to a broader spectrum of interests and topics.
- **Refined Filtering Options:** Enhancement of filtering features, allowing readers to tailor searches based on specific content attributes, genres, and author preferences.
- **Communication and Support Enhancements:** Introducing features for efficient communication and support, including real-time chat support, and comprehensive FAQs for a smoother user experience.
- **AI-Enhanced Content Quality:** Leveraging AI to enhance content quality by providing writers with insights and suggestions for improving their work, ensuring a higher standard of published content.

CHAPTER 8
BIBLIOGRAPHY

REFERENCES:

- Gary B. Shelly, Harry J. Rosenblatt, “System Analysis and Design”, 2009.
- Roger S Pressman, “Software Engineering”, 1994.
- PankajJalote, “Software engineering: a precise approach”, 2006.
- James lee and Brent ware Addison, “Open source web development with LAMP”, 2003
- IEEE Std 1016 Recommended Practice for Software Design Descriptions.

WEBSITES:

- *https://nextjs.org*
- *https://reactjs.org*
- *https://ant.design/docs/react/*
- *https://chat.openai.com/*
- *https://www.github.com/*
- *https://www.youtube.com/*
- *https://draw.io/*
- *https://medium.com/*
- *http://udemy.com/*
- *http://webnovels.com*
- *http://ranobes.top*

CHAPTER 9
APPENDIX

9.1 SAMPLE CODE

WEBSITE ENTRY POINT – INDEX.JS

```
import { useContext, useEffect, useState } from "react";
import { AuthContext } from "../context/auth";
import Head from "next/head";
import FullWidthImage from "../components/pages/FullWidthImage";
import useNumbers from "../hooks/useNumbers";
import RenderProgress from "../components/posts/RenderProgress";
import { Row, Col, Divider, Button } from "antd";
import useLatestPosts from "../hooks/useLatestPosts";
import useCategory from "../hooks/useCategory";
import Link from "next/link";
import ParallaxImage from "../components/pages/ParallaxImage";
import { ThemeContext } from "../context/theme";
import { ThunderboltOutlined } from "@ant-design/icons";
import Footer from "../components/pages/Footer";
import axios from "axios";
import useHome from "../hooks/useHome";

function Home() {
  const [auth, setAuth] = useContext(AuthContext);
  const { numbers } = useNumbers();
  const { latestPosts } = useLatestPosts();
  const { categories } = useCategory();
  const [theme] = useContext(ThemeContext);
  const { title, subtitle, fullWidthImage, setTitle, setSubtitle, setFullWidthImage } = useHome();

  const textStrokeColor = theme === "light" ? "#ffffff" : "#000";

  return (
    <>
      <Head>
        <title>IdeaWeave</title>
        <meta
          name="description"
          content="Read latest books on web development"
        />
      </Head>

      <FullWidthImage
        auth={auth.user}
        title={title}
        subtitle={subtitle}
        fullWidthImage={fullWidthImage?.url}
      />

      <Row>
        {/* posts */}
        <Col
          span={6}
          style={{ marginTop: 50, textAlign: "center", fontSize: 20 }}
        >
          <RenderProgress
            number={numbers.posts}
            name="Books"
            link="/admin/posts"
          />
        </Col>
```

```
    {/* comments */}
    <Col
     span={6}
     style={{ marginTop: 50, textAlign: "center", fontSize: 20 }}
    >
     <RenderProgress
      number={numbers.comments}
      name="Comments"
      link="/admin/comments"
     />
    </Col>
    {/* catgories */}
    <Col
     span={6}
     style={{ marginTop: 50, textAlign: "center", fontSize: 20 }}
    >
     <RenderProgress
      number={numbers.categories}
      name="Categories"
      link="/admin/categories"
     />
    </Col>
    {/* users */}
    <Col
     span={6}
     style={{ marginTop: 50, textAlign: "center", fontSize: 20 }}
    >
     <RenderProgress
      number={numbers.users}
      name="Users"
      link="/admin/users"
     />
    </Col>
   </Row>
   <Row>
    <Col span={24}>
     <ParallaxImage>
      <h2
       style={{
        textAlign: "center",
        fontSize: "74px",
        textShadow: "8px 8px 12px #000000",
        color: "#fff",
        WebkitTextStroke: `0.5px ${textStrokeColor}`,
        textStroke: `0.5px ${textStrokeColor}`,
       }}
      >
       RECENT BOOKS
      </h2>
      <Divider>
       <ThunderboltOutlined />
      </Divider>
      <div
       style={{
        textAlign: "center",
        fontSize: "15px",
        position: "relative",
        zIndex: 1,
       }}
      >
```

```
          {latestPosts.map((post) => (
            <Link href={`/post/${post.slug}`} key={post.slug}>
              <h3
                style={{
                  color: "#fff",
                  background: "rgba(0, 0, 0, 0.3)",
                  padding: "10px",
                  borderRadius: "100px",
                  margin: "10px 0",
                }}
              >
                {post.title}
              </h3>
            </Link>
          ))}
        </div>
      </ParallaxImage>
    </Col>
  </Row>
  <Row>
    <Col
      span={24}
      style={{ textAlign: "center", marginTop: 80, marginBottom: 80 }}
    >
      <Divider>CATEGORIES</Divider>
      <div style={{ textAlign: "center" }}>
        {categories.map((c) => (
          <Link href={`/category/${c.slug}`} key={c._id}>
            <Button style={{ margin: 2 }}>{c.name}</Button>
          </Link>
        ))}
      </div>
    </Col>
  </Row>
  <Footer />
 </>
 );
}
export default Home;
```

