AN INFORMATION SYSTEM DOCUMENTATION

By

 Dedan Kimani & Nathaneal Mbale

148705, 151190

A Project Proposal Submitted to the Strathmore Institute of Management and Technology in partial fulfillment of the requirements for the award of a Diploma of Business and Information Technology

Diploma in Business Information and Technology

Strathmore University

Nairobi, Kenya

April 2023

# Declaration and Approval

I declare that this work has not been previously submitted and approved for the award of a Diploma in Business Information and Technology by this or any other University. To the best of my knowledge and belief, the proposal contains no material previously published or written by another person except where due reference is made in the proposal itself.

Student’s signature:

…………………………………. [*Signature*] ………………………………….. [*Signature*]

………………………………….………….. [*Date*]

**Approval**

The Information System Project proposal of ***Dedan Kimani and Nathaneal Mupoy*** was reviewed and approved *(for examination)* by:

Supervisor’s signature:

………………………………….………….. [*Signature*]

………………………………….………….. [*Date*]

# Abstract

This documentation focuses on the design and implementation of a web-based community application for students in a university. The application aims to provide a communication platform for clubs and societies within the university, addressing the challenges of overloaded emails, lack of school event reminder features in existing systems, and a variety of communication channels. The methodology used in this project is agile, involving requirements gathering, system design, development, testing, and deployment. The system design includes use cases, entity relationship diagrams, system sequence diagrams, architectural diagrams, and wireframes. Two modules were proposed for the application: a student module and an administrator module. The system analysis and design phase involved requirements analysis, system architecture, and object-oriented diagrams, such as use cases, entity relationship diagrams, system sequence diagrams, class diagrams, and wireframes. The system implementation and testing phase focused on the implementation of the system and testing paradigms and test cases.

# Acknowledgment

We would like to appreciate our lecturers in the course who have helped us gain knowledge to implement what we have learned in the Project and our supervisor for investing as much time as they could support our work to reach the requirements and exceed our expectations.

# Table of Figures

[Figure 2. 1 (Strathmore Email Guildeline) 5](#_Toc133268313)

[Figure 2. 2 (Conceptual Framework) 9](#_Toc133268314)

[Figure 3. 1 (Agile development Process) 11](#_Toc133268317)

# List of tables

[Table 5.1 (Post creation test case) 38](#_Toc133269758)

[Table 5. 2 (Community creation test case) 39](#_Toc133269759)

[Table 5. 3 (Test case for signing up a use) 39](#_Toc133269760)

# Table of Content

Table of Contents

[Declaration and Approval ii](#_Toc133269684)

[Abstract iii](#_Toc133269685)

[Acknowledgment iv](#_Toc133269686)

[Table of Figures v](#_Toc133269687)

[List of tables vi](#_Toc133269688)

[Table of Content vii](#_Toc133269689)

[Chapter 1: Introduction 1](#_Toc133269690)

[1.1 Background 1](#_Toc133269691)

[1.2 Problem Statement 2](#_Toc133269692)

[1.3 Aim 2](#_Toc133269693)

[1.4 Specific Objective 2](#_Toc133269694)

[1.5 Justification 3](#_Toc133269695)

[1.6 Scope 3](#_Toc133269696)

[Chapter 2: Literature View 4](#_Toc133269697)

[2.1 Introduction 4](#_Toc133269698)

[2.2 Strathmore University Communication Process In clubs and Societies 4](#_Toc133269699)

[2.2.1 The communication process between the conversational parties 4](#_Toc133269700)

[2.2.2 The conversational parties 5](#_Toc133269701)

[2.2.3 The Applications Used 5](#_Toc133269702)

[2.3 Challenges of Communicating with Students about clubs and societies 5](#_Toc133269703)

[2.3.1 Overloaded Emails 6](#_Toc133269704)

[2.3.2 No School Event Reminder Feature Integration In existing Systems 6](#_Toc133269705)

[2.3.3 Variety of Communication Channels 6](#_Toc133269706)

[2.4 Review of existing systems at Strathmore University 7](#_Toc133269707)

[2.4.1 Email 7](#_Toc133269708)

[2.4.2 WhatsApp 7](#_Toc133269709)

[2.4.3 Slack 8](#_Toc133269710)

[2.5 Strathmore University Gaps in existing systems 8](#_Toc133269711)

[2.6 Conceptual Framework 9](#_Toc133269712)

[Chapter 3: Methodology 10](#_Toc133269713)

[3.1 Introduction 10](#_Toc133269714)

[3.2 Research Approach 10](#_Toc133269715)

[3.3 Agile Methodology 10](#_Toc133269716)

[3.3.1 Requirements Gathering for the System 11](#_Toc133269717)

[3.3.2 Quick System Design 12](#_Toc133269718)

[3.3.3 System Development 12](#_Toc133269719)

[3.3.4 System Testing 12](#_Toc133269720)

[3.3.4 System Deployment 12](#_Toc133269721)

[3.4 System Design 13](#_Toc133269722)

[3.4.1 Use Cases 13](#_Toc133269723)

[3.4.2 Entity Relationship Diagrams 13](#_Toc133269724)

[3.4.3 System Sequence Diagrams 13](#_Toc133269725)

[3.4.4 Architectural Diagram 14](#_Toc133269726)

[3.4.5 Wireframes 14](#_Toc133269727)

[3.5 Implementation Tools 14](#_Toc133269728)

[3.6 Proposed Modules and System Architecture 14](#_Toc133269729)

[3.6.1 Student Module 14](#_Toc133269730)

[3.6.2 Administrator Module 14](#_Toc133269731)

[Chapter 4: System Analysis & Design 15](#_Toc133269732)

[4.1 Introduction 15](#_Toc133269733)

[4.2 Requirements Analysis 15](#_Toc133269734)

[4.2.1 Functional Requirements 15](#_Toc133269735)

[4.2.2 Non-Functional Requirements 17](#_Toc133269736)

[4.3 System Architecture 18](#_Toc133269737)

[4.4 Object Oriented Diagrams 19](#_Toc133269738)

[4.4.1 Use Cases 19](#_Toc133269739)

[4.4.2 Entity Relationship Diagrams 20](#_Toc133269740)

[4.4.3 System Sequence Diagrams 21](#_Toc133269741)

[4.4.4 Class Diagram 23](#_Toc133269742)

[3.4.5 Wireframes 24](#_Toc133269743)

[Chapter 5: System Implementation and Testing 27](#_Toc133269744)

[5.1 Introduction 27](#_Toc133269745)

[5.2 Testing paradigms 27](#_Toc133269746)

[5.3 System Implementation 29](#_Toc133269747)

[5.4 Test cases 38](#_Toc133269748)

[5.4.1 Post creation test case 38](#_Toc133269749)

[5.4.2 Community creation test case 39](#_Toc133269750)

[5.4.3 Test case for signing up a user 39](#_Toc133269751)

[Chapter 6: Conclusion Recommendation and Future works 40](#_Toc133269752)

[6.1 Introduction 40](#_Toc133269753)

[6.2 Recommendation 40](#_Toc133269754)

[6.3 Future works 40](#_Toc133269755)

[References 41](#_Toc133269756)

[Appendix A: Gantt chart 45](#_Toc133269757)

# Chapter 1: Introduction

## 1.1 Background

At Strathmore University students apply for the various clubs and societies offered by the university. Which include football, basketball, rugby, archery, chess, and scribble, a French, Spanish, and Japanese club. The school offers these numerous activities to encourage self-discovery and development, nurturing service-oriented leadership. Clubs and societies are a platform for student personal development and collaboration with both Strathmore university departments and corporates' future mutual benefit.

But communication at Strathmore University affects students who wish to participate in extracurricular activities, such as additional courses, clubs, communities, or sports activities. The school office primarily uses email to communicate with students, and for students and lecturers to interact with each other, but this is a problem due to email overload. This issue is compounded by the fact that lecturers and staff members also send a high volume of emails, which can lead to email overload and unattended messages (Wells, 2018) (vdovin, 2020).

A community web-based system can solve the communication issue at Strathmore University by providing a centralized platform for communication for students. This platform will be able to notify students about upcoming events, deadlines for extracurricular activities, and other important announcements. Overall, a community web-based system can help improve communication at Strathmore University with a centralized platform for communication that is efficient and easily accessible.

The students at Strathmore University could suffer some consequences if the problem of inefficient communication is not addressed. Students who desire to join in extracurricular activities may miss out on valuable possibilities as a result of inefficient communication. Students may become frustrated and feel excluded as a result, which could have a bad effect on how they see the university in general. Additionally, students may experience email overload, which can cause them to lose track or overlook critical messages, leading to missed deadlines or opportunities to enroll in a club or society or an event (vdovin, 2020).

Additionally, if the university does not implement a community web-based system to handle the communication problem within the clubs and society. Students might start to perceive the university as unreliable or unorganized, for instance, which could harm the school's reputation. Long-term, this can result in lower student satisfaction and a drop in enrollment. Because of this, the university needs to take proactive measures to enhance communication because doing so can help minimize these undesirable effects and ultimately enhance the experience of students while attending the institution (MacDonnell, 2018).

## 1.2 Problem Statement

Strathmore University faces a significant communication issue that affects students' engagement in clubs and societies. The current system relies mainly on email, which has proven to be ineffective in providing timely and relevant information to the university community about the various clubs and societies. As a result, students miss out on important information such as registration deadlines for extracurricular activities, which ultimately leads to reduced engagement and participation in such activities (Houlis, 2022).

The problem is further compounded by email overload, which affects students and the staff managing extracurricular activities. This leads to a situation where some emails go unattended, and lecturers struggle to manage their inboxes, making it difficult for them to respond to student inquiries on time (vdovin, 2020). The lack of an efficient communication system ultimately hinders the achievement of the university's goals of promoting student engagement and participation in extracurricular activities (Clubs and socities, 2023). To address this problem, a centralized and user-friendly platform that can provide timely and relevant information to the university community, reducing email overload, and enhancing engagement and participation in extracurricular activities was created.

## 1.3 Aim

To create a web-based system to improve the engagement of students in clubs and societies at Strathmore University.

## 1.4 Specific Objective

1. To review the communication and student engagement in clubs and societies.
2. To evaluate the challenges faced in student engagement in clubs and societies.
3. To review existing systems used for students to engage in clubs and societies.
4. To design and develop a web-based system to improve student engagement in clubs and societies at Strathmore University.
5. To test the developed web application system using component, system and acceptance testing.

## 1.5 Justification

To address the issue of ineffective communication that affects students who want to participate in extracurricular activities, a web-based system was created for the underlined problems. The prevalent use of email in today's communication can result in email overload, unread messages, and missed deadlines, which can be extremely problematic for students (vdovin, 2020). The web-based solution will improve the student experience by giving them convenient access to crucial notifications, deadlines, and upcoming activities by offering a centralized platform for communication.

