COLLABLOG
WHERE CREATIVITY MEETS COMMUNITY

A PROJECT REPORT
for
Project (KCA-451)
Session (2023-24)

Submitted by

SHUBHANGINI AGRAWAL
2200290140151

Submitted in partial fulfilment of the
Requirements for the Degree of

MASTER OF COMPUTER APPLICATION

Under the Supervision of
Ms. Komal Salgotra
TEACHING ASSISTANT

Submitted to

DEPARTMENT OF COMPUTER APPLICATIONS
KIET Group of Institutions, Ghaziabad
Uttar Pradesh-201206

(MAY 2024)

CERTIFICATE

Certified that **Shubhangini Agrawal, 2200290140151** has carried out the project work having "**COLLABLOG: WHERE CREATIVITY MEETS COMMUNITY**" (**Project-KCA451)** for **Master of Computer Application** from Dr. A.P.J. Abdul Kalam Technical University (AKTU**)** (formerly UPTU), Lucknow under my supervision. The project report embodies original work, and studies are carried out by the student herself and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

Date:

Shubhangini Agrawal
(2200290140151)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date:

Ms. Komal Salgotra
Teaching Assistant
Department of Computer Applications
KIET Group of Institutions, Ghaziabad

Dr. Arun Tripathi
Head
Department of Computer Applications
KIET Group of Institutions, Ghaziabad

“Collablog: Where Creativity Meet Community”

Shubhangini Agrawal

ABSTRACT

Welcome to a digital heaven where words dance, ideas collide, and perspectives intertwine. At “Collablog”, we believe in the power of expression, the beauty of storytelling, and the magic that happens when minds converge in the boundless realm of the internet.

Our blog platform is more than just a collection of articles; it's a vibrant community where individuals from all walks of life come together to explore, learn, and engage with a myriad of topics. From insightful think pieces to practical how-tos, from thought-provoking analyses to heartwarming narratives, our platform is a melting pot of diverse voices and perspectives.

Collablog is designed to be a hub for writers, readers, and content creators to share, discover, and engage with high-quality written content. The platform boasts a user-friendly interface that emphasizes readability and ease of navigation, along with advanced text editing tools that facilitate effortless article creation and formatting.

Personalized content feeds driven by machine learning algorithms will deliver tailored reading recommendations, while social interaction features will enable users to follow writers, comment on articles, and participate in meaningful discussions.

Collablog also supports both individual blogging and collaborative publications, allowing groups of writers to publish under a unified banner, with editors curating and highlighting noteworthy content. To further support writers, detailed analytics and monetization options, such as subscription models or ad revenue sharing, will be provided. With a fully responsive design,

Collablog ensures optimal performance across various devices, making it accessible and user-friendly. Developed using modern web technologies, Collablog aims to foster a vibrant community of writers and readers, promoting the exchange of ideas and knowledge through high-quality written content

Join us on this journey as we delve into the depths of human experience, celebrate the richness of cultural diversity, and embark on a quest for understanding in an ever-changing world. Together, let's explore, discover, and connect through the power of words.

ACKNOWLEDGEMENT

Success in life is never attained single-handedly. My deepest gratitude goes to my project supervisor, **Ms. Komal Salgotra** for her guidance, help, and encouragement throughout my project work. Their enlightening ideas, comments, and suggestions.

Words are not enough to express my gratitude **to Dr. Arun Kumar Tripathi**, Professor and Head, Department of Computer Applications, for his insightful comments and administrative help on various occasions.

Fortunately, I have many understanding friends, who have helped me a lot on many critical conditions.

Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me with moral support and other kind of help. Without their support, completion of this work would not have been possible in time. They keep my life filled with enjoyment and happiness.

Shubhangini Agrawal

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

In today's digital age, communication has undergone a significant transformation. The emergence of social media, content sharing platforms, and blogging has redefined how individuals and organizations interact, share information, and express themselves. Among these, “Collablog” stand out as powerful tools that enable users to create, publish, and distribute content to a global audience with ease.

1.1 OVERVIEW

“Collablog: Where Creativity Meets Community” is a digital tool or service that enables users to create, manage, and publish content on the internet in the form of blog posts. It provides users with the necessary infrastructure and tools to write, format, and share their thoughts, ideas, stories, or expertise with a global audience.

The main aim of this application is to provide a hassle-free accessing of the posted blogs, entries, topics etc. It also used for posting the blogs, editing the blogs, deleting the posted blogs etc. It is also used for viewing and posting the others one’s blogs/posts.

It is an online service or software application that enables users to create and publish their own blogs on the internet. It typically offers features such as text editing, media embedding, categorization, scheduling, social sharing, and analytics to facilitate content creation, engagement, and audience interaction. It serves as accessible and user-friendly mediums for individuals, businesses, organizations, and professionals to establish an online presence, express themselves, and connect with like-minded individuals or communities.

. It provides individuals, businesses, and organizations with a platform to share their ideas, experiences, expertise, and stories with a global audience. “Collablog” often include features such as customization options for the appearance of the blog, social sharing capabilities to promote content across various platforms, and analytics tools to track the performance and reach of blog posts. Overall, Collablog serves as a digital hub for content creation, publication, and community engagement in the online world.

1.2 MOTIVATION AND SCOPE

1.2.1 MOTIVATION

The motivation behind developing a blog platform lies in democratizing online expression, fostering community engagement, and providing a platform for diverse voices to be heard. It's driven by a commitment to empowering individuals and organizations, regardless of their technical expertise, to share their ideas, stories, and expertise with the world. This platform aims to lower barriers to entry for content creation, offering intuitive tools and features that enable users to create, publish, and promote their content effortlessly. Moreover, the project aspires to cultivate a vibrant online community where users can connect, collaborate, and engage with one another around shared interests, passions, and causes.

1.2.2 SCOPE

In terms of scope, the blog platform project encompasses the development of a robust content management system (CMS) with a focus on user-friendliness, customization options, and search engine optimization (SEO). It includes features such as multimedia integration, theme customization, and social sharing capabilities to enhance the blogging experience. Additionally, the project prioritizes inclusivity and diversity within the blogging community, actively seeking contributions from individuals of all backgrounds and perspectives. By providing a platform that celebrates creativity, fosters dialogue, and amplifies underrepresented voices, the project aims to create a digital space that reflects the rich diversity of human experiences and perspectives

1.3 BENEFITS OF USING COLLABLOG

- **Ease of Content Creation:** Collablog typically offer user-friendly interfaces and intuitive content management systems, enabling users to create and publish blog posts quickly and easily without the need for advanced technical skills.

- **Accessibility:** With Collablog users can reach a global audience, allowing them to share their ideas, expertise, stories, or creations with people from diverse backgrounds and locations.

- **Establishing Authority:** Collablog provide users with a platform to showcase their knowledge, skills, and expertise in their chosen niche or industry, helping them establish themselves as thought leaders or authorities in their field.

- **Building a Personal Brand:** Through consistent blogging and engagement, users can build a recognizable personal brand that reflects their values, personality, and unique perspective, enhancing their credibility and visibility online.

- **Community Engagement:** Collablog often facilitate interaction and engagement among users through features such as comments, social sharing, and subscriptions, fostering a sense of community and connection around shared interests or topics**.**

1.4 FEATURES OF COLLABLOG

- **User Registration and Management:** Allow users to create accounts, manage profiles, and set preferences.
- **Content Creation and Editing:** Provide tools for writing, formatting, and editing blog posts, including text formatting options, media embedding, and draft saving.
- **Media Management:** Enable users to upload, store, and manage multimedia content such as images for inclusion in blog posts.
- **Content Organization:** Provide features for categorizing and tagging blog posts, creating hierarchical structures (such as categories and tags), and organizing posts into archives or collections.
- **Social Sharing:** Integrate with social media platforms to allow users to easily share their blog posts with their social networks.
- **Commenting System:** Include a commenting system to allow readers to leave comments on blog posts, engage in discussions, and provide feedback.
- **SEO Optimization:** Offer tools and features to optimize blog posts for search engines, meta tags, and SEO-friendly markup.
- **Analytics and Insights:** Provide built-in analytics tools to track key metrics such as page views, visitor demographics, referral sources, and engagement metrics.
- **Mobile Responsiveness:** Ensure that the blog platform and its themes are responsive and optimized for viewing on mobile devices.

1.5 HOW TO CHOOSE A BLOG PLATFORM8

When choosing a blog platform consider that the blog platform includes the following features:

- **Ease of Use**: C for a platform with an intuitive and user-friendly interface that allows you to easily create, edit, and publish blog posts without requiring extensive technical expertise.
- **Customization Options:** Look for a platform that offers a variety of customizable themes, templates, and design elements, allowing you to personalize the look and feel of your blog to match your brand or individual preferences.
- **Scalability:** Consider the scalability of the platform and its ability to accommodate the growth of your blog over time. Ensure that it can handle increased traffic, content volume, and functionality as your blog expands.
- **Content Management Features:** Evaluate the platform's content management system (CMS) and its features for organizing, categorizing, tagging, and scheduling blog posts. Make sure it meets your needs for content creation, editing, and publishing.
- **Media Support:** Check if the platform supports various types of media, including images, videos, audio files, and other multimedia content. Ensure that it provides tools for uploading, embedding, and managing media within your blog posts.
- **SEO Capabilities:** Look for a platform with built-in SEO tools and features to optimize your blog for search engines. This includes customizable URL structures, meta tags, sitemaps, and other SEO-friendly options to improve your blog's visibility and ranking in search results.
- **Social Integration:** Consider the platform's social integration capabilities, including features for social sharing, commenting, and user engagement. Ensure that it allows you to easily promote your blog posts across social media platforms and engage with your audience.

Fig 1.1: Considering Features for Choosing a Blog Platform

1.6 SOME BLOGGING PLATFORMS

- **Medium:** One of the most widely used blogging platforms. Medium is a blogging platform and social publishing network that focuses on quality content and community engagement. It offers a clean and minimalist writing interface, built-in social sharing, and the ability to reach a wide audience of readers.
- **WordPress.org:** WordPress.org is an open-source content management system (CMS) that offers extensive customization options, a vast library of plugins and themes, and robust features for blogging and website creation.
- **Blogger:** Owned by Google, Blogger is a free and easy-to-use blogging platform that allows users to create and publish blogs quickly. It offers built-in social sharing, integration with Google services, and customizable templates.

CHAPTER 2

SYSTEM ANALYSIS

System analysis is a systematic approach to understanding and improving a system by studying its various components and their interactions. This process involves a thorough examination of the current system, identification of its requirements and constraints, understanding stakeholder needs, evaluating risks, and proposing enhancements. The primary objective of system analysis is to ensure that the system effectively meets the needs of its users and operates efficiently

2.1 CURRENT SYSTEM OVERVIEW

2.1.1 PROBLEM DEFINITION

In today's digital landscape, blogging remains a vital medium for individuals and businesses to share ideas, stories, and expertise. However, despite the widespread use of blog platforms, there are significant shortcomings that hinder both creators and audiences.

Firstly, the user experience of many existing blog platforms leaves much to be desired. Navigating through complex interfaces and finding content efficiently can be frustrating for users. Moreover, with the increasing prevalence of mobile browsing, the lack of optimized experiences on smaller screens poses a significant accessibility challenge.

Customization is another area where current platforms fall short. Bloggers often struggle to personalize their sites beyond basic templates, resulting in a lack of individuality and brand identity. The limited flexibility in design options and layout customization further restricts creative expression.

Content management is a fundamental aspect of blogging, yet many platforms offer inadequate tools for content creation and editing. From cumbersome text editors to limited support for multimedia content integration, bloggers often face unnecessary obstacles in bringing their ideas to life.

Engagement tools are crucial for fostering a vibrant community around a blog, yet many platforms lack the necessary features for audience interaction. Basic functionalities like commenting systems and social sharing tools are often underdeveloped, hindering meaningful engagement between creators and readers.

Finally, search engine optimization (SEO) and content discoverability are essential for attracting new readers and growing an audience. Yet, many blog platforms offer inadequate SEO tools and fail to provide effective strategies for improving content discoverability, resulting in missed opportunities for exposure and growth.

2.2 REQUIREMENT ANALYSIS

Requirement analysis is a crucial phase in the software development lifecycle where the needs and objectives of a system or software application are identified, documented, and analysed. It involves gathering, documenting, and prioritizing the requirements that define what the system should accomplish and how it should function to meet the needs of its users and stakeholders.

2.2.1 FUNCTIONAL REQUIRMENTS

Functional requirements specify the specific functions, features, capabilities, and behaviours that a software system or application must possess to fulfil the needs of its users and stakeholders. These requirements describe what the system should do in terms of its functionality and how it should respond to various inputs or stimuli.

Functional requirements for "Collablog" typically encompass a range of features and capabilities that enable users to create, publish, manage, and interact with blog content effectively. Here are some common functional requirements for "Collablog" are:

- **User Authentication and Authorization:**
  - Users should be able to register accounts and log in securely.
  - Users can login from any third-party apps like Facebook, Gmail etc.
  - Role-based access control should be implemented to manage user permissions (authors, readers).

- **Content Creation and Management:**
  - Users should be able to create, edit, and publish blog posts.
  - Support for various content types, including text, images, videos, and embedded multimedia content.
  - Ability to categorize and tag posts for organization and easy retrieval.
- **User Interaction and Engagement:**
  - Commenting system allowing readers to leave comments on blog posts.
  - Social sharing functionality to enable users to share posts on social media platforms.
  - Like or upvote feature to allow readers to express appreciation for posts.
  - Notification system to alert users of new comments or interactions on their posts.
- **Content Discovery and Navigation:**
  - Search functionality enabling users to find specific posts or topics.
  - Tagging and categorization system for organizing and browsing content.
  - Archive or chronological browsing to explore older posts.
- **Multimedia Support:**
  - Ability to embed multimedia content such as videos, images, and audio files within blog posts.
  - Media library for managing and organizing uploaded media files.
- **SEO and Analytics:**
  - SEO optimization features to improve the discoverability of blog content on search engines.
  - Analytics dashboard providing insights into traffic, user engagement, and popular content.
- **Customization and Personalization:**
  - Personalized user profiles with options to add a bio, profile picture, and social media link
  - Users can edit their profiles as and when they want

2.2.2 NON-FUNCTIONAL REQUIREMENTS

Non-functional requirements (NFRs) specify criteria that describe how a system should behave, rather than what functions the system should perform. They focus on the characteristics of the system such as performance, security, usability, reliability, scalability, and other quality attributes. Unlike functional requirements which describe specific features and functionalities of the system, non-functional requirements define the overall behaviour, constraints, and qualities that the system must exhibit to meet user expectations and adhere to industry standards.

For a blog platform, non-functional requirements are just as important as functional requirements to ensure that the platform operates effectively, securely, and meets user expectations. Here are some non-functional requirements that could apply to a blog platform:

- **Performance:**
  - The platform should load blog posts and pages within a reasonable time frame, typically under 3 seconds.
  - It should be able to handle a large number of concurrent users without significant degradation in performance.
  - The database queries should be optimized to minimize response time.
- **Reliability:**
  - The platform should have a high uptime, with a target availability of at least 99.9%.
  - It should include mechanisms for backup and disaster recovery to prevent data loss.
  - Error handling and logging should be robust to identify and address issues promptly.
- **Scalability:**
  - The platform should be able to handle increasing traffic and content without a significant decrease in performance.
  - It should support horizontal scaling, allowing additional servers to be added as needed to distribute the load.
- **Security:**
  - User authentication and authorization mechanisms should be implemented securely to prevent unauthorized access.

- The platform should encrypt sensitive user data, such as passwords and personal information.

- **Usability:**
  - The user interface should be intuitive and easy to navigate, with clear labeling and consistent design elements.
  - Accessibility features should be implemented to accommodate users with disabilities.
  - The platform should support multiple devices and screen sizes, including mobile responsiveness.
- **Compatibility:**
  - The platform should be compatible with popular web browsers (e.g., Chrome, Firefox, Safari) and operating systems.
  - It should support various content formats (e.g., text, images, videos) and integrate with third-party services (e.g., social media sharing, analytics).
- **Regulatory and Compliance:**
  - The platform should comply with relevant data protection regulations, such as GDPR or CCPA, regarding user privacy and data handling.
  - It should include features for users to manage their data and privacy settings effectively.

2.3 FEASIBILITY STUDY

A feasibility study is a comprehensive analysis conducted to assess the viability of a proposed project or business idea. It aims to determine whether the project is technically, economically, legally, operationally, and schedule-wise feasible. The study provides detailed insights and evaluations to help decision-makers understand the potential outcomes, risks, and benefits of proceeding with the project. Regular monitoring and adaptation to user needs will be crucial for sustained growth and success.

2.3.1 TECHNICAL FEASIBILITY

The technical feasibility of developing “Collablog” involves assessing several key areas.

Firstly, the platform must offer a clean and intuitive user interface, ensuring ease of navigation and a responsive design that works

seamlessly across web and mobile devices. A robust content management system (CMS) is essential, featuring a rich text editor for writers, efficient content organization through tagging and categorization, and built-in SEO tools for enhancing discoverability.

Scalability is another critical factor; the infrastructure must handle high traffic volumes and support a growing user base without performance issues, potentially utilizing a microservices architecture to efficiently manage different platform components. Security is paramount, requiring data encryption, secure authentication and authorization mechanisms, and regular security audits to protect against vulnerabilities.

For the technology stack, a combination of HTML, CSS, and JavaScript, with frameworks like React.js for the frontend, is ideal. The backend can be powered by Node.js, with Firebase as the database. Hosting on cloud-based solutions like AWS or Google Cloud or Firebase ensures reliability and scalability. Overall, this comprehensive technical approach ensures the feasibility and sustainability of "Collablog".