ADMIN DASHBOARD – INDEX.JS

```
import { Row, Col, Divider } from "antd";
import AdminLayout from "../../components/layout/AdminLayout";
import RenderProgress from "../../components/posts/RenderProgress";
import useNumbers from "../../hooks/useNumbers";

function Admin() {
 const {numbers} = useNumbers();

 return (
  <AdminLayout>
   <Row>
    <Col span={24}>
     <Divider>
      <h1 style={{ marginTop: 50 }}>Statistics</h1>
     </Divider>
    </Col>
   </Row>

   <Row>
```

```
      {/* posts */}
      <Col
        span={12}
        style={{ marginTop: 50, textAlign: "center", fontSize: 20 }}
      >
        <RenderProgress
          number={numbers.posts}
          name="Books"
          link="/admin/posts"
        />
      </Col>
      {/* comments */}
      <Col
        span={12}
        style={{ marginTop: 50, textAlign: "center", fontSize: 20 }}
      >
        <RenderProgress
          number={numbers.comments}
          name="Comments"
          link="/admin/comments"
        />
      </Col>
    </Row>

    <Row>
      {/* catgories */}
      <Col
        span={12}
        style={{ marginTop: 50, textAlign: "center", fontSize: 20 }}
      >
        <RenderProgress
          number={numbers.categories}
          name="Categories"
          link="/admin/categories"
        />
      </Col>
      {/* users */}
      <Col
        span={12}
        style={{ marginTop: 50, textAlign: "center", fontSize: 20 }}
      >
        <RenderProgress
          number={numbers.users}
          name="Users"
          link="/admin/users"
        />
      </Col>
    </Row>
  </AdminLayout>
 );
}

export default Admin;
```

VIEW POSTS – POSTS.JS

```
import { useState, useEffect } from "react";
import axios from "axios";
import { Row, Col, Card, Avatar, Button, Carousel, Input, Divider } from "antd";
import Head from "next/head";
import Link from "next/link";
import useCategory from "../hooks/useCategory";

const { Meta } = Card;
const { Search } = Input;

export const Posts = ({ posts }) => {
  const [allPosts, setAllPosts] = useState(posts);
  const [originalPosts, setOriginalPosts] = useState(posts);
  const [total, setTotal] = useState(0);
  const [page, setPage] = useState(1);
  const [loading, setLoading] = useState(false);
  const [mostRecommended, setMostRecommended] = useState(null);
  const [searchTerm, setSearchTerm] = useState("");
  const [visiblePosts, setVisiblePosts] = useState([]);
  const [carouselPosts, setCarouselPosts] = useState([]);
  const postsPerPage = 8;
  const {categories} = useCategory();

  useEffect(() => {
    getTotal();
  }, []);

  useEffect(() => {
    setOriginalPosts(posts);
    setAllPosts(posts);
    setVisiblePosts(posts.slice(0, postsPerPage));
  }, [posts]);

  useEffect(() => {
    findMostRecommended();
    updateCarouselPosts();
    updateVisiblePosts();
  }, [allPosts, page]);

  const getTotal = async () => {
    try {
      const { data } = await axios.get("/post-count");
      setTotal(data);
    } catch (err) {
      console.error(err);
    }
  };

  const updateCarouselPosts = () => {
    const top5 = allPosts
      .filter((post) => post.commentCount > 0)
      .sort((a, b) => b.commentCount - a.commentCount)
      .slice(0, 5);
    setCarouselPosts(top5);
  };

  const updateVisiblePosts = () => {
    const startIndex = (page - 1) * postsPerPage;
    const endIndex = startIndex + postsPerPage;
```

```
  const newVisiblePosts = allPosts.slice(startIndex, endIndex);
  setVisiblePosts(newVisiblePosts);
 };

 const loadMore = () => {

  const nextPage = page + 1;
  if ((nextPage - 1) * postsPerPage < total) {
   setPage(nextPage);
   const startIndex = (nextPage - 1) * postsPerPage;
   const endIndex = startIndex + postsPerPage;
   const newVisiblePosts = allPosts.slice(startIndex, endIndex);
   setVisiblePosts((prevVisiblePosts) => [
    ...prevVisiblePosts,
    ...newVisiblePosts,
   ]);
  }
 };

 const loadPrevious = () => {
  const prevPage = page - 1;
  setPage(prevPage);
  const startIndex = (prevPage - 1) * postsPerPage;
  const endIndex = startIndex + postsPerPage;
  const newVisiblePosts = allPosts.slice(startIndex, endIndex);
  setVisiblePosts(newVisiblePosts);
 };

 const handleSearch = async (value) => {
  setSearchTerm(value);
  const result = originalPosts.filter((post) =>
   post.title.toLowerCase().includes(value.toLowerCase())
  );
  setAllPosts(result);
  setPage(1);
  updateVisiblePosts();
 };

 const findMostRecommended = () => {
  const postsWithComments = allPosts.filter((post) => post.commentCount > 0);

  if (postsWithComments.length > 0) {
   const recommendedPost = postsWithComments.reduce((prev, current) =>
    current.commentCount > (prev ? prev.commentCount : 0) ? current : prev
   );
   setMostRecommended(recommendedPost);
  } else {
   setMostRecommended(null);
  }
 };

 const handleCategoryClick = (slug) => {
  router.push(`/category/${slug}`);
 };

 return (
  <>
   <Head>
    <title>Recent New Releases</title>
    <meta description="Exciting new Books to invigorate your creative mind." />
   </Head>
```

```
<div
  style={{
    maxWidth: "1200px",
    margin: "0 auto",

    textAlign: "center",
    marginTop: 70,
    padding: "0 20px",
    overflowX: "hidden",
  }}
>
  <h2>Recommended</h2>
  <Carousel autoplay style={{ maxWidth: "100%" }}>
    {carouselPosts.map((post, index) => (
      <Link href={`/post/${post.slug}`} key={post.slug}>
        <Card
          hoverable
          style={{ width: "100%", marginTop: 20 }}
          cover={
            <Avatar
              shape="square"
              style={{ height: "300px" }}
              src={post.coverImage?.url || "images/default.jpg"}
              alt={post.title}
            />
          }
        >
          <div
            style={{
              position: "absolute",
              top: 10,
              left: 10,
              backgroundColor: "rgba(0, 0, 0, 0.5)",
              padding: "4px 8px",
              color: "white",
            }}
          >
            Rank {index + 1}
          </div>
          <div style={{marginBottom:20}}><Meta title={post.title} /></div>

        </Card>
      </Link>
    ))}
  </Carousel>
</div>
<Divider>Categories</Divider>
<div
  style={{
    display: "flex",
    justifyContent: "center",
    flexWrap: "wrap",
    margin: "20px 0",
  }}
>
  {categories.map((c) => (
    <Link href={`/category/${c.slug}`} key={c._id}>
      <Button
        style={{
          margin: "0 8px 8px 0",
```

```
          background: "#468570",
          color: "white",
        }}
      >
        {c.name}

      </Button>
    </Link>
  ))}
</div>
<div
  style={{
    textAlign: "center",
    marginTop: 50,
    padding: "0 20px",
    overflowX: "hidden",
  }}
>
  <h2>Recent New Releases</h2>
  <Search
    placeholder="Search for books"
    onSearch={handleSearch}
    enterButton
    style={{ maxWidth: 300, margin: "0 auto" }}
  />
</div>

<Row
  gutter={16}
  style={{ marginTop: 20, padding: "0 20px", overflowX: "hidden" }}
>
  {visiblePosts.map((post) => (
    <Col xs={24} sm={12} md={8} lg={6} xl={6} key={post.slug}>
      <Link href={`/post/${post.slug}`}>
        <Card
          hoverable
          style={{ width: "100%", maxWidth: "300px", marginBottom: 20 }}
          cover={
            <Avatar
              shape="square"
              style={{ height: "200px" }}
              src={post.coverImage?.url || "images/default.jpeg"}
              alt={post.title}
            />
          }
        >
          {mostRecommended && mostRecommended._id === post._id && (
            <div
              style={{
                position: "absolute",
                top: 0,
                right: 0,
                backgroundColor: "gold",
                padding: "4px 8px",
                color: "black",
              }}
            >
              Most Recommended
            </div>
          )}
```