All community members at Strathmore University stand to gain from this engagement in many ways. Fewer emails will be sent out, which will prevent students from missing or ignoring vital communications. The ability to communicate with students in a more effective and structured manner would be advantageous to instructors and students as well. The web-based solution will increase the university's overall communication efficiency, improving student satisfaction, extracurricular activity participation rates, and academic success. In conclusion, Strathmore University's adoption of a web-based system is an essential step in enhancing communication and enhancing the overall student experience.

## 1.6 Scope

By using a website, Strathmore University is able to address the issue of insufficient communication with students regarding clubs and societies, which has an impact on their involvement to participate. The website will primarily focus on Strathmore University students and instructors collaborating with the school, no outside parties will be involved. With the use of tools such as GitHub which aided the collaboration in coding between programmers together with techniques and methods to which are procedures used for designing, developing, documenting and maintaining programs. The limitation that came with using a tool like GitHub is that it is difficult for beginners because the various things that can be done on the platform making it confusing and complicated (Clouds, 2021).

# Chapter 2: Literature View

## 2.1 Introduction

This This chapter explains in detail the process of communication between the clubs and societies and the students and its evolution, the challenges of communicating about clubs and societies with students, a review of related communication channels used to communicate with the students about clubs and societies, gaps in existing communication channels, and a description of the conceptual framework for the proposed web-based system5 for communication between people of the same club or society which includes a well-structured diagram.

## 2.2 Strathmore University Communication Process In clubs and Societies

The following processes are used to demonstrate the communication process in clubs and societies at Strathmore University. They include rules and guidelines, a communication process, conversational participants, and the application implemented. The identified information will be discussed in the following subtitles.

### 2.2.1 The communication process between the conversational parties

Communication at Strathmore University is implanted widely by the use of emails. With this, there are certain rules and regulations to be followed. An email is always sent and received formally. It must always contain the student's admission number, and both first name and surname to identify themselves and their titles if lecturers or administrators. The sender should always have a greeting to the receiver. A subject line and body are also important which explains why someone is sending their email. An example that illustrates the guidelines at Strathmore University which was provided by Strathmore University is as follows:

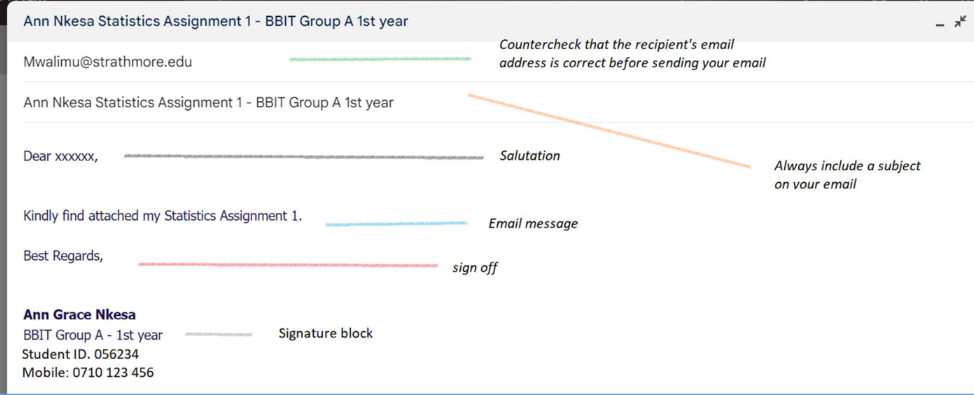


Figure 2. 1 (Strathmore Email Guildeline)

### 2.2.2 The conversational parties

The people involved in the communication around Strathmore University comprises Students, Lectures, and Instructors who are involved in the clubs and societies. These parties need to communicate to build connections with each other and enable them to express their needs and experiences surrounding the clubs and societies community. It gives them the chance to communicate their views, share information and express their emotions (professionals, 2022).

### 2.2.3 The Applications Used

The applications implemented at Strathmore University used to communicate about these clubs and societies include Email, WhatsApp, Slack, and Zoom. Each of these is used for recruiting new members, announcing events, and communicating with each other in the same club or society. Supporting the communication between the University and students enables collaboration among applicants in the University such as Students and Instructors (Staff, 2019) (Young, 2021).

## 2.3 Challenges of Communicating with Students about clubs and societies

The next subtitles discuss the communication challenges and consequences in the University externally and internally. These include overloaded emails, language barriers, unorganized communication, and unscheduled events.

### 2.3.1 Overloaded Emails

During communication, while using email the application can overload. This means too many emails have been sent and haven't been attended to by a receiver. With this there are consequences. Such as unattended emails because of the numerous emails being sent in one spot (vdovin, 2020).

There are signs of overloaded emails which are according to research by David Campbell whereby a person constantly checking their emails, deleting more emails than you actually need, spending more time organizing and sorting your emails than actually responding or reading, feeling behind and can’t catch up (Campbell, 2022).

There are certain consequences to having an overloaded email towards people. Such as having demotivation to read the emails, follow up on them, or prioritize other things that can be done in a short period than going through serval emails (Campbell, 2022).

### 2.3.2 No School Event Reminder Feature Integration In existing Systems

Due to numerous different events, there is no reminder feature in the existing systems. To remind students about open spots in some clubs and societies. Due to this some students can miss these open stops offered and must wait until there are slots. Causing frustration and a loss of interest. Events and occasions are also missed on the calendar as a result of a busy work schedule (gegok12, 2022).

### 2.3.3 Variety of Communication Channels

This means communication-based on clubs and societies is spread across different channels which leads to unorganized communication as they can't be kept on track. Because of this, there are a lot of drawbacks to communication. Communication is spread in many different ways at Strathmore and there isn't a centralized area for all students to communicate effectively at Strathmore. Since there are several communication channels implemented at Strathmore which include Email, WhatsApp, Slack, and Microsoft Teams. With these communication tools, some cons are present these cons consist of; There may be too much information, you could get information overload if you have too many channel subscriptions, and separating personal life from work is challenging (depending on the tool you are using) According to Lisette on a Podcast (Superpowers, 2018).

## 2.4 Review of existing systems at Strathmore University

### 2.4.1 Email

Email, sometimes known as electronic mail, is a digital messaging service that enables users to send and receive messages online. Since its initial introduction in the 1970s, it has grown to be among the most popular means of communication in the world. Users can transmit documents, pictures, text messages, and other types of files to one or more recipients through email. Email is accessible through a number of email service providers, including Apple's iCloud, Microsoft Outlook, and Google's Gmail (vdovin, 2020).

You must first set up an email account with a service provider, such as Google, Yahoo, or Microsoft, to utilize email. You can enter your inbox, create a new message, and send it to the recipient's email address after you have an account. Email is a flexible communication method since messages can include text, photos, and other attachments. The problem of spam or undesired communications, which can clog up your inbox and make it difficult to identify critical messages, is one of the biggest obstacles to using email. Miscommunication is another issue because it can be challenging to express tone and context in written words. In addition, email can occasionally be exposed to security risks like phishing attempts, which can compromise critical data (vdovin, 2020).

### 2.4.2 WhatsApp

In addition to text messages, voice, and video conversations, and file sharing with both individuals and groups, WhatsApp is a cross-platform messaging application. Jan Koum and Brian Acton, two ex-Yahoo workers, introduced the app in 2009. Facebook purchased the app in 2014 for $19 billion. One of the most popular messaging apps in the world, WhatsApp had over 2 billion monthly active users as of February 2021 (Ceci, 2023).

Users must download the WhatsApp app from the Google Play Store or Apple App Store and register an account using their phone number to access the service. They can begin chatting with their contacts who have WhatsApp installed on their devices after their account has been set up. Only the sender and recipient can view the messages because the app uses end-to-end encryption to preserve user privacy. The program has drawn criticism for its data sharing and privacy restrictions, though. The fact that WhatsApp is owned by Facebook has sparked concerns about data sharing and privacy, particularly given Facebook's history of gathering and disclosing user data, according to a 2021 article by Wired (NEWMAN, 2021).

### 2.4.3 Slack

Slack is a real-time communication and file-sharing tool for teams that are hosted in the cloud. It was introduced in 2013 and has since grown to be one of the most well-liked communication platforms for remote teams. Users of Slack can establish channels for various subjects, tasks, or teams and communicate via text, voice, or video conversations. The platform also enables file sharing, app integration, and task automation using bots (Knapp, 2021).

Users need to set up a workspace and create an account with their email address to use Slack. After setting up their workplace, they may invite their team members to join and begin chatting. The platform is accessible via desktop or mobile apps, and web browsers and both free and premium plans are available. Additionally, Slack offers a variety of customization choices, like themes, emoji’s, and shortcuts, to let users tailor their experience (Creative, 2019).

## 2.5 Strathmore University Gaps in existing systems

The difference between adopting a web-based system and messaging apps like email, WhatsApp, and Slack to improve poor communication at Strathmore University can be substantial and depends on several different aspects. Here are some possible variations:

Purpose: The web-based system's main function is usually to manage participants in a club or society, allow communication between users of the same community, and register or admit to a certain group. Email, WhatsApp, and Slack are just a few of the messaging apps that are more adaptable and may be used for a variety of communication styles.

Features: Web-based systems may include elements designed expressly for the clubs and societies, like an event calendar and awards record. These exact functionality may not be available in messaging applications, which instead provide basic messaging features like text, image, and file sending.

Access: Certain individuals, such as students, and instructors, may only be able to access a web-based system with specified login information such as their school admissions numbers and ids. Wider audiences may find it easier to use messaging apps like email, WhatsApp, and Slack, yet access to certain discussions or groups may still be limited.

## 2.6 Conceptual Framework

A conceptual framework is a detailed diagram demonstrating how a system works that includes all processes involved in the system. The system will contain three major entities, this includes a student, an administrator, and a database system. A student should log into the website using their Strathmore email and should be able to join a community and should be able to make posts or delete their posts on the community they have joined. The administrator should be able to sign in to the site and to be able to create or delete a community or a post. The users' (administrator and students) information is encrypted and stored in a database. The figure below shows the conceptual framework of the proposed web-based system.

