A logo on a black background

Description automatically generated

**COMP1682 – FINAL YEAR PROJECT REPORT**

**Project:** Student Contribution for Annual University Magazine

**Student Name:** Do Khac Quan

**Student ID:** 001358195

**Instructor**:

**Date**:

**Acknowledgement**

In acknowledgement section, I would like to take this opportunity to express my deep gratitude to people who have supported the “Cuisine Hub” project.

First and foremost, I would like to express thanks to my family who have been there with unwavering support, encouragement, and understanding throughout this journey. Their unwavering support has been an important source of inspiration for me and helped me through some difficult moments.

I also want to extend special thanks to my project instructors, who provided invaluable guidance, feedback, and technical support throughout the research and development process. Their unwavering dedication and committed teaching have enabled me to advance and develop professionally. Finally, I want to express my thanks to myself for my hard work, and dedication to this project. Each achievement is the product of continuous work and unwavering dedication.

**Table of Contents**

[1. Introduction 5](#_Toc165619938)

[2. Literature Review 5](#_Toc165619939)

[2.1. The evolution and development of “Social Networking” 5](#_Toc165619940)

[2.2. Overview and Compare Technologies. 11](#_Toc165619941)

[2.2.1. Overview of Front-End Stacks 11](#_Toc165619942)

[2.2.1.1. HTML / CSS / JavaScript 11](#_Toc165619943)

[2.2.1.2. JavaScript Libraries and Frameworks 12](#_Toc165619944)

[2.2.1.3. Comparison between Front-End Stacks 13](#_Toc165619945)

[2.2.2. Overview of Back-End Stacks 15](#_Toc165619946)

[2.2.2.1. Back-End Programming Language 15](#_Toc165619947)

[2.2.2.2. Database 16](#_Toc165619948)

[2.2.2.3. Comparation between Back-End Stacks 17](#_Toc165619949)

[2.3. Overview and Comparison between Methodologies 20](#_Toc165619950)

[2.3.1. Overview of Methodologies 20](#_Toc165619951)

[2.3.2. Comparation Methodologies 21](#_Toc165619952)

[2.4. Choosing Technologies and Methodology 24](#_Toc165619953)

[2.4.1. Technologies of “Cuisine Hub” Project: 24](#_Toc165619954)

[2.4.2. Methodology of “Cuisine Hub” Project: 26](#_Toc165619955)

[3. Requirement Analysis 27](#_Toc165619956)

[3.1. Similar Application 27](#_Toc165619957)

[3.1.1. Dev Community 27](#_Toc165619958)

[3.1.2. GitHub Discussions 31](#_Toc165619959)

[3.2. Requirements Analysis 34](#_Toc165619960)

[3.2.1. Functional Requirements 34](#_Toc165619961)

[3.2.2. Non-Functional Requirements 36](#_Toc165619962)

[3.2.3. Data requirements 37](#_Toc165619963)

[3.2.4. Development requirements 37](#_Toc165619964)

[3.2.4.1. User Stories 38](#_Toc165619965)

[3.2.4.2. Product Backlog 39](#_Toc165619966)

[3.2.4.3. Sprint Timebox 40](#_Toc165619967)

[4. Software Design 41](#_Toc165619968)

[4.1. Application Architecture 41](#_Toc165619969)

[4.2. Use Case Diagram 45](#_Toc165619970)

[4.3. Class Diagram 47](#_Toc165619971)

[4.4. Database Diagram 48](#_Toc165619972)

[4.5. Activities Diagram 50](#_Toc165619973)

[4.6. Wireframe 52](#_Toc165619974)

[5. Software Implementation 55](#_Toc165619975)

[5.1. Development Environment 55](#_Toc165619976)

[5.2. Code Structure 55](#_Toc165619977)

[5.3. Technical Problems and Solutions 58](#_Toc165619978)

[5.4. Development Notes 59](#_Toc165619979)

[6. Evaluation 60](#_Toc165619980)

[6.1. Final Web Screenshots 60](#_Toc165619981)

[6.2. Test Plan 60](#_Toc165619982)

[6.2.1. Scope 60](#_Toc165619983)

[6.2.2. Test Methodology 61](#_Toc165619984)

[6.2.4. Test Resource and Environments 62](#_Toc165619985)

[6.3. Test Case 62](#_Toc165619986)

[6.3. Test Logs 74](#_Toc165619987)

[7. Conclusion 76](#_Toc165619988)

[References: 77](#_Toc165619989)

**List of Figure**

[Figure 1: Most popular social networks worldwide as of July 2023 7](#_Toc165118288)

[Figure 2: The number of Threads sign-ups worldwide as of July 10,2023 8](#_Toc165118289)

[Figure 3: Dev Community (Source: https://dev.to/ ) 26](#_Toc165118290)

[Figure 4: GitHub Discusstion (Source: https://github.com/orgs/community/discussions) 30](#_Toc165118291)

**List of Table**

[Table 1: Comparation table of JavaScript libraries and framework 13](#_Toc165118300)

[Table 2: Comparison table of Back-End programming language 16](#_Toc165118301)

[Table 3: Comparison table of database 18](#_Toc165118302)

[Table 4: Advantages and Disadvantages of Waterfall 20](#_Toc165118303)

[Table 5: Advantages of Disadvantage of Agile 21](#_Toc165118304)

[Table 6: Advantages and Disadvantage of DevOps 22](#_Toc165118305)

[Table 7: Summary Table of Chosen Technologies 25](#_Toc165118306)

# 1. Introduction

“Cuisine Hub” – a social networking application designed to provide an experience that goes beyond simply discovering interesting recipes or dining ideas, but also fosters meaningful connections between individuals.

My application is not just a platform for sharing stunning food photos and insightful reviews; it is a dynamic community where individuals can meet, interact, and exchange ideas and cooking tips. With smart search features and advanced filters, users will easily find ideas or favorite recipes from traditional recipes to exotic specialties from around the globe. Users can also make new friends and share experiences with others.

This report embarks on the evaluation and development of the social network application “Cuisine Hub”. Detailed reports will start from a literature review that delves into the evolutionary trajectory of social networking, followed by comparative analyses of possible technologies and methodologies. Following is requirements analysis delineates the scope, functional, and non-functional requirements essential for project alignment. Project management strategies, adopting an agile methodology, are explained through user stories, product backlog, and sprint lists. Design considerations encompass system architecture, use case diagram, activity diagrams, and wireframe. Implementation details include tools selection, development environment setup, and technical problems and solutions. Evaluation encompasses results analysis with final application screenshots, testing methodologies include test plan, test case, test log, and test result evaluation. A conclusion will be written down to ensure a comprehensive understanding of the social networking applications’ development lifecycle.

# 2. Literature Review

## 2.1. The evolution and development of “Software Development”

In the era of digital development, the phrase “social media” is widely used because it has appeared in almost all areas from life to career. “Social Media” is a term that describes a set of online technologies things such as knowledge, documents, information. The purpose of these platforms is to create opportunities for individuals and organizations to communicate and bond over time. According to (Partrick, 2022), social media is not only a collection of technologies but also a culture and social phenomenon that has changed the way people communicate, express themselves, search the information and travel through space. This can be clearly seen through the development of social networking platforms such as Facebook, Twitter, TikTok, they have developed very strongly enough to change the nature of business and marketing of business. Currently, this brings up the argument that the appearance of social media has rapidly changed human communication in the digital age.

In the information mentioned in (Partrick, 2022)’s article, the journey to develop social networks began with forms of online communication in the 1960s and 1970s-the years in the early day of computers. Researchers have discovered a way to enable digital communication using a Compatible Time-Sharing System (CTSS) that allows users to leave messages on shared computers. While still rudimentary, the experiment opened a breakthrough for digital communication innovations to come. In 1970, 1980, the birth of Bulletin Board System (BBS) – a platform that enabled users to dial into a computer system via modem, where users could participate in conversations based on documents, files, and online communities. One of the earliest BBSs, CBBS (Computerized Bulletin Board System), was created in 1978 by Ward Christensen and Randy Suess, allowing users to post messages, upload, and download files, and participate in discussions on a variety of different topics. BBSs was a solid foundation that paved the way for the appearance of interactive and community-focused features of other social networking platforms.

Moving forward to the 1980s and 1990s the appearance of two online services, CompuServe, and Prodigy, greatly supported the development of digital media. In 1979, the online service CompuServe was founded, providing not only access to news and information. In CompuServe’s forums, users can participate in conversations on a variety of topics creating strong engagement with online community. CB Simulator was one of the first services that allowed users to chat in real time and appeared in 1980. In 1988, Prodigy – a platform providing many services from news, forums, shopping online, email was founded by IBM and Sears. With Prodigy’s community, users could chat and share ideas, and this. Feature prefigured the distinction between interactive capabilities in later social media platform.

In 1990, the appearance of the American Online (AOL) – a online service with Instant messaging (IM) function became one of the online services with the largest number of users. In 1997, AOL innovated and announced AIM functionality that enabled users to communicate with friends and contacts in real time and allowed users to have group chats – function that would be the first step pioneered chat rooms and group discussions that would later become an indispensable part of other social networking platforms. It is undeniable that AIM is a harbinger of the emergence of new communication in the digital age.

Before and after the 2000s, there were pioneering social networking platforms. One of the names that can be mentioned is Classmate.com created by Randy Conrads in 1995. This was one of the first websites that focused on connecting people with classmates. It allows users to create profiles, list their academic history, and search for others who attended the same school. Move on to another platform is SixDegrees.com created founded by Andrew Weinreich in 1997. Using this platform, users can set up profiles, list friend requests and messages sent. The inspiration for creating this platform comes from the phrase “Six Degrees” which means that people around the world are connected through a network of acquaintances. The rise of Friendster launched in 2002 by Jonathan Abrams also created a great influence on the social networking platform when in addition to functions such as: creating profiles, friend requests and the ability to create connections with other users at its peak and has since grown in popularity.

While there were still scalability issues and technical difficulties, it cannot be denied that Friendster at that time played an important role in illustrating the possibility of human connection through digital technologies. If Friendster brought great influence on major social networking platforms in 2000, Myspace is considered the first largest social network launched in 2003 by two people are Tom Anderson and Chris DeWolfe. With functions like creating and personalizing profiles, it was perfect for young musicians at the time. Myspace’s growth was based on the growing music industry and creating connections to society. It can be said that Myspace has created a huge influence from social networks to the entertainment industry thanks to the formation of social networks and music marketing. “Facebook”- a platform created by Mark Zuckerberg and a few friends in 2004, was initially used for Harvard students, but after a while, it spread to other schools and eventually is also released to the public. Which features such as using university email addresses to verify accounts, “Wall” to post messages and new updates, “New Feed” to highlight content and “Like” button – shows interaction between posts. What’s specials connections with friends, family and acquaintances and form there, creates rapid global expansion. By 2008, Facebook had surpassed Myspace in the number of active users, creating a certain foothold in other social networking platforms. Right after Facebook, there were many other platforms that developed and created other influences such as: The emergence of Twitter created the trend of blogging or YouTube created a social network specializing in video sharing.

It is now 2023, this is the year when technology has developed to a new level, many technology products have been created to help people’s social lives, and socials networks in general have had a lot of changes changing dynamics as the tendency to use it in users ‘daily lives. According to (Statista, 2023), the chart below displays the data of most popular social networks worldwide as of July 2023, ranked by number of monthly active users:

A screenshot of a graph

Description automatically generated

Figure 1: Most popular social networks worldwide as of July 2023

(Source: https://www.statista.com/statistics/272014/global-social-networks-ranked-by-number-of-users/)

From that data chart above, Facebook is the first social network to surpass one billion registered accounts and currently has 2.9 billion monthly active users. Meta platforms owns four of the largest media platforms like Facebook Messenger, WhatsApp, Instagram, and Facebook, each will over a billion monthly active users. Not only that, from the platforms listed above, the origins are all form the United States and China.

In addition to leading platforms with more than 100 million users originating from the US, other Chinese social networking services such as: WeChat, QQ, or video sharing application in their respective regions due to the impact of context and content.

