Development Of A Publish-Subscribe System For Mountain Top University Staff And Student

By

ADELEYE, Daniel Ifeoluwa.

17010301028

A PROJECT SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE AND MATHEMATICS, COLLEGE OF BASIC AND APPLIED SCIENCES,

IN PARTIAL FULLFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF DEGREE OF BACHELOR OF SCIENCE IN COMPUTER SCIENCE

2021

DECLARATION

I hereby declare that this project has been written by me and is a record of my own research work. It has not been presented in any previous application for a higher degree of this or any other University. All citations and sources of information are clearly acknowledged by means of reference.

ADELEYE, DANIEL IFEOLUWA

Date

CERTIFICATION

This is to certify that the content of this project entitled '**Development of a Publish/Subscribe System for Mountain Top University'** was prepared and submitted by **ADELEYE DANIEL IFEOLUWA** in partial fulfillment of the requirements for the degree of **BACHELOR OF SCIENCE IN COMPUTER SCIENCE.** The original research work was carried out by him under by supervision and is hereby accepted.

______________________________ (Signature and Date)

Mr J.A. Balogun

Supervisor

______________________________ (Signature and Date)

Dr. M.O. Adewole

Coordinator, Department of Computer Science and Mathematics

DEDICATION

God Almighty, my creator, my strong pillar, my source of inspiration, wisdom, knowledge, and insight, I commit my work to you. Throughout this program, He has provided me with strength, and I have only been able to fly on His wings. I also dedicate my effort to my boss, Mr. Jeremiah Balogun, who has supported me along and ensured that I finish what I started. To my buddies Olatubosun Tobiloba, Obe Israel, Amosun Jumoke and Yusuf Hannah and co who have been affected in every manner possible by this quest. Thank you. My affection for you all can never be quantified. God bless you.

ACKNOWLEDGEMENT

My sincere thankfulness goes to God, who has supplied everything necessary for the completion of this project and the purpose for which it was undertaken. There was never a time of scarcity or desire. In the course of my study, He was there for me, taking care of everything that could have slowed me down and empowering me even at my most trying moments. I am grateful to Dr. D.K. Olukoya, the Chancellor of this magnificent university, for following God's calling and birthing and nourishing the vision of this wonderful institution, Mountain Top University, where I have been prepared to fulfil my destiny as a ruler and a ruler of nations. I commend the Vice Chancellor, Professor Elijah Ayolabi, for his excellent leadership as a leader and a father in this institution.

This work would not have been possible without the assistance and constructive criticism of my supervisor, Mr. Jeremiah Balogun, whose contribution and constructive criticism inspired me to put out the type of effort that I have put forth to make this work as unique as possible. Because of him, I have had the opportunity to participate in legitimate study and have gained a greater understanding of the subject matter. Sir, you will remain in my heart forever. Finally, my uttermost admiration also goes to my parents, Mr. and Mrs. Adeleye who tirelessly created the foundation for my education giving it everything it requires.

ABSTRACT

The aim of this study was to build a social network system that allows educators to share content while also allowing students to join to one or more educators on the system and gain access to their content as well as receive notifications whenever new content is uploaded by any of the educators the subscribers subscribe to.

A literature review was conducted in order to discover and evaluate current publish/subscribe systems. The system's user and system needs were established through informal discussions with system users. UML diagrams, such as use case, sequence, and class diagrams, were used to specify the system design. React Native, Node.js, JavaScript, and Firebase were used to develop the system.

The system's result revealed the deployment of the database for storing data alongside the mobile application. The system was able to distinguish if the user was a writer or a reader as a results of the research.

The study concluded that using the system deals with the issue of uncertainty or lack of direction in the regarding career path for final academic year or after, as a consequence of the knowledge gap between academic curriculum and practical career criteria.

Keywords: *Publish subscribe, topic-based, content based, information system.*

TABLE OF CONTENTS

LIST OF FIGURES

Page

CHAPTER ONE

INTRODUCTION

1.1 Background of Study

Information dissemination and communication is generally strictly formatted in academic environments and only a few years ago academic institutions joined the rest of the world in adopting social networking sites for sharing information and subsequently foster less formal information dissemination and communication methods. Recent research work have begun to shape and redefine the ways educators and students communicate for academic purposes. Many study have proved the adoption of online interactions between educators and students to be more productive and satisfactory.