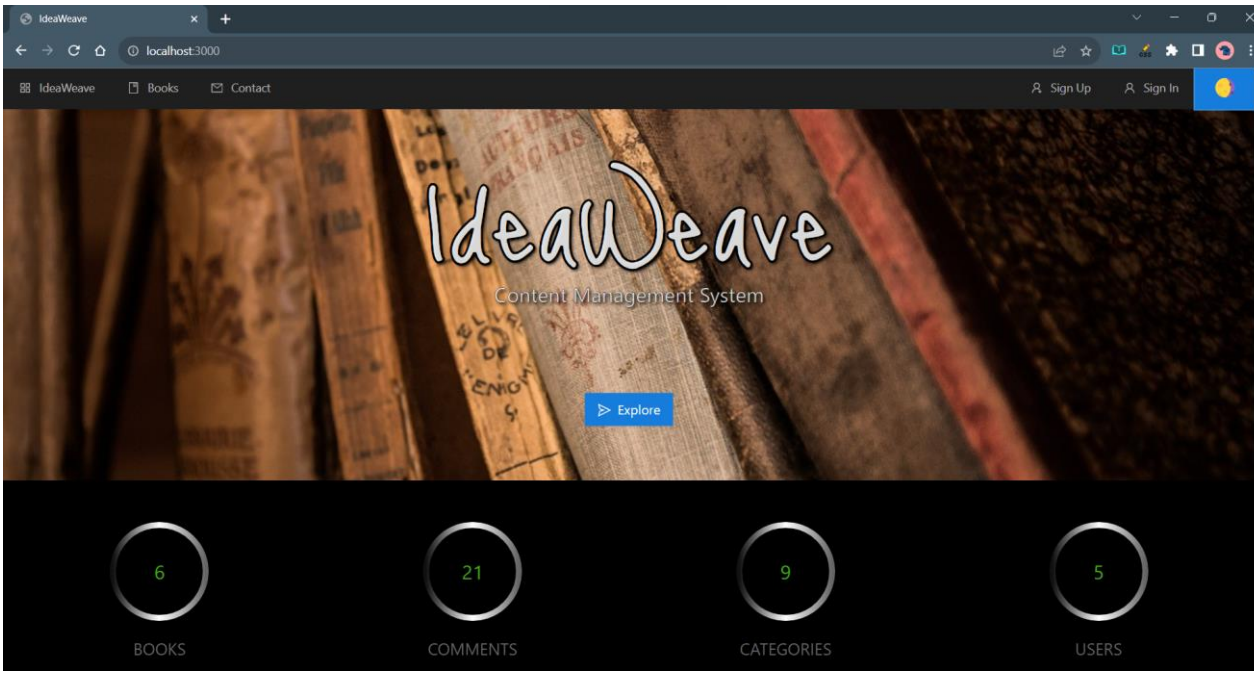
```
            {carouselPosts.findIndex((book) => book._id === post._id) !==
             -1 && (
             <div
              style={{
               position: "absolute",
               top: 0,
               left: 0,
               backgroundColor: "green",
               padding: "4px 8px",
               color: "white",
              }}
             >
              Top{" "}
              {carouselPosts.findIndex((book) => book._id === post._id) +
               1}
             </div>
            )}
            <Meta title={post.title} />
           </Card>
          </Link>
         </Col>
        ))}
      </Row>
      <div
       style={{
        textAlign: "center",
        marginTop: 20,
        padding: "0 20px",
        overflowX: "hidden",
       }}
      >
       <Button
        onClick={loadPrevious}
        disabled={page === 1}
        style={{ marginRight: "10px" }}
       >
        Previous
       </Button>{" "}
       <Button
        onClick={loadMore}
        disabled={page * postsPerPage >= total}
        loading={loading}
        style={{ marginBottom: 50, marginLeft: "10px" }}
       >
        Load More
       </Button>
      </div>
     </>
    );
};
export async function getServerSideProps() {
  const { data } = await axios.get(`${process.env.API}/posts/1`);
  return {
    props: {
      posts: data,
    },
  };
}

export default Posts;
```