According to (Bravo, M, 2023), the authors outline the key milestones, emerging trends, and transformative changes that will define the evolution of social media in 2023. More specifically, the author pointed out the rise of the new app “Threads” – an application within the Meta platform, a game-changing development, poised to disrupt Twitter’s dominance. What makes “Threads” special lies in its unique ability to support real-time conversations in a more realistic way, thereby enhancing the user experience. Clearly, “Threads” is aiming to serve the needs for meaningful interaction and rich digital experiences of modern social media users. According to data from Statista, the chart below is the number of Threads sign-ups worldwide as of July 10,2023:

A graph with blue bars

Description automatically generated

Figure 2: The number of Threads sign-ups worldwide as of July 10,2023

(Source: https://www.statista.com/statistics/1398663/global-threads-users/)

Based on Statista, “Threads” is present in more than 100 countries, excluding the European Union. Within just two hours of its release, two million people had signed up for Thread, and within 24 hours, 30 million people were using the platform. As of July 10, the number of subscribers has reached 100 million. Users can post up to 500 characters on posts and offer many features comparable to Twitter and are verified through the user's Instagram account.

In addition to the appearance of the “Threads” application, there is a change in privacy and data protection of users when using social networking platforms. Leading platforms are following the trend of “empowering and regulating social media users”. This problem occurs when the surrounding context changes, users feel that data is not secure and always make demands for better control to platform founders. Platforms have come up with a solution by proactively adopting stronger data protections such as empowering users to manage their privacy preferences and providing clearer insights into their data. usage and data collection. With this measure, social platforms are creating a digital environment that is trustworthy and puts users first. (Bravo, M, 2023).

Furthermore, content censorship to create an appropriate balance on social platforms is becoming increasingly stricter to limit the posting of false information and harmful content. The social network is planning to invest in AI-driven algorithms and create dedicated human moderation teams that are efficient and consistent. For harmful information, platforms not only delete but also actively promote authentic and trustworthy sources of information, cooperate with fact-checking organizations, and implement transparency measures. to promote users' sense of trust in social networks (Bravo, M, 2023).

One of the things that dominates social media platforms is video and live streaming. In today’s constantly evolving context, social media platforms realize the power and influence of videos, so they focus on creating video content that attracts a large audience. With some leading platforms in 2023, video, stories and live features are indispensable because users can experience real-time, express creativity, and engage viewers in a unique way. Soon, users will always expect video content to continue to dominate as the king of social media, allowing connection, inspiration, and entertainment from a new perspective. (Bravo, M, 2023).

The fact that businesses use social networking platforms to increase marketing capabilities to users is no longer strange until now. From a business perspective there is always a formula for successful social marketing which is the combination of social commerce and influencer marketing. For social networking platforms, integrating social commerce features, users can easily learn, explore, shop, and transact, creating an engaging experience. It is this convergence of content and commerce that has paved the way for the growth of influencers to expand the reach and engagement of brands. For this reason, influencers play an important role in deeply influencing brand awareness and user behavior (Bravo, M, 2023). One of the innovations that will increase user experience to a new level soon is the integration of augmented reality (AR) and virtual reality (VR). With the power of AR, users will have more interaction thanks to filters, lenses and effects that help posted content come alive in creative directions from users. With VR, users can experience virtual journeys, virtual meetings, and events to explore new environments. Overall assessment of these new technologies, it can be said that it will not only narrow the gap between the physical world and the digital world but also exploit boundless creativity, surpassing imagination in social media field. (Bravo, M, 2023).

In conclusion, the innovation and growth of the social media sector in 2023 has truly reshaped the way users connect and interact online. With the appearance of "Threads", it is undeniable that the application has brought a complete change in social media when content is built on Twitter. Additionally, concerns about speech rights and data security have become a priority, pushing media companies to increase protections and give users more control over their information. their cause. In addition, content moderation is also very necessary and important to create a harmonious balance between freedom of speech and preventing harmful information. Not only that, but social commerce has also created remarkable growth, helping users shop directly on social media platforms. The integration of augmented reality (AR) and virtual reality (VR) technology will create a new upgrade in user experience, providing more creative and vivid content. With a forward-looking perspective, it can be said that social media will create a push for real connection and empower users to create positive impacts, meaningful interactions. This means helping social networking platforms bring more value and meet the diverse needs of users. Furthermore, with the continued advancement of technology, media will increasingly open a future that is more connected, inclusive, and meaningful.

## 2.2. Overview and Compare Technologies.

### 2.2.1. Overview of Front-End Stacks

The front-end tech stack is what users see and interact with on their screens. It is all about making websites and apps easy to use, with great designs and smooth navigation. Essentially it is responsible for designing how things look, where they go and how browsing mechanisms of websites or web applications.

#### 2.2.1.1. HTML / CSS / JavaScript

HTML, CSS, JavaScript are three principal elements of a front-end stack. Specifically:

* **HTML (Hyper Text Markup Language)**: HTML is used to create and format content for Web pages. HTML defines the structure of a web page using tags as markup elements. HTML is often combined with CSS to adjust the appearance and layout of a Web page, as well as JavaScript to add dynamic interactive functions.
* **CSS (Cascading Style Sheet):** CSS helps to simplify the process of making pages presentable. Developers can apply a style to web page and use CSS independently without relying on HTML that makes up each web page. CSS also helps to redesigned websites very quicky. Without the use of extensive coding, CSS supports different-looking web pages and the only thing that changes are the content.
* **JavaScript:** JavaScript is an interpreted multi-paradigm, and cross-platform programming language used in Web Development. JavaScript is often used to change Web page content, control HTML and CSS, handle user events, and communicate with web services. With common libraries and frameworks like React Angular and Vue, JavaScript has become more popular when it comes to web application development.

#### 2.2.1.2. JavaScript Libraries and Frameworks

JavaScript libraries and frameworks are essential tools for web development, providing pre-written code and structure to streamline the process of building interactive and dynamic web applications. They offer a wide range of functionality, from simplifying DOM manipulation to managing complex data flow and rendering components efficiently.

Below are some common libraries and framework in JavaScript based on (GeekforGeek, 2024):

* **React.js:** React is a JavaScript library made by Facebook and a lots of individuals developer in community for building user interfaces. React is also known for its high-performance thanks to its use of a virtual DOM, which reduces the time required to re-render the view when the state changes. In addition, React offers support for server-side rendering which the support from Next.js framework, which can improve performance of web application by allowing data to be fetched before the view is rendered on the client side.
* **Vue.js:** Vue is a JavaScript framework for building user interface and single-page applications. Vue was created by Evan You and now a core developer keeps it up to date. Vue.js offers a gradually adaptable structure centered on declarative rendering and component assembly. It comprises essential libraries engineered for simplicity yet flexibility. The core library emphasizes the view layer exclusively, facilitating seamless integration with other libraries or ongoing projects.
* **Ember.js**: Ember is a component-based JavaScript framework that was created in 2011 by Yehuda Katz. Ember aims to enhance web development productivity and enjoyment by furnishing a complete toolset encompassing the entire development workflow. This includes a router, view layer, and data management library, ensuring a comprehensive approach to development. Ember is a service or component that can be invoked at any moment and is easily accessible across in Ember project. Furthermore, the browser’s URL and Ember’s routing mechanism are tightly integrated. Ember maps URLs to a route handler via its router. To load data, show templates, and indicate the state of the application, the router can match the current URL to other routes.

#### 2.2.1.3. Comparison between Front-End Stacks

Standing in front of many different Front-End technologies including programming languages, libraries, and frameworks, the author came up with an idea that is to consider what currently have and base it to make possible options. "Cuisine Hub" is an independently developed project, so prioritizing the use of familiar technologies is the first thing to do. The author has time with JavaScript and uses it for some projects. Based on that, for "Cuisine Hub" this time, there is no reason why the author does not use JavaScript as the main programming language. Besides, with the JavaScript extensive ecosystem of libraries and frameworks such as React.js, Vue.js or Ember.js, selecting a library or framework offers some advantages:

Firstly, it enhances efficiency by providing pre-built functionalities and components, reducing development time, and enabling focus on unique features. Secondly, it promotes consistency through standardized coding patterns, aiding in codebase comprehension and maintenance. Additionally, frameworks are designed with scalability in mind, facilitating project growth without compromising performance. Moreover, they benefit from large and active developer communities, ensuring ongoing support, knowledge sharing, and updates. Furthermore, they often include security features to mitigate vulnerabilities and handle cross-browser compatibility issues, saving time in testing and debugging. Lastly, comprehensive documentation and resources help developers get started quickly and learn best practices.

From the information provided above, in section 2.2.1.3, the author gives to reader a comparation table about JavaScript’s libraries and framework for considering and choosing the suitable one for “Cuisine Hub” project.

Three objects to be compared include React.js, Vue.js and Ember.js with some criteria including learning curve, flexibility and customization, community support, performance, compatibility, and integration and scalability. Detail of comparison in below:

Table 1: Comparation table of JavaScript libraries and framework

|  |  |  |  |
| --- | --- | --- | --- |
| **Criteria** | **React.js** | **Vue.js** | **Ember.js** |
| **Learning Curve** | Moderate. JSX syntax may have a learning curve | Easy. Simple syntax and clear documentation | Moderate. Convention over configuration approach |
| **Flexibility and Customization** | Highly flexible, allows for component-based architecture and easy integration with other libraries | Flexible, supports component-based architectures, and offer a variety of plugins for customization | Moderate flexibility with a strong emphasis on conventions. Offers built-in features for routing, data management, and more. |
| **Community Support** | Large and active community with extensive resources, tutorials, and third-party libraries | Active community with growing support and a wide range of resources. | Stable community with dedicated contributors and extensive documentation. |
| **Performance** | High performance due to virtual DOM and efficient rendering mechanisms | High performance with a lightweight core and optimized rendering | High performance with a focus on fast initial load times and rendering optimizations. |
| **Compatibility and Integration** | Compatible with most modern browsers and supports seamless integration with other libraries and tools | Compatible with modern browsers and integrates well with other libraries and tools | Compatible with modern browsers and offers seamless integration with various tools and libraries. |
| **Scalability** | Highly scalable with support for building complex applications and easy integration with state management solutions like Redux | Scalable for both small and large applications with built-in features for state management and routing | Highly scalable with built-in features for handling data management, routing, and progressive enhancement. |

### 2.2.2. Overview of Back-End Stacks

The back-end tech stack constitutes the server-side of software development. Comprising the inner mechanisms of a website or app concealed from users. The back-end stack includes these elements: programming language and database.

#### 2.2.2.1. Back-End Programming Language

When it comes to selecting a back-end programming language, there are multitude of programming languages available for web development. To make an informed choice, the author based on (Rick, J.,2022) and identified the top three most popular programming languages for backend development include JavaScript, Python, and Java. Prior to conducting the comparison, below is a brief overview of these programming languages:

* **JavaScript**: JavaScript is known as client-side scripting languages used for developing interactivity and user experience within web browsers. But the appearance of Node.js – a runtime environment that allows the execution of JavaScript code outside the browser makes itself become a language that can handle both client-side and server-side.
* **Python**: Normally, Python is widely used for developing the server side for building robust and scalable back-end applications. In this context, Python is often use with web frameworks such as Django, Flash, and Pyramid which are frameworks that provide a structure for building web applications and handling server-side logic, database interactions, and other back-end functionalities.
* **Java:** Java is known as a widely programming language that was used for back-end development for many years because of its portability, strong typing, and extensive ecosystem. These things make Java become a suitable programming language for building large-scale, enterprise-level applications.

#### 2.2.2.2. Database

A database is a structured collection of data that is organized and stored in a way that allows for efficient retrieval, updating, and management. It serves as a central repository for storing and managing information, making it easier to access and manipulate data as needed. In a database, data is typically organized into tables, which consist of rows and columns. Each row represents a record, while each column represents a specific attribute or field of the data. By organizing data in this structure format, databases enable users to perform complex queries, analyze data, and generate reports.