However, there is a growing and urgent need for information sharing and communication systems that specifically suits the preferences of the students and at the same time afford the educators a chance to a wider audience of student. Such system will allow educators to effectively impact students academically, socially and perhaps psychologically through information sharing and timely informal communication channels.

A publish/subscribe (pub/sub) system is a mechanism for disseminating data from one person to another. Data is published by publishers, and subscribers receive data that is relevant to them. Publishers and subscribers are separate entities that do not need to know anything about one another. There are several sorts of Pub-Sub systems, including content-based, type-based, and topic-based systems. Content-based membership is a more adaptive, but also more complicated perspective in the Pub-Sub conspiracies. In this communication system, a person can act as a publisher or a subscriber of data. Publishers create material, which is consumed by subscribers, according to the research. The major semantic distinguishing feature of pub/sub is the way notifications move from senders to receivers: receivers are not explicitly addressed by publishers, but are instead addressed indirectly by the content of notifications. Subscriptions for specific notifications are issued separately from the publishers that generate them, and subscribers are then asynchronously informed for all material, supplied by any publisher, that matches their subscription. Both publishers and subscribers connect with a single entity, the notification service, which records all subscriptions linked with individual subscribers, receives all notifications from publishers, and distributes all published notifications to the appropriate subscribers. As a consequence, information is exchanged between publishers and subscribers who do not know one other personally. One of the pub/sub paradigm's primary characteristics is anonymity.

This study aims to allow subscribers and publishers create a relationship that is needed to foster learning and necessary guidance such that a subscriber can not only access content submitted by a publisher but can also communicate directly with the publisher and develop further interest through working relationship with the publisher.

1.2 Statement of Problem

This study aims to solve the problem of ambiguity or indecisiveness in the choice of career path for student in their final year or after, due to the knowledge gap between academic curriculum and practical career requirement. The students hence will be able to via the information shared by the lecturers acclamation urge a subject or field of interest and build a career prospect through the indirect mentorship from the lecturers.

1.3 Aim and Objectives

The aim of this study is to develop a social network platform that can be used by educators for sharing content (articles, projects, links and other learning resources) then allowing students to subscribe to one or more educator on the system and get access to the content posted by the educator as well as receive notification each time a content is submitted by any of the educators the students subscribe to.

The specific objectives are to;

i. identify the requirements of the proposed system

ii. specify the design of the system

iii. implement the system

iv. test the system

1.4 Methodology of the Study

In order to meet up with the aforementioned objectives of the study, the following methods will be adopted:

a. A review of literature was done in order to identify and understand existing publish-subscribe systems

b. The user and system requirements of the system were identified from the system users via informal interview

c. The system design was specified using the unified modelling language (UML) diagrams such as: Use case, Sequence diagram, and Class diagram as well as system architecture.

d. The database was implemented using MY SQL technology via Firebase services and the Frontend will be implemented using React Native Mobile application framework.

e. The system was tested manually through simulations using lecturers and student from the department of computer science, mountain top university for security and performance.

1.5 Significance of Study

Various publisher-subscriber model applications exist that allow information sharing in different areas and institution but this study will be significant in the following ways:

a. It will create an avenue for continuous learning among students outside the classroom.
b. It will foster better educator and student's relationship particularly in higher institutions of learning by bridging the communication gap.
c. It will serve as a means for educators to inspire and invariably mentor student who have interest in their fields or related works.

1.6 Scope and Limitation

This study does not attempt to redefine the publish-subscribe model by removing the anonymity feature between subscriber and publisher but rather will focus on using the model to suit the case study of the university environment to achieve the academic and social objectives. This project will only develop a system that is immediately adoptable within the university in view based on their prevalent situation as it may need to be further tweaked to suit the academic and social realities in universities in other climes.

1.7 Definition of Terms

Content: Any piece of information irrespective of its form (speech, writing, visual arts etc.) or media through directed towards an audience or end-user through which they can derive value.

Educator: An educator is a person who teaches, informs and inspires others. He/she is responsible for modelling and demonstrating effective learning as well as disseminate content and knowledge.