2.3.2 BEHAVIOURIAL FEASIBILITY

Behavioural feasibility assesses how users and stakeholders will react to a new system, focusing on their acceptance, usability, and adaptability. For "Collablog", this means understanding and predicting user behaviour, preferences, and needs. The target audience includes writers and content creators who seek a platform to share their stories and expertise, readers looking for high-quality content, and organizations aiming to publish content for engagement and marketing purposes. Ensuring user acceptance involves providing an intuitive, user-friendly interface that makes content creation and discovery easy and enjoyable. The platform should offer incentives, such as revenue sharing and visibility opportunities, to attract and retain high-quality writers.

Additionally, features like community engagement tools and personalized content recommendations will enhance the user experience and encourage active participation. By addressing these factors, the platform can achieve high levels of user satisfaction and engagement, ensuring its long-term success and sustainability. Personalized content recommendations based on user preferences and behaviour can enhance the user experience by making content discovery more relevant and engaging. By focusing on these aspects, the platform can achieve high levels of user satisfaction and engagement, ensuring its long-term success and sustainability.

2.3.3 OPERATIONAL FEASIBILITY

Operational feasibility for “Collablog” involves establishing practicality in its day-to-day operations, resource management, and sustainability. To effectively manage the platform, a well-structured team is essential. This team typically includes frontend and backend developers, a UI/UX designer, testers, content moderators, editors, and customer support agents. These roles ensure smooth development, content management, and user support.

Operational processes such as content moderation and user support need to be meticulously planned. Implementing automated tools for content screening and establishing clear guidelines for human moderators can streamline content management

Infrastructure plays a critical role in operational feasibility. Reliable cloud-based hosting solutions, coupled with robust backup and recovery systems, ensure high availability and data integrity.

In conclusion, operational feasibility for a Medium-like website relies on a well-coordinated team, efficient processes, and scalable infrastructure. By focusing on these aspects, the platform can be effectively managed, ensuring sustained growth and user satisfaction.

2.3.4 SCHEDULING FEASIBILITY

The project was envisioned to unfold across distinct phases, each meticulously planned to facilitate smooth progress:

First, the Planning and Requirement Analysis phase spanned a month. During this time, we delineated the project's scope, objectives, and deliverables. Following this groundwork, the Design and Prototyping phase extended over two months. Here, we delved into wireframing and prototyping, shaping the user interface (UI) and user experience (UX) design. The Development phase unfolded across four months, representing the most intensive stage of the project. Our frontend developers meticulously translated the finalized UI design into code, leveraging HTML, CSS, and JavaScript frameworks. Simultaneously, backend developers embarked on constructing the server-side logic, database architecture, and APIs. Rigorous unit testing and code reviews underscored our commitment to quality and reliability. With the foundation laid, the Testing and Quality Assurance phase ensued for one month. Here, we subjected the website to exhaustive functional and usability testing, scrutinizing features and interactions to uncover any latent issues. Armed with invaluable feedback, we swiftly implemented bug fixes and optimizations to enhance the website's robustness and user experience. As the project

neared completion, the Deployment and Launch phase spanned an additional month. During this crucial period, we meticulously prepared the website for deployment to production servers. Beta testing with a select user base provided invaluable insights, allowing us to fine-tune our offering. Finally, orchestrating the official launch of the website, coupled with strategic marketing and promotion endeavours, marked the culmination of our efforts. Throughout this journey, careful resource allocation, proactive risk management, and steadfast communication and collaboration underpinned our progress. By adhering to established timelines and milestones, we navigated complexities with poise, ensuring the project's timely execution and eventual success.

2.3.5 ECONOMICAL FEASIBILITY

For economic feasibility, Economic analysis or cost/benefits analysis is the most frequently used technique for the effectiveness of a proposed system. it is a procedure to determine the benefits and savings that are expected from the proposed system and compare them with cost if the benefits outweigh the costs, a decision is taken to design and implement the system. otherwise, further justification or alternative in the proposed system will have to be made if it is to have a chance of being approved this is an ongoing effort that improves in accuracy at each phase of a system life cycle.

- Cost-Benefit Analysis: Assessing the potential costs involved in implementing a food ordering system against the anticipated benefits such as increased revenue, improved operational efficiency, and customer satisfaction.
- Market Demand: Analyzing the demand for online food ordering services in the target market, including factors like demographics, consumer preferences, and competitor analysis to gauge the revenue potential.
- Cost Reduction: Identifying opportunities to reduce costs through automation, streamlining processes, and optimizing resource allocation within the food ordering system.
- Risk Assessment: Conduct a thorough risk assessment to identify potential economic risks such as market volatility, regulatory changes, competitive pressures, and technological disruptions, and develop mitigation strategies accordingly.

CHAPTER 3

SYSTEM DESIGN

Designing is the most important phase of software development. It requires a careful planning and thinking on the part of the system designer. Designing software means to plan how the various parts of the software are going to achieve the desired goal. It should be done with utmost care because if the phase contains any error, then that will affect the performance of the system, as a result it may take more processing time, more response time, extra coding workload etc.

Software design sits at the technical kernel of the software engineering process and is applied regardless of the software process model that is used. After the software requirements have been analysed and specified, software design is the first of the three technical activities Designing, Coding and Testing that are required to build and verify the software. Each activity transforms information in such a manner that ultimately results in validated computer software.

3.1 DESIGN GOALS

Design goals of "Collablog" includes:

- User-Friendly Interface: Ensure easy navigation and intuitive tools for content creation and management.
- Responsive Design: Enable seamless access across various devices.
- Content Discovery: Facilitate easy content discovery through search, filters, and recommendations.
- Engagement Features: Foster community interaction with comments, likes, and social media integration.
- Performance and Speed: Optimize loading times for a smooth user experience.
- Accessibility: Design with accessibility standards to accommodate all users.
- Scalability: Build infrastructure to handle increasing traffic and content volume.
- Security: Implement robust measures to protect user data and prevent cyber threats.

3.2 USE CASE DIAGRAM

A use case diagram is a graphical representation of the interactions between users (actors) and a system, showcasing the various ways users interact with the system to achieve specific goals. It illustrates the functionalities or features of the system from a user's perspective and helps in understanding the system's behaviour in different scenarios.

Key components of a use case diagram include

- Actors: Actors represent users or external systems interacting with the system being modelled. They are depicted as stick figures or simple shapes outside the system boundary.
- Use Cases: Use cases represent specific functionalities or actions that the system provides to fulfill the needs of its users. They are depicted as ovals or ellipses within the system boundary and are connected to actors via lines to indicate interaction.
- Relationships: Relationships between actors and use cases illustrate the interactions between them. These relationships are represented by lines connecting actors to use cases.

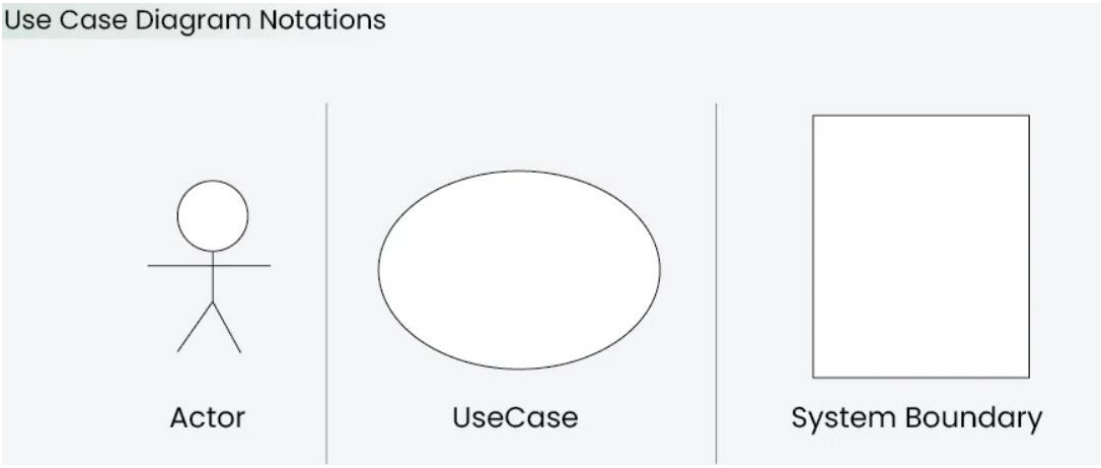


Fig 3.1 Use Case Diagram Notations

Use case diagrams are valuable tools for capturing and communicating the requirements of a system, providing stakeholders with a clear understanding of how users interact with the system and the functionalities it offers to meet their needs. They serve as a foundation for further analysis, design, and implementation of the system.

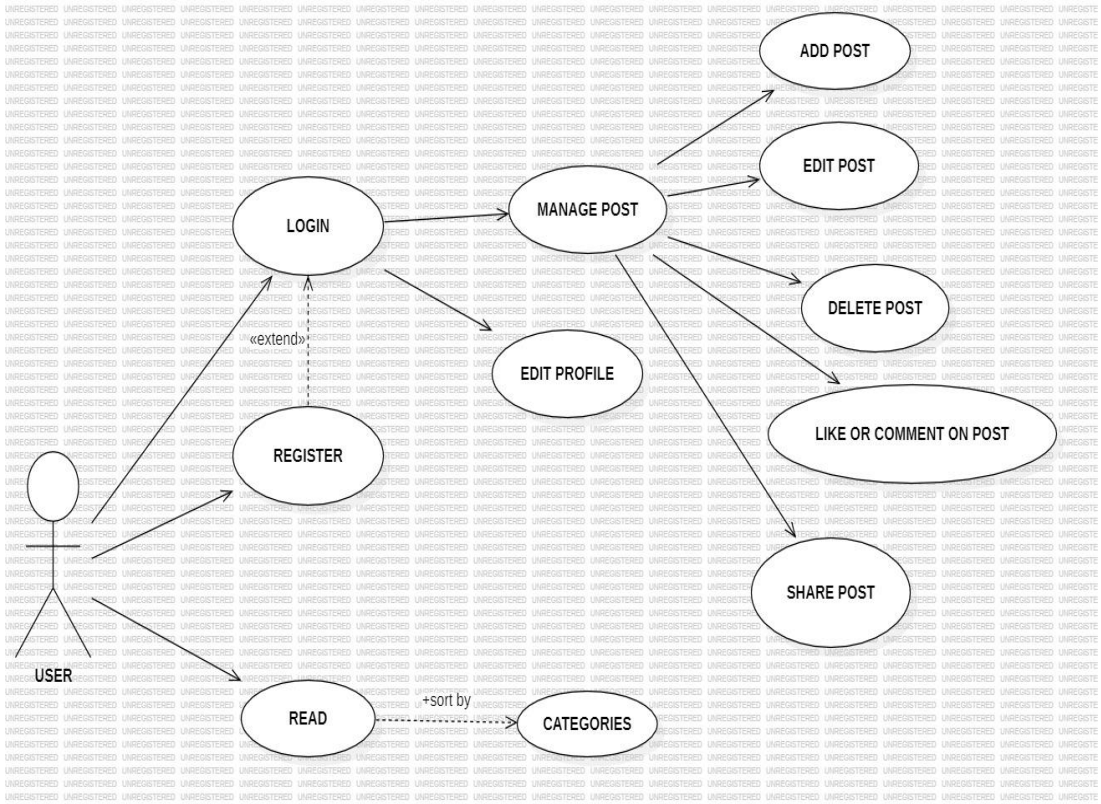


Fig 3.2 Use Case Diagram of "Collablog"

3.3 ENTITY RELATIONSHIP DIAGRAM

An Entity-Relationship (E-R) diagram is a graphical representation used to model the data and relationships within a database system. It visually depicts the entities (or objects), attributes, and relationships between entities. E-R diagrams are widely used in database design to illustrate the logical structure of a database.

Key components of an E-R diagram include:

- Entities: Entities are objects or concepts in the real world that are represented in the database. Each entity is depicted as a rectangle in the diagram.
- Attributes: Attributes are properties or characteristics of entities that describe them further. They are depicted as ovals connected to their respective entities.
- Relationships: Relationships represent the associations between entities. They indicate how entities are related to each other.They are depicted as diamond shapes connected to the related entities by lines. The cardinality and optionality of relationships can be indicated using symbols or annotations.

- Keys: Keys are attributes or combinations of attributes that uniquely identify each entity within an entity set. In an E-R diagram, primary keys are usually underlined or denoted by a key symbol.

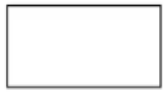
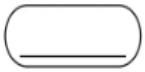
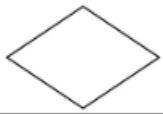
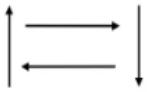
	**Rectangle**. It represents the entities, the things about which we seek information.
	**Week Entity.** It depends on another entity to exist.
	**Ellipse.** It shows attributes that are properties of the entity.
	**Primary Key / Attribute.** It is the unique, distinguishing property of the entity.
	**Multivalve Attribute.** It can have more than one value.
	**Derived Attribute.** It is based on another attribute to exist.
	**Diamond.** It shows relationships that provide the structure which draws information from multiple entities.
	**Arrows.** They identify the flow i.e. movement of information in a flow chart.

Fig 3.3 E-R Diagram Notations

E-R diagrams provide a visual representation of the database schema, enabling database designers to understand the structure of the data and relationships between entities. They serve as a blueprint for database implementation and help ensure the integrity and efficiency of the database design.

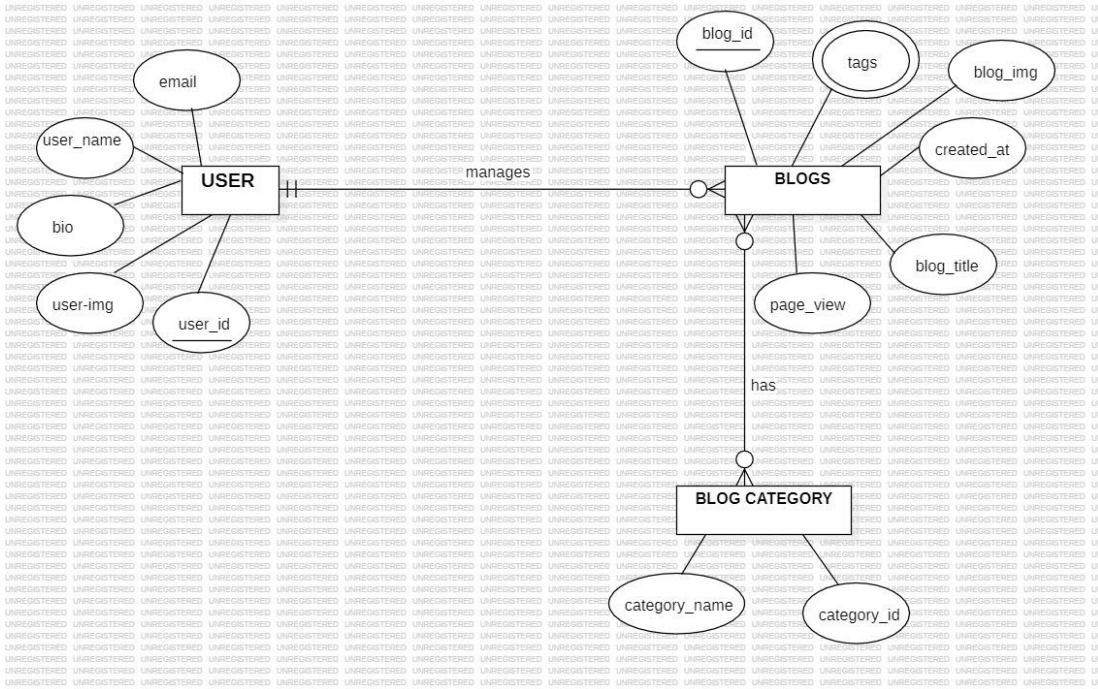


Fig 3.4 E-R Diagram for "Collablog"

3.4 ACTIVITY DIAGRAM

An activity diagram is a type of diagram in Unified Modelling Language (UML) that visually represents the workflow or activities within a system or process. It illustrates the sequence of actions or flow of control in a system, showing how different activities interact and the paths they take. Activity diagrams are particularly useful for modelling the logic of complex operations, business processes, and workflows.

Activity Diagrams are used to illustrate the flow of control in a system and refer to the steps involved in the execution of a use case. We can depict both sequential processing and concurrent processing of activities using an activity diagram ie an activity diagram focuses on the condition of flow and the sequence in which it happens.

- We describe what causes a particular event using an activity diagram.
- An activity diagram portrays the control flow from a start point to a finish point showing the various decision paths that exist while the activity is being executed.
- They are used in business and process modeling where their primary use is to depict the dynamic aspects of a system.