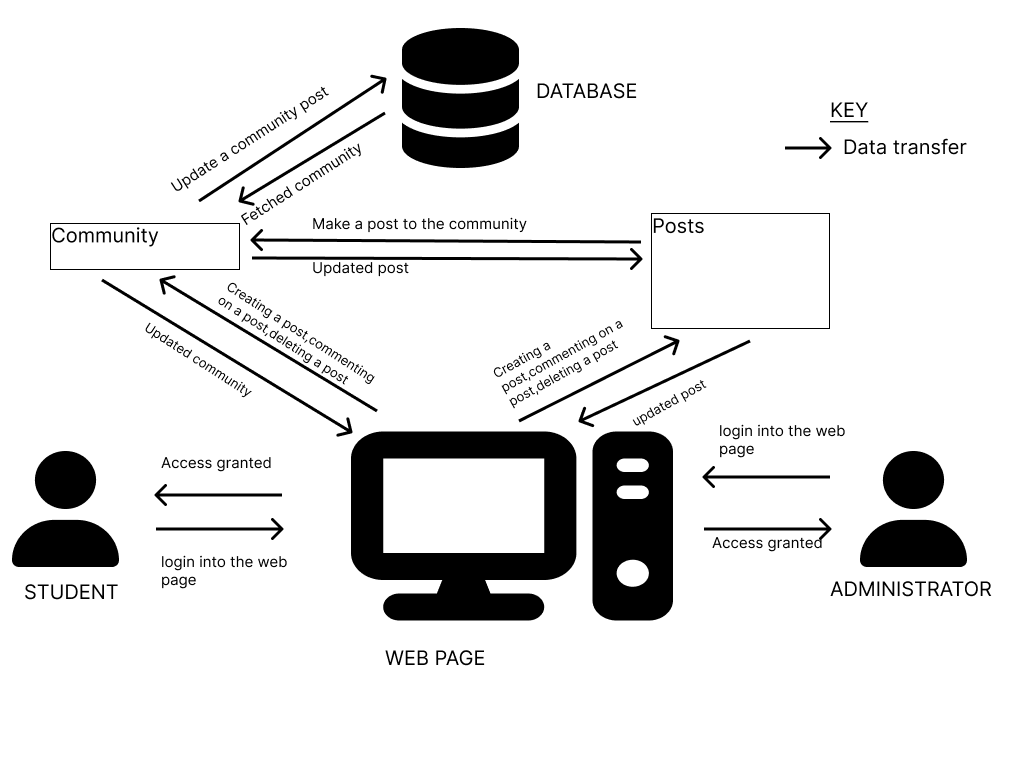


Figure 2. 2 (Conceptual Framework)

# Chapter 3: Methodology

# 3.1 Introduction

This chapter entails the research approach that was chosen and implemented which was object-oriented programming. The system methodology Approach that was used is called agile with its steps of requirements gathering, quick design, development, testing, and deployment. That will build the community web application to meet user requirements and its functionalities that are included in the chapter. The chapter also involves describing modules on how users interact with the system. Together with a Gantt chart that illustrates the duration of activities being performed.

# 3.2 Research Approach

The actions that must be taken to guarantee a project's success would be specified using an object-oriented programming (OOP) methodology by building a class hierarchy that reflects the various project phases. The Project class, which would be at the top of the hierarchy, would have attributes like the project name, start and finish dates, budget, and other pertinent data. This class would also have methods that outline the project's major goals as well as the key actions needed to attain those goals. There would be a number of subclasses representing the various project phases, such as Planning, Design, Development, Testing, and Deployment, below the Project class. Each of these subclasses would include specialized methods and attributes that specify the tasks and products connected with that specific project phase. It is simpler to structure and manage complicated projects utilizing an OOP approach, as well as to adjust and update the project plan as necessary. A more effective project outcome can be attained by ensuring that all stakeholders understand the project's goals and the processes needed to achieve them. This can be achieved with the aid of an OOP approach (Half, 2023).

# 3.3 Agile Methodology

Agile is a project management approach that places a strong emphasis on adaptability, teamwork, and iterative development. Agile is founded on the idea that responding to change is more important than adhering to a plan, in contrast to traditional project management techniques that emphasize drafting a thorough plan in advance and carrying it out to completion. Agile teams, then, work in brief iterations, or "sprints," and frequently solicit input to improve their output. Agile encourages teamwork and open communication with stakeholders, which speeds up problem-solving and decision-making. Faster delivery, better quality, more customer happiness, and the capacity to adjust to shifting requirements and market situations are all advantages of Agile. Agile is a very effective methodology for developing projects that satisfy client needs and change the environment (Paterska, 2021). The figure below shows the agile development process.



Figure 3. 1 (Agile development Process)

(Agile , n.d.)

### 3.3.1 Requirements Gathering for the System

The Agile development process started with the requirements gathering phase where requirements for the proposed system will be collected by the development team. This involves precisely defining the system features which are generated by interviewing the system users. It is done to gather information about what they expect from the proposed system. This phase also involves developing a System Requirements document that defines all requirements of the system for the users (Holmes, 2023). The proposed web-based system involves interviewing students in clubs and societies and potential members who which to join a club or society to gather the requirements of the new system.

### 3.3.2 Quick System Design

This stage was the creation of the system's fundamental design. By creating a wireframe design which is a method for presenting a system's design and organizational structure visually. Users will receive a brief overview of the proposed system thanks to this design. Integrating a wireframe will insure users’ needs are met and will be clear and quick (HANNAH, 2022). Since the development team will use feedback from students and instructors, who want to participate in clubs and societies to identify areas for improvement, this design will serve as the foundation for establishing additional needs for the system.

### 3.3.3 System Development

This phase involved developing a web application system from the quick design and modifying it based on the feedback gathered in the quick design phase from the clubs and societies that wish to join. To create a web-based system, the development team will alter the quick design and the requirements gathered. A web application system will be flexible across various devices, and good at protecting data for its users (Khamooshi, 2019). This system will we built to meet the requirements and functionality of the system users. The proposed system will incorporate java script as the programming language as it simplifies responsive design and enhances user’s experiences (Ozanich, 2022). With an environment IDE accompanying it called visual studio code. Because it eases editing, building and debugging (Why did we build Visual Studio Code?, 2023).

### 3.3.4 System Testing

To determine whether the user criteria had been met, a set of tests will be performed on the community web-based system. Three levels of testing will be performed: component testing, which will test individual components; system testing, which will test the system as a whole; and acceptance testing, which will use customer requirements to determine whether the system satisfies their needs and fix any flaws (Hamilton, 2023).

### 3.3.4 System Deployment

The final stage of the agile development stage is the deployment stage which means that once the app is ready. It can be released by the development team to the cloud or an on-pretmise server. This indicates that the product is available to customers and in use. Software release, installation, testing, deployment, and performance monitoring are some of the most frequent software deployment procedures.

## 3.4 System Design

System design defines the modules, interfaces, and data for a system to satisfy specified requirements. It is the application of a systems template to the actual system development. In this phase, the system requirements and logical description of the entities, relationships, and attributes of the data that are documented in the Requirements Analysis phase will be further refined into a final community web application system. Object Oriented Analysis and Design diagrams will be in its design for the community web application. This includes a use case diagram, Entity Relationship diagram, activity diagram, sequence diagram, architectural and wireframes.

### 3.4.1 Use Cases

The diagram shows how the users interact with the community web application. This includes the flow of processes and how each process interacts with an actor in the system (GeeksForGeeks, 2021).

### 3.4.2 Entity Relationship Diagrams

The relationships between entity sets contained in a database are displayed in an entity relationship diagram (ERD). In this context, an object or piece of data qualifies as an entity. A group of related entities is known as an entity set. The attributes of these entities can specify their characteristics. An ERD diagram exemplifies the logical structure of databases, in this case, the community web application, by specifying the entities, their properties, and the interactions between them (GeeksforGeeks, 2020).

### 3.4.3 System Sequence Diagrams

A system sequence diagram is a visual representation of the events that external actors produce, their logical sequence, and potential inter-system interactions for a scenario or use case. System sequence diagrams provide a visual representation of each of the community web application's use cases (GeeksForGeeks, Unified Modeling Language (UML) | Sequence Diagrams, 2022).

### 3.4.4 Architectural Diagram

The method of specifying a group of hardware and software components, as well as their interfaces, in order to create the foundation for the creation of a computer system (GeeksForGeeks, Software Engineering | Architectural Design, 2022).

### 3.4.5 Wireframes

A method for presenting a system's design and organizational structure visually. Integrating a wireframe will insure users’ needs are met and will be clear and quick (HANNAH, 2022).

## 3.5 Implementation Tools

1. VS Code IDE - This is the Visual Studio Integrated Development Environment (IDE) that was used to develop the system.
2. Java Script - This is the scripting language that was used to code the system.
3. CSS - Cascading Style Sheets which helped design and style the community web application system.
4. GitHub - A platform for collaboration and version control of code
5. Drawa.io- To aid draw software diagrams in object oriented programming

# 3.6 Proposed Modules and System Architecture

### 3.6.1 Student Module

This module gives access to the system services to the user. A student can register or log in to the system. This will then let them post and participate in topics and conversations around the interests of the community. With permission, one can join or create a group and also view events and add reminders to coming up events. Users can share images, movies, and other types of media relevant to the interests of the group in this module.

### 3.6.2 Administrator Module

They are responsible for making posts on the web page for major events, deleting posts, deleting comments, creating communities and deleting communities. The administrator can view information and analysis on user behavior, engagement with the community, and other metrics that can be utilized to raise the performance of the community.

# Chapter 4: System Analysis & Design

## 4.1 Introduction

This text outlines the functional and non-functional requirements for a web-based community application, as well as the system architecture and object-oriented diagrams. To help visual the system and also look if the underlined requirements are meet or are underlined in the diagrams to know how the system functionalities work.

## Requirements Analysis

### 4.2.1 Functional Requirements

Functional requirements are to specify the requirements for a product, including its features and functions (AltexSoft, 2021). The functional requirements for the community web application include:

For a web community, the main administrator should have various functional requirements to manage the platform effectively. Firstly, the main admin should be able to create posts, edit or delete them as needed. Secondly, the main admin should have the ability to create a new community on the platform and delete communities that are no longer needed. Additionally, the main admin should have the power to delete comments posted by users, and to create comments in response to user posts. They should also be able to promote users to community admin status and remove them as community admins if needed. Lastly, the main admin should be able to delete user accounts from the platform as necessary to maintain community standards. These functional requirements are essential for the main admin to effectively manage and moderate the web community.

For a web community, community admins should have various functional requirements to manage their personal accounts, join and leave communities, create and delete posts and comments, and receive and delete notifications from the community. Firstly, community admins should be able to create an account on the platform and log in with their credentials. They should also have the ability to delete their account if necessary. Secondly, community admins should be able to create a community and delete the community they made. They should also have the option to join communities that interest them and unjoin communities they no longer wish to be part of. Thirdly, community admins should receive notifications from the community they joined and be able to delete them as necessary. Finally, community admins should have the ability to create posts, delete posts they made, create comments on people's posts and their own, and delete comments they have made or that others have made. These functional requirements are critical for community admins to effectively manage their accounts, interact with the community, and engage in meaningful discussions.