Publisher: A publisher is a client in a pub-sub model who creates messages or content and sends to specific users based on their subscriptions.

Student: A student studying in a school or other academic establishment who is learning with the intention of gaining knowledge. It is a person who is studying; it is a person who is dedicated to learning.

Subscriber: A subscriber is a client in a pub-sub model who receives content or message from one or more publishers based on his/her subscriptions.

Subscription: Subscription is the indication of interest in the contents made available by a particular publisher in a pub-sub model.

UML: is a general-purpose, developmental modeling language for software engineering that aims to give a common approach to depict a system's architecture.

CHAPTER TWO

LITERATURE REVIEW

2.1 Information System

The terminology "information system" has two meanings: one for its functionality and the other for its framework. From a functional perspective; an information system is a technologically implemented medium with the intention of recording, storing, and disseminating linguistic expressions as well as for the supporting of inference making. From a structural perspective; an information system consists of a collection of people, processes, data, models, technology and partly formalized language, forming a cohesive structure which serves some organizational purpose or function. (Management Information Systems (MIS) 2011)

Information management is described as an organization's capacity to handle information. Creating, preserving, retrieving, and making available the appropriate information as quickly as possible, that is information, at the right moment, in the hands of the right people, at the right location for use in decision-making at the lowest cost, in the finest medium (Langemo, 1980). Although information systems is regarded as the most important discipline, all current research is focused on it, and enterprises struggle to develop unique technologies and facilities to support it. The information that was assessed as the soul of these systems is the generator component of this region. As an outcome, high performance success necessitates determining the sorts of IS and the work style. Information systems (IS) are an integrated environment of hardware, software, and people that primarily serves the purpose of collecting and processing data into valuable information by employing a set of procedures on data collection. As a result, information is derived from data by IS procedures, and the distinction between data and information is clarified. The information is the final data of processing, while the data is raw materials. (Hasan, 2018)

Processed data can be characterized as information. Raw data that has been processed to give a meaning is a simple definition of information. Data that has been purified to the point where it can be utilized for analysis is referred to as information. Information, as contrast to data, is organized, clear, and has shape and meaning. It can be linked to any context depending on how it is examined. Information is a meaning that a person may convey or extract from a representation of facts or a concept, and this is done by applying existing conventions to the field of study in question. (Zoikoczy, 1981). An information system is a set of interconnected components that collect, store, and analyze data, as well as deliver information, knowledge, and digital commodities. Businesses and other organizations utilize information systems to run and manage their operations, communicate with customers and suppliers, and compete in the market. (Zwass, 2020)

2.1.1 Types of information system

There are several types of information systems that apply to specific business needs or business types and can majorly be classified into the following:

a. Transaction processing system

A transaction includes all product and service purchases and sales, as well as any everyday business transactions or operations necessary to run a firm. The frequency and types of transactions executed vary depending on the business and the firm's size/scope. Billing clients, making bank deposits, keeping track of new recruit information, inventory counts, and keeping track of client-customer relationship management data are all examples of typical transactions. All contractual, transactional, and customer relationship data is kept safe and accessible through a transaction processing system. It also helps with sales order processing, payroll, shipping, sales management, and other normal tasks that keep companies functioning. By establishing a TPS, organizations may improve the dependability and quality of their user/customer data while lowering the chance of human error. (Christiansen, 2021)

b. Office automation systems

The system includes of both hardware and software solutions that allow data to be transferred across systems without the need for human involvement. Accounting, data management, training, facility management, and other administrative activities are all simplified and automated through office automation. Going paperless isn't the only benefit of an office automation system. For instance, every work product and customer interactions should be stored in a document management system. However, an office automation system provides much more to businesses than simply being green by reducing paper usage. It's a powerful tool for automating processes, identifying inefficient workflows, and assisting with informed decision-making. (Eisner, 2020)

c. Operations support system

The end client inserts data into a system, which is subsequently utilized to generate information products for organizations and individuals, such as reports. An operation support system is what this type of system is called. The purpose of the operation support structure is to simplify business transactions, manage production, aid internal and external communication, and maintain the organization's central database. The operation support system includes a transaction processing system, a processing control system, and an integrated information system. (Juneja, 2015)