There are different types of databases, including relational databases, which use tables to store data and establish relationships between them. NoSQL databases, which are designed for handling unstructured or semi-structured data and offer more flexibility. In the range of project, the author gives to readers two database management systems belong two different types of databases include PostgreSQL in relational database and MongoDB in NoSQL database.

* **PostgreSQL**: It is a reliable and powerful relational database management system (RDBMS). Postgres is known for organizing data neatly into tables and following strict rules to keep everything in order. It is commonly chosen for applications requiring structured data management, where reliability, robustness, and adherence to standards are critical.
* **MongoDB**: MongoDB is a flexible and scalable NoSQL database. It stores data in a JSON-like format, making it easy to handle unstructured or changing data. It scales horizontally to manage large volumes of data and high traffic. MongoDB is commonly used in applications requiring flexible data models, such as content management systems, real-time analytics, caching layers, and applications with rapidly changing requirements.

### 2.2.2.3. Comparation between Back-End Stacks

In the realm of web development, and specifically in the domain of server-side (back-end) development, numerous programming languages have been devised to fulfill this need. When selecting the suitable programming language for the “Cuisine Hub” project, the author must weigh various criteria such as programming paradigm, community and ecosystem, security, learning curve, performance, and scalability. To break down these criteria comprehensively, the author has created a comparison table encompassing three prominent programming languages: JavaScript, Python and Java, as outlined in section 2.2.2.1. Details of the comparison table in below:

Table 2: Comparison table of Back-End programming language

|  |  |  |  |
| --- | --- | --- | --- |
| **Criteria** | **JavaScript** | **Python** | **Java** |
| **Programming Paradigm** | Multi-paradigm: functional, object-oriented, imperative | Multi-paradigm: object-oriented, imperative, procedural | Object-oriented, imperative, procedural |
| **Community and ecosystem** | Vast community and ecosystem with numerous frameworks and libraries supporting both front-end and back-end | Large and active community with a rich ecosystem of libraries and tools such as Django and Flask | Strong community support with extensive ecosystem of libraries, frameworks, and tools such as Spring, Hibernates. |
| **Security** | Requires diligent attention to security practices, particularly | Emphasize readability and ease of use, which can aid in code review | Strong emphasis on security with features like static typing and memory management enhancing security |
| **Learning Curve** | Moderate learning cure, especially for beginners due to asynchronous programming and event-driven nature | Generally easy to learn and beginner friendly. Syntax is clear and concise | Sightly steeper learning curve due to strict syntax and verbosity. Java concepts like threading and memory management can be complex |
| **Performance** | Fast execution in browser environments due to just-in-time compilation. Can be slower in server-side if compared with Java | Generally slower execution compared to Java. Performance can be enhanced through optimization and third-party | Known for its performance due to its static typing and complier optimizations. But startup time can be slower due to JVM startup |
| **Scalability** | Scalable for small to medium-sized application, but may face challenges with larger, complex system | Scalable for various sizes of applications but may face challenges with performance and scalability issues in large-scale systems. | Scalable for enterprise-level applications with support for distributed computing and robust concurrency features. |

When discussing Back-End development, it is essential to address the choice of database. Section 2.2.2.2 provides an overview of the database. To select the suitable database for the “Cuisine Hub” project, the author will create a comparison table between PostgreSQL, which represents the relational database management system, and MongoDB, which represents the NoSQL database management system. The comparison will be based on criteria including data model, scalability, performance, security, availability and fault tolerance, ease of use and administration, and community support.

Table 3: Comparison table of database

|  |  |  |
| --- | --- | --- |
| **Criteria** | **PostgreSQL** | **MongoDB** |
| **Data Model** | Relational: Use tables with rows and columns, supporting SQL queries | NoSQL: Documents-oriented, stores data |
| **Performance** | Excellent for complex queries and transactions, especially in ACD compliance scenarios | Good performance for read-heavy workloads and large-scale distributed deployment. |
| **Security** | Strong security features including SSL encryption, authentication methods role-based access control, and row-level security | Robust security features including authentication mechanisms, access control field-level encryption, and auditing capabilities. |
| **Fault Tolerance** | Supports high availability through streaming replication synchronous and asynchronous replication, and automatic failover with tools like Patroni or repeg. | Buil-in replication with automatic failover sharding for horizontal scaling, and multi-documents ACID transactions for data consistency. |
| **Ease of Use and Administration** | May have a steeper learning curve for beginners due to SQL complexity but offers robust administration tools and GUIs like pgAdmin | Generally easier to learn and use, particularly for developers familiar with JavaScript and JSON. Offers intuitive query language and flexible schema design |
| **Community Support** | Strong community support with active development, extensive documentation, and third-party extensions | Large and active community with numerous resources, tutorials and forums for assistance and learning |

## 2.3. Overview and Comparison between Methodologies

### 2.3.1. Overview of Methodologies

In software development, methodologies are structured approaches or frameworks that guide the processes involved in creating software. These methodologies provide a set of practices, principles, and techniques to manage the complexities of software development efficiently.

Some common methodologies include:

* **Waterfall**: Waterfall is a linear and sequential approach where each phase of the software development lifecycle (SDLC) includes requirements, design, implementation, testing, deployment, maintenance is completed before moving on to the next phase.
* **Agile**: Agile methodologies emphasize iterative development, collaboration, and customer feedback. Some Agile frameworks could be mentioned such as:
  + **SCRUM**: SCRUM is an Agile framework that divides the development process into small iterations called sprint. It involves cross-function teams working collaboratively to deliver incremental improvements within a short time frame.
  + **Kanban**: Kanban is another Agile framework that focuses on visualizing the workflow and limiting work in progress (WIP). It uses a Kanban board to visualize tasks and their status, enabling teams to optimize their workflow continuously.
* **DevOps**: DevOps combines software development (Dev) and IT operations (Ops) to streamline the entries software delivery lifecycle. It emphasizes automation, collaboration, and continuous integration and delivery (CI/CD) to enable faster and more reliable software release.

### 2.3.2. Comparation Methodologies

In this section, the author will provide the advantages and disadvantages of each methodology to evaluate and consider which method is suitable for “Cuisine Hub” project.

Table 4: Advantages and Disadvantages of Waterfall

|  |  |
| --- | --- |
| **Advantages of Waterfall** | **Disadvantage of Waterfall** |
| **1. Clear Structure**: The sequential nature provides a clear roadmap for development. | **1. Limited Flexibility**: Little room for changes once the project moves to the next phases. |
| **2. Well-Defined Requirements**: Requirements are established upfront, reducing ambiguity | **2. High Risk**: Any misunderstanding or errors I requirements can lead to costly changes later. |
| **3. Easy to Understand and Mange**: Simple and easy to understand, making it easier to manage some projects. | **3. Limited Customer Involvement**: Customers have limited opportunities for feedback until end of the project |
| **4. Stable Scope and Budget**: Scope, timeline, and budget are typically well-defined at the beginning of the project. | **4. Late testing**: Testing is performed at the end, increasing the risk of identifying issues late in the development cycle. |
| **5. Documentation**: Extensive documentation is produced at each stage, aiding in traceability and compliance | **5. Inflexible to Change**: Changes in requirements are difficult and costly to accommodate once development has progressed |
| **6. Suitable for Small, Well-Understood Project**: Works well for projects with stable requirements and predictable outcomes | **6. Customer Dissatisfaction**: Limited customer involvement can lead to dissatisfaction if the final product doesn’t meet expectations. |

The waterfall methodology has its strengths in providing structure and clarity but also comes with limitations, particularly regarding flexibility and customer involvement.

Table 5: Advantages of Disadvantage of Agile

|  |  |
| --- | --- |
| **Advantages of Agile** | **Disadvantage of Agile** |
| **1. Flexibility and Adaptability**: Agile allows for changes and adjustment throughout the development process. | **1. Complexity:** Agile methodologies can be more complex to implement and require a mindset shift for some teams. |
| **2. Customer Collaboration**: Agile encourages continuous customer involvement and feedback, leading to higher customer satisfaction. | **2. Resource Intensive**: Agile requires active participation from stakeholders and dedicated involvement, which can be resource-intensive |
| **3. Faster Time to Market**: Incremental development and shorter interactions enable faster delivery of working software. | **3. Lack of Predictability**: Agile projects can be less predictable in terms of scope, timeline, and budget due to changing requirements. |
| **4. Early and Continuous Delivery of Value**: Value is delivered incrementally, allowing stakeholders to see tangible results early in the process. | **4. Emphasis on individuals over processes:** Agile relies heavily on empowered teams and individual interactions, which may not be suitable for all organizations. |
| **5. Improved Quality:** Continuous testing and integration lead to higher-quality software with fewer defects | **5. Difficulty in Scaling**: Agile methodologies can be challenging to scale to large or distributed teams and complex projects. |
| **6. Better Risk Management:** Risks are identified early and addressed iteratively throughout the project lifecycle. | **6. Requires Experiences and Discipline**: Agile requires a disciplined approach and experienced practitioners to be successful |

Agile methodologies offer many benefits, including flexibility, customer collaboration, and faster time to market. However, they also present challenges such as complexity, unpredictability, and resource intensity.

Table 6: Advantages and Disadvantage of DevOps

|  |  |
| --- | --- |
| **Advantage of DevOps** | **Disadvantage of DevOps** |
| **1. Faster Time to Market**: DevOps enables faster delivery of software through continuous integration and continuous delivery (CI/CD) pipelines | **1. Organizational Resistance**: Implemented DevOps may face resistance from traditional siloed organizational structures and cultures. |
| **2. Improved Collaboration**: DevOps promote collaboration between development, operations, and other stakeholders, breaking down silos and improving communication. | **2. Complexity**: Implementing DevOps practices and tools can be complex, requiring changes to processes, tools, and culture. |
| **3. Increased Stability and Reliability:** Automation and infrastructure as code (IaC) improve the stability and reliability of systems by reducing manual errors and inconsistencies. | **3. Skills Gap**: DevOps requires a diverse skill set encompassing development, operations, automation, and collaboration, which may be challenging to find or develop. |
| **4. Continuous Feedback and Reliability**: DevOps encourages a culture of continuous improvement through feedback loops, enabling teams to respond quickly to changes and customer feedback | **4. Security Concerns**: Rapid deployment and automation in DevOps can introduce security risks if not property managed and integrated into the workflow |
| **5. Scalability**: Dev Ops practices support scalability by automating deployment, scaling infrastructure, and managing resources dynamically | **5. Tooling Complexity**: Dev Ops involves a plethora of tools for automation, monitoring, and deployment, which can lead to tool sprawl and integration challenges. |

## 2.4. Choosing Technologies and Methodology

### 2.4.1. Technologies of “Cuisine Hub” Project:

“Cuisine Hub” project is a social networking application project, after considering and revieing many resources and combining them with the comparison table had been conducted in Table 1. The author decided React.js will be the Front-End library use for Front-End Development for some of reason:

* **High Performance**: React utilizes Virtual DOM, which improves performance by updating only the necessary parts of the DOM instead of the entire structure. This reduces rendering time and enhances user experiences.
* **Flexible Structure**: React has flexible and easily scalable structure which can break down the UI into independent components, making the code readable, maintainable, and reusable.
* **Large Library**: React has a large and integrates many useful libraries such as Redux, React Router, and Material-UI. This handles complex issues and speeds up the development process.
* **Good SEO Support**: React provides tools and libraries to optimize websites for search engines (SEO), making it easier to search and rank higher in search engines results page.
* **Strong community support**: With thousands of React users and developers worldwide, finding solutions for specific issues is easy.

In summary, React.js is a good choice for a social networking project because of its flexibility, performance, and strong community support.