For a web community, students should have various functional requirements to manage their personal accounts, join and leave communities, interact with the community, and receive and delete notifications. Firstly, students should be able to create an account on the platform and log in with their credentials. They should also have the ability to delete their account if necessary. Secondly, students should have the option to join communities that interest them and unjoin communities they no longer wish to be part of. Thirdly, students should receive notifications from the community they joined and be able to delete them as necessary. Additionally, students should have the ability to make comments on posts and delete comments they have made. These functional requirements are essential for students to effectively participate in the community and engage in meaningful discussions with other members.

The server is responsible for connecting the database to the web page making sure requests coming through it are correct, authentifying the users and encrypting the data sent to the database.

The database is an organization of related data, also referred to as structured data .The database will store the user’s information and the posts that they make.

### 4.2.2 Non-Functional Requirements

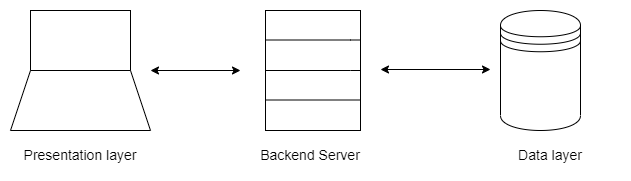
Non-functional requirements enumerate the system's fundamental characteristics. They are sometimes referred to as qualities (AltexSoft, 2021). The functional requirements for the community web application include:

Authentication and authorization: Before granting access to sensitive data or functionality, the system should demand user authentication. To prevent illegal access, the system should additionally impose access rules based on user roles and permissions (Point, 2021). For example a student and an administrator won’t access the community web application the same way. The administrator will require additional safety measures as they control the system and hold important information about the system users.

Encryption: Encryption should be used by the system to safeguard sensitive information like user passwords. It is important to implement encryption to enhance the security of communication between client apps, web-applications and servers that helps to protect sensitive data and private information for system users who are students and instructors (File, 2021).

Usability: The community application system can quickly traverse a usable application's interface. Additionally, users can comprehend how the application arranges its content and are aware of how to access sites like the settings page. Users can also understand a feature's purpose and capabilities with ease. They might, for instance, expect that pressing a button with an image of a magnifying glass will activate a search bar (Team, 2022).

## System Architecture



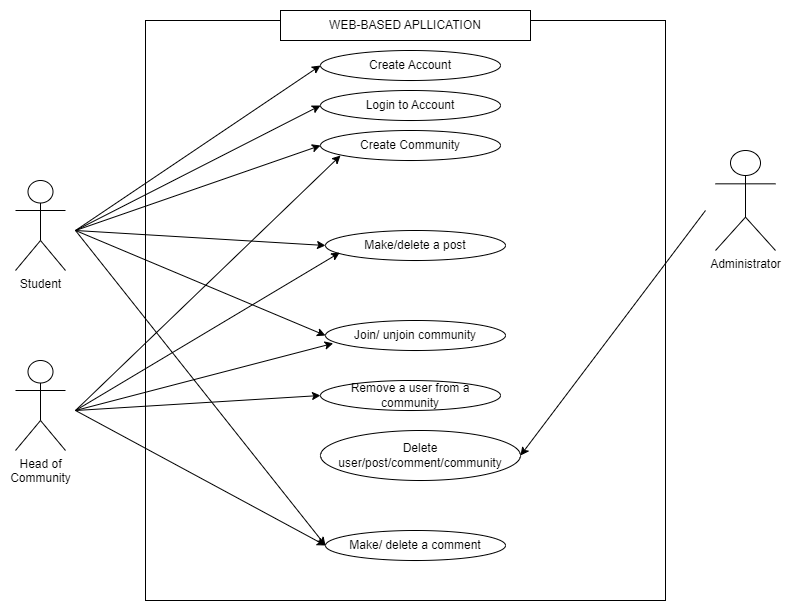
This is the layer where a user can make request to the server which can be accepted or rejected. This are can be a post request to the posts server, community server or user’s server.

This is the layer connects the data layer and the presentation layer this layer is responsible for accepting or rejecting request coming from the presentation layer ,this is where a user is verified ,passwords are encrypted and tokens are generated for the user. Then requests to the server are also verified to be coming from the correct user.

This is the database which is responsible for storing user’s information in three different entities that is posts, community and user.

## Object Oriented Diagrams

### Use Cases

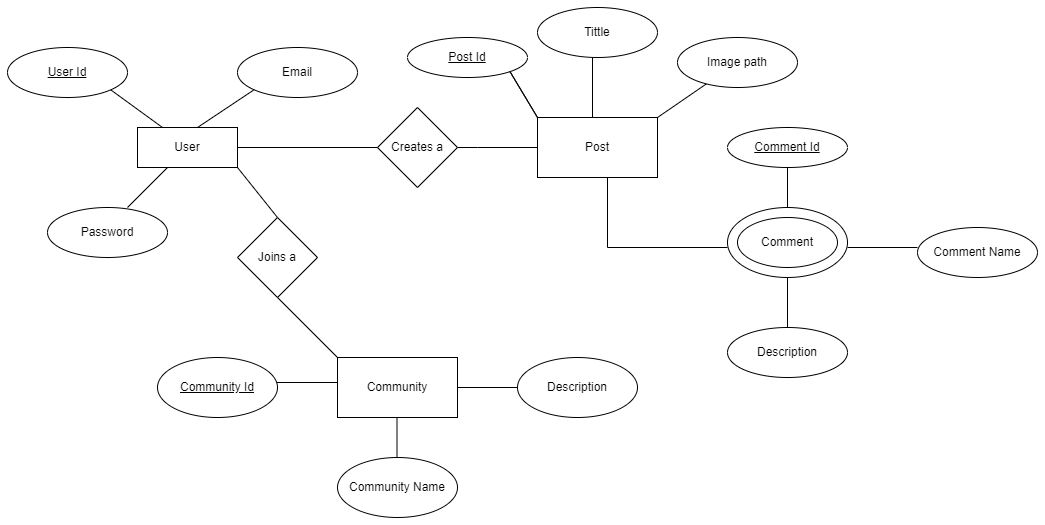


A student should be able to create an account, then use the account to log into the web page. Student can choose which community they want to join and also unjoin the ones they are part of a student can make a post to a community they are part of and they can delete the post they have made .Moreover , they can comment on people’s posts and delete the comment they have made.

A head of community can creating a community, join/unjoin a community and make a post on a community .Moreover they are able to delete people’s posts on the community and also should be able to add or delete users to the community.

An admin should be able to delete users account, post, comments and communities.

### Entity Relationship Diagrams

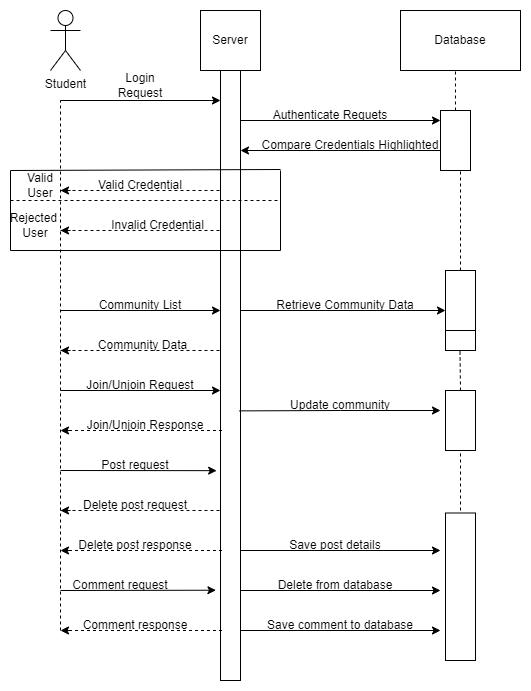


A user is identified by a unique value on the database that is the user\_id, every user also has an Email and a password which is unique to the user.

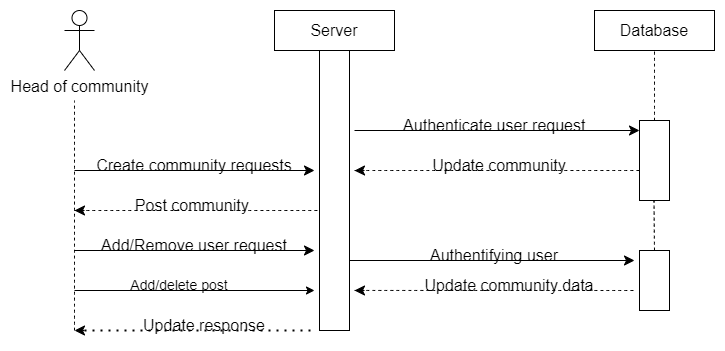
A user is able to join a community which has a unique identifier community ID which is used to search for a community, on the other hand it has a community name and a description about the community.

A user can make a post which has a unique identifier post ID, title, image Path which is optional on the post a comment can be made which take a comment ID which is the post ID, a comment title and a comment description.

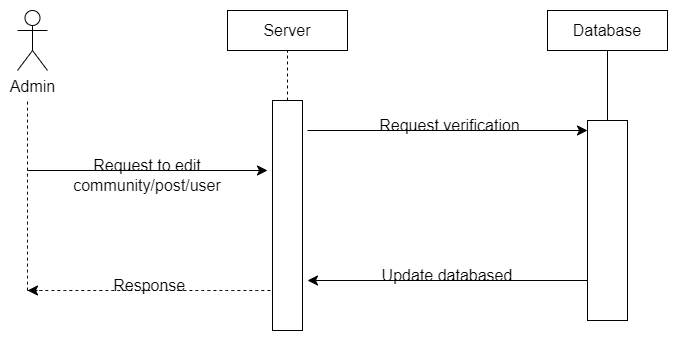
### System Sequence Diagrams



A student should be able to create an account, then use the account to log into the web page .When log in the user information must be authentified, using the email and the user token, then the credentials are compared and an error is prompted to the user if he the credentials are invalid; access is granted if the credentials are correct. A list of communities is displayed to the user where the student can choose which community they want to join and also unjoin the ones they are part of .A student can make a post to a community they are part of and they can delete the post they have made. Moreover, they can comment on people’s posts.