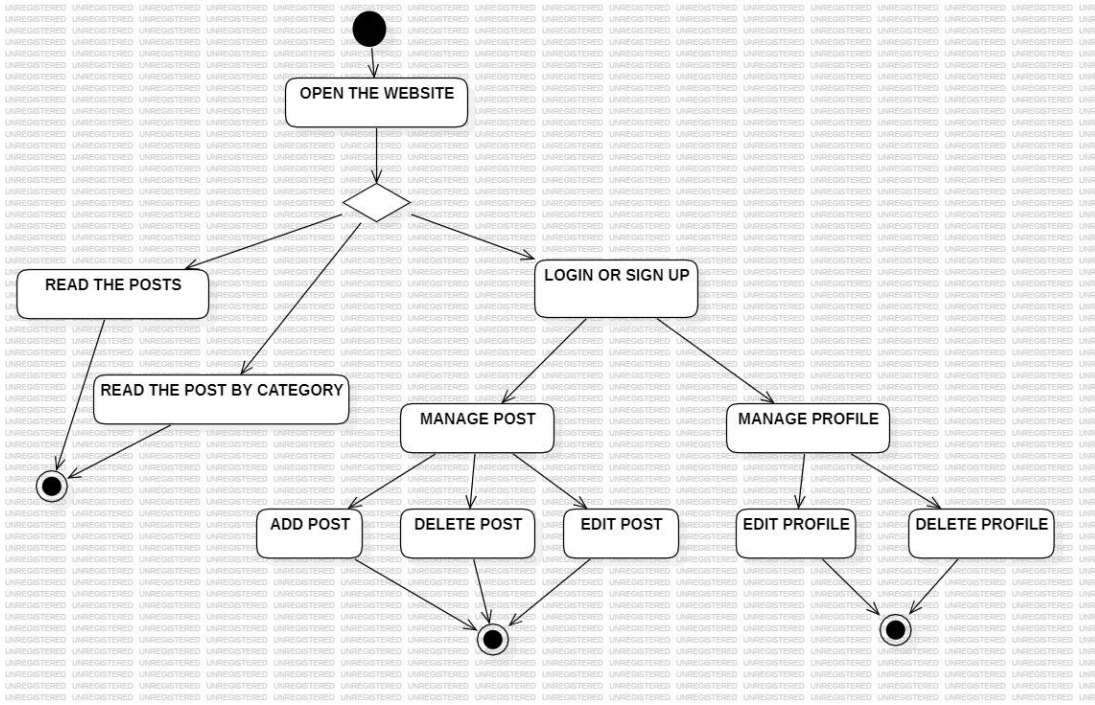


Fig 3.5 Activity Diagram for “Collablog”

CHAPTER 4

TECHNOLOGY USED AND SETUP

4.1 HARWARE REQUIREMENTS

Hardware requirements for a project refer to the specific physical components or devices needed to support the project's objectives. These requirements can vary significantly depending on the nature of the project. These requirements include the computing power, storage, network connectivity, memory etc.

The hardware requirements for accessing "Collablog" is enlisted in the table below:

Table 4.1: Hardware Requirements

S. No.	Description of system requirement
1	5 GB or more Hard disk.
2	8 GB RAM.
3	Core i5 7^{th} gen or above processor.
4	50 Mbps or more internet connectivity

1. 5 GB or more Hard disk:

This specifies the storage requirement for the PC. It should have a hard disk with a capacity of 5 gigabytes (GB) or more. This is where you store your operating system, software applications, and data. A PC with a 5 GB or larger hard disk provides ample storage for the operating system, software applications, and user data. This ensures smooth performance, accommodates growing storage needs, and allows for data backups and future expansion

2. **8 GB RAM:**

This sets the minimum random-access memory (RAM) requirement for the PC. It should have at least 8 gigabytes of RAM. RAM is essential for running applications and the operating system efficiently**.** Having 8 GB of RAM ensures smooth performance for running multiple applications simultaneously, faster operation of the operating system, seamless multitasking, and readiness for resource-intensive tasks like gaming or video editing.

3. **Core i5 7th gen or above processor:**

This specifies the processor requirement for the PC. It should have an Intel Core i3 processor or a more powerful one. The processor is a crucial component that determines the computer's overall speed and performance. A Core i5 7th gen or higher processor ensures strong performance and responsiveness for the PC. This Intel processor provides ample computing power for everyday tasks, multitasking, and even some demanding applications like photo or video editing. It offers a balance of speed, efficiency, and reliability, making it suitable for most users' needs.

4. **50 Mbps or more internet connectivity**

A 50 Mbps or higher internet connection ensures fast and reliable access to online resources, data, and collaborative tools required for the project. With this speed, users can quickly download and upload files, stream multimedia content, participate in video conferences, and collaborate with team members in real-time. It provides a seamless online experience, reducing wait times and enhancing productivity, particularly for projects that rely heavily on cloud-based services, remote collaboration, or data-intensive tasks.

4.2 SOFTWARE REQUIREMENTS

Software requirements for a project outline the specific programs, platforms, and functionalities needed to achieve project goals. They detail essential software components such as operating systems, development tools, databases, and any specialized software necessary for project execution. These requirements serve as a roadmap for software selection, development, integration, and testing throughout the project lifecycle.

The software requirements of “Collablog” are enlisted in the following software requirements table:

Table 4.2 Software Requirements

S. No.	Description	Type
1	Operating System	Windows 10 or 11
2	Front End	Vite, React JS, Vanilla JS, Tailwind CSS
3	Back End	Node JS
4	Database and Storage	Firebase
5	IDE	VS Code
6	Browser	Chrome, Firefox, Edge

1. Operating System

The specified operating system requirement for the project development environment is either Windows 10 or 11, with the option to use a newer version if available. This ensures compatibility with the latest software development tools, libraries, and frameworks required for the project. Additionally, it provides a consistent and stable platform for developers to create, test, and deploy the project's software components. By standardizing the operating system environment, it facilitates collaboration among team members and simplifies software configuration management, version control, and troubleshooting processes throughout the project lifecycle.

2. Front End

The frontend of a project is what users interact with directly, including the interface, design, and user experience. The goal is to create an intuitive, visually appealing, and responsive interface that guides users seamlessly through the application.

- **React JS**

  React.js is a JavaScript library for building user interfaces. It simplifies UI development by breaking it down into reusable components and managing updates efficiently with a virtual DOM. Developers use JSX to write UI components, enabling a declarative approach to defining UIs based on application state

- **Vanilla JS**

  Vanilla JS refers to using pure JavaScript without additional libraries or frameworks. It's lightweight, flexible, and helps developers focus on core JavaScript concepts. It's

great for projects where simplicity, performance, and browser compatibility are priorities.

- **Tailwind CSS**

  Tailwind CSS is a utility-first CSS framework that streamlines frontend development by providing a set of pre-defined utility classes for styling HTML elements. It prioritizes simplicity, responsiveness, and customization, making it easy for developers to create modern and responsive user interfaces efficiently.

- **Vite**

  Vite is a fast build tool for modern web development, specifically designed to optimize the development experience for frontend projects. It leverages native ES modules in modern browsers to deliver instant server startup and rapid hot module replacement (HMR). With Vite, developers can build and serve frontend applications quickly, enhancing productivity and facilitating a smooth development workflow.

3. Back End

The backend of a project comprises server-side code, databases, and APIs that handle data processing, business logic, authentication, security, and communication with clients. It powers the functionality of the application, manages data storage, and ensures that users can interact with the system securely and efficiently.

- **Node JS**

  Node.js is a runtime environment that allows you to use JavaScript for server-side development. It efficiently handles many connections simultaneously due to its event-driven and asynchronous nature. This makes it ideal for building scalable applications like real-time chat apps, APIs, and microservices. Using the same language for both frontend and backend simplifies development, and the npm ecosystem provides a vast array of modules to speed up the development process. Node.js is built on the fast V8 JavaScript engine, ensuring high performance.

4. Database and Storage

The database in a project provides essential data management capabilities, allowing for the storage, retrieval, and organization of data. Integrating a database involves installing the necessary drivers, connecting

to the database, defining schemas and models, and using these models in the application logic.

- **Firebase**

  Firebase is a Google platform offering real-time database and storage solutions for app development. The Realtime Database stores data as JSON and syncs it instantly across clients, while Cloud Firestore organizes data into documents and collections, supporting complex queries and offline access. Cloud Storage allows secure, scalable handling of user files like photos and videos. Firebase simplifies backend integration with easy setup and strong security, making it ideal for real-time applications and scalable project.

5. IDE

An Integrated Development Environment (IDE) is a software tool that combines various features to facilitate programming tasks. It typically includes a code editor, compiler/interpreter, debugger, build automation tools, version control integration, and project management capabilities. IDEs increase developer productivity by providing a centralized environment for coding, debugging, and managing projects, ultimately leading to more efficient software development.

- **Visual Studio Code**

  Visual Studio Code (VS Code) is a highly popular and versatile integrated development environment (IDE) developed by Microsoft. It's favoured by developers across various platforms and programming languages due to its extensive features and flexibility. Visual Studio Code (VS Code) is the preferred integrated development environment for coding.

6. Browser

Accessing a project via a web browser allows users to interact with web-based applications or websites. It provides accessibility, cross-platform compatibility, user-friendly interfaces, security features, scalability, and easy updates, making it a crucial aspect of modern computing. Mozilla Firefox, Google Chrome, Microsoft Edge, any of the browsers can be used to access the software.

4.3 INSTALLATION OF SOFTWARE REQUIREMENTS

i. Visual Studio Code (VS Code)

To install Visual Studio Code (VS Code):

- Download: Go to the official VS Code website (https://code.visualstudio.com) and download the installer for your operating system (Windows, macOS, or Linux).
- Run Installer: Once the download is complete, run the installer executable file. After the installation completes, launch Visual Studio Code from your system's applications menu or desktop shortcut.

ii. Node JS

To install Node.js:

- Download: Visit the official Node.js website (https://nodejs.org) and download the appropriate installer for your operating system (Windows, macOS, or Linux).
- Run Installer: Open the downloaded installer file. Finish the installation process. The installer will automatically add Node.js and npm (Node Package Manager) to your system PATH.
- Verify Installation: Open a terminal or command prompt and run the following commands to verify the installation:
    - node -v
    - npm -v

iii. Vite

To install Vite, execute the following lines on the terminal

- vite build
- npm create vite@latest .

iv. React JS

To install React JS, execute the following lines on the terminal

- npm create vite@latest my-project -- --template react
- cd my-project

To install various react components used in the project

- npm install moment
- npm install react-quill

- npm install react-tagsinput
- npm install react-share
- npm install react-router-dom react-icons
- npm install --save react-tag-input

v. Tailwind CSS

To install Tailwind CSS, execute the following lines on the terminal

- npm install -D tailwindcss postcss autoprefixer
- npx tailwindcss init -p

vi. Firebase

To install Firebase, execute the following on terminal

- npm install firebase
- npm login

4.4 DATABASE SETUP

Firestore organizes data into documents and collections, supporting complex queries and offline access. To setup the firestore database follow the following steps

- Create Firebase Project:
  - Go to the Firebase Console.
  - Click "Add project" and follow the prompts to create a new project.

- Enable Firestore:
  - In the Firebase Console, select your project.
  - Click on "Firestore Database" in the left-hand menu.
  - Click "Create database" and follow the setup prompts, choosing either production or test mode.

- Install Firebase SDK:
  - In your project directory, run: npm install firebase

- Initialize Firestore:

  In your project code, initialize Firestore:

  ```
  const firebase = require('firebase/app');

  require('firebase/firestore');
  ```

```
const firebaseConfig = {
apiKey: "YOUR_API_KEY",
authDomain: "YOUR_PROJECT_ID.firebaseapp.com",
projectId: "YOUR_PROJECT_ID",
storageBucket: "YOUR_PROJECT_ID.appspot.com",
messagingSenderId: "YOUR_MESSAGING_SENDER_ID",
appId: "YOUR_APP_ID"
};
firebase. initializeApp(firebaseConfig);
const db = firebase. firestore ();
```

4.5 STORAGE SETUP

Firebase Storage allows secure, scalable handling of user files like photos and videos. To use the firebase storage, we first need to setup the storage. Follow the following steps to set up the firebase storage

- In the Firebase Console, select your project.
- Click on "Storage" in the left-hand menu.
- Click "Get Started" and follow the prompts to set up Firebase Storage in your project.

CHAPTER 5

TESTING

5.1 SOFTWARE TESTING

Software testing is an integral component of the software development lifecycle (SDLC) that aims to identify and rectify defects within a software application. This process safeguards the quality and functionality of the software before its release. Through a systematic approach, software testing verifies if the software adheres to the predetermined requirements and delivers the intended functionalities.

Testing methodologies encompass various techniques designed to uncover errors, inconsistencies, or missing features. These techniques can be broadly categorized into functional and non-functional testing. Functional testing assesses if the software performs its designated tasks as specified in the requirements documents. Non-functional testing, on the other hand, evaluates aspects like performance, usability, security, and compatibility.

By meticulously examining the software, testers can pinpoint potential issues that could hinder user experience or system stability. These issues may include program crashes, incorrect data handling, or security vulnerabilities. Early detection of such defects allows developers to address them promptly, preventing them from propagating into later stages of development.

Software testing not only identifies shortcomings but also validates if the software fulfils the established requirements. This verification process ensures the software is aligned with its intended purpose and delivers the value proposition envisioned during the initial planning stages.

Furthermore, testing plays a pivotal role in enhancing the overall quality of the software. Through rigorous testing procedures, the software's performance, usability, and security are demonstrably improved. Performance testing evaluates the responsiveness and stability of the software under various load conditions. Usability testing focuses on how users interact with the software, identifying any design flaws or areas for improvement. Security testing safeguards the software from unauthorized access, data breaches, and other malicious attacks A strategic approach to software testing has the generic characteristics:

- Testing Begins at the module level and works “outwards” towards the integration of the entire computer-based system.
- Different testing techniques are appropriate at different points of time.
- Testing and debugging are different activities, but debugging must be accommodated in the testing strategy

5.2 TESTING METHODLOGIES

Various testing methodologies were employed to ensure comprehensive coverage and validation of the software:

5.2.1 UNIT TESTING

The module interface is tested to ensure that information properly flows into and out of the program unit under test. The unit testing is normally considered as an adjunct step to coding step. Because modules are not a standalone program, drivers and/or stubs software must be developed for each unit. A driver is nothing more than a “main program” that accepts test cases data and passes it to the module. A stub serves to replace the modules that are subordinate to the modules to be tested. A stub may do minimal data manipulation, prints verification of entry and returns.

Approaches used for Unit Testing were:

- **Functional Test**: Each part of the code was tested individually and the panels were tested individually on all platforms to see if they are working properly.
- **Performance Test:** These determined the amount of execution time spent on various parts of units and the resulting throughput, response time given by the module.
- **Stress Test:** A lot of test files were made to work at the same time in order to check how much workloads can the unit bear.
- **Structure Test**: These tests were made to check the internal logic of the program and traversing particular execution paths.

5.2.2 INTEGRATION TESTING

Integration testing is the next step after unit testing in the software development lifecycle (SDLC). It focuses on verifying how different software units or modules work together as a whole. Imagine you've meticulously crafted individual bricks for a building (unit testing), and integration testing is like checking how these bricks fit together to form a sturdy wall. If they all work individually, they should

work when we put them together. The problem of course is "putting them together". This can be done in two ways:

- **Top-down integration:** Modules are integrated by moving downwards through the control hierarchy, beginning with main control module are incorporated into the structure in either a depth first or breadth first manner.
- **Bottom-up integration:** It begins with construction and testing with atomic modules i.e. modules at the lowest level of the program structure. Because modules are integrated from the bottom up, processing required for the modules subordinate to a given level is always available and the need of stubs is eliminated.

Testing includes Verification and Validation

- **Verification** is a process of confirming that software meets its specification.
- **Validation** is the process of confirming that software meets the customer's requirements.

5.2.3 SYSTEM TESTING

System testing is a type of software testing that evaluates the overall functionality and performance of a complete and fully integrated software solution. It tests if the system meets the specified requirements and if it is suitable for delivery to the end-users. This type of testing is performed after the integration testing and before the acceptance testing.

System Testing is a type of software testing that is performed on a complete integrated system to evaluate the compliance of the system with the corresponding requirements. In system testing, integration testing passed components are taken as input. The goal of integration testing is to detect any irregularity between the units that are integrated together. System testing detects defects within both the integrated units and the whole system. The result of system testing is the observed behaviour of a component or a system when it is tested.

System Testing Process:

System Testing is performed in the following steps:

- Test Environment Setup: Create testing environment for the better-quality testing.
- Create Test Case: Generate test case for the testing process.
- Create Test Data: Generate the data that is to be tested.

- Execute Test Case: After the generation of the test case and the test data, test cases are executed.
- Defect Reporting: Defects in the system are detected.
- Regression Testing: It is carried out to test the side effects of the testing process.
- Log Defects: Defects are fixed in this step.
- Retest: If the test is not successful then again test is performed.

5.2.4 ACCEPTANCE TESTING

It is formal testing according to user needs, requirements, and business processes conducted to determine whether a system satisfies the acceptance criteria or not and to enable the users, customers, or other authorized entities to determine whether to accept the system or not.

Acceptance Testing is the last phase of software testing performed after System Testing and before making the system available for actual use.

5.3 TESTING PROCESS

5.3.1 TEST SUITE

A test suite is a collection of test cases designed to validate that a software program performs as expected. It is an organized and systematic approach to testing, ensuring that different parts of the application are tested comprehensively. Test suites can vary in size and complexity depending on the scope of the testing required.

Key Components of a Test Suite:

- Test Case IDs
- Test Cases
- Setup Instructions
- Execution Steps
- Expected Results

These Test Cases have been developed using the manual Testing technique.

Test Case ID	Description	Preconditions	Test Steps	Expected Result
TC001	User Registration	User is on the registration page	1. Enter a valid email address.2. Enter a valid password. 3. Click "Register".	User should see a confirmation message and be redirected to the home page.
TC002	User Login	User is on the login page and has a registered account	1. Enter registered email. 2. Enter correct password.3. Click "Login".	User should be redirected to the home page and see a welcome message.
TC003	Login with Invalid Credentials	User is on the login page	1. Enter invalid email.2. Enter incorrect password. 3. Click "Login".	Error message should be displayed indicating invalid credentials.
TC004	Create a New Post	User is logged in and on "New Post" page	1. Enter post title. 2. Enter post content. 3. Click "Publish".	Post should be created and user redirected to post's detail page.
TC005	Edit an Existing Post	User is logged in and on "Edit Post" page for existing post	1. Modify post title. 2. Modify post content. 3. Click "Save".	Post should be updated with new title and content, and user redirected to post's detail page.