Similar with Fron-End, comparation table in Table 2 had been imported for choosing Back-End programming languages. After considering each criteria include programming paradigm, community and ecosystem, security, learning curve, performance, and scalability, the author decided to use JavaScript as the Back-End programming language for the “Cuisine Hub” project for some reason below:

* **Language Consistency**: Using JavaScript for both Front-End and Back-End reduces complexity in language management. This makes development and maintenance easier.
* **Node.js**: Node.js is a popular platform for Back-End development using JavaScript. It provides high performance and scalability, making it suitable for social applications with large user bases and real-time processing requirements.
* **Robust Ecosystem**: JavaScript has a strong ecosystem of libraries and frameworks for Back-End development such as Express.js, Nest.js and Fastify. These tools provide necessary features for building and managing complex applications like social networks.
* **Strong Community**: JavaScript has a large and diverse community, meaning developers can easily find support, documentation, and solutions for complex issues that may be encountered during development.
* **Real-time Support**: JavaScript and Node.js support building Real-time applications easily. This makes implanting features like live chat and real-time notification easier.

In summary, JavaScript is an ideal choice for the backend of a social networking project, providing performance, scalability, and strong community support.

For the database, after considering the comparation table in Table 3 based on some criteria including data model, performance, security, fault tolerance, ease of use and administration, community support. The author decided to use MongoDB as the database management system based on some reason below:

* **Flexible Schema**: MongoDB is a NoSQL database, which means it doesn’t require a predefined schema. This flexibility allows developers to store data of verifying structures without the need to alter the existing schema, making it ideal for dynamic and evolving applications like “Cuisine Hub”.
* **Scalability:** MongoDB is designed to scale out horizontally, meaning developers can distribute data across multiple servers to handle large amounts of traffic and data. This ability to scale is essential for a social networking app that could undergo swift expansion and verifying levels of user engagement.
* **High Performance**: MongoDB uses internal memory for storing working sets, which allows for faster access to data. Additionally, it supports indexes and various query optimization techniques, enabling efficient data retrieval, making it suitable for handling the real-time data demands of “Cuisine Hub.
* **Rick Query Language**: MongoDB supports a powerful query language that allows for complex queries, aggression, and data manipulations. This flexibility in querying makes it easier to retrieve and analyze data for various features of “Cuisine Hub”.

In summary, MongoDB is a suitable choice for “Cuisine Hub” due to its flexible schema, scalability, high performance, and rich query language.

Table 7: Summary Table of Chosen Technologies

|  |  |  |
| --- | --- | --- |
| **Front-End** | **Back-End** | **Database** |
| React.js | JavaScript (Node.js) and Express.js | MongoDB |

### 2.4.2. Methodology of “Cuisine Hub” Project:

In tables 4,5, and 6, the author provided advantages and disadvantages of each methodologies include Waterfall, Agile and DevOps. After considering and researching, the author decided to apply Agile (SCUM) to “Cuisine Hub” project based on some reason below:

* **Iterative and Incremental Development**: Agile Scrum promotes iterative and incremental development, allowing to deliver working software in short cycles called sprints. This approach enables the author to receive feedback from stakeholders regularly and make necessary adjustments throughout the project, ensuring that the final product meets user requirements and expectations.
* **Flexibility and Adaptability**: Agile Scrum embraces change and uncertainty, providing flexibility to accommodate evolving project requirements and priorities. This adaptability is crucial for “Cuisine Hub” where features, user needs, and market trends quickly to new opportunities and challenges.
* **Focus on Delivering Value**: Agile priorities delivering value to users through the early and frequent delivery of working software. By focusing on delivering high priority working software.

In summary, Agile Scrum methodology is well-suited for “Cuisine Hub” as it promotes iterative and incremental development, flexibility, and adaptability, and focus on delivering value.

|  |
| --- |
| **Methodology** |
| Agile Methodology with Scrum Framework |

# 3. Requirement Analysis

## 3.1. Similar Application

### 3.1.1. Dev Community

A screenshot of a computer

Description automatically generated

Figure 3: Dev Community

(Source: <https://dev.to/> )

Dev Community is an online platform where developers and technologies can connect, share ideas, ask questions, and discuss various topics related to software development, programming languages, tools, and technology trends.

The author chooses Dev Community as a similar application based on some point of view below:

* **Active Community**: Dev Community boats act active and engaged user base of developers and technologists from around the world. This the author will have access to a diverse range of perspectives, expertise, and experiences to draw from.
* **Knowledge Sharing**: The platform is centered around knowledge sharing. Users regularly post articles and technology. This wealth of information can serve as a valuable resource for “Cuisine Hub.”
* **Networking Opportunities**: Dev Community provides opportunities for networking and building connections with other professionals in the field. Engaging with the community can help the author establish relationships with like mindset individuals, potential collaborators, or even future employers.
* **Inspiration and Motivation**: Browsing through the posts and discussions on Dev Community can be inspirational and motivating. Seeing what others are working on, learning about new technologies, and reading success stories can help fuel the author’s creativity and drive.

Secondly, the author would mention the user interface of Dev Community, below are some reviews from personal point of view:

|  |  |
| --- | --- |
| **Dev Community User Interface Components** | |
| **Components** | **Author’s Point of View** |
| **Header** | The header contains essential elements such as site logo, search bar, and button for account creation and logging in. These elements are easily accessible and prominent |
| **Navigation** | The navigation menu provides quick access to various sections of the websites, including Home, Podcasts, Videos, Tags, and more. This makes it easy for users to explore different content categories. |
| **Content Feed** | The main content feed displays posts from the community, organized by relevance, latest, and top. Each post is presented with a title, author, date, and tag, making it easy to scan through the content. |
| **Featured Challenges** | There are prominent banners promoting ongoing challenges, which can attract user’s attention and encourage participation. |
| **Sidebar** | The sidebar offers additional navigation options, including links to Dev Help, Forum Shop, Advertise on Dev, and more. It provides supplementary resources and functionalities for users. |
| **Visual Design** | The overall color scheme and visual design are pleasing to the eye. The use of contracting colors for button and links helps them stand out, enhancing usability. |
| **Engagement Features** | The inclusion of hashtags and comment counts encourages user engagement and participation in discussions. |

Overall, the user interface of Dev Community seems well-designed, with a focus on usability and community engagement. It provides a platform that is easy to navigate and encourages users to interact with content and each other.

Thirdly, the author would mention the features of Dev Community, and all opinions below are just come from the personal point of view:

|  |  |
| --- | --- |
| **Dev Community Features** | |
| **Features** | **Author’s Point of View** |
| **User Authentication** | Dev Community allows users to create accounts and log in, enabling personalized experiences and interactions. |
| **Content Feed** | Dev Community features a feed where users can browse articles, discussions, and posts shared by the community. Content is organized by relevance, latest, and top to carter to different user preferences |
| **Navigation Menu** | A comprehensive navigation menu offers easy access to different sections of the websites, including Home, Podcast, Videos, Tags, and more |
| **Hashtag and Tags** | Users can tag their posts with relevant hashtags, allowing for easier categorization and discovery of content. Tags are also used to group similar content together. |
| **Commenting System** | Dev Community includes a commenting system that allows users to engage in discussions on posts. The number of comments is displayed each post, indicating its level of engagement |
| **Search Bar** | A search bar at the top of the page allows users to search for specific topics, articles, or discussions. |
| **Profile Settings** | Users likely have access to profile setting where they can manage their account information, preferences, and notification settings |

Finally, a table of comparison between advantages and disadvantages of Dev Community is drawn by author in below. And again, all of options include above is just come from author’s point of view:

|  |  |
| --- | --- |
| **Advantages of Dev Community** | **Disadvantage of Community** |
| Active community engagement | Limited visibility into content quality |
| Diverse range of content topics | Potential for information |
| User-friendly interface | Lack of advanced filtering options |
| Promotes community participation | Dependency on user-generated content |
| Provides platform for sharing knowledge | Potential for echo chamber effects |

### 3.1.2. GitHub Discussions

A screenshot of a computer

Description automatically generated

Figure 4: GitHub Discusstion

(Source: <https://github.com/orgs/community/discussions>)

GitHub Discussion is a feature within GitHub repositories that allows developers to engage in threaded discussion, ask questions, and collaborate on projects.

The author chooses GitHub Discussions as a similar application based on some point of view below:

* **Community Interaction**: GitHub Discussions allows developers to engage in discussion, share knowledge, and collaborate with others in the community.
* **Focus on Specialized Topics**: GitHub Discussion focuses specific topics related to software development, open-source projects, and technology, catering to a niche audience of developers and technologists.
* **Content Sharing**: Users can share various types of content on GitHub Discussions, including questions, answers, ideas, and feedback, like how users share posts, photos, and videos on traditional social networking applications.

Secondly, the author would mention the user interface of GitHub Discussion, below are some reviews from personal point of view:

|  |  |
| --- | --- |
| **GitHub Discussion User Interface Components** | |
| **Components** | **Author’s point of view** |
| **Header** | The header contains the GitHub logo and navigation links =, providing easy access to different of the GitHub platform |
| **Categories and Discussion** | The main content are displays categories on the left and discussion on the right |
| **Discussion Cards** | Each discussion is presented as a card, showing the title, category, author, number of replies, and last activity. This card-based layout makes it easy to scan through discussion and identify topics of interest |
| **Search Bar** | A search bar at the top allows users to search for specific discussions or topics, enhancing usability and discoverability |
| **Filter Options** | User can filter discussion based on criteria such as label, sort order, and activity, giving them more control over how they view and interact with content. |
| **New Discussion Button** | A button “New Discussion” encourages users to start new discussions, contribution to community participants and content generation. |

Thirdly, the author would mention some of features of Dev GitHub Discussion, and all opinions below are just come from the personal point of view:

|  |  |
| --- | --- |
| **GitHub Discussion Features** | |
| **Features** | **Author’s point of view** |
| **User Authentication** | GitHub Discussions likely include user authentication functionality, requiring users to login with their GitHub Account to participate in discussion. This ensures that users are authenticated and authorized to access and contribute to discussion, enhancing security and accountability within the platform. |
| **Categories** | GitHub Discussion are organized into categories, allowing users to easily navigate and find relevant topics of interest. |
| **Seach Functionality** | A search bar enables users to search for specific discussions or topics, helping them quickly find relevant content |
| **New Discussion Creation** | “New Discussion” button that allows users to create new discussions, contributing to community engagement and content creation. |
| **Notifications** | Users likely receive notifications for new replies or mentions in discussions, helping them stay updated on relevant activity. |
| **User Roles and Permission** | GitHub Discussions likely includes features for managing user roles and permission, allowing administrator to manage discussions and enforce community guidelines. |

These features of GitHub Discussions contribute to a robust and user-friendly discussion platform that fosters collaboration, knowledge sharing, and community engagement among developers and technologies.

Finally, a table of comparison between advantages and disadvantages of Dev Community is drawn by author in below. And again, all of options include above is just come from author’s point of view:

|  |  |
| --- | --- |
| **Advantages of GitHub Discussion** | **Disadvantage of GitHub Discussion** |
| Seamless Integration with GitHub | Limited customization options for discussion layout |
| Organized by Categories | Dependency on GitHub ecosystem for usage |
| Search Functionality | Potential for spam or irrelevant discussions |
| Filtering Options | Limited accessibility features |
| Notifications | May require familiarity with GitHub interface |
| User Authentication |  |

The table above outlines the strengths and weaknesses of GitHub Discussions, highlighting its integration with GitHub, organized discussion structure, and search functionality as advantages, while noting limitations such as dependency on the GitHub ecosystem and potential moderation challenges.

## 3.2. Requirements Analysis

### 3.2.1. Functional Requirements

Based on (GeekforGeek, 2024), functional requirements are specific requirements regarding function and activities that a system, product, or project must perform. These requirements describe the features, behaviors, and specific functions that users expect from the product or system. They are detailed descriptions of what needs to be done, including the functions to be performed, characteristics of the user interface, business processes and data requirements.