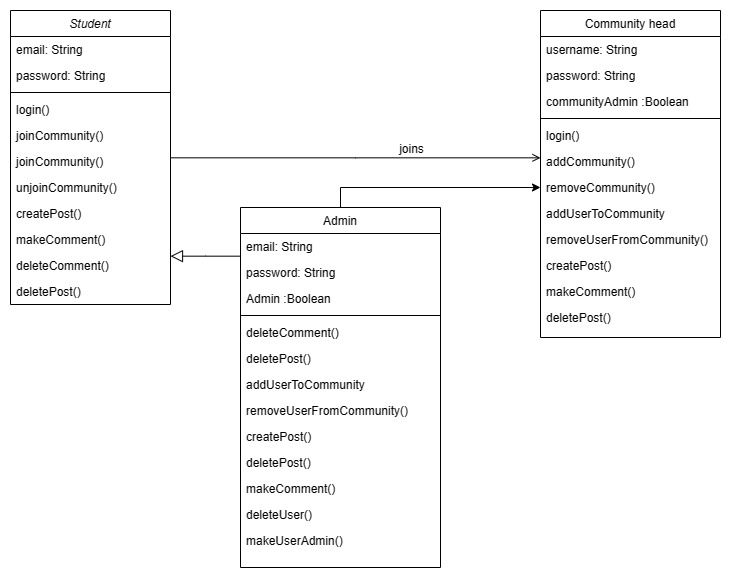


A head of community is an admin capable of creating a community and making post on that community .Moreover they are able to delete people’s posts on the community and also should be able to add or delete users to the community.



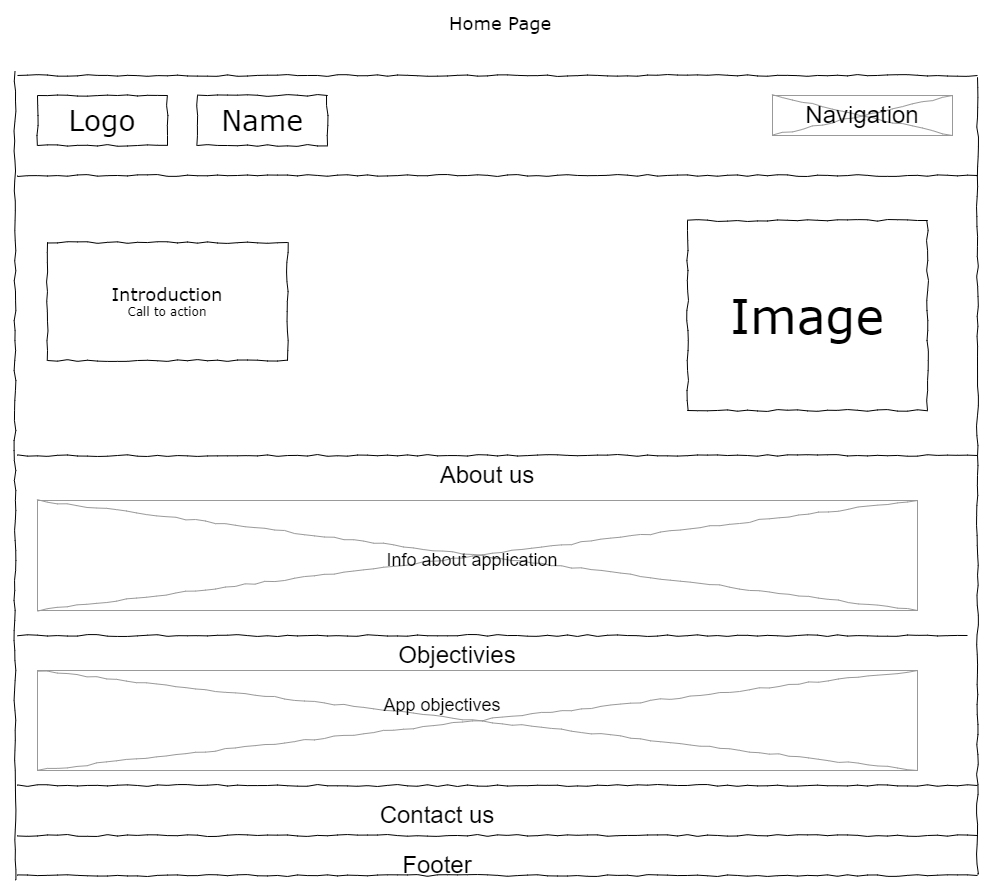
An admin should be able to delete users account, post, comments and communities.

### Class Diagram

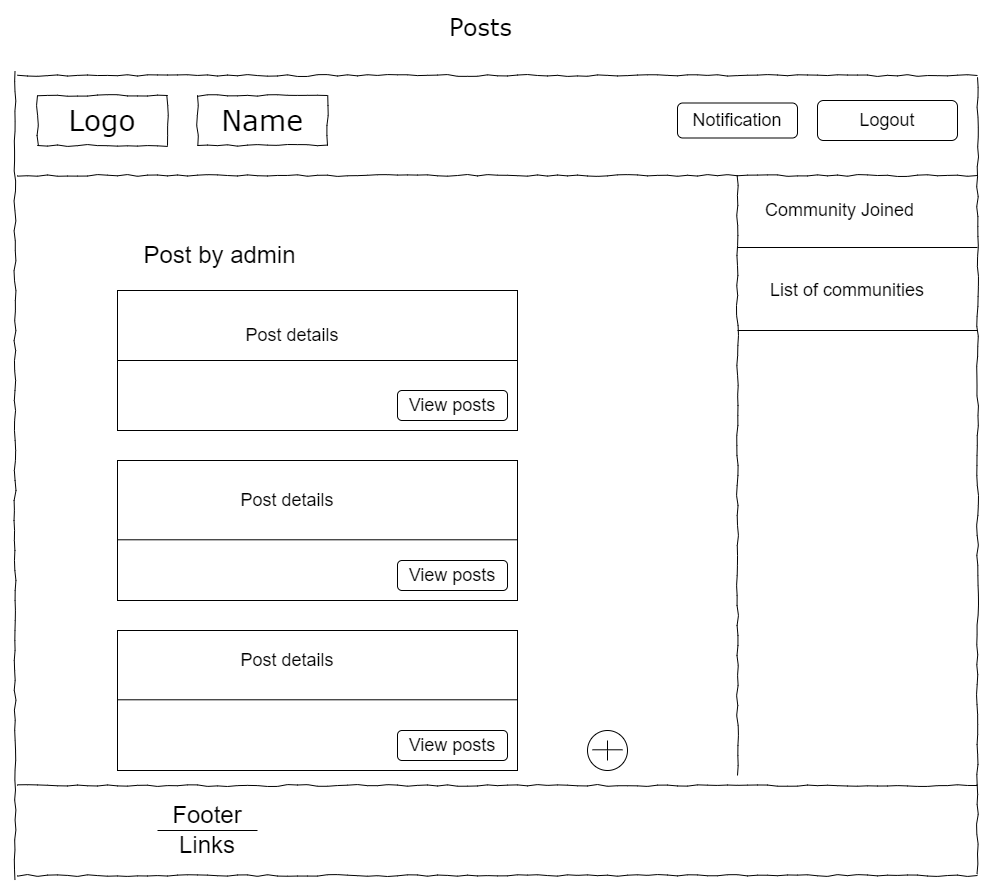


This class diagram describes the relationships between three classes: Student, Community Head, and Admin. Each class has several attributes and methods. The Student class has attributes such as email and methods such as joinCommunity() and createPost(). The Community Head class has attributes such as username and methods such as removeUserFromCommunity(). The Admin class has attributes such as email and methods such as makeUserAdmin(). Each class has its own set of methods for creating, deleting, and managing posts, comments, users, and communities. The relationships between the classes allow for effective management of communities and users within those communities.