CONTACT – CONTACT.JS

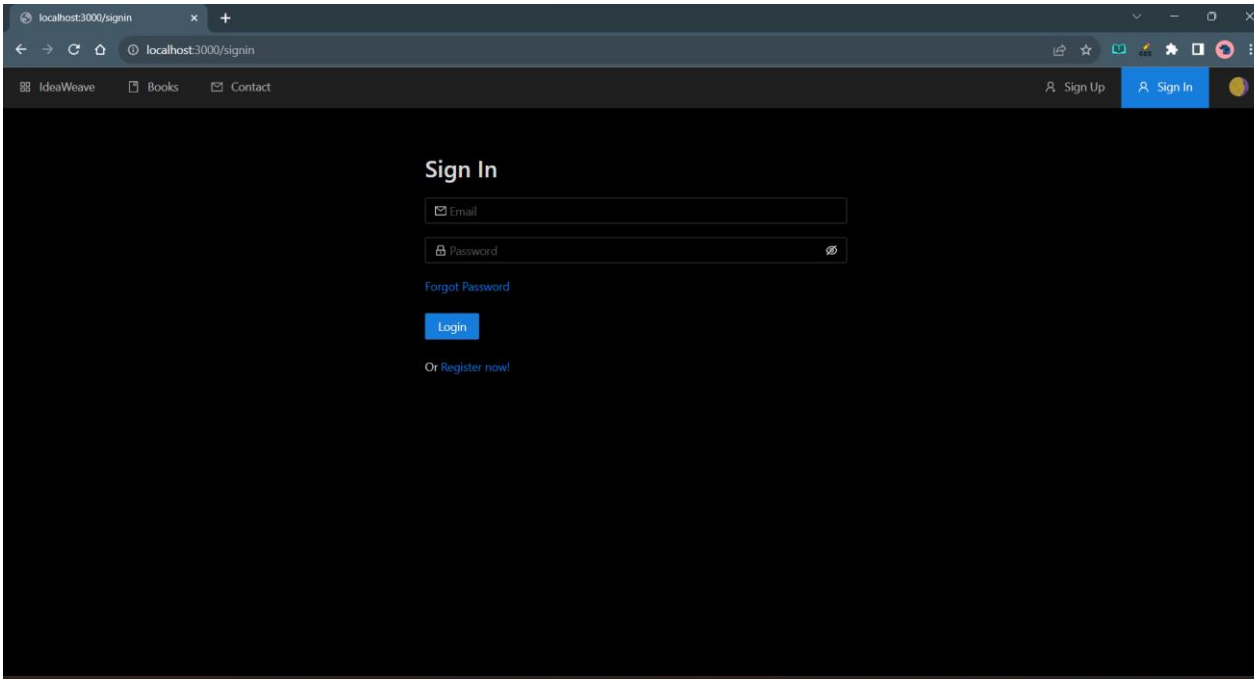
```
import { useState } from "react";
import { Form, Input, Button, Col, Row } from "antd";
import { UserOutlined, MailOutlined } from "@ant-design/icons";
import axios from "axios";
import { toast } from "react-hot-toast";
import { useRouter } from "next/router";

function ContactForm() {
  const [loading, setLoading] = useState(false);
  const router = useRouter();
  const [form] = Form.useForm();

  const onFinish = async (values) => {
    setLoading(true);
    try {
      const { data } = await axios.post("/contact", values);
      if (data?.error) {
        toast.error(data?.error);
      } else {
        toast.success("Your message has been sent");
        form.resetFields();
      }
    } catch (err) {
      console.error("Error:", err);
      toast.error("Email failed. Try again.");
    } finally {
      setLoading(false);
    }
  };

  const formItemLayout = {
    labelCol: {
      xs: { span: 24 },
      sm: { span: 8 },
    },
    wrapperCol: {
      xs: { span: 24 },
      sm: { span: 16 },
    },
  };

  return (
    <Row justify="center" align="middle" style={{ height: "100vh" }}>
      <Col span={12}>
        <h1>Contact</h1>
        <Form
          {...formItemLayout}
          name="contact_form"
          form={form}
          onFinish={onFinish}
          initialValues={{
            remember: true,
          }}
        >
          <Form.Item
            label="Your Name"
            name="name"
            rules={[
              {
```

```
            required: true,
            message: "Please enter your name",
          },
        ]}
      >
        <Input prefix={<UserOutlined />} placeholder="Your Name" />
      </Form.Item>

      <Form.Item
        label="Your Email"
        name="email"
        rules={[
          {
            required: true,
            type: "email",
            message: "Please enter a valid email address",
          },
        ]}
      >
        <Input prefix={<MailOutlined />} placeholder="Your Email" />
      </Form.Item>

      <Form.Item
        label="Your Message"
        name="message"
        rules={[
          {
            required: true,
            message: "Please enter your message",
          },
        ]}
      >
        <Input.TextArea placeholder="Write your message here..." />
      </Form.Item>

      <Form.Item wrapperCol={{ offset: 8, span: 16 }}>
        <Button type="primary" htmlType="submit" loading={loading}>
          Submit
        </Button>
      </Form.Item>
    </Form>
  </Col>
 </Row>
 );
}

export default ContactForm;
```