Below are the functional requirements table of “Cuisine Hub” project.

|  |  |  |
| --- | --- | --- |
| **No.** | **Functions** | **Description** |
| **1** | User Registration | User should be able to create an account by providing necessary such as username, email, and password |
| **2** | Profile Management | Users should be able to edit their profiles, including adding or changing profile pictures, updating personal information. |
| **3** | Post Creation | User should be able to create new posts, including text, images, videos, and share them with their network |
| **4** | New Feeds | Users should have a personalized news feed that displays posts from users they follow, sorted by relevance or recency. |
| **5** | Like and Comment | Users should be able to like and comment on posts made by other users in their network. |
| **6** | Friend Request | Users should be able to send and accept friend requests to connect with other users on the platform |
| **7** | Messaging | Users should be able to send private messages to other users. |
| **8** | Notifications | User should receive notifications for activities such as likes, comment, friend request, and messages. |
| **9** | Search | Users should be able search for other users, posts, or topics within the app |
| **10** | Privacy Settings | Users should have control over their privacy settings, including who can view their profiles, posts, and personal information. |

### 3.2.2. Non-Functional Requirements

Based on (GeekforGeek, 2024), non-functional requirements describe the attributes or qualities that a system must have rather than specific behaviors or functions. These requirements focus on aspects such as performance, usability, security, scalability, and reliability.

|  |  |  |
| --- | --- | --- |
| **No.** | **Aspects** | **Description** |
| **1** | Performance | The app should load quickly, with page loading in under 3 seconds, and response time for user actions (e.g., linking a post, sending a message) should be under 1 second |
| **2** | Scalability | The app should be able to handle concurrent users without significant performance degradation. It should support at least 10,000 active users simultaneously. |
| **3** | Security | User data should be encrypted both in transit and at rest. The app should implement secure authentication mechanisms, protect against common web vulnerabilities (e.g., XSS, CSRF), and have a robust user authorization system. |
| **4** | Usability | The user interface should be intuitive and user-friendly, catering to users of all ages and technical proficiency levels. It should follow accessibility standards to ensure that it is usable by individuals with disabilities. |
| **5** | Reliability | The app should be stable and reliable, with minimal crashes or downtime. It should gracefully handle errors and provide informative error message to users when issues occur |
| **6** | Compatibility | The app should be compatible with major web browsers (Chrome, Firefox, Safari, Edge) and be responsive across various devices (desktops, tablets, smartphone) and screen sizes. |

### 3.2.3. Data requirements

Data requirements specify the types of data that the system needs to store, process, and manage. Below are the table outline some data requirements for “Cuisine Hub” project:

|  |  |  |
| --- | --- | --- |
| **No** | **Data Entity** | **Attributes** |
| **1** | Users | - User ID, Username, Email. Password (hashed), Avatar, Location, Professional. Friends |
| **2** | Post | User ID, Description, Image, Likes. Comment ID |
| **3** | Comment | User ID, Post ID. Comment, likes |
| **4** | Verification | User ID, Token, created At, expired at |
| **5** | Friend Request | Request To, Request From, Request Status |

### 3.2.4. Development requirements

The chosen methodology for “Cuisine Hub” project is Agile Scrum. Details of reason why is Agile and its advantage for social networking application like “Cuisine Hub” has been explained in section 2.4.2. In this section, the author provides how the project is managed by collecting user stories, setup product backlog and sprint timebox.

#### 3.2.4.1. User Stories

|  |  |  |
| --- | --- | --- |
| **Use Case ID** | **User Story** | **Acceptance Criteria** |
| **US01** | As a user, I want to log in to my account | As a user, I want to be able to log in to my account to access the features of the social networking app. |
| **US02** | As a user, I want to register for a new account | As a user, I want to be able to register for a new account to experience and user the social networking app. |
| **US03** | As a user, I want to have email verification for account security | As a user, I want to have the option for email verification during the registration process to enhance the security of my account and prevent unauthorized access. |
| **US04** | As a user, I want to create and edit my profile | As a user, I want to be able to create and edit my personal profile including avatar and some other information |
| **US05** | As a user, I want to search posts | As a user, I want to be able to search for posts based on keywords or topics of interest to quickly find relevant content within the app. |
| **US06** | As a user, I want to interact with posts | As a user, I want to be able to like, comment other user’s posts to show interacts and engage with them |
| **US07** | As a user, I want to manage my posts | As a user, I want to be able to manage my post including viewing, editing, or deleting. |
| **US08** | As a user, I want to have a dark mode option | As a user, I want to be able to switch to a dark mode theme within the app to reduce eye strain and improve visibility, especially in low-light environment. |
| **US09** | As a user, I want to send friend requests | As a user, I want to be able to send friend request to other users on the platform, allowing me to connect with them and expand my social network |
| **US10** | As a user, I want to accept or decline friend request | As a user, I want to be notified when I receive friend requests and can accept or decline them, giving me control over who can connect with me on the platform |
| **US11** | As a user, I want to receive friend suggestions | As a user, I want to receive friend suggestions based on mutual connections, interests, or other relevant criteria, helping me discover and connect new people on the platform |

#### 3.2.4.2. Product Backlog

|  |  |  |
| --- | --- | --- |
| **Use Case ID** | **Priority** | **Sprint** |
| **US01** | Must Have | 1 |
| **US02** | Must Have | 1 |
| **US03** | Must Have | 1 |
| **US04** | Must Have | 1 |
| **US05** | Must Have | 1 |
| **US06** | Must Have | 1 |
| **US07** | Must Have | 1 |
| **US08** | Should Have | 2 |
| **US09** | Should Have | 2 |
| **US10** | Should Have | 2 |
| **US11** | Should Have | 2 |

#### 3.2.4.3. Sprint Timebox

Table 8: Sprint Timebox

|  |  |  |  |
| --- | --- | --- | --- |
| **Sprint Number** | **Sprint Duration** | **Start Date** | **End Date** |
| Sprin 01 | 3 weeks (21 days) | March 4 2024 | March 24 2024 |
| Sprint 02 | 2 weeks (14 days) | March 25 2024 | April 7 2024 |
| Sprint 03 | 2 weeks (14 days) | April 8 2024 | April 21 2024 |

The table outlines the sprint schedule for “Cuisine Hub” project, likely in an agile software development context.

The table consists of four columns:

* **Sprint number**: This column lists the sequential number assigned to each sprint in the project. Sprint numbers are typically used for tracking and referencing specific iterations of development.
* **Sprint Duration**: This column specifies the duration of each sprint in terms of week and days. For example, the first sprint (Sprint 01) has a duration of 3 weeks, equivalent to 21 days.
* **Start Date:** This column indicates the start date for each sprint. It shows when each sprint begins, allowing developers to plan and coordinate work accordingly.
* **End Date**: This column displays the end date of each sprint. It shows when each sprint is scheduled to conclude, providing a clear timeline for completing the designated tasks and deliverables.

# 4. Software Design

## 4.1. Application Architecture

A diagram of a computer

Description automatically generated

Figure 5: Overview Architecture Diagram

The diagram above is an overview of application architecture. It includes:

|  |  |
| --- | --- |
| **Components** | **Descriptions** |
| **User** | Represented end-user who interact with the application through the client-side interface |
| **Client-Side (React.js)** | This is the user interface portion that end-users interact with directly. React.js is a popular JavaScript library used to build interactive user interface. |
| **Web Server** | Act as web server responsible for processing requests from the client and responding with corresponding data. In this case, Node.js serves as the web server. |
| **Database (MongoDB)** | Represents the MongoDB database, a popular non-relational database management system used for web applications. |
| **File System** | Include static resources such as HTML, CSS, and images used in the user interface. |

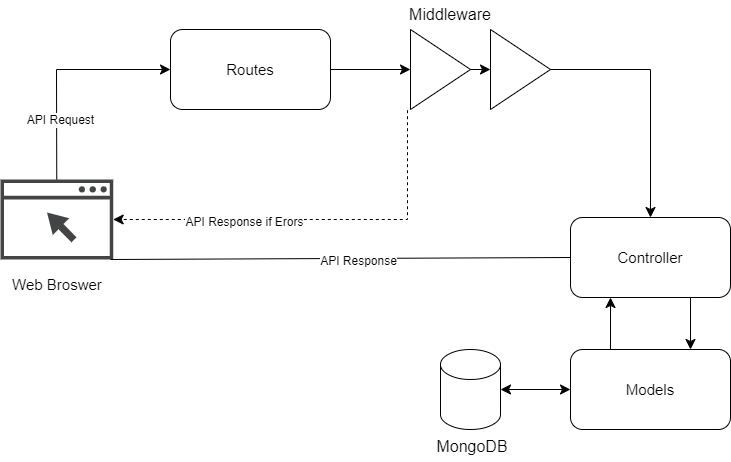


Figure 6: Web Server Workflow

|  |  |
| --- | --- |
| **Components** | **Description** |
| **Web Browser** | This is the client component that sends an API request to the Server. The request is typically initiated by a user interacting with a web application |
| **Routes** | Routes are used to handle different API endpoints. Each route is associated with a specific controller function that is executed when the route is accessed. |
| **Middleware** | Middleware functions are used to handle requests and responses in a server-side application. They can be used for tasks such as authentication, logging, and error handling. |
| **Controller** | The controller is the component that handles the main logic of the application. It receives a request from the route, processes it, and return a response |
| **API Response** | This is the request that is sent back to the client (web browser) after the server has processed the request. The response typically includes a status code header, and a body. |
| **Models** | Models are used to define the structure of the data in the application. They provide a way to interact with the database |

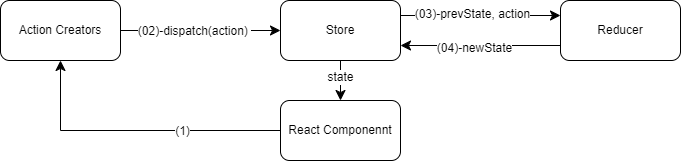


Figure 7: Redux Workflow

The diagram above depicts the data flow in a typical Redux architecture within a React application. Below is the breakdown:

|  |  |
| --- | --- |
| **Components** | **Description** |
| **React Component** | This represents the user interface components in a React application. These components interact with the Redux store through action and receive updated state from the store. |
| **Action Creators** | Action creators are functions that create and return action objects. Actions are payloads of information that send data from application to Redux store. They are the only source of information for the store. |
| **Dispatch** | Dispatch is a function provided by Redux that is used to dispatch actions to the Redux store. When an action is dispatched. It flows through the middleware and then into the reducer. |
| **Store** | The redux store holds the state tree of the application. It is a single source of truth that manages the application state. When an action is dispatched, the store passes the current state and the action to the reducer. |
| **Reducer** | Reducers are pure functions that specify how the application’s state changes in response to action sent to the store. They take the previous state and an action and return to the new state. The new state changes in response to action sent to the store. They take the previous state and an action and return to the new state. The new state is then stored in the Redux Store |

## 4.2. Use Case Diagram

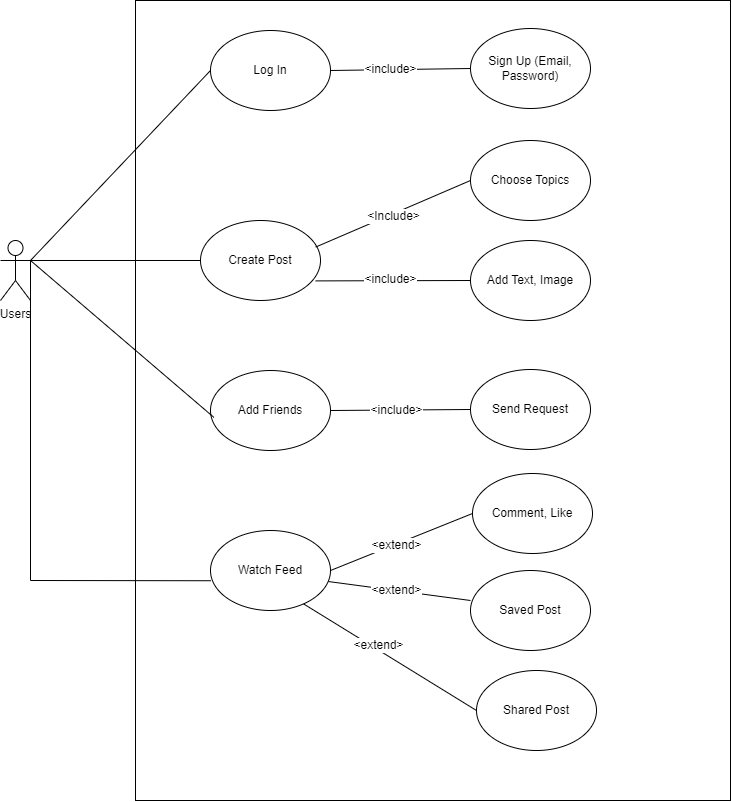


Figure 8: User Case Diagram (User Role)

|  |  |
| --- | --- |
| **Use Case** | **Description** |
| **Users** | The primary actors in the application, who can log in or sign up to access the platform. |
| **Log In** | The process by which users access their existing accounts on the platform |
| **Sign Up** | The process by which new users create an account on the platform, providing their email and password |
| **Choose Topics** | Users can select specific topics of interest to customize their feed and content |
| **Create Post** | Users can create their own posts, adding text or images |
| **Watch Feed** | Users can view a stream of content from other user, based on their chosen topics |
| **Saved Post** | Users can save posts for later viewing or reference |
| **Add Friends** | Users can add other users as friends, allowing them to view each ‘other’s content and interact more directly. |
| **Send Request** | Users can send friend to other users, who can then choose to accept or decline |
| **Comment, Like** | Users can engage with posts by leaving comments or liking them |
| **Shared Post** | Users can share posts from other users, who can then choose to accept or decline. |