TC006	Delete a Post	User is logged in and viewing their post	1. Click "Delete" button on post. 2. Confirm deletion.	Post should be deleted and user redirected to home page; post should no longer be visible.
TC007	Add a Comment to a Post	User is logged in and viewing a post detail page	1. Enter comment. 2. Click "Submit".	Comment should be added and displayed below the post.
TC008	View User Profile	User is logged in	Click on user profile link.	User should be redirected to their profile page displaying their information and list of posts.
TC009	Search Posts	User is on home page	1. Enter keyword in search bar. 2. Click "Search" or press Enter.	List of posts matching keyword should be displayed.
TC010	Responsive Design Check	User is on any page of the application	1. Resize browser window to different screen sizes. 2. Check layout and functionality.	Application should adapt to different screen sizes, remain usable and visually appealing.

Table 5.1 Test Cases for “Collablog”

5.3.2 AUTOMATION TESTING

The Test Suite's test cases have been tested using the automation testing tool Playwright Automating the testing of a website using Playwright in TypeScript involves several steps. Below is a general outline of how you can approach this task:

1. Installation

First, you need to install Playwright and TypeScript in your project:

- npm install playwright @types/node typescript ts-node --save-dev

2. Setting up TypeScript

Initialize TypeScript in your project:

- npx tsc --init

This will generate a tsconfig.json file where you can configure TypeScript options.

3. Write Scripts

- **TC001: User Registration**

```
const { test, expect } = require('@playwright/test');

test('User Registration', async ({ page }) => {

await page.goto('http://localhost:3000/register');

await page.fill('input[name="email"]',
'testuser@example.com');

await page.fill('input[name="password"]', 'password123');

await page.click('button[type="submit"]');

await expect(page).toHaveURL('http://localhost:3000/');

await expect(page.locator('text=Registration
successful')).toBeVisible();

});
```

- **TC002: User Login**

```
test('User Login', async ({ page }) => {

await page.goto('http://localhost:3000/login');

await page.fill('input[name="email"]',
'registereduser@example.com');
```

```
await page.fill('input[name="password"]', 'password123');
await page.click('button[type="submit"]');
await expect(page).toHaveURL('http://localhost:3000/');
await expect(page.locator('text=Welcome')).toBeVisible();
});
```

- **TC003: Login with Invalid Credentials**

```
test('Login with Invalid Credentials', async ({ page }) => {
await page.goto('http://localhost:3000/login');
await page.fill('input[name="email"]',
'invaliduser@example.com');
await page.fill('input[name="password"]',
'wrongpassword');
await page.click('button[type="submit"]');
await expect(page.locator('text=Invalid
credentials')).toBeVisible();
});
```

- **TC004: Create a New Post**

```
test('Create a New Post', async ({ page }) => {
await page.goto('http://localhost:3000/login');
await page.fill('input[name="email"]',
'registereduser@example.com');
await page.fill('input[name="password"]', 'password123');
await page.click('button[type="submit"]');
await page.goto('http://localhost:3000/new');
await page.fill('input[name="title"]', 'Test Post');
await page.fill('textarea[name="content"]', 'This is a test
post.');
await page.click('button[type="submit"]');
await expect(page).toHaveURL(/\/post\/\d+/);
await expect(page.locator('text=Test Post')).toBeVisible();
});
```

- **TC005: Edit an Existing Post**

```
test('Edit an Existing Post', async ({ page }) => {
await page.goto('http://localhost:3000/login');
await page.fill('input[name="email"]', 'registereduser@example.com');
await page.fill('input[name="password"]', 'password123');
await page.click('button[type="submit"]');
await page.goto('http://localhost:3000/post/1/edit');
await page.fill('input[name="title"]', 'Updated Test Post');
await page.fill('textarea[name="content"]', 'This is the updated test post.');
await page.click('button[type="submit"]');
await expect(page).toHaveURL(/\/post\/1/);
await expect(page.locator('text=Updated Test Post')).toBeVisible();
});
```

- **TC006: Delete a Post**

```
test('Delete a Post', async ({ page }) => {
await page.goto('http://localhost:3000/login');
await page.fill('input[name="email"]', 'registereduser@example.com');
await page.fill('input[name="password"]', 'password123');
await page.click('button[type="submit"]');

await page.goto('http://localhost:3000/post/1');
await page.click('button.delete-post');
await page.click('button.confirm-delete');
await expect(page).toHaveURL('http://localhost:3000/');
await expect(page.locator('text=Test Post')).not.toBeVisible();
});
```

- **TC007: Add a Comment to a Post**

  test('Add a Comment to a Post', async ({ page }) => {

  await page.goto('http://localhost:3000/login');

  await page.fill('input[name="email"]', 'registereduser@example.com');

  await page.fill('input[name="password"]', 'password123');

  await page.click('button[type="submit"]');

  await page.goto('http://localhost:3000/post/1');

  await page.fill('textarea[name="comment"]', 'This is a test comment.');

  await page.click('button[type="submit"]');

  await expect(page.locator('text=This is a test comment.')).toBeVisible();

  });

- **TC008: View User Profile**

  test('View User Profile', async ({ page }) => {

  await page.goto('http://localhost:3000/login');

  await page.fill('input[name="email"]', 'registereduser@example.com');

  await page.fill('input[name="password"]', 'password123');

  await page.click('button[type="submit"]');

  await page.click('a[href="/profile"]');

  await expect(page).toHaveURL('http://localhost:3000/profile');

  await expect(page.locator('text=User Profile')).toBeVisible();

  });

- **TC009: Search Posts**

  test('Search Posts', async ({ page }) => {

  await page.goto('http://localhost:3000/');

  await page.fill('input[name="search"]', 'Test');

```
await page.press('input[name="search"]', 'Enter');
await expect(page.locator('text=Search Results')).toBeVisible();
await expect(page.locator('text=Test Post')).toBeVisible();
});
```

- **TC010: Responsive Design Check**

```
test('Responsive Design Check', async ({ page }) => {
await page.goto('http://localhost:3000/');
const sizes = [
{ width: 1200, height: 800 },
{ width: 1024, height: 768 },
{ width: 768, height: 1024 },
{ width: 375, height: 667 },
];
for (const size of sizes) {
await page.setViewportSize(size);
await page.waitForTimeout(1000); // wait for layout to adjust
await expect(page.locator('header')).toBeVisible();
await expect(page.locator('footer')).toBeVisible();
}
});
```

4. Running the Test Cases

You can run your TypeScript test script using ts-node:

- npx ts-node your_test_script.ts

CHAPTER 6

IMPLEMENTATION

Once the system was tested, the implementation phase started. A crucial phase in the system development life cycle is successful implementation of new system design. Implementations simply mean converting new system design into operation. This is the moment of truth the first question that strikes in every one's mind that whether the system will be able to give all the desires results as expected from system. The implementation phase is concerned with user training and file conversion.

The term implementation has different meanings, ranging from the conversion of a basic application to a complete replacement of computer system Implementation is used here to mean the process of converting a new or revised system design into an operational one. Conversion is one aspect of implementation. The other aspects are the post implementation review and software maintenance. There are three types of implementations

- Implementation of a computer system to replace a manual system
- Implementation of a new computer system to replace an existing one
- Implementation of a modified application to replace an existing one

6.1 MODULES IMPLEMENTATION

In computer software, a module is an extension to a main program dedicated to a specific function. In programming, a module is a section of code that is added in as a whole or is designed for easy reusability

The proposed system of "Collablog: Where Creativity Meets Community" has the following modules

1. User authentication
2. Content Creation
3. Content Reading and Category Reading
4. Sharing
5. Like and Commenting
6. Follow
7. Searching