d. Management information systems

Managers at all levels benefit from information systems, from those in charge of small work group schedules and budgets to those in charge of the organization's long-term plans and budgets. Management reporting systems deliver routine, detailed, and voluminous information reports tailored to each manager's areas of responsibility. These systems are most commonly used by first-level supervisors. Such reports, in general, focus on past and present actions rather than anticipating future outcomes. Reports may only be issued automatically in extraordinary circumstances or at the request of management to minimize information overload. (Zwass, 2020)

e. Decision support systems

Data is analyzed by a decision support system to assist managers in making choices. It gathers and saves the information needed for management to make the best decisions at the right moment. A DSS can be used by a bank management, for example, to assess changing loan trends and determine which yearly loan targets to meet. The information system includes decision models that assess and synthesize large volumes of data and present it in an understandable visual fashion. Because a DSS is interactive, management may simply add or delete data and ask relevant questions. This provides proof for mid-management to make the best judgments feasible in order for the company to meet its goals. (Christiansen, 2021)

f. Executive information systems

Executive information systems (EIS) present a variety of critical facts in a highly simplified and accessible format, generally through a graphical digital dashboard. Senior executives, on the other hand, generally rely on a variety of informal sources of information, making formal, computerized information systems only marginally useful. Nonetheless, the chief executive officer, senior and executive vice presidents, and the board of directors require this support in order to monitor the company's performance, analyze the business climate, and establish strategic objectives for the future. These executives must, in particular, compare their company's performance to that of its rivals and research regional or national economic trends. Executive information systems are frequently customized and employ a variety of media formats, allow users to "dig down" from broad overviews to more specific data. (Zwass, 2020)

2.2 Publish/subscribe

It's a well-known consideration in the design in which writers use a set of writings to produce a set of information, while users use a set of subscriptions to express their interests. Publishers use a set of publications to disseminate a collection of information, and subscribers use a set of subscriptions to describe their interests. When a publication is received, the system confirms receipt and distributes the publication to the subscribers who fulfill the criteria. The publish/subscribe method divides time, space, and flow between publishers and subscribers, which reduces program complexity and resource consumption.

Subscribers to the publish/subscribe approach frequently receive just a percentage of the total number of messages published. Filtering is the process of choosing which communications to receive and then processing them once they have been received.

Topic-based filtering and content-based filtering are the two most popular forms of filtering.

Joining a group that has a topic of interest is how you subscribe to a topic-based system. All of the group's members get copies of publications that are related to the topic. As a result, both publishers and subscribers must specify which group they want to subscribe to. Topic-based systems, as well as group communication and event notification systems, have been around for a long time. Because no prior knowledge of the subscribers' interests is required, content-based publish/subscribe systems link subscriptions with publications. As an outcome, systems are more versatile and helpful, as subscribers may more accurately describe their interests using a set of predicates. To create this sort of system, millions of publications must be matched with subscribers in a thorough and effective manner. Publish/subscribe (pub/sub) strategies are an attractive communication paradigm to explore when designing large-scale school notification systems. (Jahan, 2013)

The following challenges of a publish/subscribe system:

a. Relay Nodes

One distinguishing feature of Peer-to-Peer pub/sub systems is that they use a generic overlay network (e.g., DHT, tree, full-mesh) rather than projecting the social graph in the Peer-to-Peer overlay network. Due to the fact that social users are not always directly connected in the Peer-to-Peer overlay network, message dissemination in Peer-to-Peer pub/sub systems is dependent on peers (also known as relay nodes) who may or may not be interested in the message.

b. High traffic

Recent pub/sub systems attempt to simplify the routing tree design and focus on the construction of the Peer-to-Peer overlay network in order to improve message dissemination efficiency. Each peer has a finite number of connections that can be maintained, and these connections are chosen without using the social graph or social interactions. As a corollary of the high social interactions and the lack of social integration in the overlay design, the generated Peer to Peer overlay network has load balancing issues, with a percentage of their colleagues having high traffic overhead compared to the rest of the peers.

c. Dissemination Latency

Each peer in the Peer-to-Peer overlay network has a unique upload and download bandwidth profile. Because each peer has a limited number of connections, keeping a Peer-to-Peer connection with a low bandwidth rate increases dissemination latency, affecting the overall performance of the Peer-to-Peer pub/sub system.

d. Dynamic environment

It is critical for the OSN's success to provide a fail-safe peer-to-peer pub/sub system with minimal disruption to social friends' communication. The architecture of a churn-resistant Peer to Peer pub/sub system has been studied in OMen; however, OMen falls short of identifying each social user's online activity, which introduces additional latency because establishing a Peer-to-Peer connection requires a Multi-Path TCP connection.