## 4.3. Class Diagram

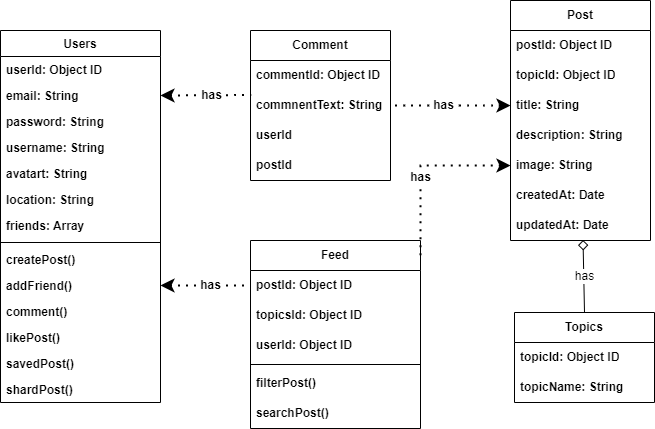
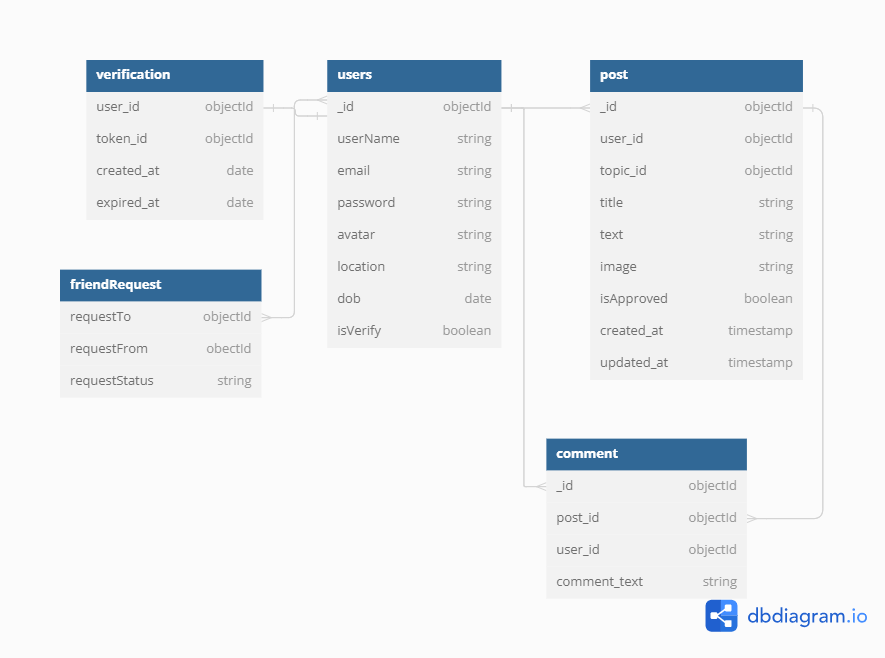


Figure 9: Class Diagram

|  |  |
| --- | --- |
| **Class** | **Description** |
| **Users** | This is an object type representing individual users of the social media platform. Each user has various attributes, such as ID (userId, email, username, password, location, and a list of friends). Users can also perform various actions, such as creating posts, adding friends, commenting on posts, liking posts, saving posts, sharing posts, and viewing their feed. |
| **Comment** | This is an object type representing a comment on a post. Each comment has an ID, the text of the comment, and the ID of the user who made the comment. |
| **Post** | This is an object type representing a post on social media platform. Each post has an ID, the title of the post, the text of the post, the ID of the user who made the post, the ID of the topic the post belongs to, the image associated with the post, and the date and time the post was created and last updated. Users can interact with posts by commenting, liking, saving, and sharing |
| **Topics** | This is an object type representing a topic on the social media platform. Each topic has an ID and a name. Topics can have posts associated with them, and users can follow topics to see posts related to that topic in their feed. |

## 4.4. Database Diagram



The diagram appears to represent a database schema, outlining the structure and relationships between different tables / entities in a database.

Below is a breakdown of each entity and its attributes:

|  |  |
| --- | --- |
| **Entity** | **Attribute** |
| verification | user\_id (object ID), token\_id (object ID), created\_at (date), expired\_at (date) |
| users | \_id (object ID), firstName (string), lastName (string), email (string), password (string), avatar (string), location (string), dob (date), isVerify (boolean) |
| friendRequest | requestTo (objectID), requestFrom (object ID), requestStatus (string) |
| post | \_id (object ID), user\_id (object ID), title (String), text (string), image (string), isApproved (boolean), created\_at (date)m updated\_at (date). |
| comment | \_id (object ID), post\_id (object ID), user\_id (object ID), comment\_text (string) |

## 4.5. Activities Diagram

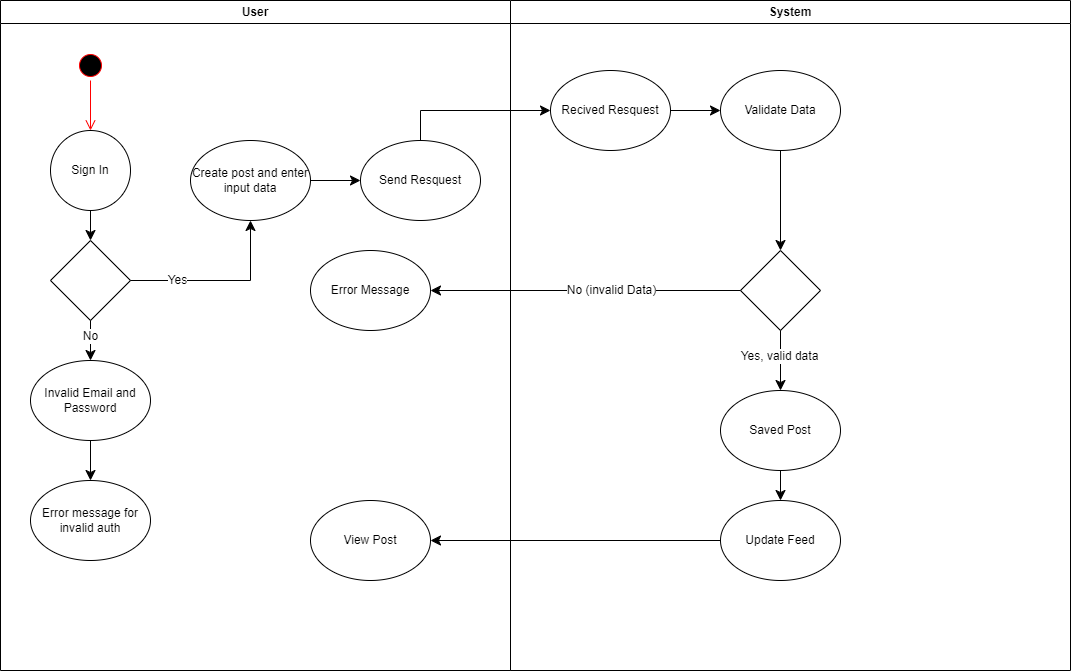


Figure 10: Activities Diagram for create post function.

The activity diagram above illustrates the interactions between a user and a system when the user creates a post. Below is the explanation:

1. User Activities:

* Sign In: The user begins by signing into the system.
  + If the credentials are invalid, an error message for invalid email and password is displayed.
* Create post and enter input data: The user created a post and enters the necessary data.
* View Post: The user views a post.

2. System Activities

* Receive Request: The system receives a request from the user.
* Send Request: The system sends the request to perform certain actions.
* Validate Data: The system validates the received data.
  + If the data is invalid, an error message is generated.
* Update Feed: The system updates the feed with new posts.
* Save Post: The system saves the post if the data is valid.

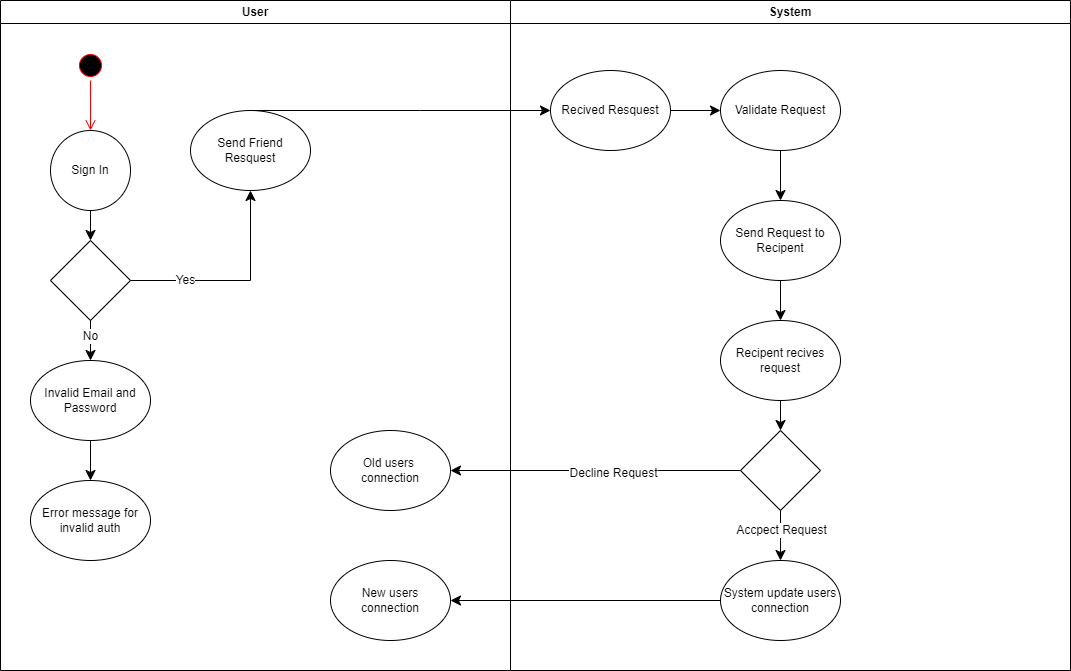


Figure 11: Activities Diagram for Friend Request

The activity diagram above depicts the interactions between user and system when user send a friend request. Below is an explanation:

1. User Activities:

* Sign In: The user initiates by signing into the system.
  + If the credentials are invalid, an error message for invalid email and password is shown.
* Send Friend Request: The user established a connection with old users.
* No connection: The user has no connection with the recipient.
* New users’ connection: The user establishes a new connection recipient if request has accepted.

2. System Activities:

* Receive Request: The system receives a request from the user.
* Validate request: The system validates the received request.
* Send request to recipient: If the request is valid, the system sends the request to the intended recipient.
* Recipients receive request: The system notifies the recipient about the received request.
* Decline Request: The recipient has the option to decline the request.
* Accept Request: The reception accepts the request.
* System update users connection: The system updates the user connection based on the accepted or declined requests.