6.2 FLOWCHARTS

6.2.1 USER AUTHENTICATION

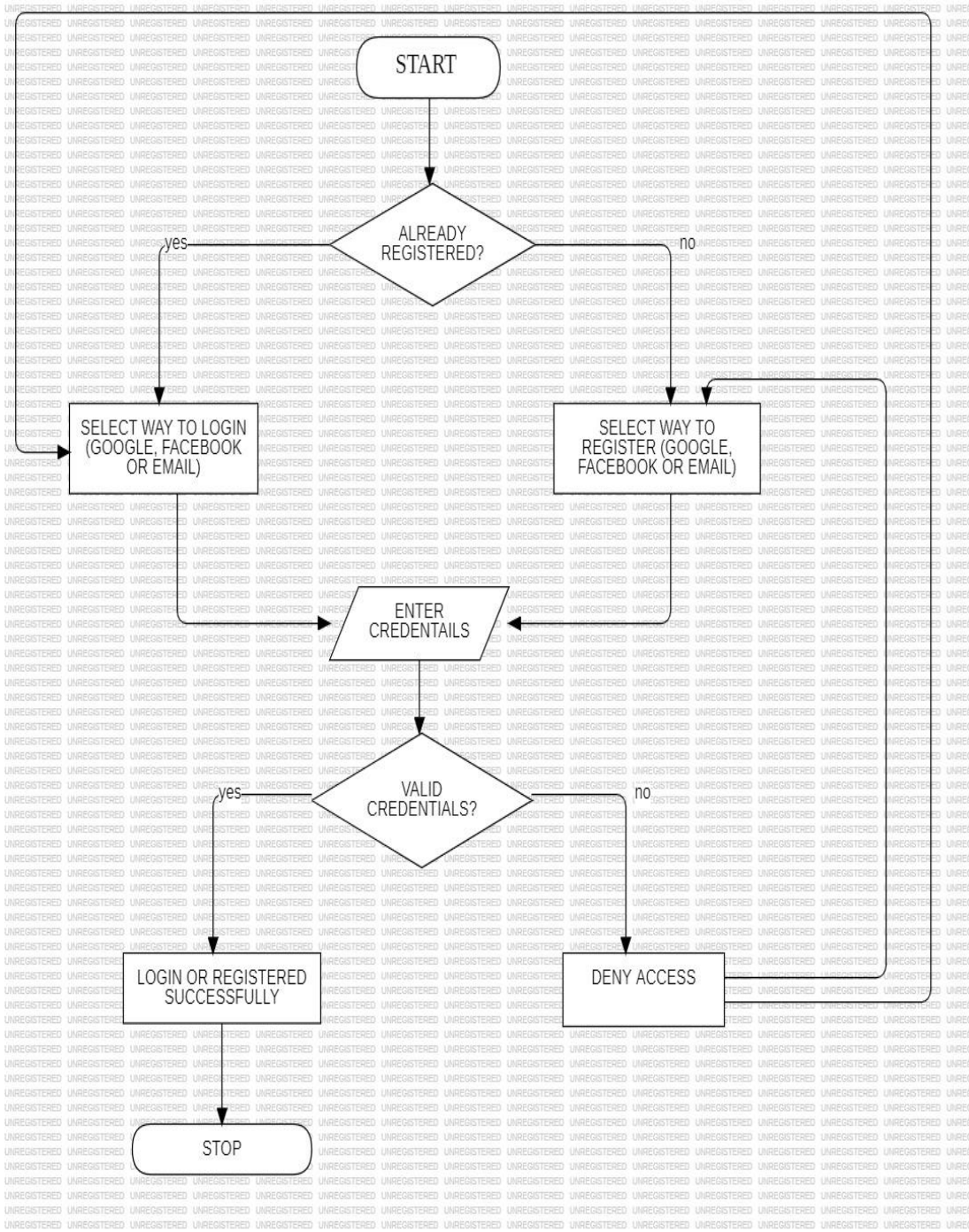


Fig 6.1: Flowchart of User Authentication Module

6.2.2 CONTENT CREATION

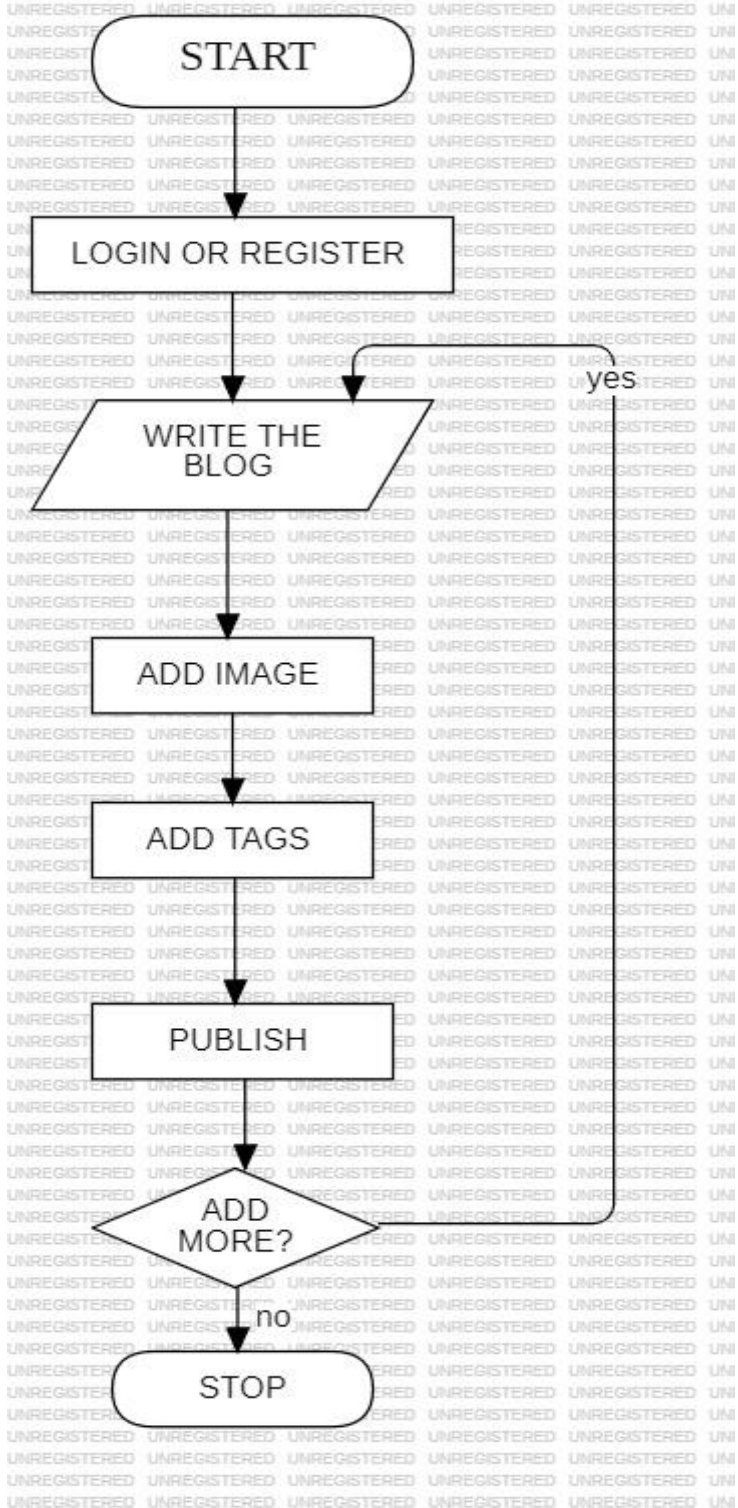


Fig 6.2: Flowchart of Content Creation Module

6.2.3 CONTENT READING

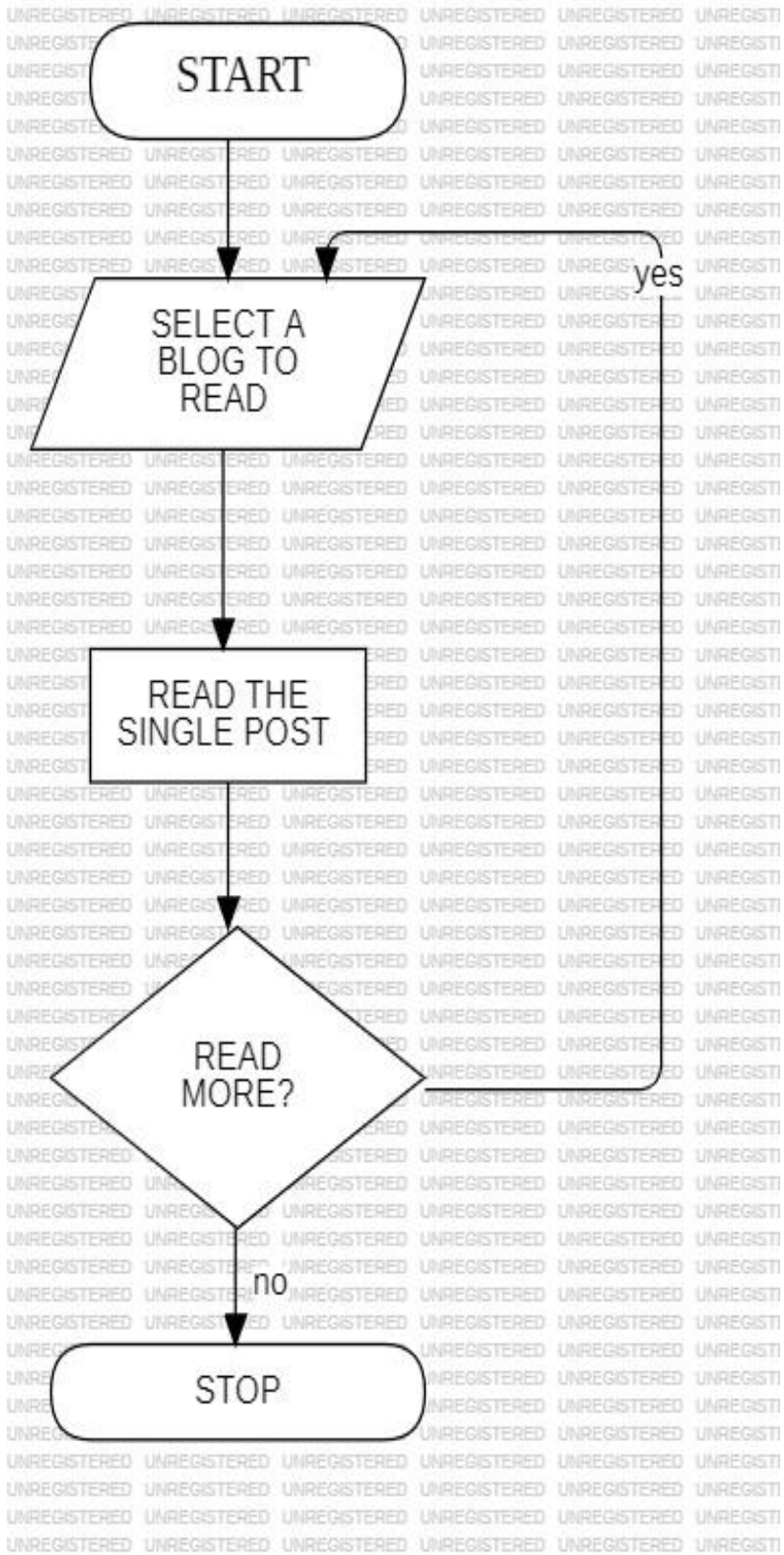


Fig 6.3: Flowchart for Content Reading Module

6.2.4 SHARING

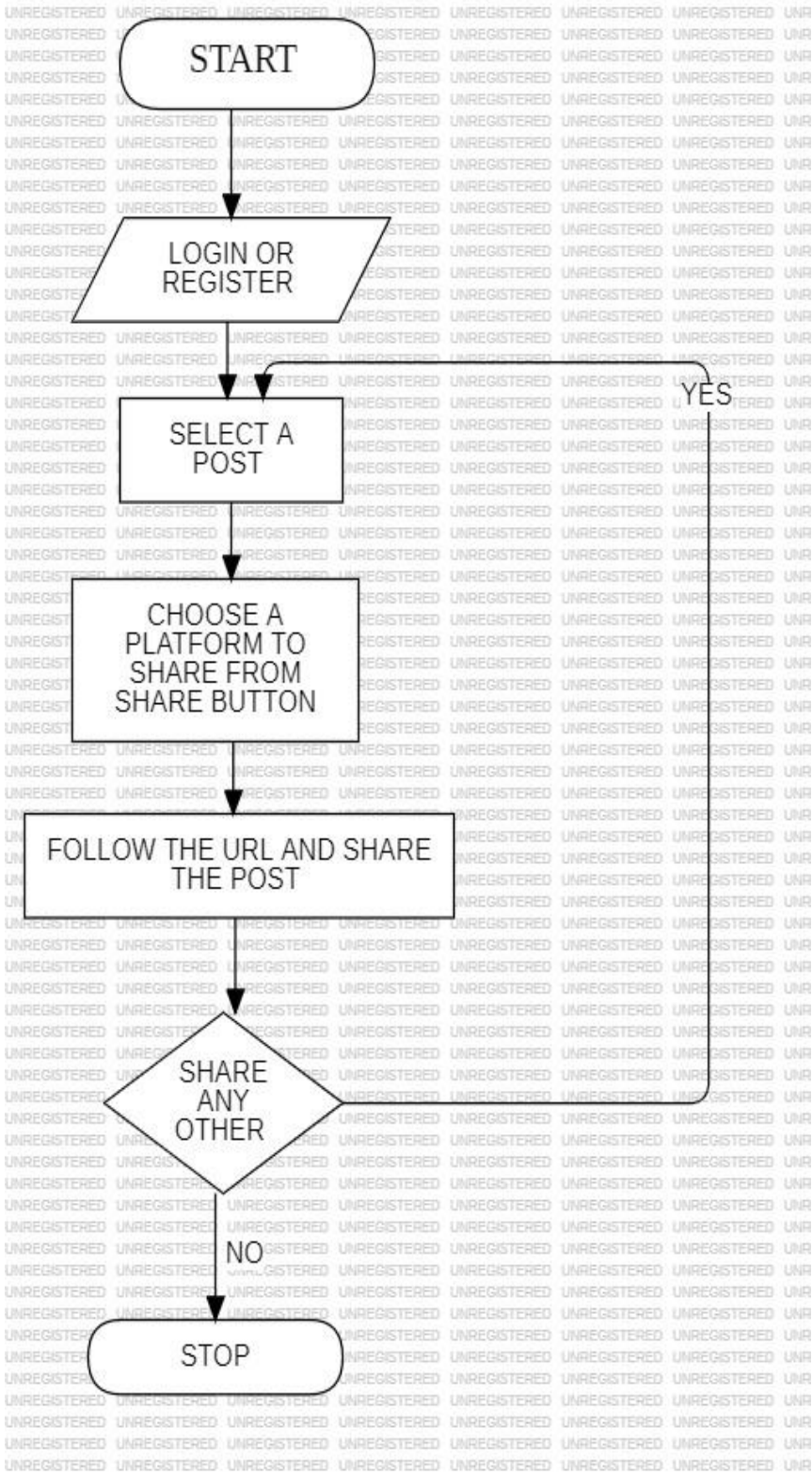


Fig 6.4: Flowchart for Sharing Module

6.2.5 LIKE AND COMMENTING

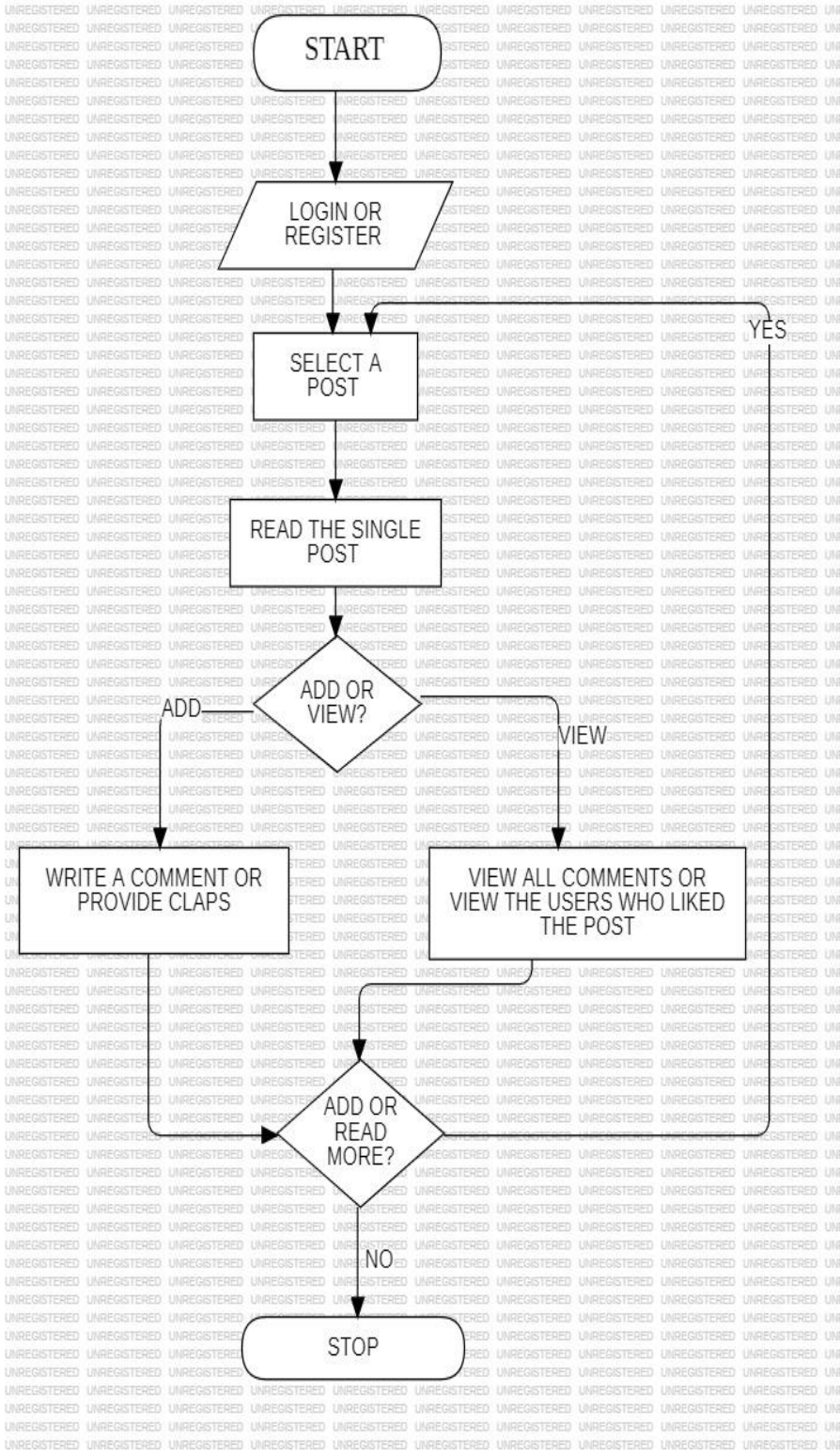


Fig 6.5: Flowchart of Like and Commenting Module

6.2.6 CATEGORY READING

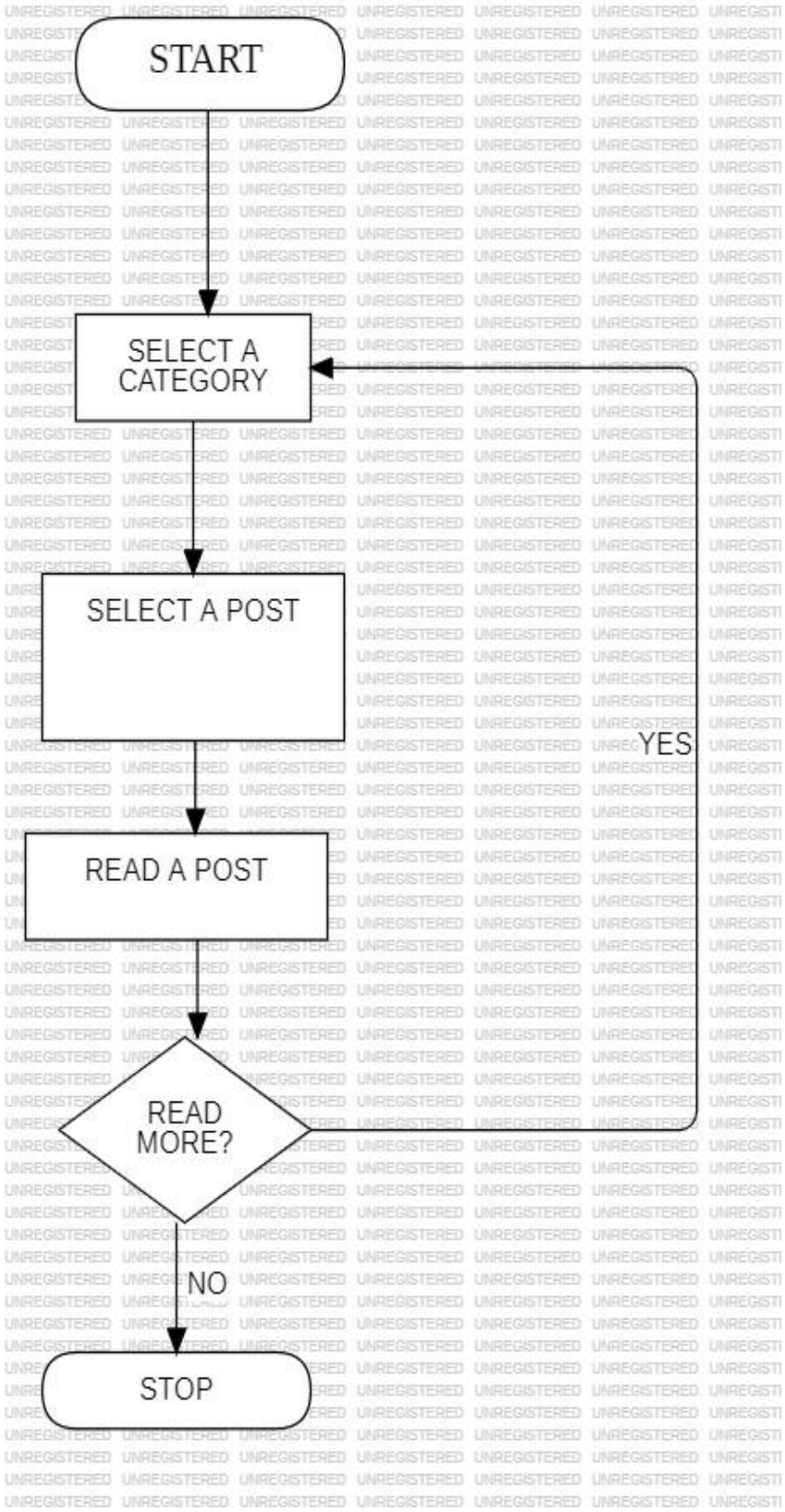


Fig 6.6: Flowchart of Category Reading Module

6.2.7 SEARCH

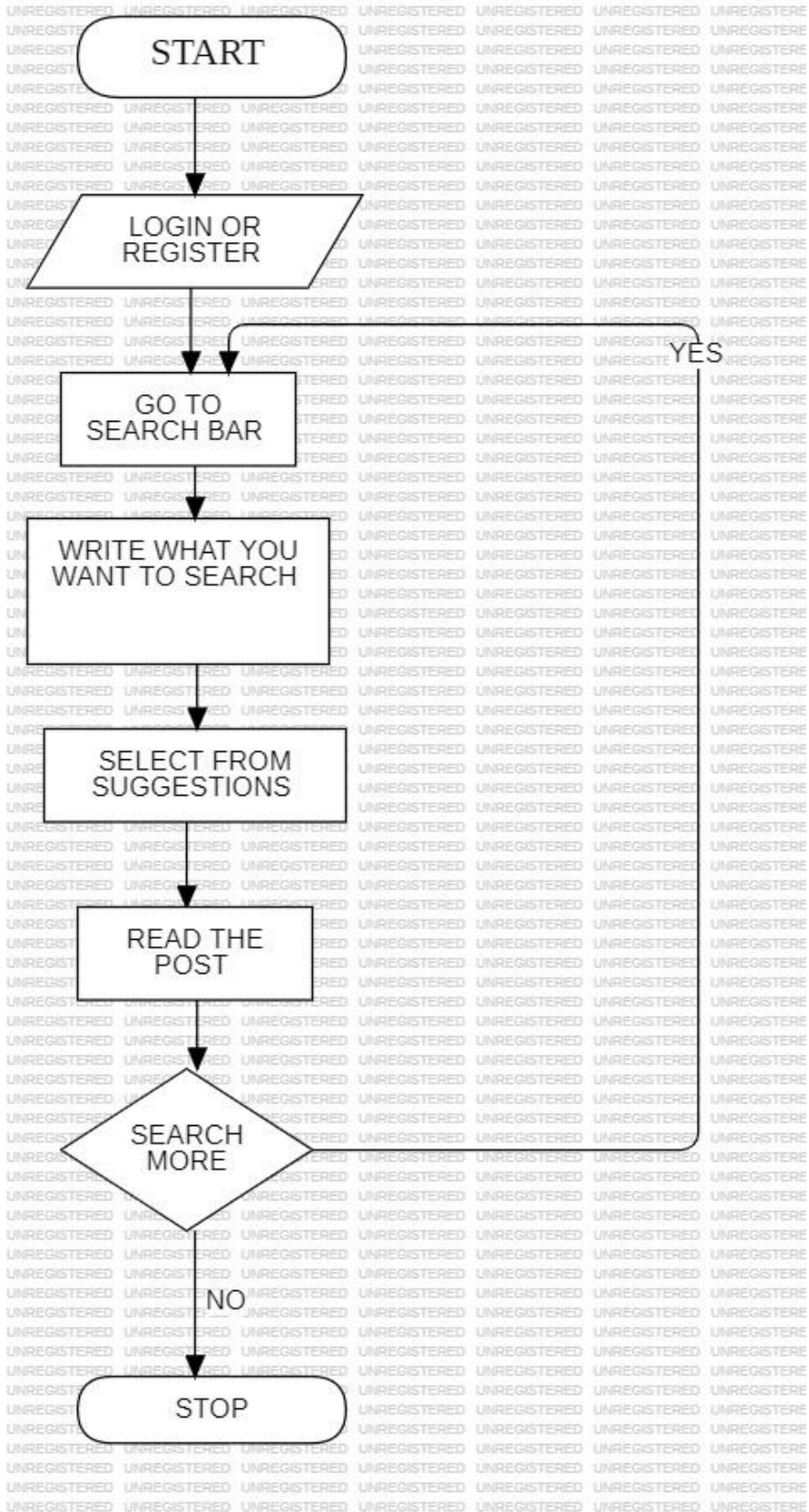


Fig 6.7 Flowchart of Search Module

6.2.8 FOLLOW

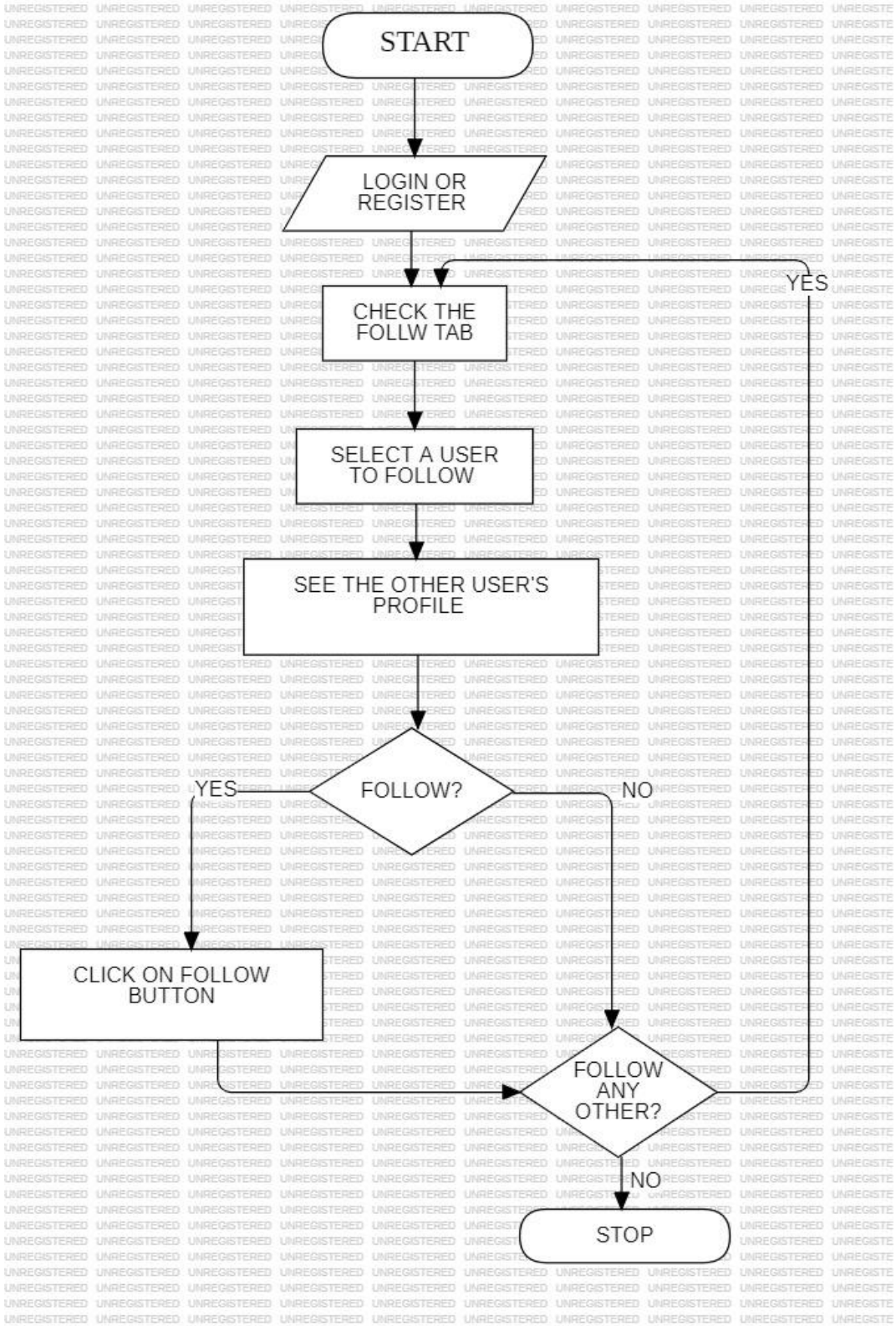


Fig 6.8 : Flowchart of Follow Module

CHAPTER 7

DEPLOYMENT

7.1 OVERVIEW OF DEPLOYMENT PROCESS

Deployment is the pivotal phase in the software development lifecycle where the developed application or system is made accessible and operational for its intended users. It involves the process of transferring the finalized software from the development environment to the production environment, where it can be utilized in real-world scenarios. Deployment encompasses a series of steps, including configuring servers, setting up databases, transferring files, and ensuring the proper functioning of the application. Depending on the nature of the software, deployment may involve deploying to local servers, cloud platforms, or third-party hosting services. The goal of deployment is to make the software available to users while ensuring its stability, reliability, and security. It marks the culmination of development efforts and the beginning of the software's operational phase, where it can deliver value to its users and stakeholders. Successful deployment requires careful planning, coordination, and testing to mitigate potential risks and ensure a smooth transition to the production environment.

The deployment process involves preparing the application by reviewing and testing the code, setting up the production environment, and configuring servers. The code is then transferred to the production server, dependencies are installed, and the application is deployed. Database migrations and seeding are performed as needed, and configurations, including SSL/TLS for security, are set up. After deployment, smoke tests and user acceptance testing are conducted. Monitoring, logging, and alert systems are established for ongoing maintenance. A rollback plan with backups is also prepared to revert to a previous version if necessary. This ensures the application is accessible, functional, and secure for end-user. “Collablog” is deployed using Firebase Hosting.

7.2 FIREBASE DEPLOYMENT