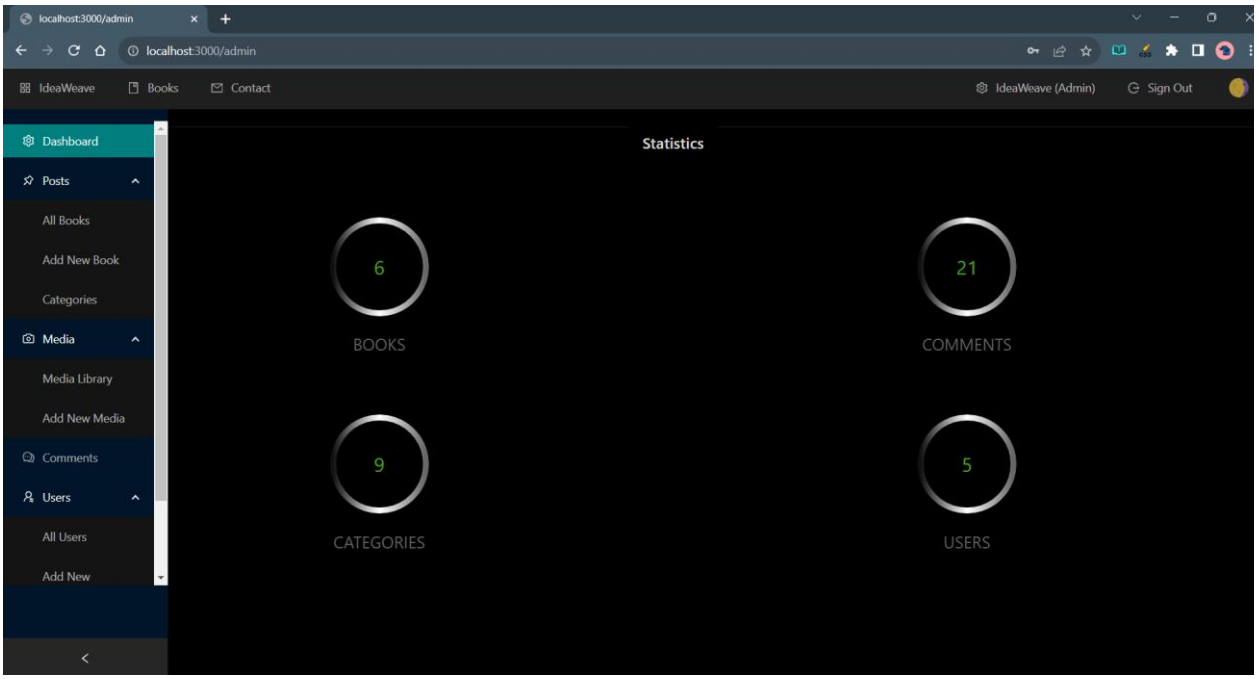
9.2 SCREEN SHOTS



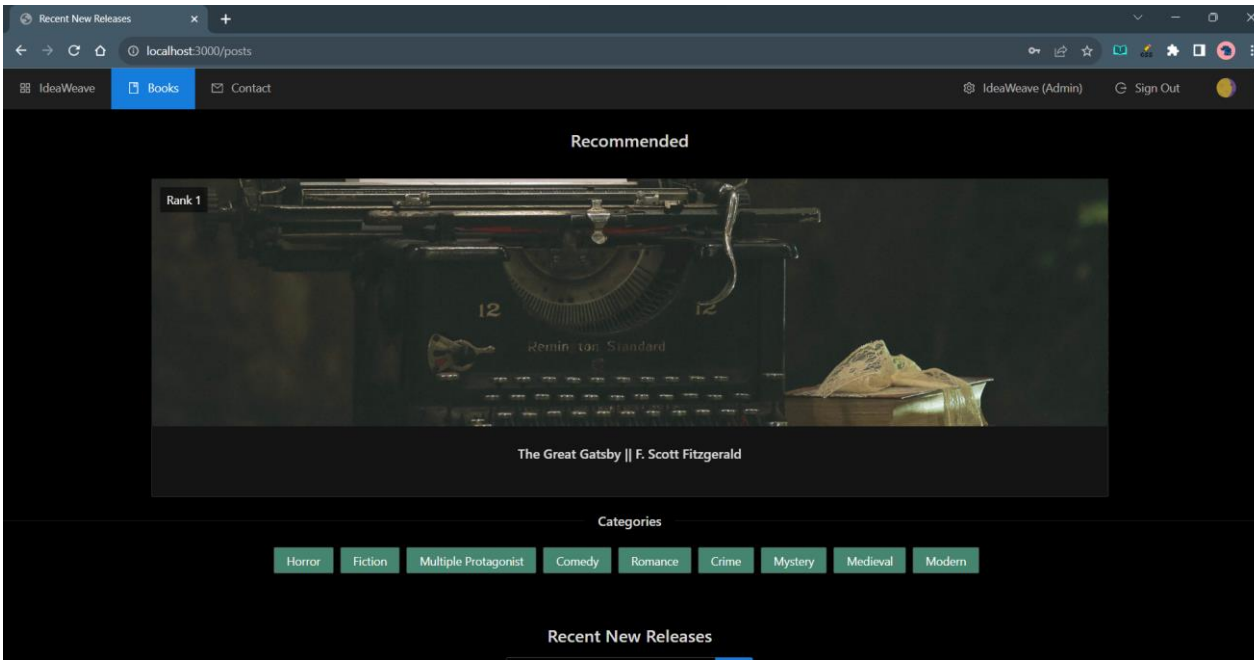
Screenshot 9.2.1: *Homepage*



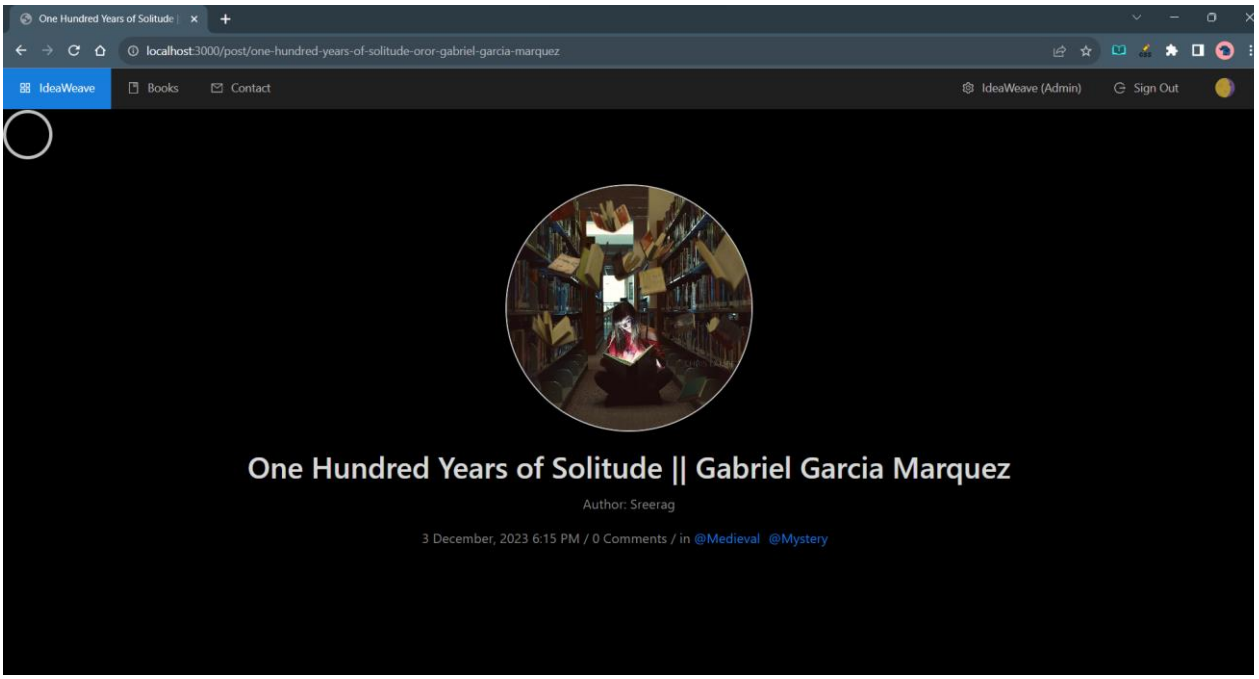
Screenshot 9.2.2: *Sign in*