## 4.6. Wireframe

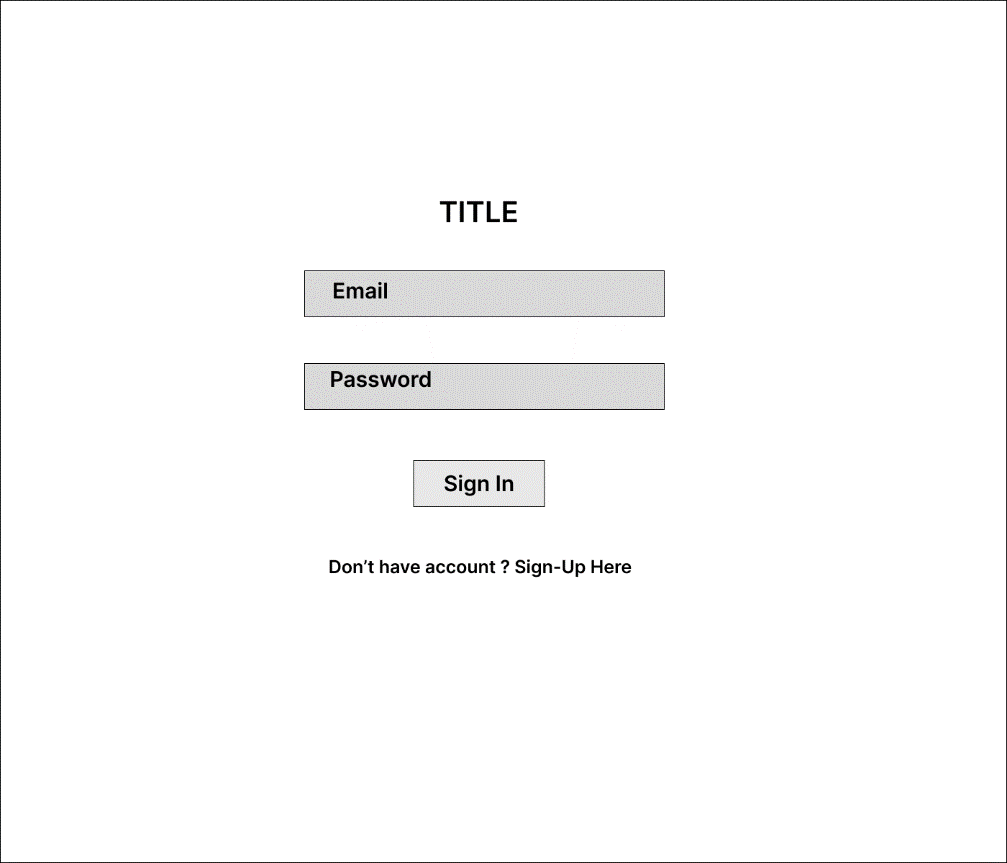


Figure 12: Login Page in Wireframe

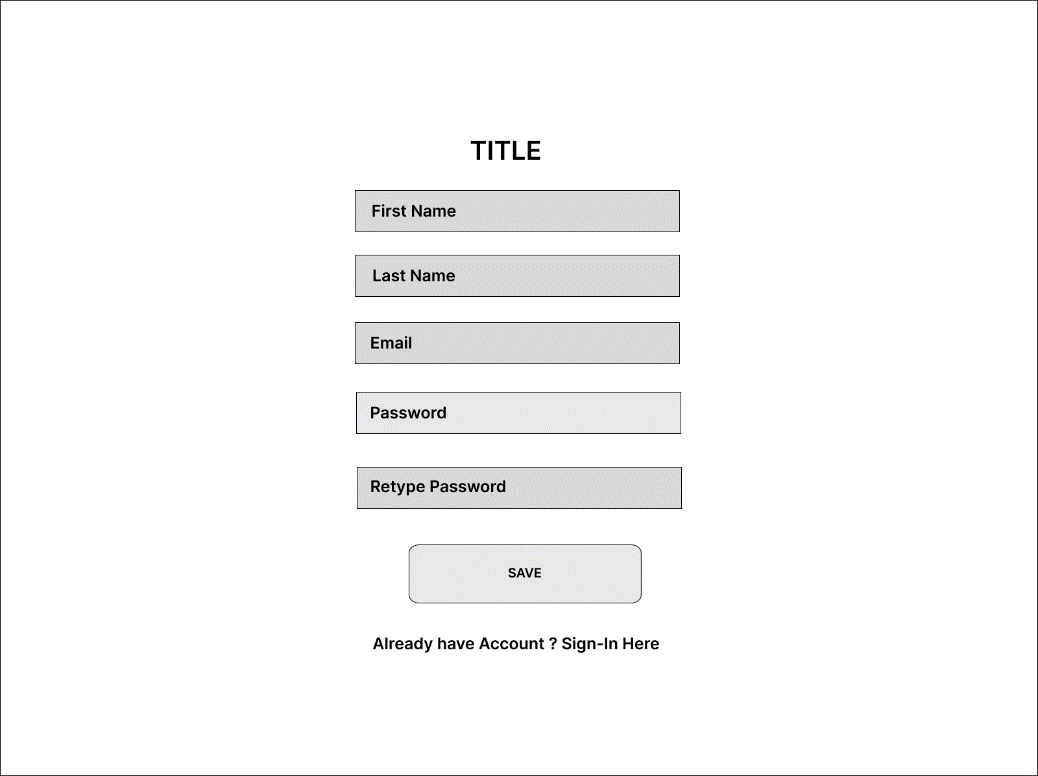


Figure 13: Sign in Page in Wireframe



Figure 14: User Page in Wireframe

# 5. Software Implementation

## 5.1. Development Environment

|  |  |
| --- | --- |
| **Development Environment** | **Description** |
| **Programming Language and Framework** | JavaScript with MERN Stack: MongoDB, Express.js, React.js, and Node.js |
| **Development Tools** | Visual Studio Code (VS Code) for coding and debugging |
| **Servers and Database** | Utilizing MongoDB Atlas for Database and Render for server hosting |
| **Source code Management** | Utilizing Git and GitHub for version control and source code management |
| **Deployment Environment** | Deploying with Render for streamlined deployment and hosting |

## 5.2. Code Structure

A screenshot of a computer

Description automatically generated

Figure 15: Back-End Code Structure

|  |  |
| --- | --- |
| **Directory / File** | **Description** |
| **BACKEND** | Main directory |
| **src** | Source Code directory |
| **config** | Configuration files |
| **controllers** | Controllers for handling request and response |
| **middleware** | Middleware functions for request processing |
| **models** | Data models representing the structure processing |
| **routes** | Route definition for handling HTTP requests |
| **utils** | Utilities Functions |
| **. env** | Environment variables configuration file |
| **node\_modules** | Directory for installed dependencies via npm |
| **.gitignore** | File specifying intentionally untracked files for Git |
| **index.js** | Entry point file the application |
| **package-lock.json** | Dependency locking file generated by npm |
| **package.json** | Metadata file for the project, including dependencies |

A screenshot of a computer

Description automatically generated

Figure 16: Front-End Code Structure

|  |  |
| --- | --- |
| **Directory/File** | **Description** |
| **node\_modules** | Directory for installed dependencies via npm |
| **src** | Source Code directory |
| **assets** | Static assets (image, fonts, etc) |
| **components** | Reuseable React components |
| **layouts** | Layout-specific components or templates |
| **pages** | Page-specific components or routes |
| **redux** | Redux state management library code |
| **routers** | Routing configuration or router components |
| **styles** | CSS or other styling files |
| **App.jsx** | Main application component |
| **main.jsx** | Entry point or main component |
| **.gitignore** | File specifying intentionally untracked files for Git |
| **Index.html** | Main HTML file for the application |
| **package-lock.json** | Dependency locking file generated by npm |
| **package.json** | Metadata file for the project, including dependencies |
| **vite.config.js** | Vite development server configuration |

## 5.3. Technical Problems and Solutions

|  |  |
| --- | --- |
| **Technical Problems** | **Solution** |
| Decreasing application performance with large data load | Use pagination or lazy loading techniques to only load necessary data at the required time. |
| Handling asynchronous processing issues | Utilize Promises or async / await to handle asynchronous tasks in a clear and readable manner. |
| Security risks from Cross-site Scripting (XSS) vulnerabilities | Employ security libraries’ like Helmet.js to protect the application from XSS attacks |
| Versioning and release management | Utilize version control tools like Git and Continuous Integration / Continuous Deployment (CI / CD) techniques for automated releases. |
| Slow response times from external API calls | Implement caching mechanisms to store responses from external APIs and reduce response time |
| Code complexity leading to maintenance challenges | Refactor complex code into smaller, modular components and utilize design patterns for better maintainability. |
| Cross-browser compatibility issues in web development | Use CSS and JavaScript libraries that offer cross-browser support and perform throughout testing across different browsers. |

## 5.4. Development Notes

To install and run “Cuisine Hub” project, follow the below steps:

|  |  |
| --- | --- |
| Step | Content |
| 1 | Download source code from: <https://github.com/Quan2409/cuisinehub> |
| 2 | Install development environment and necessary dependencies |
| 3 | Open Terminal and redirect to folder contain source code |
| 4 | Run command “npm install” to install dependencies package |
| 5 | Run command “npm run dev” to start the application |

# 6. Evaluation

## 6.1. Final Web Screenshots

## 6.2. Test Plan

### 6.2.1. Scope

|  |  |  |
| --- | --- | --- |
| **Test Category** | **In-Scope** | **Out of Scope** |
| User Authentication and Authorization | User registration, login, logout | Single Sign-In (SSO) integration |
| Profile Management Testing | Profile creation, editing | Social media analytics integration |
| Post Creation and Interaction Testing | Creating posts, like, comments | Advertisements management |
| Mobile application Testing | UI/UX, offline capabilities | Wearable device compatibility |
| Security Testing | Penetration testing, encryption | Compliance audits |
| Performance and Scalability Testing | Response times, scalability | Hardware stress testing |
| Integration and Compatibility Testing | Third-party API integration | Lagacy system integration |
| Sending Friend Request | Sending friend request to other users | Sending friend requests via third-party integrations |
| Receiving and handling friend requests | Receiving and responding to friend requests | Automatically accepting or rejecting friends’ requests |
| Friend List Management | Adding accepted requested to the user’s friend list | Automatic organization or categories of friends |
| Error handling and Edge Cases | Handling invalid or expired friend requests | Advanced validation of friend request data or metadata |

### 6.2.2. Test Methodology

**White Box**: White Box Testing is a software testing method in which the tester is aware of the thing being tested is internal structure, design, and implementation. The implementation and impact of the code are evaluated.

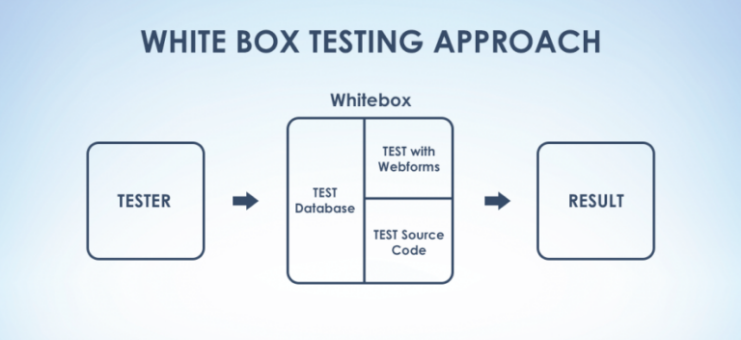


Figure 17: White Box Testing

**Black Box**: Black Box Testing is a type of software testing in which the tester has no knowledge of the item’s internal structure, design, or implementation. Only the outward design and construction are evaluated.