Firebase deployment streamlines the process of making web applications or mobile apps accessible to users worldwide. Leveraging Firebase Hosting, deployment becomes seamless, enabling developers to focus more on building features rather than managing servers. Firebase Hosting offers a fast and secure platform for hosting static and dynamic content, with features like automatic SSL certification and CDN integration for efficient content delivery. Deploying to Firebase involves using the Firebase Command Line Interface (CLI) to initialize, configure, and deploy the project. Once deployed, Firebase provides a hosting URL where the application is accessible. Additionally, Firebase offers other services like Firestore for database needs, Firebase Authentication for user authentication, and Firebase Storage for file storage, all of which can be seamlessly integrated into the deployed application. Firebase's real-time database capabilities ensure that updates made to the application are immediately reflected to users, making it ideal for applications requiring real-time data synchronization. Overall, Firebase deployment simplifies the deployment process, ensuring scalability, reliability, and security, while allowing developers to focus on building exceptional user experiences.

7.3 DEPLOYMENT PROCESS

Collablog is deployed using the Firebase Hosting Component. The following steps were followed to deploy the project:

- Prepare Your Project: Make sure your project is fully developed and ready for deployment. This includes optimizing code, handling environment variables, and ensuring all dependencies are up to date.
- Install Firebase Tools: If you haven't already, install the Firebase command-line tools (firebase-tools) globally on your machine using npm. Run the following line on the terminal
    - npm install -g firebase-tools
- Login to Firebase: Use the Firebase CLI to log in to your Firebase account:
    - firebase login
- Initialize Firebase Project: Navigate to your project directory and initialize a Firebase project:
    - firebase init
- Follow the prompts to select Firebase Hosting and your project from the Firebase console. Choose the appropriate options for your project configuration.

- Build Your Project: Generate a production build of your project. In your project directory, run:
    - npm run build
- Deploy to Firebase Hosting: Once the build is complete, deploy your project to Firebase Hosting:
    - firebase deploy
- Verify Deployment: After deployment, Firebase will provide you with a hosting URL where your website is now live. Visit the provided URL to verify that your website is successfully deployed and functional.
- Configure Domain (Optional): If you have a custom domain, you can configure it to point to your Firebase Hosting site. Follow the instructions provided by Firebase to set up custom domain hosting.
- Continuous Deployment (Optional): Set up continuous integration and deployment (CI/CD) pipelines using tools like GitHub Actions or GitLab CI to automate the deployment process whenever changes are pushed to your repository.
- Monitor and Maintain: Regularly monitor your deployed website for any issues or performance bottlenecks. Update your website as needed and maintain its functionality over time.