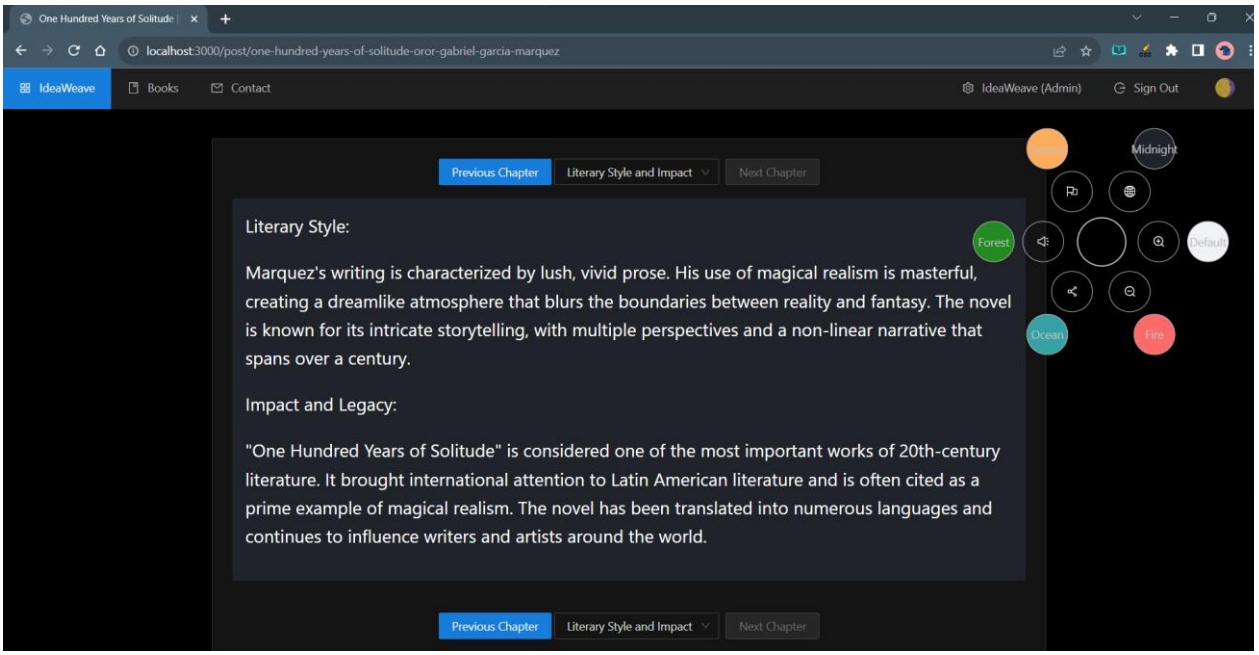
Screenshot 9.2.3: *Admin Dashboard*



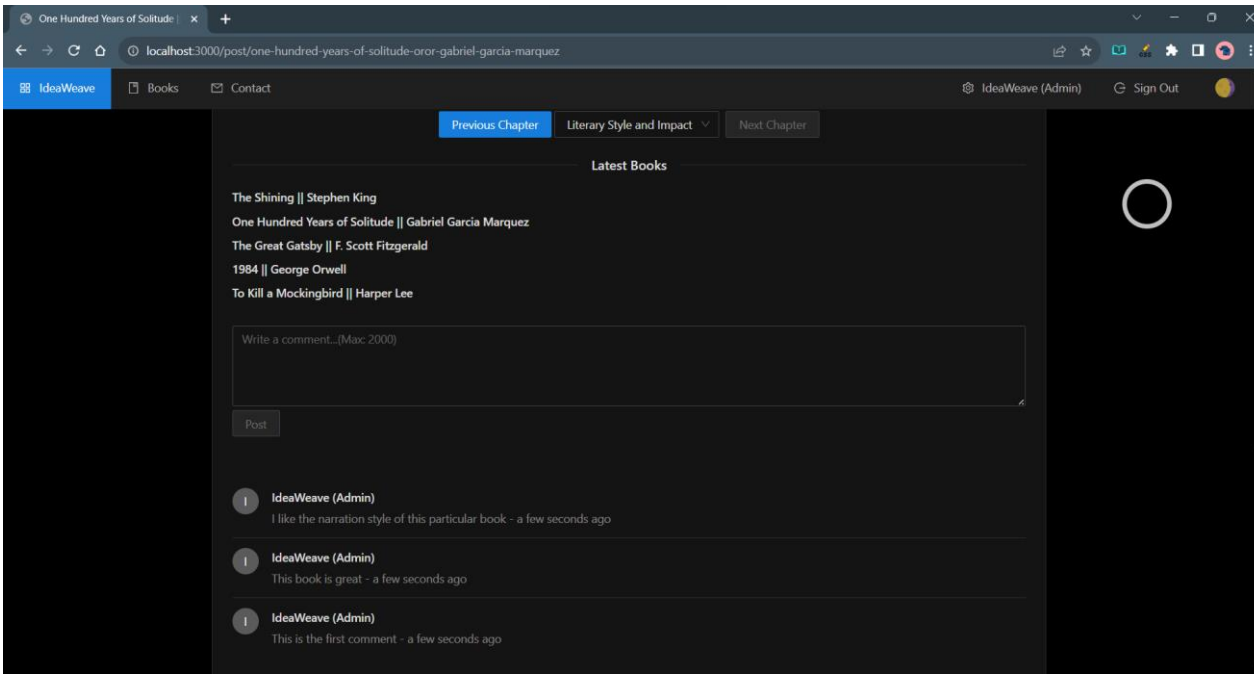
Screenshot 9.2.4: *Book List*



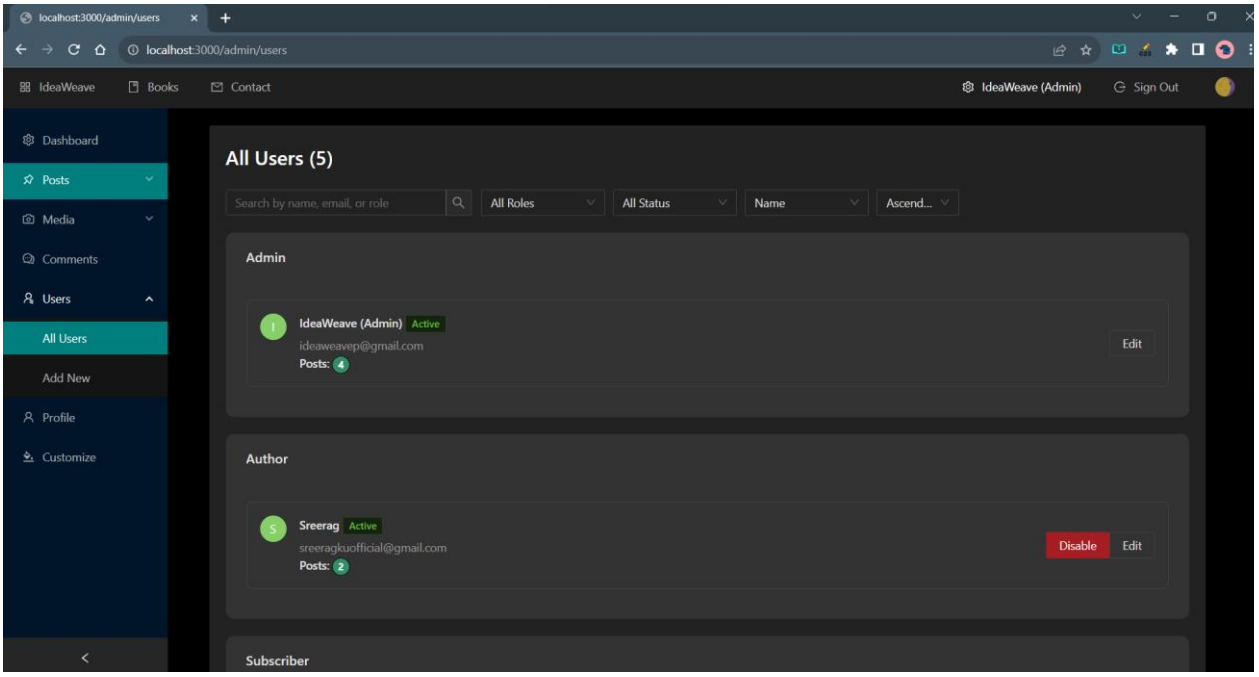