Figure 18: Black Box Testing

Different between Blackbox Testing and White Box Testing:

|  |  |
| --- | --- |
| **Black Box Testing** | **White Box Testing** |
| Functional Testing | Check code coverage, navigation completeness |
| Less exhaustive | More exhaustive |
| Nothing is known about internal structure | Know about internal structure |
| Code implementation is not needed | Code implementation is necessary |
| Behavior Testing | Logic Testing |
| Easier | More difficult |

### 6.2.4. Test Resource and Environments

|  |  |
| --- | --- |
| **Resource** | **Description** |
| **Test Plan** | Document outlining the scope and approach of testing |
| **Test Cases** | Detailed scenarios and steps of testing |
| **Test Data** | Sample user profiles, posts, comment, ... |
| **Test Environment** | Local development environment with test database |
| **Test Tools** | Postman for API testing, Selenium for automated UI testing, Mocha for backend testing, |
| **Test Documentation** | Test Logs |
| **Test Device** | Desktops, laptops, smartphone for testing |
| **Test Servers** | Local Server for hosting test environment |

## 6.3. Test Case

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Description** | **Implementation Step** | **Expected Result** | **Actual Result** | **Status (Pass or Fail)** |
| **Test Login** | | | | | |
| **TC-01** | Verify successful login with valid credentials | 1. Navigate to the login page  2. Enter valid email  3. Enter valid password  4. Click on the “Login” button | User should be redirected to the home page | Success | Pass |
| **TC-02** | Verify error message for invalid email | 1. Navigate to the login page  2. Enter invalid email  3. Enter valid password  4. Click on the “Login” button | Error message should be displayed indicating invalid username | Success | Pass |
| **TC-03** | Verify error message for invalid password | 1. Navigate to the login page  2. Enter valid email  3. Enter invalid password4. Click on the “Login” button. | Error message shou be displayed indicating invalid password | Success | Pass |
| **TC-04** | Verify error message for empty email | 1. Navigate to the login page  2. Enter blank email  3. Enter valid password  4. Click on the “Login” button | Error message should be displayed indicating empty username field | Success | Pass |
| **TC05** | Verify error message for empty password | 1. Navigate to the login page  2. Enter valid email  3. Enter blank password  4. Click on the “Login” button | Error message should be displayed indicating empty password field | Success | Pass |
| **Test Register** | | | | | |
| **TC06** | Verify successful registration with valid input | 1. Navigate to the registration page  2. Enter valid first name, last name, email, password  3. Retype the same password  4. Click on the “Register” button | User should be successfully registered and redirected to the login page | Success | Pass |
| **TC07** | Verify errors message for empty first name | 1. Navigate to the registration page  2. Enter blank first name  3. Enter valid last name, email, password  4. Retype the same password  5. Click on the “Register” button | Error message should be displayed indicating empty first name field | Success | Pass |
| **TC08** | Verify errors message for empty last name | 1. Navigate to the registration page  2. Enter valid first name, email, password  3. Enter blank last name  4. Retype the same password  5. Click on the “Register” button | Error message should be displayed indicating empty last name field | Success | Pass |
| **TC09** | Verify error message for mismatched passwords | 1. Navigate to the registration page  2. Enter valid first name and last name, email, password  3. Enter different password in the retype field  4. Click on the “Register” button | Error message should be displayed indicating password mismatch | Success | Pass |
| **TC10** | Verify sending of verification email | 1. Complete the registration process  2. Check registered email inbox | A verification email should be received containing a verification link | Success | Pass |
| **Test Create Post** | | | | | |
| **TC11** | Verify successful creation of a post | 1. Navigate to the “Create Post” page.  2. Enter post content  3. Upload any attached media  4. Click on the “Post” button | Post should be successfully created and displayed on the user’s profile/feed | Success | Pass |
| **TC12** | Verify error message for empty post content | 1. Navigate to the “Create Post” page  2. Leave the post content field empty  3. Click on the “Post” button | Error message should be displayed indicating empty post content field | Success | Pass |
| **TC13** | Verify error message for uploading invalid image file types | 1. Navigate to the “Create Post” page  2. Attempt to upload a file with an invalid image format (e.g.: docx)  3. Enter post content  4. Click on the “Post” button | Error message should be displayed indicating invalid file format | Success | Pass |
| **Test Interact Post** | | | | | |
| **TC14** | Verify commenting on a post | 1. Navigate to a post  2. Enter a comment in the comment field  3. Click on the “Comment” button or press Enter | Comment should be successfully added to the post | Success | Pass |
| **TC15** | Verify liking a post | 1. Navigate a post  2. Click on the “Like” button | Post should be successfully liked | Success | Pass |
| **TC16** | Verify unlinking a post | 1. Navigate to a post  2. Click on the “Like” button | Like should be successfully removed from the post | Success | Pass |
| **TC17** | Verify editing own post | 1. Navigate to a post that the user has already liked  2. Click on the “Edit” button | User should be able to edit the content of the post and save changes | Success | Pass |
| **TC18** | Verify deleting own post | 1. Navigate to a post created by the user  2. Click on the “Delete” button or select “Delete” option from the post menu | Post should be successfully deleted from the user’s profile/feed | Success | Pass |
| **TC19** | Verify viewing comments on a post | 1. Navigate to a post  2. Scroll down to view comments or click on the “View” link if available | Comments related to the post should be displayed | Success | Pass |
| **TC20** | Verify replying to a comment | 1. Navigate to a post with comments  2. Click on “Reply” button next to a comment | User should be able to reply to the selected comment | Success | Pass |
| **TC21** | Verify deleting own comment | 1. Navigate to a post with comments  2. Click on the “Delete” button next to the user’s own comment | Comment should be successfully deleted from the post | Success | Pass |
| **TC22** | Verify editing own comment | 1. Navigate to a post with comments  2. Click on the “Edit” button next to the user’s own comment | User should be able to edit the content of their comment and save changes | Success | Pass |
| **TC23** | Verify liking a comment | 1. Navigate to a post with comment  2. Click on the “Like” button next to a comment | Comment should be successfully liked | Success | Pass |
| **TC24** | Verify unlinking a comment | 1. Navigate to a post with comments where the user has already liked a comment  2. Click on the “Unlike” button next to the liked comment | Like should be successfully removed from the comment |  |  |
| **Test Friend Request** | | | | | |
| **TC25** | Verify sending a friend request | 1. Navigate to the profile of another user  2. Click on the “Add Friend” button | Friend request should be sent to the other user | Success | Pass |
| **TC26** | Verify accepting a friend request | 1. Navigate to the “friend Request” section  2. Accept the pending friend request | User should become friends with the sender of the friend request | Success | Pass |
| **TC27** | Verify declining a friend request. | 1, Navigate to the “Friend Requests” section  2. Decline the pending friend request | User should not become friends with the sender of the friend request | Success | Pass |
| **TC28** | Verify viewing friend list | 1. Navigate to the user’s profile or friend list  2. Click on the “Friends” tab or similar | List of friends should be displayed. | Success | Pass |
| **Test Responsive** | | | | | |
| **TC29** | Monile Device Access | 1. Open the app on various smartphones  2. Rotate devices to test landscape and portrait modes | Platform should adapt to different screen, screen sizes and orientations | Success | Pass |
| **TC30** | Tablet Access | 1. Access the platform using different tablets  2. Vary screen sizes and resolutions | Platform should provide a seamless experience on tablets | Success | Pass |
| **TC31** | Desktop Access | 1. Open the platform on different desktop or laptop computers  2. Vary screen sizes and resolutions | Platform should offer an optimized experience on desktops. laptops | Success | Pass |
| **TC32** | Browser Compatibility | 1. Access the platform on various web browsers  2. Test across different devices | Platform should render correctly and function consistently on all browsers | Success | Pass |

## 

## 6.3. Test Logs

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Test Result** | **Test Date** | **Comments / Note** |
| TC01 | Pass | March 20 2024 | None |
| TC02 | Pass | March 20 2024 | None |
| TC03 | Pass | March 20 2024 | None |
| TC04 | Pass | March 20 2024 | None |
| TC05 | Pass | March 20 2024 | None |
| TC06 | Pass | March 20 2024 | None |
| TC07 | Pass | March 20 2024 | None |
| TC08 | Pass | March 20 2024 | None |
| TC09 | Pass | March 20 2024 | None |
| TC10 | Pass | March 20 2024 | None |
| TC11 | Passs | April 5 2024 | None |
| TC12 | Passs | April 5 2024 | None |
| TC13 | Passs | April 5 2024 | None |
| TC14 | Passs | April 5 2024 | None |
| TC15 | Passs | April 5 2024 | None |
| TC16 | Passs | April 5 2024 | None |
| TC17 | Passs | April 5 2024 | None |
| TC18 | Passs | April 5 2024 | None |
| TC19 | Passs | April 5 2024 | None |
| TC20 | Passs | April 5 2024 | None |
| TC21 | Passs | April 5 2024 | None |
| TC22 | Passs | April 5 2024 | None |
| TC23 | Passs | April 5 2024 | None |
| TC24 | Passs | April 5 2024 | None |
| TC25 | Pass | Aoril 10 2024 | None |
| TC26 | Pass | Aoril 10 2024 | None |
| TC27 | Pass | Aoril 10 2024 | None |
| TC28 | Pass | Aoril 10 2024 | None |
| TC29 | Pass | Aoril 10 2024 | None |
| TC30 | Pass | Aoril 10 2024 | None |
| TC31 | Pass | Aoril 10 2024 | None |
| TC32 | Pass | Aoril 10 2024 | None |

# 7. Conclusion

In conclusion, the report delvesinto various aspects crucial for the development of the "Cuisine Hub" project, which aims to provide a comprehensive platform for culinary enthusiasts. Through an extensive literature review, the evolution and development of social networks were explored, providing valuable insights into the landscape within which Cuisine Hub operates. The comparison and selection of technologies and methodologies underscore the importance of making informed choices to ensure the project's success. Moreover, the analysis of requirements, encompassing various dimensions such as data, functionality, and development, lays the groundwork for a robust and effective system. The software design phase further refines the project's architecture, outlining key components and structures essential for implementation. The implementation section addresses technical challenges and solutions, emphasizing the practical aspects of translating design into reality. Finally, the evaluation phase encompasses testing and feedback, crucial for refining and enhancing the platform's usability and performance. Together, these components provide a comprehensive roadmap for the development and evaluation of Cuisine Hub, poised to offer a compelling solution for culinary enthusiasts worldwide.

# References:

1. Patrick Mutabazi, 2022. The Evolution of social media: How did it begin, and where could it go next? [Online]. Available at: <https://www.linkedin.com/pulse/evolution-social-media-how-did-begin-where-could-go-next-mutabazi/>. Accessed at: 10 October 2023.
2. Matt Bravo, 2023. The evolution of social media in 2023: A Paradigm shift in digital connectivity [Online]. Available at: <https://www.freniklabs.com/the-evolution-of-social-media-in-2023-a-paradigm-shift-in-digital-connectivity/> Accessed at: 11 October 2023.
3. Statista, 2023. Most people social networks worldwide as of July 2023, ranked by number of monthly active users [Online]. Available at: <https://www.statista.com/statistics/272014/global-social-networks-ranked-by-number-of-users/> (Accessed at: 11 October 2023)
4. Statista, 2023. Number of Threads sign-ups worldwide as of July 10,2023 [Online]. Available at: <https://www.statista.com/statistics/1398663/global-threads-users/> . Accessed at: 11 October 2023.
5. Rick, J., 2022. The Top 8 Best Languages for Backend Development [Online]. Available at: <https://www.couchbase.com/blog/backend-languages/> Accessed at: 11 October 2023.
6. Statistics & Data, 2023. Most Popular Backend Frameworks [Online]. Available at: <https://statisticsanddata.org/data/most-popular-backend-frameworks-2012-2023/> Accessed at: 11 October 2023.
7. GeekforGeek, (2024). Top 10 Most Popular JavaScript Frameworks for Web Development [Online]. Available at: <https://www.geeksforgeeks.org/most-popular-javascript-frameworks-for-web-development/> (Accessed at: 11 October 2023).
8. GeekforGeek, (2024). Functional vs Non-Functional Requirements [Online]. Available at: <https://www.geeksforgeeks.org/functional-vs-non-functional-requirements/> (Accessed at: 13 January 2023)