CHAPTER 8

PROJECT SNAPSHOTS

8.1 HOME PAGE

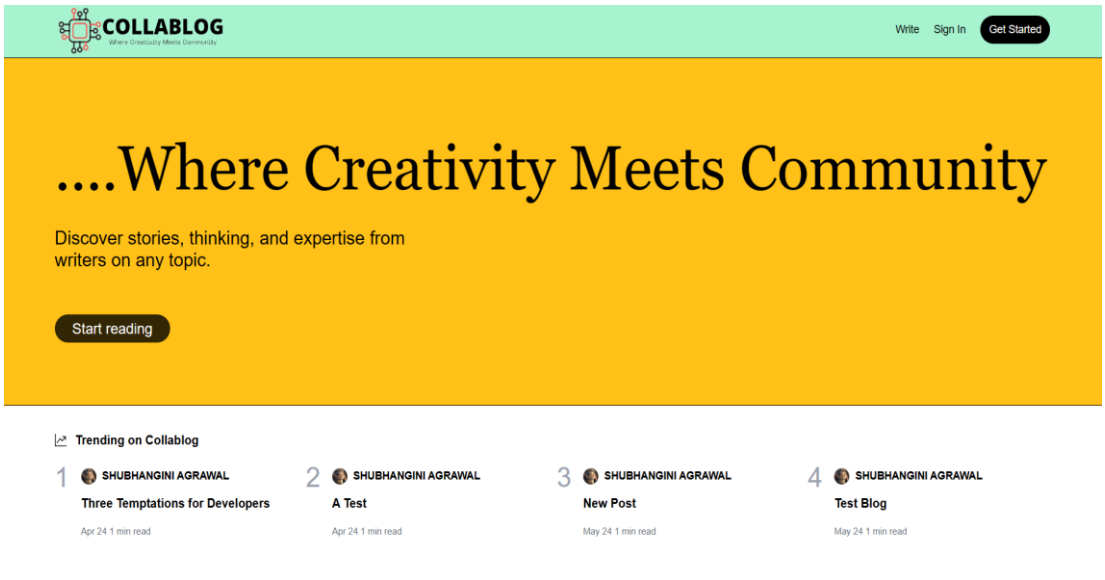


Fig 8.1 Home Page

8.2 READ SINGLE POST

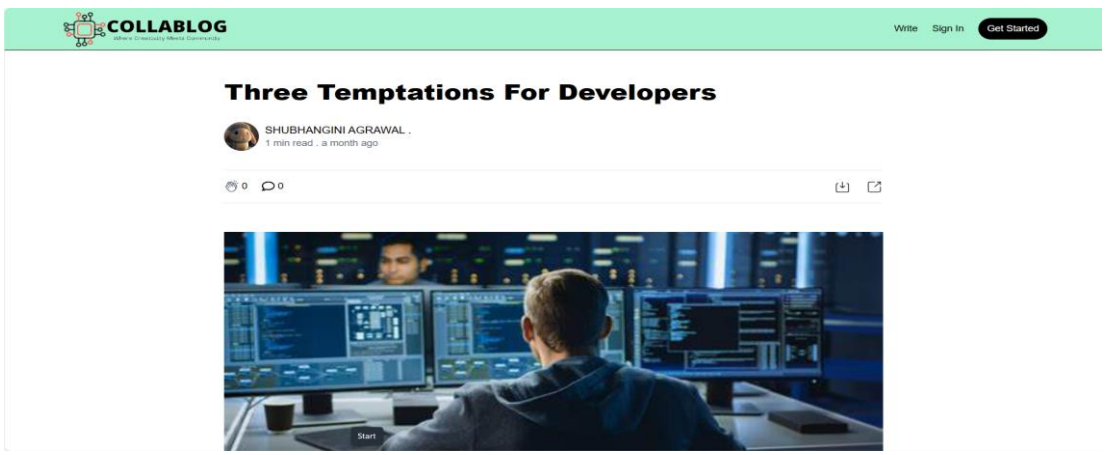


Fig 8.2 Read Single Post

8.3 READ POST BY CATEGORY

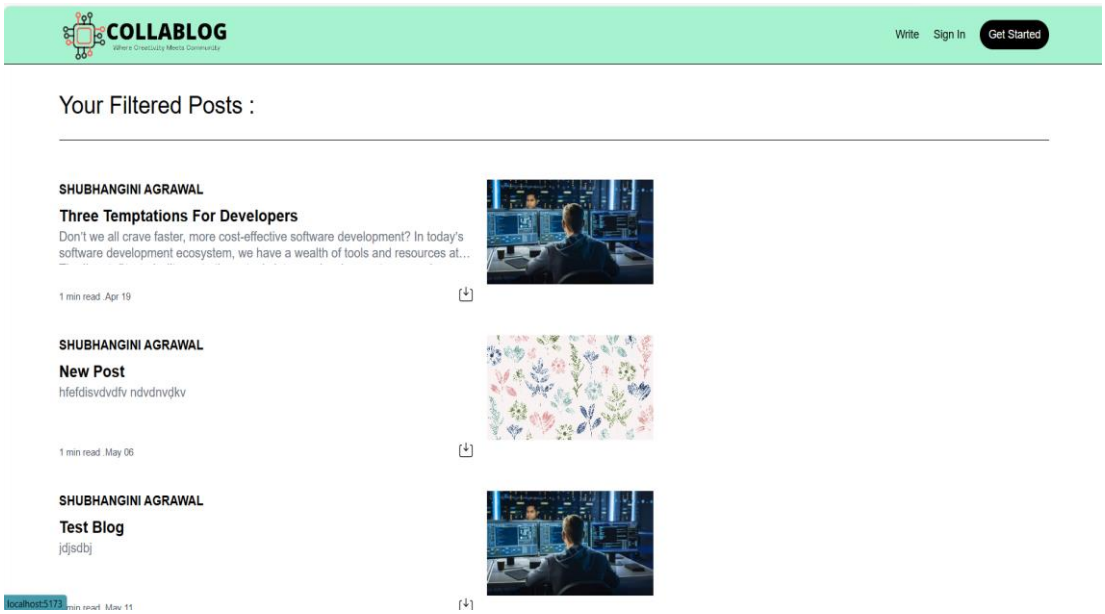


Fig 8.3 Read Post by Category

8.4 RECOMMENDED POST

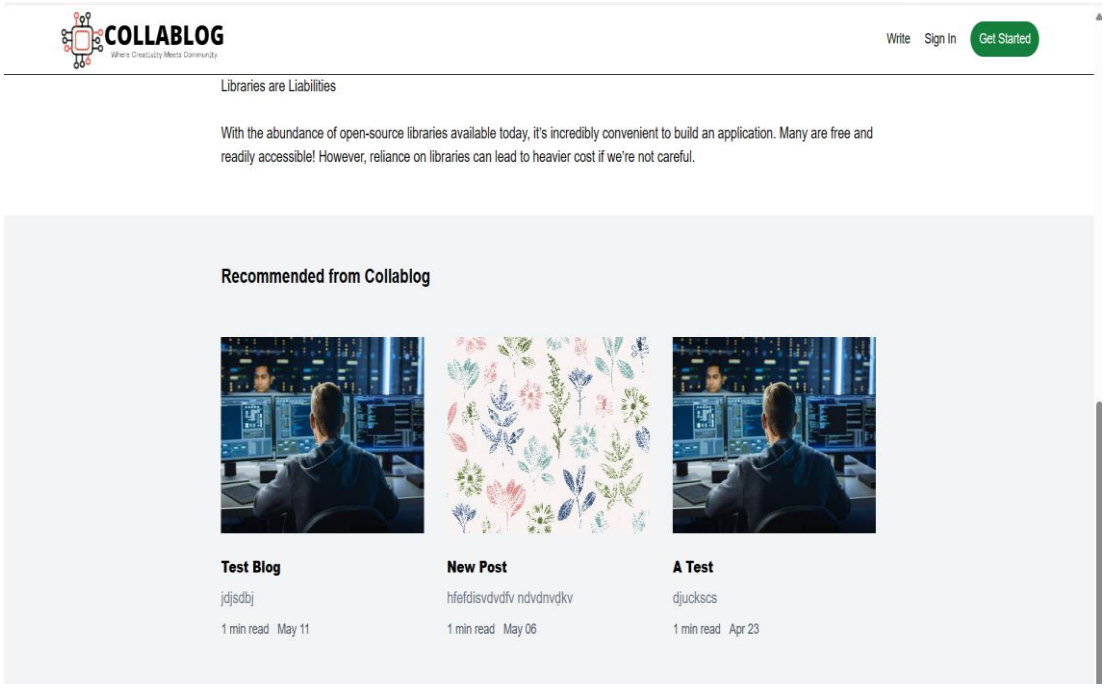


Fig 8.4 Recommended Post

8.5 REGISTER

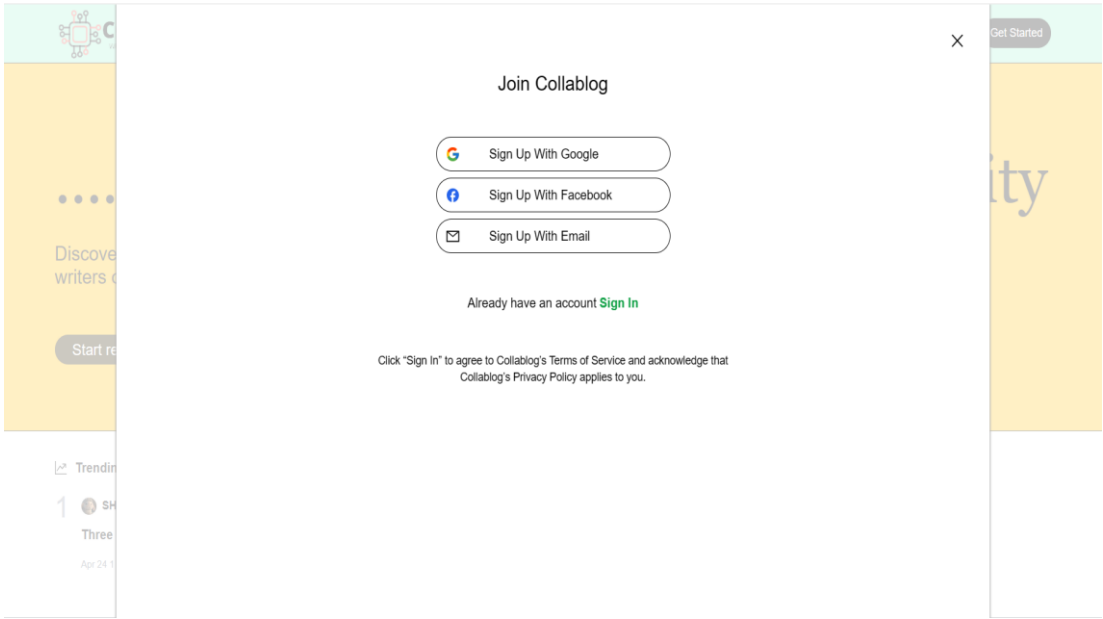


Fig 8.5 Register in Collablog

8.6 LOGIN

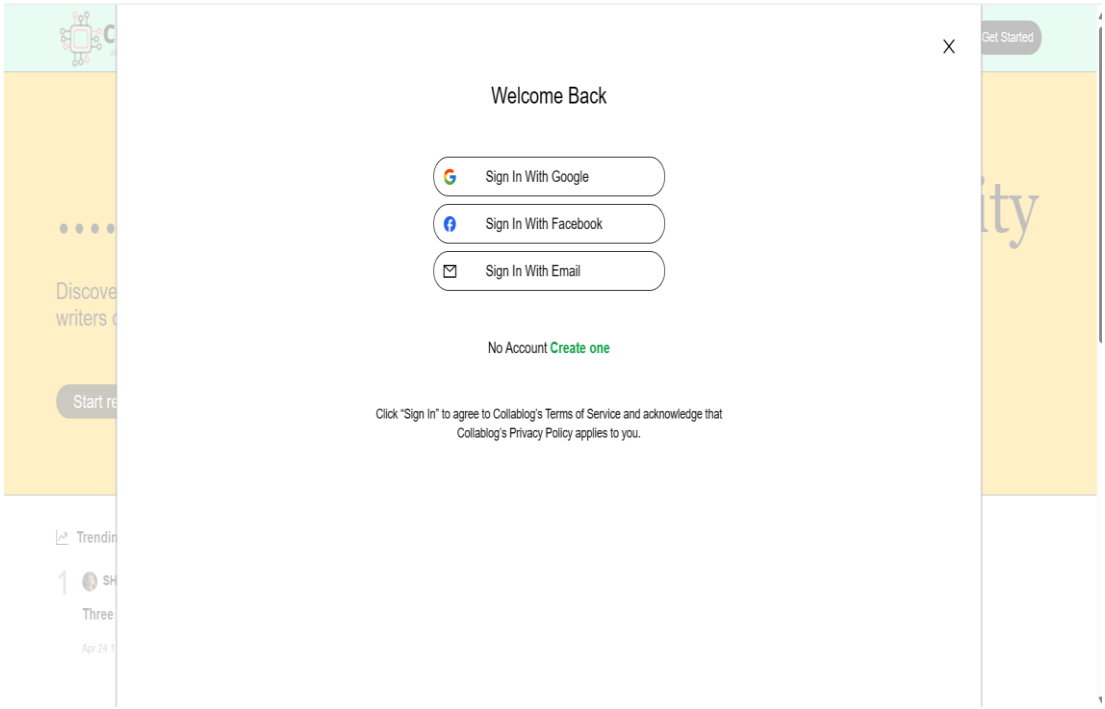


Fig 8.6 Login to Collablog

8.7 HOME PAGE AFTER LOGIN

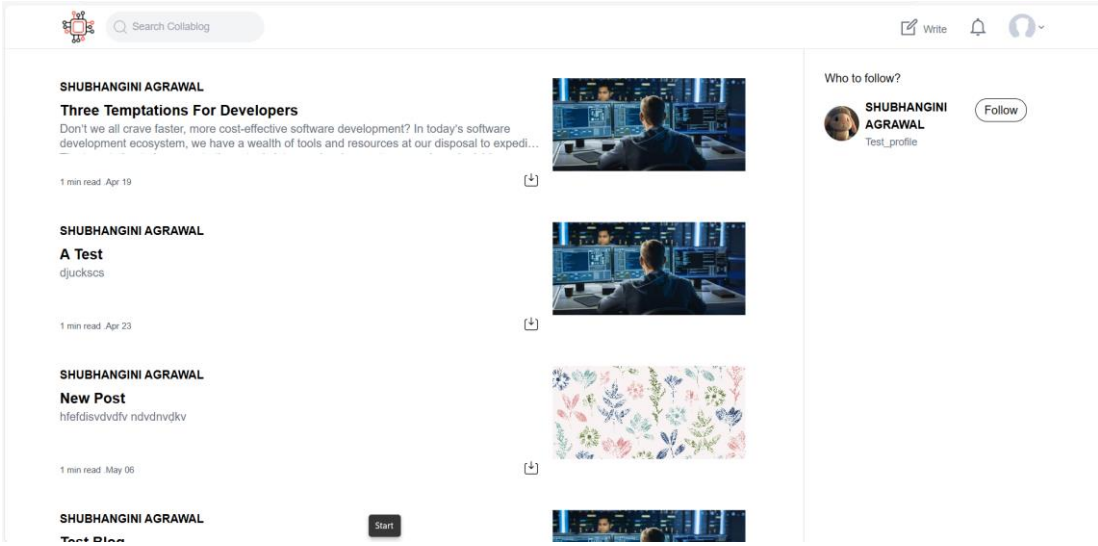


Fig 8.7 Home Page After Login

8.8 CREATE POST

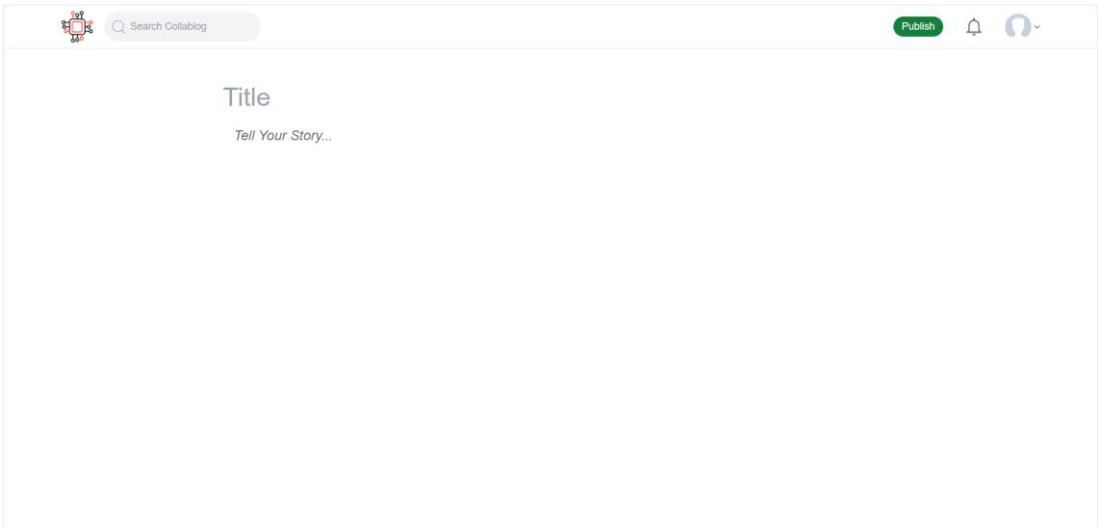


Fig 8.8 Create Post

8.9 STORY PREVIEW BEFORE PUBLISH

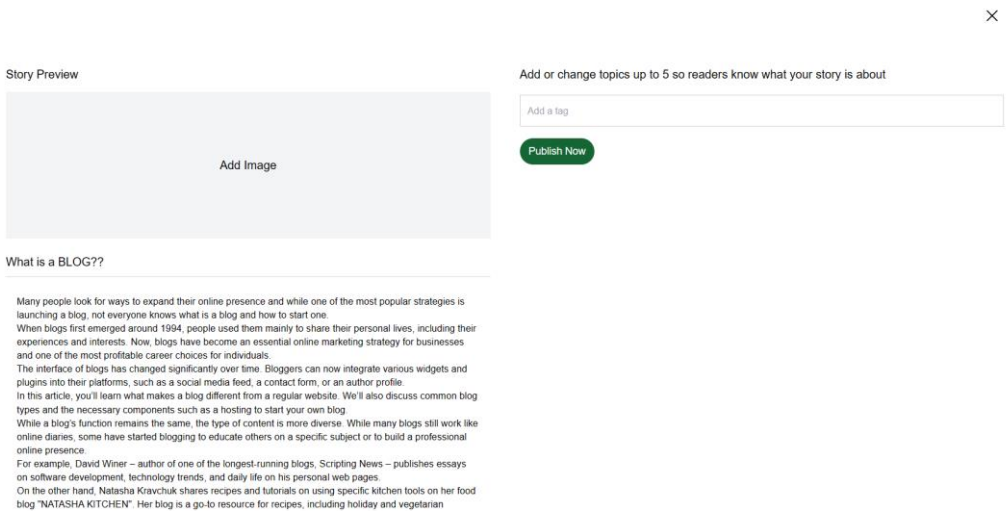


Fig 8.9 Story Preview Before Publishing

8.10 FOLLOW USER

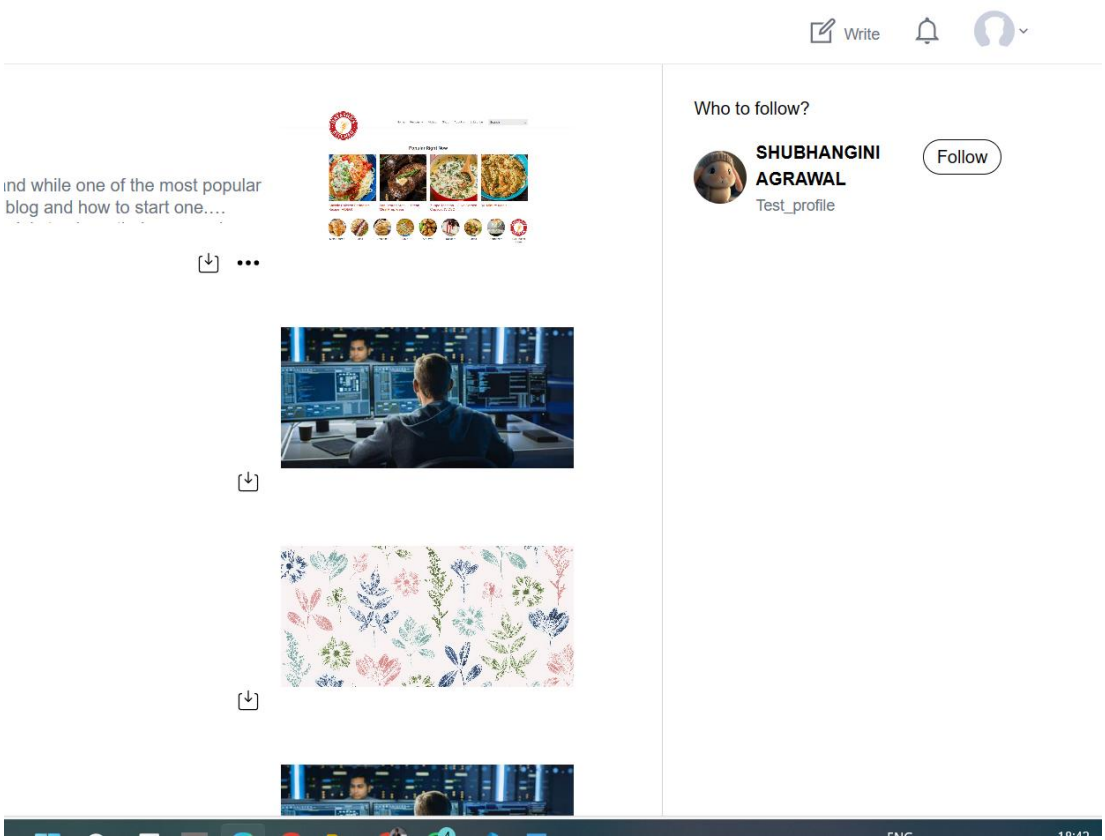


Fig 8.10 Follow User

8.11 ADD COMMENT

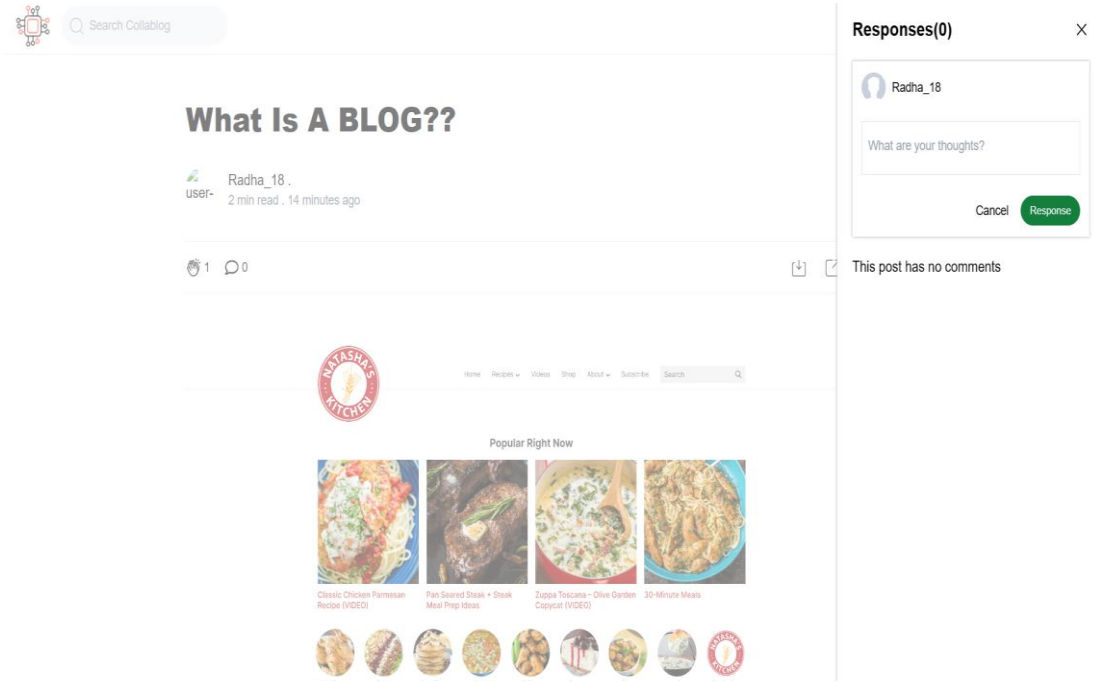


Fig 8.11 Add Comment

8.12 EDIT POST

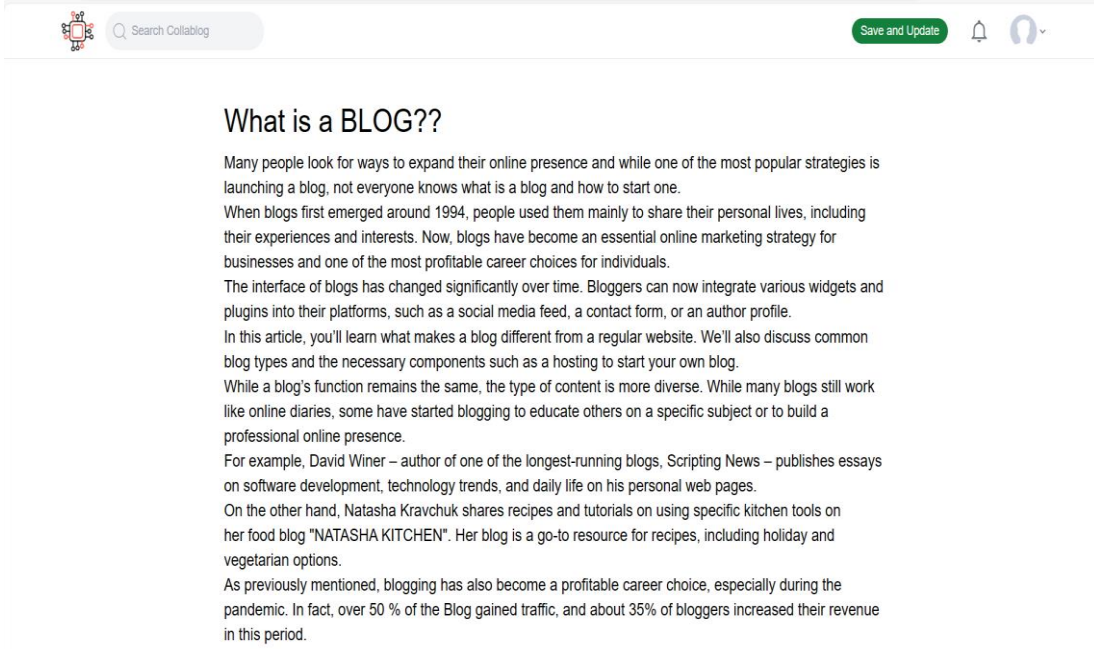


Fig 8.12 Edit Post

8.13 DELETE POST

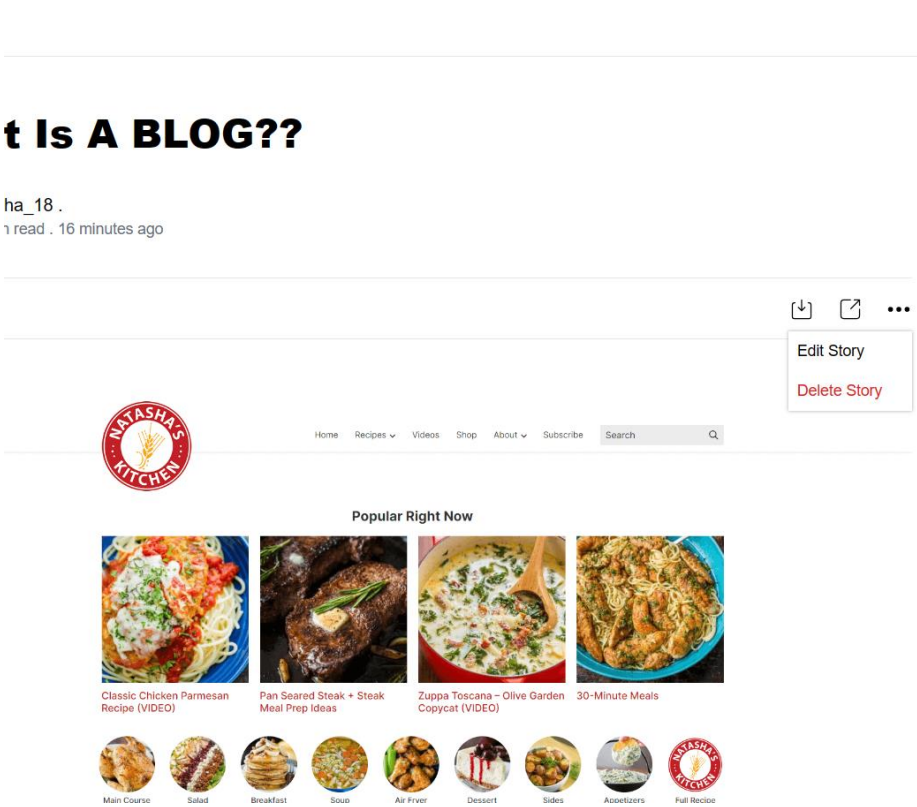


Fig 8.13 Delete Post

8.14 SHARE POST

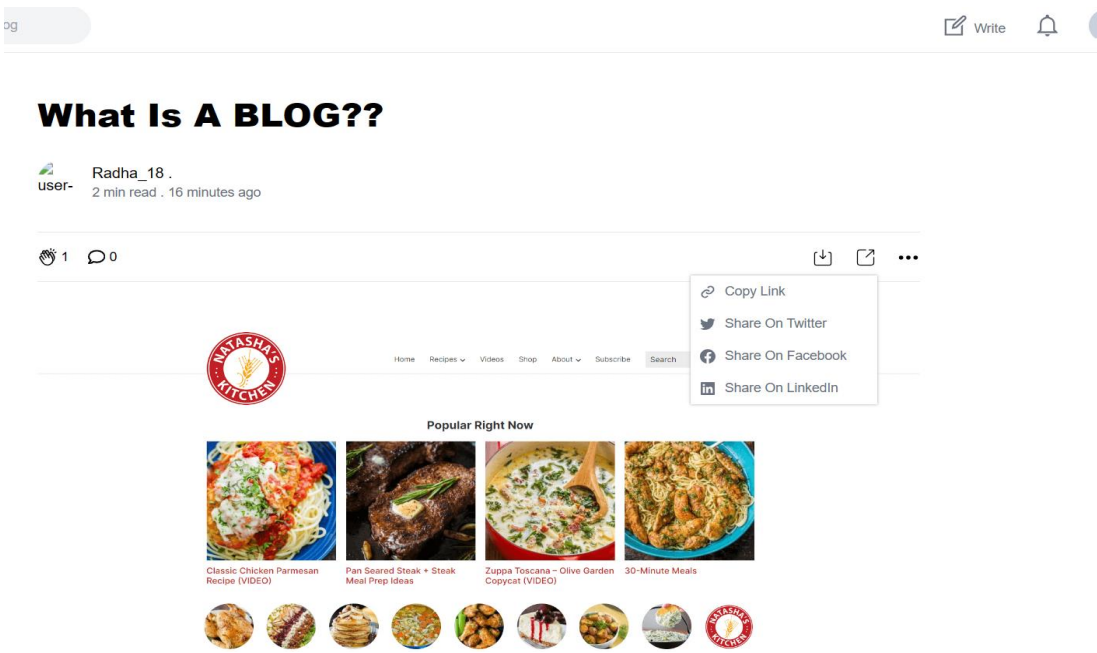


Fig 8.14 Share Post

8.15 USER'S PROFILE VIEW

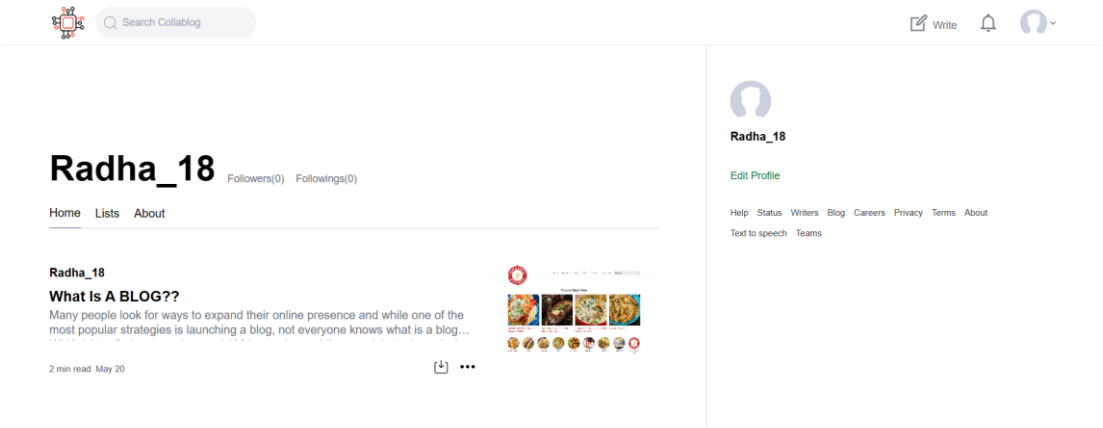


Fig 8.15 User's Profile View

8.16 EDIT USER'S PROFILE

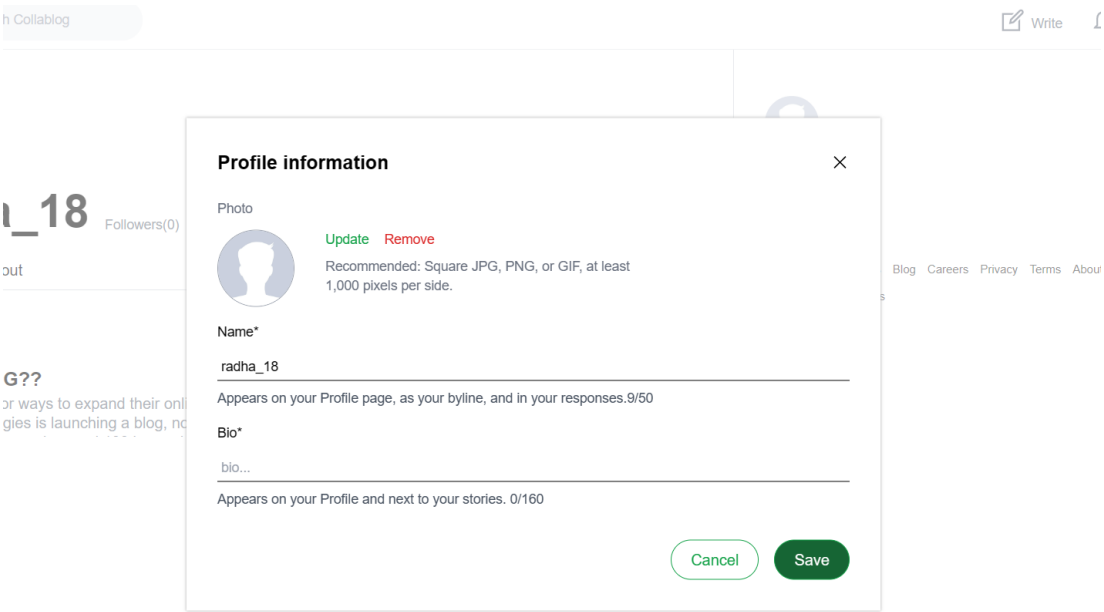


Fig 8.16 Edit User's Profile

8.17 SEE SAVED POST

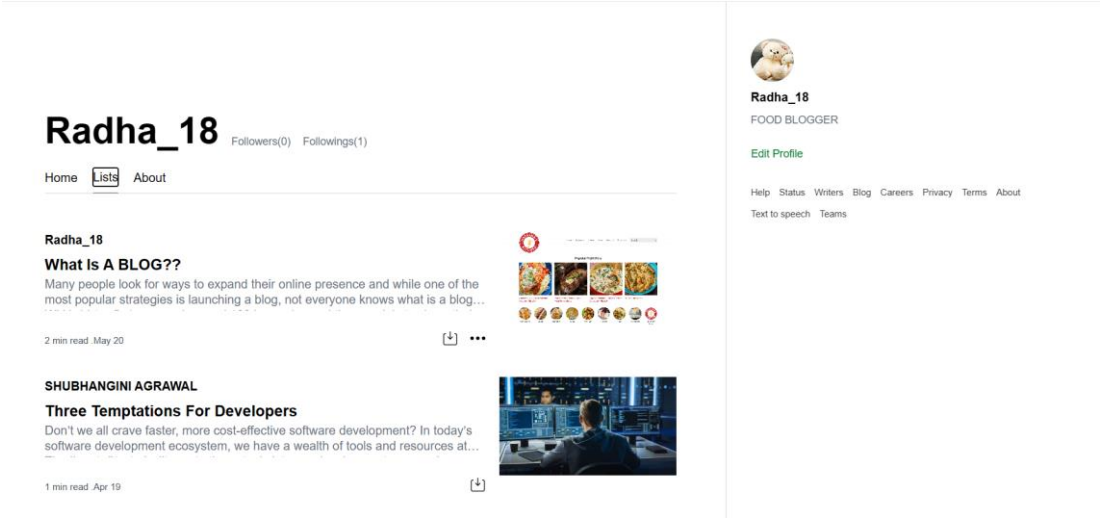


Fig 8.17 See Saved Post

8.18 SEARCH POST

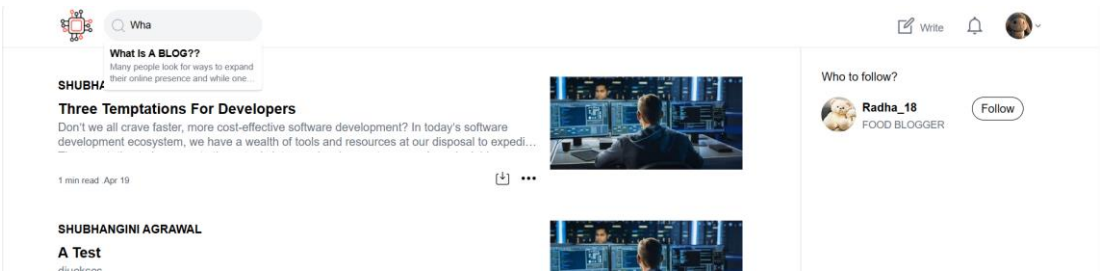


Fig 8.18 Search Post

8.19 TRENDING POST

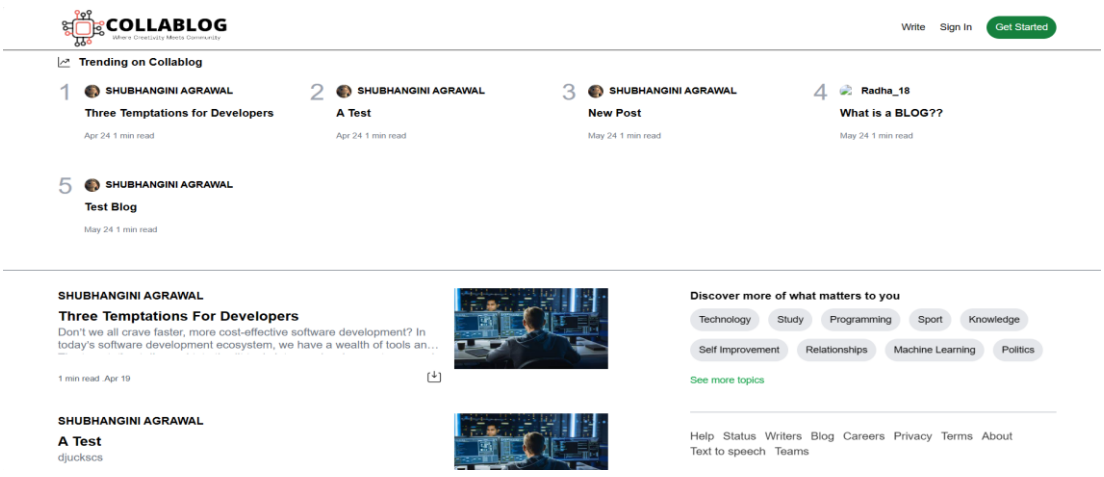


Fig 8.19 Trending Post

CHAPTER 9

CONCLUSION

8.1 LIMITATIONS OF THE PROJECT

Due to less knowledge in particular fields and limited time we were not able to fulfil all our expectations that we expected we could do while the project got started. We hope these limitations are considerable. Some of the project limitations are:

- **Scalability:** Firebase and Firestore are suitable for small to medium applications but might face challenges in scaling for a very large user base and high traffic volumes, potentially leading to performance issues and increased costs.
- **SEO:** Single Page Applications (SPAs) built with React may have SEO limitations compared to server-rendered applications, affecting the discoverability of content by search engines.
- **Advanced Features Complexity:** Implementing advanced features like live notifications, rich text editors, and collaborative editing can be complex and may require additional tools and libraries.
- **Real-Time Capabilities:** Ensuring real-time updates (e.g., live comments, post updates) can be challenging, particularly in maintaining consistency and handling conflicts.
- **Accessibility and User Experience:** Achieving a polished UI/UX and ensuring full accessibility for users with disabilities requires thorough testing and adherence to best practices, which can be challenging to implement comprehensively.

8.2 CONCLUSION

The development of the "Collablog: Where Creativity Meets Community" website using Vite, React, and Firebase has been a significant achievement, showcasing the effective integration of modern web development technologies to create a robust and scalable web application. Throughout the

project, we successfully replicated essential blog platform functionalities, such as user authentication, content creation, and content management. By utilizing Vite, we significantly enhanced our development workflow with its fast hot module replacement (HMR) and optimized build process, resulting in increased productivity. React's component-based architecture facilitated the creation of a dynamic and responsive user interface, ensuring a seamless user experience. Firebase played a crucial role in handling real-time data operations, including authentication, database management, and hosting, which allowed for efficient data synchronization and immediate updates. This integration provided users with an engaging and interactive experience. Furthermore, Firebase Authentication ensured secure user sign-up and sign-in processes, while Firestore provided a reliable solution for managing user-generated content. Overall, the combination of Vite, React, and Firebase not only met our project goals but also established a solid foundation for scalability and performance, making "Collablog" a successful and efficient web application.