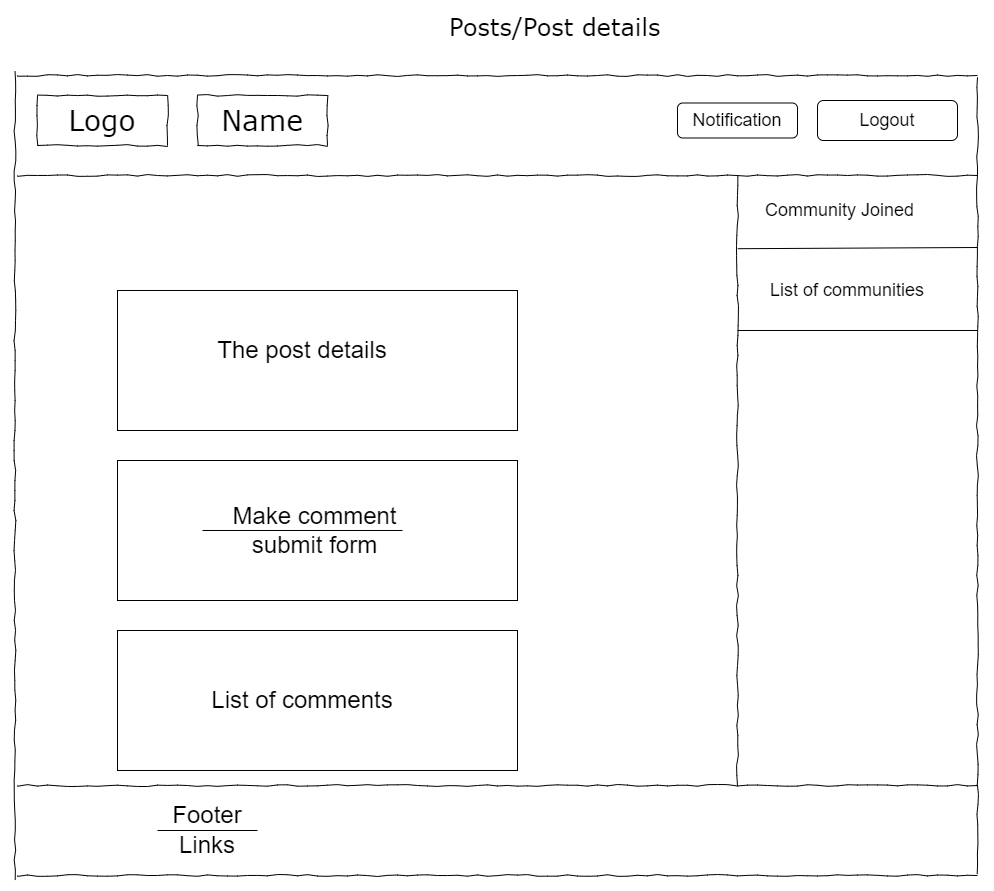
### 3.4.5 Wireframes



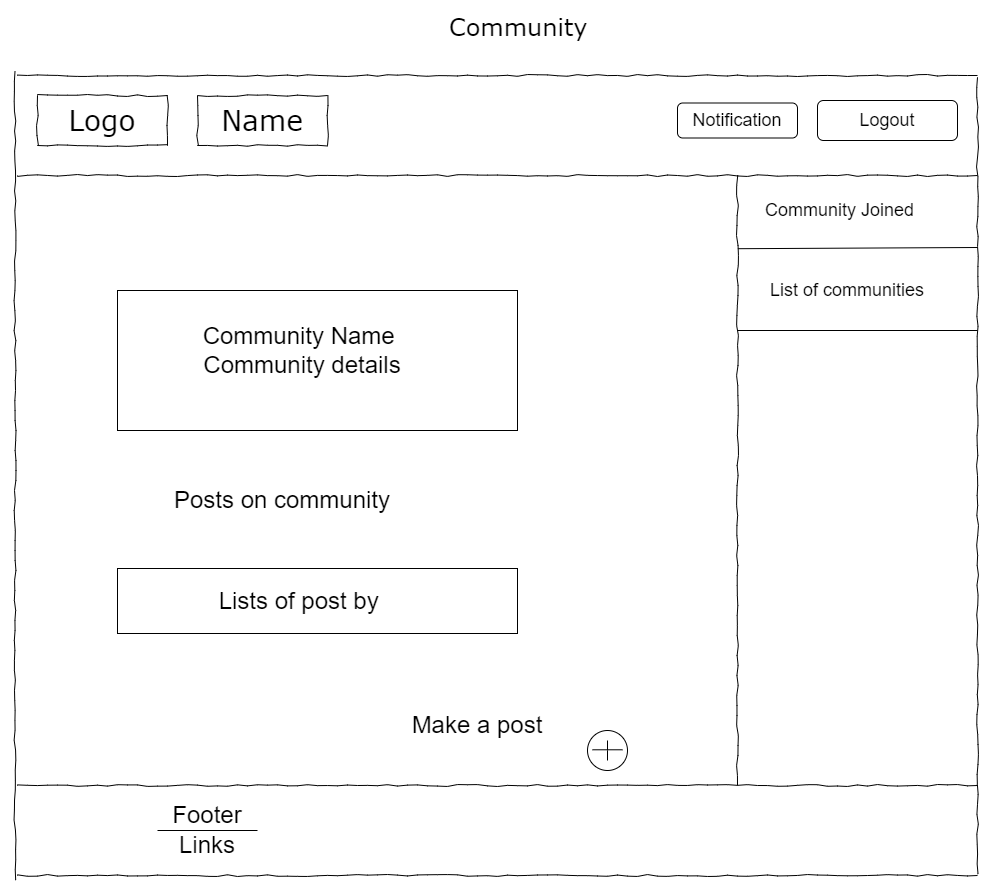
On this page, it is where a user is introduced to the application and its functionalities, the reason it was developed, and how it can be beneficial to people. Moreover, it has a call to action button which takes one to the functional page. Which is the posts page but a user must have an account. A user can sign in or create an account that they can use to access the page. The page also consists of the application objectives and a footer with links to the developer's portfolio.



On the post page a list of posts are displayed from various communities .To its right there is a list of communities on can join and communities one has already joined .One can view post which would direct them to the post details page .But if one is an admin they should be able to delete a post. An admin can also create a post by clicking on the ‘+’ button.



Then in the next post page the post info are displayed and one can make a comment they have made and also delete any comment they make and also be able to reply to people’s comments .To its right there is a list of communities on can join and communities one has already joined .



On the community page the community info is displayed and one can make a post to that community which is exclusive to that community.

# Chapter 5: System Implementation and Testing

## 5.1 Introduction

Chapter 5 of the documentation focuses on API testing, specifically in the areas of API security and functionality. The chapter begins with a brief introduction to the importance of testing in ensuring the reliability and security of APIs. It then presents two main test scenarios: API security and API functionality. The API security scenario includes test cases related to JWT token authentication and user password handling, while the API functionality scenario covers testing the synchronization of application state with the database using Context API. Each test case is outlined with its respective preconditions, test steps, test data, expected results, and actual results. The chapter provides a comprehensive overview of the testing approach and methodology for API security and functionality, which are critical aspects of API development and deployment.

## 5.2 Testing paradigms

Software testing is a software engineering process for quality assessment and improvement. Software testing would be a crucial step to ensure that the application meets the defined requirements and expectations of its users. The objectives of system testing would include:

(1) Correct Functionality: System testing would involve verifying that all the functionalities of the web application are working correctly. This would include testing features such as user authentication, post creation and retrieval, commenting, creation and retrieval of communities, and notification functionalities to ensure that they are functioning as expected.

(2)Incorrect data: System testing would also aim to identify any faults or defects in the system by purposefully trying to make different parts of the application fail. This involves feeding the web application with the wrong data to see how it responds.

(3)Requirement Conformance: System testing would assess whether the social media platform meets the defined functional and non-functional requirements. This would involve verifying that the system meets the specifications, design, and functionality as outlined in the proposal, ensuring that it meets the expectations of its users.

(4) Security testing: The system must be tested so it is not vulnerable to security risks, this was done in multiple ways which are by the architecture of the API, using a user token, protecting the user routes, encrypting user passwords and storing important information in eny file.

By conducting thorough system testing, the social media platform can be validated for its correct functionality, fault identification, and requirements conformance. This would ensure that the system is reliable, secure, and efficient in improving communication among students, staff, and faculty at Strathmore University.

So the Application Programming Interface API was tested using POSTMAN. A representational state transfer (REST) API was used to assign routes to different information on the database with three main routes for the user, posts, and community.

## 5.3 System Implementation

The proposed solution for improving communication at Strathmore University through a dedicated social media platform can be implemented using the MERN stack (MongoDB, Express, React, Node.js) for the web application development, along with Firebase for handling image uploads. React was used to make instant changes as a user interact with the page which provides a user-friendly interface for users to access the social media platform, node js was used to type javascript on the server side, express is used to start the local application and MongoDB was used to store community data, user data, and posts data. Node Package Manager (Npm) was used in the installation of multiple packages jwt web token which was used to authentify users, mongoose which is used to access the MongoDB database, and Firebase used for accessing the firebase storage.

Security was reinforced by using a JWT web token, hence making sure everyone who would like to access some data needed a jwt web token, using a JWT web token prevents users from accessing the database even if it’s from a third-party software.

Here is the code snippet for its implementation.

//express app

const app = express()

//Global Middleware uses json

app.use(express.json())

//logs the route called

app.use((req, res, next) => {

console.log(req.path, req.method)

next()

})

//routes

app.use('/user', userRoutes)

app.use('/post', postRoutes)

app.use('/community', communityRoutes)

Moreover a user password is never returned to the user, a token is created for the user which expires in 30 days (a variable defined in the code), the user token is what is used to access routes on the API. This token is generated when a user registers.

The route protection using a middleware which checks if a user’s token is correct and throws an error whenever it is wrong.

const jwt = require('jsonwebtoken')

const User = require('../models/userModel')

const requireAuth = async (req , res , next) => {

//Verify authentication

const { authorization } = req.headers

if(!authorization) {

return res.status(401).json({error: 'Authorization token required'})

}

const token = authorization.split(' ')[1]

try {

const {\_id} = jwt.verify(token, process.env.SECRET)

req.user = await User.findOne({\_id}).select('\_id')

next()

}catch(error){

console.log(error);

res.status(401).json({ error: "Request is not authorized"})

}

}

module.exports = requireAuth

Security is implemented in multiple ways, most importantly everything to do with security has been written on the server side meaning it cannot be seen by the end user.

Encryption of the user passwords on the database has been implemented using [bcrypt](https://www.npmjs.com/package/bcrypt) making sure user passwords are not stored the way they are type. Moreover bcrypt uses a key which is statically type in an .env file. Here is an example of how a password is stored.  
“$2b$10$TGwMFpx3Bar.YgldhLDxROFltMdVfXB2ndCGcWqVEtlW.mOPO0mya”.

Functionality of the application was tested by testing its API’s .The context API was used to make sure that the state of the application was in sync with the state of the database, this was done by using different context API’s for different components, that is Authentication context, comment Context, community Context and Post Context. The context Api servers the purpose of updating any change that occurs on the database.

Here is an example of the context API:

export const PostReducer =(state, action) =>{

switch (action.type) {

case 'SET\_POSTS':

return {

posts: action.payload

}

case 'CREATE\_POST':

return {

posts: [ action.payload, ...state.posts]

}

case 'DELETE\_POST':

return {

posts: state.posts.filter((post) => post.\_id !== action.payload.\_id)

}

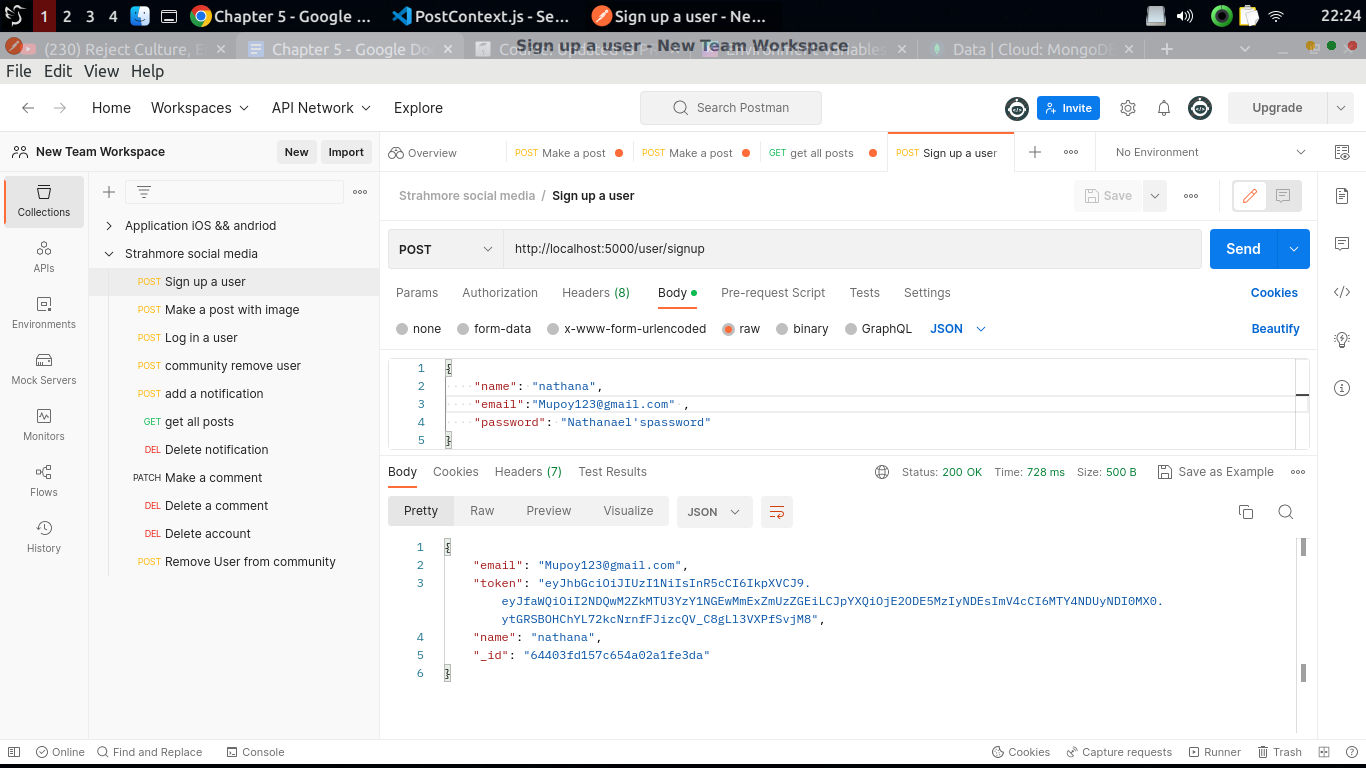
default:

return state

}

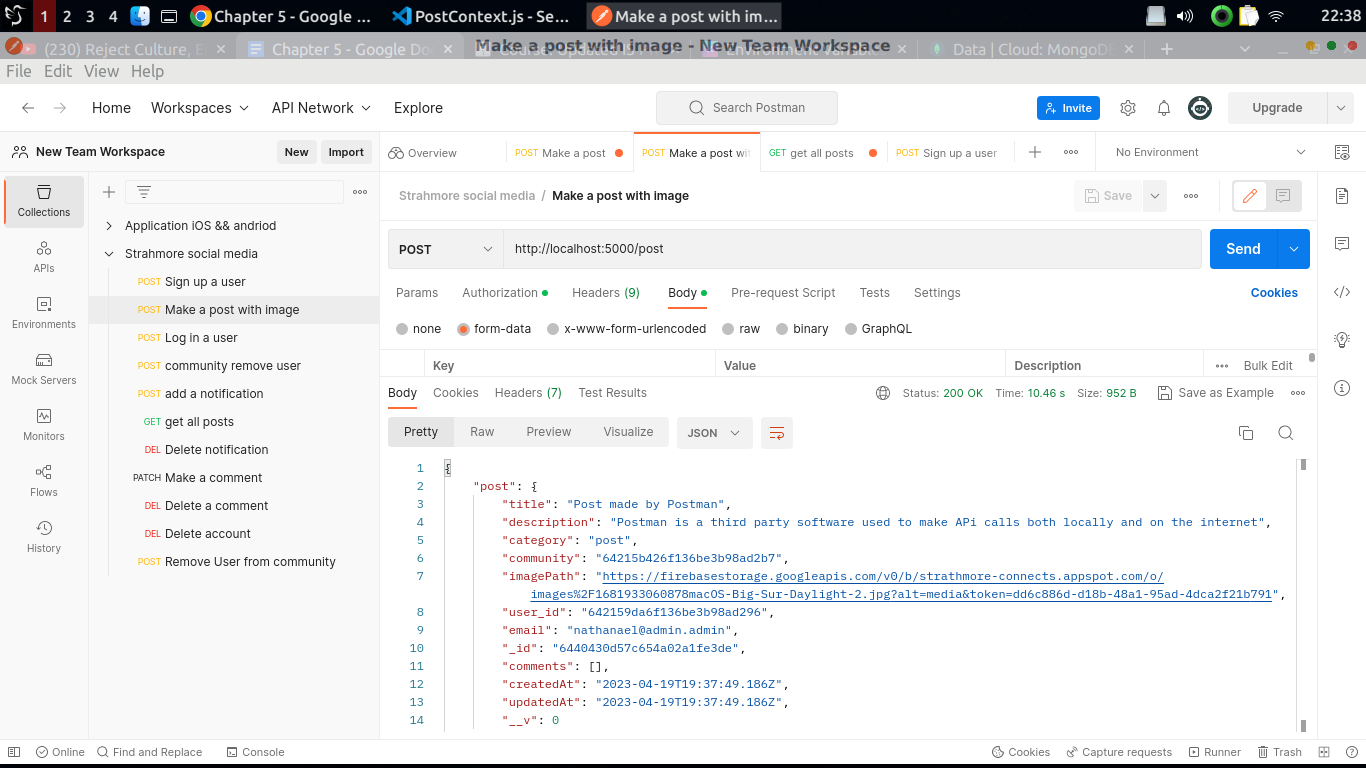
}

The second way was using postman to test the live REST API .So multiple functionalities were tested which are signing up a user, making a post, log in a user, remove a user from a community ,making a community ,deleting a community ,making a comment and deleting a comment.

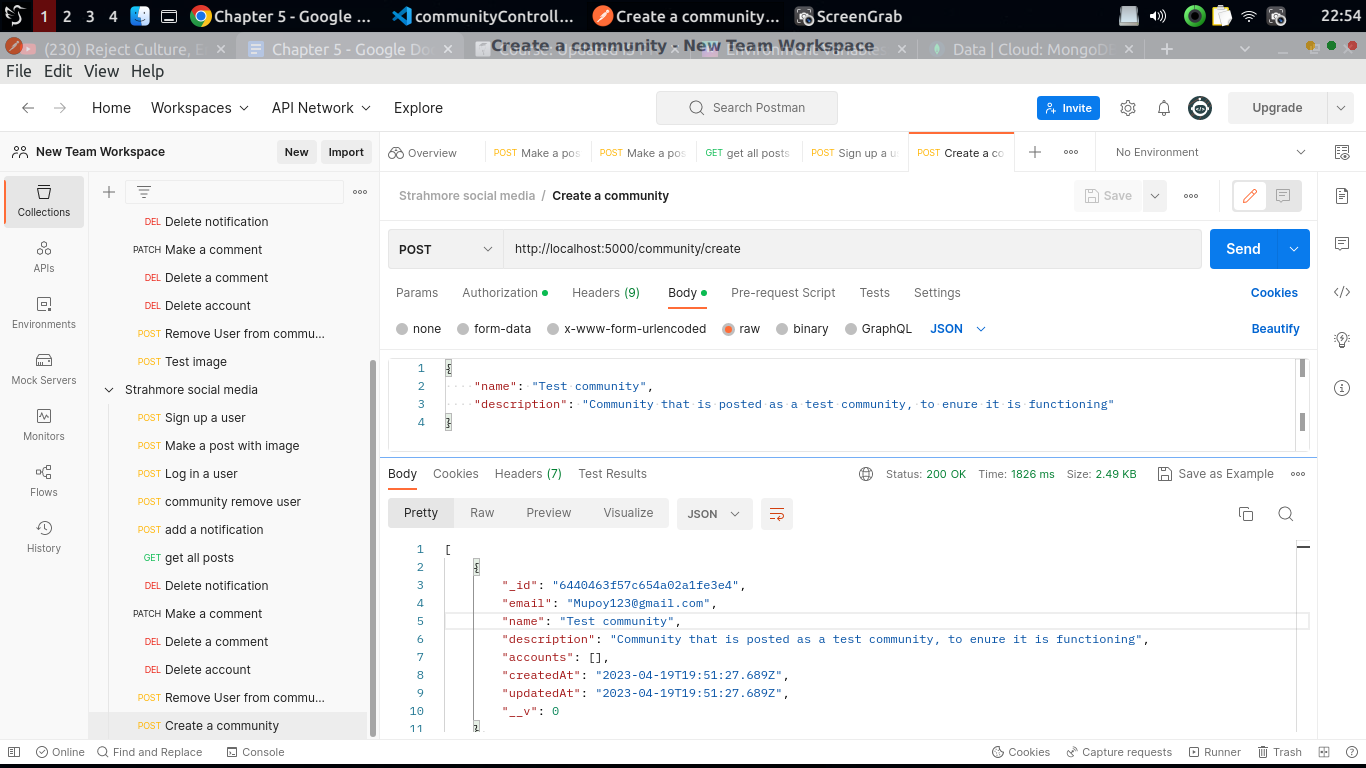
So here is an example of a user registered to the user route (http://localhost:5000/user/signup):  
  


In the example above the api was feed with a name, a password and an email and appropriately returned a user token and a user id, that has been created on the database.

So here is an example of a user registered to the post route (http://localhost:5000/post):



In the example above the API was feed with a title, a description, category, email, an image, a user id, a community and an email and appropriately returned a post object that has been created on the database.

Here is an example on the community route (http://localhost:5000/post/create):  


In the example above the API was feed with a name and a description it appropriately returned a community object that has been created on the database.

The correctness of every data is handled by mongodb automatically this is done by wrapping all the databases made to the database through the server are wrapped in a try catch statement hence is handled by mongodb in most cases .Here is an example of a function making a request to the database.

//create a new post

const createCommunity = async (req, res) => {

//defines parameters for the data to be inputted in the database

const { name, description } = req.body

console.log(req.user)

const user = await User.findOne({ \_id: req.user })

if (!user) throw Error('Invalid user account')

const email = user.email

//adds doc to db

try {

if (!name || !description) throw Error('Please input the required fields')

if (!email) throw Error('You need to use you email')

//finds community by name given

const existingCommunity = await Community.findOne({ name })

//checks if community exits

if (existingCommunity) {

throw Error("Community already exists")

} else {

//adds the document to the datbase

await Community.create({ name, description, email })

//returns all communities

const community = await Community.find({}).sort({ createdAt: -1 })

res.status(200).json(community)

}

} catch (error) {

res.status(400).json({ error: error.message })

}

}

In this function the error is send back to the frontend to be handled when an error is throw by the server.

## 5.4 Test cases

The following tests were carried out to find out how the system was working.

### 5.4.1 Post creation test case

| Test Scenario | Test Case | Pre-conditions | Test Steps | Test Data | Expected Results | Actual Results | Pass/ Fail |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Making a post | To test a user creating a post on the web page | A user must be registered to a community before making a post | A user fills in the required details to make post onto a community | Title: “Community”,  a description : “More on the community” | A post has been created onto the database and a post object from the database is returned | A post is made on the database | pass |

Table 5.1 (Post creation test case)

### 5.4.2 Community creation test case

| Test Scenario | Test Case | Pre-conditions | Test Steps | Test Data | Expected Results | Actual Results | Pass/ Fail |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Making a community | To test the creation of community | A user must be an admin so as to be able to make changes on a community | A user fills in the required details to make community | name: “Community name”,  a description : “Description on a community” | A community has been created onto the database and a community object from the database is returned | A community is made on the database | pass |

Table 5. 2 (Community creation test case)

### 5.4.3 Test case for signing up a user

| Test Scenario | Test Case | Pre-conditions | Test Steps | Test Data | Expected Results | Actual Results | Pass/ Fail |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Registering a user | To test a the creation of a user | A user nust not have been registered before ,meaning their email should not be in use | A user fills in the required details | name: “Nathanael”,  Email: “nathaniel@strathmore.edu”,  password: “123” | A post has been created onto the database and a post object from the database is returned | A post is made on the database | pass |

Table 5. 3 (Test case for signing up a use)

# Chapter 6: Conclusion Recommendation and Future works

## 6.1 Introduction

In conclusion, the project aimed to address the persistent communication issues at Strathmore University through the development of a dedicated social media platform for the university community. The problem statement highlighted the lack of awareness among students about school activities, resulting in reduced collaborations and decreased participation in events. The project's main conclusion reveals that the social media platform has successfully provided a space for students, staff, and faculty to connect, share information, and foster a sense of belonging, ultimately improving communication, engagement, and participation within the university community. The discovered truth from executing the project is that an exclusive social media platform has the potential to significantly enhance communication and engagement among students and other members of the Strathmore University community. This conclusion aligns with similar projects and research that have emphasized the importance of social media in facilitating communication and fostering community engagement in educational institutions.