Screenshot 9.2.5: *Book Front*



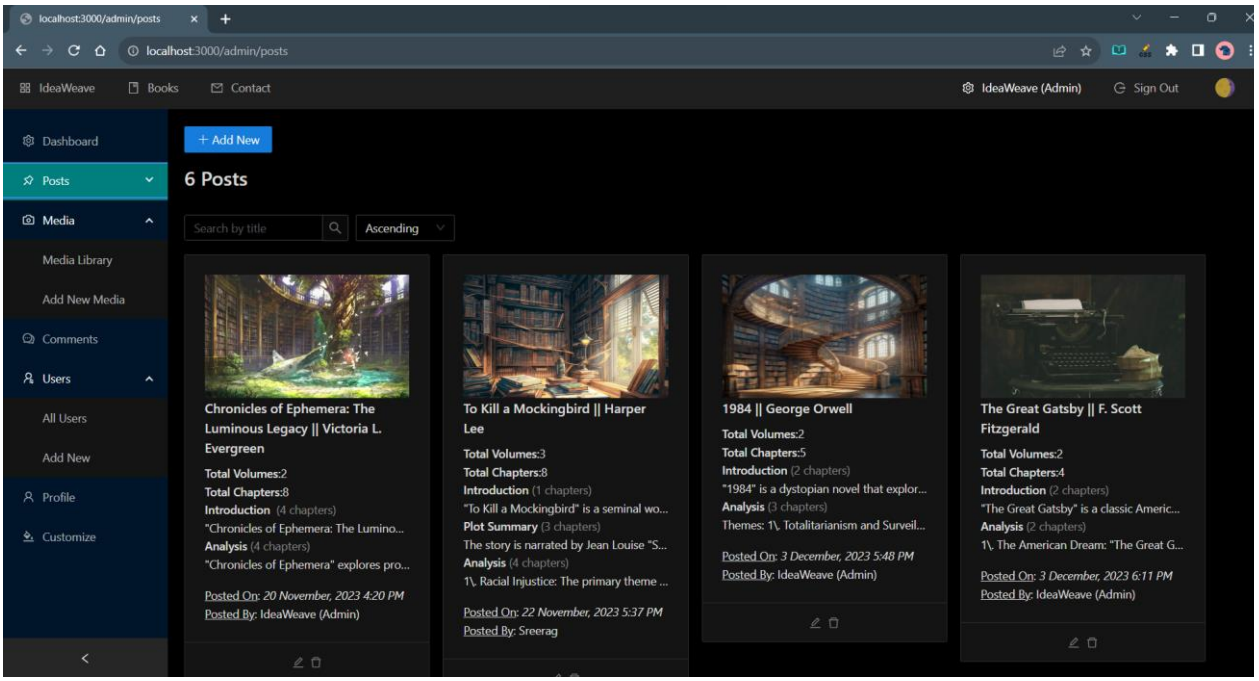
Screenshot 9.2.6: *Book Content and Feature Bubble*



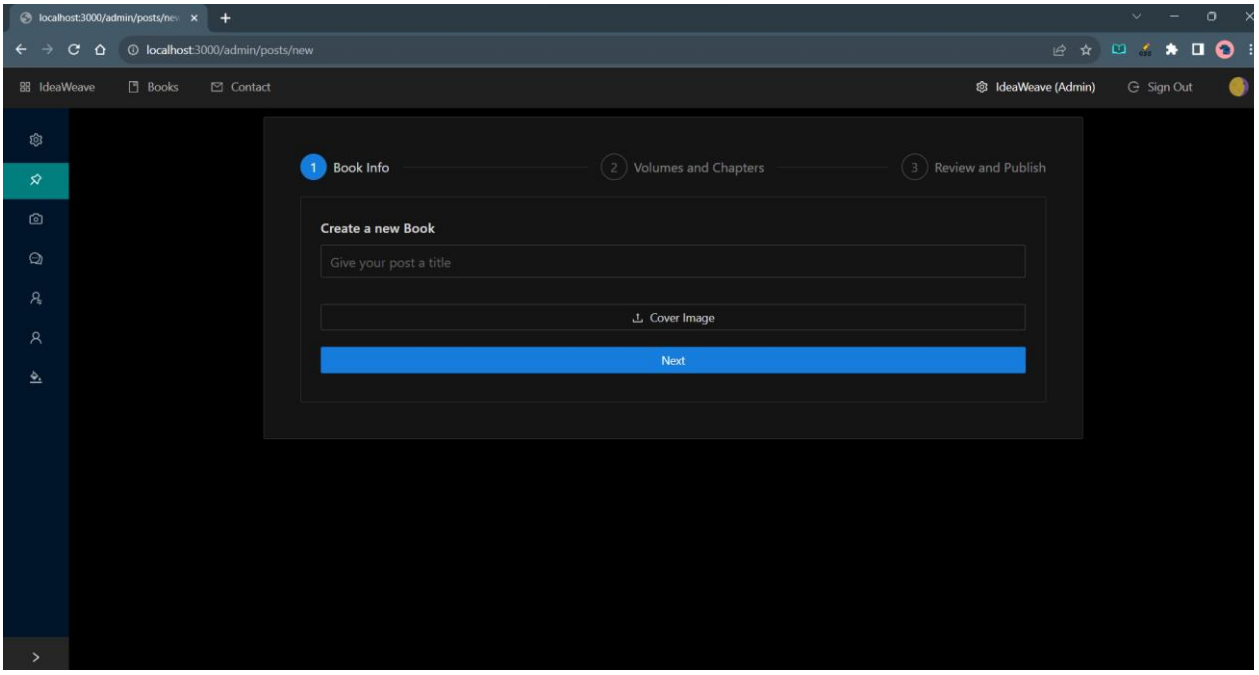
Screenshot 9.2.7: *Comments and Latest Book in Last Chapter*



Screenshot 9.2.8: *All Users page with search, sort, edit and disable*



Screenshot 9.2.9: *All Books Published*



Screenshot 9.2.10: *Multi-Stage Content Creation*