Some advantages of a Publish–subscribe system according to (Gibb, 2019) are:

a. Loose coupling

Subscribers are only tangentially related to publishers, thus they don't need to be aware of their presence. Publishers and subscribers are allowed to disregard system topology since the content is the major emphasis. Even if the other isn't there, both can continue to function normally. The client can't send messages to the server unless the server process is active, and the server can't receive messages unless the client is operating under the classic closely linked client–server paradigm.

b. Scalability

Publishers and subscribers are often physically and temporally separated in many pub/sub systems. Middleware analysts frequently deactivate a publisher in such pub/sub systems so that the subscriber may work through the backlog (a form of bandwidth throttling). Scalability, on the other hand, has a cap. The possibility of a load spike or delay rises as the number of nodes and messages grows. Moreover, message passing issues, such as: Regardless of whether the message was received or not, a publisher may only deliver messages for a limited time because the publisher has no way of knowing who is listening, the benefits of the pub/sub model are sometimes swamped by message delivery issues.

Publish-Subscribe Components:

A number of components have been taken into consideration when developing the publish, /subscribe model. A description of how those components are organised and connect to othersis provided in the following discussion.

a. User

The User component is the most important component of the model, and it is subdivided into two subcomponents: the Subscriber component and the Publisher component. The User component maintains information about both publishers and subscribers, such as their names, addresses, states, zip codes, and email addresses.

b. Subscriber

The Subscriber component inherits the attributes name, address, state, zip, email, and so on from the parent component User, which is itself an inherited attribute. It possesses the property "has Interest," which allows it to form a connection with the component Interest. The publish/subscribe model tells the subscriber of a recent publication that meets any of his/her many levels of interest, which is determined by the subscriber's various degrees of interest.

c. Publisher

The Publisher component, like the Subscriber component, inherits the properties name, address, state, zip, email, and so on from the parent component User, which is the same as the Subscriber component. In ways to construct a connection with the Publication component, it has the property "publishes." For example, if a

Publisher publishes any form of news (for example, news on music), the Subscribers whose interests coincide with the news receive a notification of this publication.

2.2.2 How Publish-Subscribe Pattern Works:

The core of software design patterns is the creation of reusable groupings of modules and their relationships. These modules are often represented as classes or objects in a UML design diagram. Modules, on the other hand, are shown in modern architectural patterns as bigger, self-executing processes scattered throughout distributed systems. To fully appreciate the advantages of the Pub/Sub design, you must first comprehend the underlying pattern upon which an information system is formed, and then trace its progression into a distributed system. Publish–subscribe is a communications pattern in which message senders, known as publishers, categorize published messages into classes without knowing which subscribers, if any, may exist. Subscribers register an interest in one or more classes and only get communications that are relevant to them, with no awareness of whose publishers, if any, they are receiving messages from. The publish–subscribe paradigm is a cousin of the message queue paradigm, and it's usually part of a broader message-oriented middleware system.

The pub/sub and message queue concepts are supported by most messaging system APIs; for example, Java Message Service (JMS). This design improves network scalability and generates a more dynamic network architecture, but it limits the ability to modify the publisher and the nature of the data published.

In the publish-subscribe approach, subscribers often only receive a portion of the total messages published. The process of choosing messages for reception and processing is known as filtering. The two most popular methods of filtering are topic-based and content-based. In a topic-based system, messages are published to "topics," or specified logical channels. Subscribers to a topic-based system get all communications posted to the subjects to which they have subscribed. The publisher is in control of deciding which topics subscribers will be able to access. In a content-based system, messages are only delivered to subscribers if their properties or content fit the subscriber's limitations. The subscriber is in charge of categorizing the messages.