## 6.2 Recommendation

The application can be upgraded, to bundle code faster by the use of next js instead of react also one can completely switch to only using Firebase as their only database rather than just using it as third-party software. The user notifications can be implemented much better by using a server to send notifications to students all at once, instead of sending them progressively from one user account this would help solve the communication issue better.

## 6.3 Future works

The backend could have been developed using Rust since it has been proven to be very fast, this is backed up by the fact that Discord a platform with multiple users accessing at the same time has switched to rust to increase performance (the link for the research). Moreover, the frontend could have been written in typescript which makes javascript a statically typed language, hence forcing the programmer to write quality code

# References

AltexSoft. (2021, July 23). *Functional and Nonfunctional Requirements: Specification and Types*. Retrieved from AltexSoft: https://www.altexsoft.com/blog/business/functional-and-non-functional-requirements-specification-and-types/#:~:text=They%20fall%20into%20two%20large,also%20known%20as%20quality%20attributes.

Atatus. (2022, June 13). *Software Deployment*. Retrieved from Atatus: https://www.atatus.com/glossary/software-deployment/

Bojic, A. (2022, January 20). *How to effectively communicate and collaborate with third parties in business*. Retrieved from Pumble: https://pumble.com/blog/third-party-business-communication/

Campbell, D. (2022, September 28). *Email Overload: How to Manage It*. Retrieved from right inblox: https://www.rightinbox.com/blog/email-overload

Ceci, L. (2023, February 6). *WhatsApp monthly usage in selected markets worldwide as of 3rd quarter 2021*. Retrieved from Statista: https://www.statista.com/statistics/291540/mobile-internet-user-whatsapp/#:~:text=As%20of%20July%202021%2C%20WhatsApp,of%202021%20in%20India%20alone.

Choice, C. (n.d., n.d. n.d.). *Agile* . Retrieved from Citizen Choice: https://citizenchoice.in/course/software-project-management/Chapter%202/5-agile-methods

Clouds, C. (2021, November 19). *The Advantages and Disadvantages of Using GitHub*. Retrieved from Code Clouds: https://www.codeclouds.com/blog/advantages-disadvantages-using-github/#:~:text=Potential%20Drawback%3A%20Difficult%20To%20Use%20For%20Beginners&text=New%20users%20can%20find%20the,and%20arguments%20in%20the%20software.

*Clubs and socities*. (2023). Retrieved from Strathmore University Student Affairs: https://susa.strathmore.edu/our-services/clubs-and-societies/

Code, V. S. (2023, February 2). *Why did we build Visual Studio Code?* Retrieved from Visual Studio Code: https://code.visualstudio.com/docs/editor/whyvscode#:~:text=Edit%2C%20build%2C%20and%20debug%20with,selection%2C%20snippets%2C%20and%20more.

Creative, T. (2019, October 24). *Pros and Cons of Slack*. Retrieved from Texas Creative: https://texascreative.com/blog/pros-and-cons-slack

Eddie. (2020, February 11). *Email vs. Messenger: Which communication software is right for your business?* Retrieved from Flock Blog: https://blog.flock.com/email-vs-messenger#:~:text=Another%20key%20benefit%20of%20modern,across%20multiple%20emails%20or%20attachments.

File, T. (2021, June 13). *What is data encryptiona nd why is it important?* Retrieved from Titan File: https://www.titanfile.com/blog/what-is-data-encryption-and-why-is-it-important/

gager, A. (2022, September 30). *Efficiency and Effectiveness: Know the Difference*. Retrieved from Facilities Net: https://www.facilitiesnet.com/maintenanceoperations/article/Efficiency-and-Effectiveness-Know-the-Difference--17835#:~:text=Efficiency%20and%20effectiveness%20are%20not,producing%20a%20desired%20result%3B%20success.

GeeksforGeeks. (2020, July 17). *ER Diagram of a Company*. Retrieved from GeeksForGeeks: https://www.geeksforgeeks.org/er-diagram-of-a-company/

GeeksForGeeks. (2021, September 6). *Use Case Diagram for Online Banking System*. Retrieved from GeeksForGeeks: https://www.geeksforgeeks.org/use-case-diagram-for-online-banking-system/

GeeksForGeeks. (2022, December 2). *Software Engineering | Architectural Design*. Retrieved from GeeksFOrGeeks: https://www.geeksforgeeks.org/software-engineering-architectural-design/

GeeksForGeeks. (2022, May 2). *Unified Modeling Language (UML) | Activity Diagrams*. Retrieved from GeeksForGeeks: https://www.geeksforgeeks.org/unified-modeling-language-uml-activity-diagrams/

GeeksForGeeks. (2022, May 6). *Unified Modeling Language (UML) | Sequence Diagrams*. Retrieved from GeeksForGeeks: https://www.geeksforgeeks.org/unified-modeling-language-uml-sequence-diagrams/

gegok12. (2022, May). *School Event Management System*. Retrieved from gegok12: https://gegok12.com/school-event-management-system/#:~:text=It%20helps%20the%20teachers%2C%20parents,for%20a%20better%20learning%20outcome.

Half, R. (2023, January r5). *Advantages of OOP*. Retrieved from Robert Half: https://www.roberthalf.com/blog/salaries-and-skills/4-advantages-of-object-oriented-programming

Hamilton, T. (2023, January 7). *Levels of Testing in Software Testing*. Retrieved from GURU99: https://www.guru99.com/levels-of-testing.html#:~:text=4%20Levels%20of%20Testing&text=Integration%20Testing%20%3A%20checks%20the%20data,met%20as%20per%20its%20delivery.

HANNAH, J. (2022, December 27). *What Exactly Is Wireframing? A Comprehensive Guide*. Retrieved from CareerFoundry: https://careerfoundry.com/en/blog/ux-design/what-is-a-wireframe-guide/

Holmes, T. (2023, January 16). *What is a System Requirement Specification?* Retrieved from Easy Tech Junkie: https://www.easytechjunkie.com/what-is-a-system-requirement-specification.htm

Houlis, A. M. (2022, Febuary 13). *7 Problems Good Communicators Are Able to Avoid*. Retrieved from Fairy God Boss: https://fairygodboss.com/career-topics/lack-of-communication#:~:text=Because%20ineffective%20communication%20can%20create,might%20just%20feel%20overall%20defeated.

Khamooshi, P. (2019, December 20). *The benefits of using web-based applications*. Retrieved from GEEKS: https://www.geeks.ltd.uk/insights/blog/the-benefits-of-using-web-based-applications

Knapp, M. T. (2021, July 1). *What Is Slack and How Does It Work?* Retrieved from Life Wire: https://www.lifewire.com/what-is-slack-4685683

MacDonnell, R. P. (2018, June 25). *10 Ways to Make the College Experience Better for All Students*. Retrieved from JFF: https://www.jff.org/points-of-view/10-ways-make-college-experience-better-all-students/

NEWMAN, L. H. (2021, January 8). *WhatsApp Has Shared Your Data With Facebook for Years, Actually*. Retrieved from Wired: https://www.wired.com/story/whatsapp-facebook-data-share-notification/

Ozanich, A. (2022, March 7). *What Is JavaScript & Why Is It Important?* Retrieved from Hub Spot: https://blog.hubspot.com/website/what-is-javascript

Paterska, P. (2021, September 9). *10 Most import beefits of Implementing Agile*. Retrieved from EL Passion: https://www.elpassion.com/blog/10-benefits-of-agile

Point, S. (2021, July 19). *What Is the Difference Between Authentication and Authorization?* Retrieved from Sail Point: https://www.sailpoint.com/identity-library/difference-between-authentication-and-authorization/

professionals, P. (2022, Febuary 7). *The Importance of communication* . Retrieved from Cipp: https://www.cipp.org.uk/resources/news/the-importance-of-communication.html#:~:text=In%20our%20daily%20life%2C%20communication,We%20all%20need%20to%20communicate.

Staff, T. (2019, June 9). *30 Of The Best Digital Collaboration Tools For Students*. Retrieved from Teach Thought: https://www.teachthought.com/technology/best-digital-collaboration/

Superpowers, C. (2018). Pros and Cons of communication styles [Recorded by Lisette]. The Hague, The Neverlands.

Team, I. E. (2022, June 25). *9 Nonfunctional Requirements Examples*. Retrieved from indeed: https://www.indeed.com/career-advice/career-development/non-functional-requirements-examples

Tony Oquendo, Chris Wagner. (2022, September 22). *5 Benefits of Cloud Communications in Higher Education*. Retrieved from Ed Tech: https://edtechmagazine.com/higher/k12/article/2022/09/5-benefits-cloud-communications-higher-education

vdovin, A. (2020, Febuary 18). *The Advantages and Disadvantages of Email for Communications in a Company*. Retrieved from Desk Alerts Internnal communications solution: https://www.alert-software.com/blog/the-advantages-and-disadvantages-of-email

VoiceSnap Team. (2022, July 26). *College Communication App- The Friend Every Student Needs*. Retrieved from Voice Snap: https://www.google.com/search?q=how+can+universities+upgrade+communication+with+apps&rlz=1C1VDKB\_enKE1010KE1010&ei=pAvcY5WoJo-ckgX\_up6gBQ&ved=0ahUKEwiViqOFxvf8AhUPjqQKHX-dB1QQ4dUDCA8&uact=5&oq=how+can+universities+upgrade+communication+with+apps&gs\_lcp=Cg

Wells, K. (2018, January 9). *8 Reasons Email Communication is Ineffective in the Workplace*. Retrieved from fsd: https://fsd.servicemax.com/2018/01/09/8-reasons-email-is-ineffective-in-workplace/

Young, N. (2021, December 20). *Best Collaborative Learning Tools for 2023*. Retrieved from Teach Floor: https://www.teachfloor.com/blog/best-collaborative-learning-tools-2022

Zulia, U. o. (2019, Novemebr 12). *Language barriers to effective communication*. Retrieved from redalyc: https://www.redalyc.org/journal/279/27962177008/html/#:~:text=Language%20barriers%20are%20significant%20because,and%20lives%20of%20the%20people.

# Appendix A: Gantt chart

The appendix will contain supplementary material providing a more comprehensive understanding of the research problem or it is information that is too cumbersome to be included in the body of the paper.