8.3 LESSONS LEARNT

During the creation "Collablog" using Vite, React, and Firebase, we learned several key lessons:

- **Thorough Planning and Requirement Gathering:** Proper initial planning and clear requirement definitions are critical to ensure project scope is well-defined and all essential features are covered, preventing scope creep and guiding development efforts effectively.
- **Effective Tool Selection:** Choosing the right tools, such as Vite for rapid development and Firebase for comprehensive backend services, can significantly enhance productivity and streamline the development process. Evaluating and selecting tools that align with project needs is crucial.
- **Component-Based Architecture and Modularity:** Utilizing React's component-based architecture simplifies UI development and maintenance. Breaking the user interface into reusable, modular components makes the codebase more manageable and scalable.
- **Real-Time Data Handling and Security:** Implementing real-time data features with Firebase Firestore enhances user interactivity, while Firebase Authentication provides robust security measures. Ensuring efficient data synchronization and secure user authentication is vital for application reliability and user trust.
- **Performance Optimization and Scalability:** Using tools like Vite for performance optimization (e.g., fast hot module replacement) improves development efficiency.

8.4 FUTURE SCOPE

By focusing on these future enhancements, the Collablog can evolve into a more robust, scalable, and user-friendly platform, capable of meeting the needs of a growing user base and providing a superior blogging experience.

- Enhanced Scalability:
    - Serverless Architecture: Transition to a more scalable backend infrastructure, such as AWS Lambda or Google Cloud Functions, to handle increased traffic and user load more efficiently.
    - Database Optimization: Implement database sharding and indexing strategies to improve performance and scalability of data operations in Firestore.
- Improved SEO:
    - Server-Side Rendering (SSR): Incorporate Next.js or a similar framework to enable server-side rendering, improving SEO and initial load times.
    - Static Site Generation (SSG): Use static site generation for frequently accessed pages to enhance performance and search engine visibility.
- Advanced Features:
    - Rich Text Editor: Integrate a more advanced rich text editor that supports media embedding, markdown, and real-time collaborative editing.
    - Analytics Dashboard: Develop a comprehensive analytics dashboard for writers to track post performance, readership metrics, and engagement statistics.
- Real-Time Capabilities:
    - WebSockets: Implement WebSockets for real-time updates on comments, likes, and post edits to enhance user interactivity.
    - Push Notifications: Enable push notifications for important events, such as new comments on posts or follower activity, to keep users engaged.
- Cost Management and Optimization:
    - Cost Monitoring Tools: Integrate tools like Firebase's cost management and alert systems to monitor usage and prevent unexpected billing.
    - Alternative Databases: Explore alternative database solutions, such as PostgreSQL or MongoDB, to manage costs more effectively while meeting performance needs.

BIBLIOGRAPHY

1. **React Official Documentation**

   React. (n.d.). Retrieved from https://reactjs.org/docs/getting-started.html

2. **Tailwind CSS Official Documentation**

   Tailwind CSS. (n.d.). Retrieved from https://tailwindcss.com/docs

3. **Vite Official Documentation**

   Vite. (n.d.). Retrieved from https://vitejs.dev/guide/

4. **Firebase Official Documentation**

   Firebase. (n.d.). Retrieved from https://firebase.google.com/docs

5. **Playwright Official Documentation**

   Playwright. (n.d.). Retrieved from https://playwright.dev/docs/intro

6. **JavaScript Tutorial**

   W3Schools. (n.d.). JavaScript Tutorial. Retrieved from https://www.w3schools.com/js/

7. **Building a Blog with React and Firebase**

   Smashing Magazine. (2020). How To Build A Blog Using React And Firebase. Retrieved from https://www.smashingmagazine.com/2020/03/building-blog-react-firebase/