Four core concepts make up the pub/sub model:

a. **Topic** – An intermediate gateway that manages a database of customers to whom publishers can send messages.

b. **Message** – A publisher who is unaware of the subscribers sends sequential information to a subject.

c. **Publisher** – The application that publishes a message to a topic

d. **Subscriber** – An application that registers itself with the desired topic in order to receive the appropriate messages.

Some systems support a hybrid of the two; publishers post messages to a topic, while readers can choose to explore one or more subject areas via content-based subscriptions.

2.2.3 Social networking

People create content, share it, bookmark it, and network at a quick pace on social media, which has increased in popularity as a category of online communication over the last few years. Because of its ease of use, speed, and reach, social media is fast influencing public debate in society and establishing trends and agenda in a wide range of topics ranging from the environment and politics to technology and the entertainment industry. In only ten years, the online world has undergone significant transformation. Thanks to the creation of social media, young men and women are now exchanging thoughts, feelings, personal information, photographs, and videos at an astonishingly fast rate. (Huberman, 2010).

Platforms for digital networking are now used by 73% of wired American teenagers (Oberst, 2010). The concepts of social media are shared by Martn, (2008) and Lusk, (2010). To them, Digital networking refers to the usage of Facebook, blogs, Twitter, Myspace, and LinkedIn for conversation and the sharing of photographs and videos. However, for the purposes of this study, social media is defined as individuals using Facebook, Twitter, Skype, Myspace, and Yahoo Messenger to communicate, share ideas, and share images and videos. The increased use of social networking websites has become an international phenomena in recent years. What began as a hobby for a few computer savvy people has turned into a social norm and way of life for people all over the world. Teenagers and young people, in particular, have embraced these sites as a way to interact with their friends, share information, reinvent their identities, and showcase their social lives. (Boyd, 2007). Social networking is the practice of using the internet to link individuals with friends, family, and acquaintances. Social networking sites aren't necessarily about meeting new people online, but that does happen. Instead, they are mostly about engaging with real-life friends, family, and acquaintances. Facebook, Twitter, and Myspace are the most well-known social media platforms. These services allow you to exchange photographs, videos, and information, as well as schedule events, chat, download music, and play online games like Scrabble and Chess. (Australian Communications Consumer Action Network (ACCAN), 2010).

2.2.3 History of social networking

Social networks, contrary to common opinion, have a long and storied history. In 1971, the first email was sent, and it is believed that this was the beginning of social networking. For a variety of reasons, the majority of people appreciated it. Emails were a considerably more cost-effective and practically instantaneous way of interacting with individuals all across the world. People were motivated by the email to think of new methods to communicate even faster. The Bulletin Board system was created in 1978 by Ward Christensen and Randy Suess to allow users to transmit data via phone lines. On a regular basis, users might interact with one another about planned meetings, make announcements, and share data. Several applications were presented to the globe in the next decade, notably the World Wide Web. The arrival of the World Wide Web prompted many people to create new ways of sharing knowledge not only locally but internationally. Geocities, widely recognized as the world's first social networking site, was founded by Beverly Hills Internet. Users may create their own webpages on Geocities depending on their particular hobbies and preferences. (Al-Jenaibi, 2016)

2.3 Software Development Life Cycle (SDLC)

The system development life cycle encompasses the whole process of creating, implementing, and retiring information systems, from inception to analysis, design, implementation, and maintenance to disposal. Despite the fact that there are several SDLC models and approaches, they always share a set of processes or phases. To provide effective protection for the data that the system will communicate, process, and store, information security must be included in the SDLC for any SDLC model. Businesses may balance security demands for agency data and assets with the cost of security controls and mitigation approaches across the SDLC by using the risk management process to develop systems. Critical assets and activities, as well as systemic weaknesses, are identified through risk management methods. Risks are frequently shared within an organization and are not limited to certain system architectures. (Radack, 2009).SDLC is used majorly in several engineering fields, and industrial fields as a framework to help manage, plan, and control the process of developing a system. It is a continuous process that starts from the moment a decision has been made to launch a project, to when the system has met the user requirements and is ready to be deployed.

2.3.1 Software development life cycle processes

The software development life cycle gives a sequence of processes to be followed to design and create a software product efficiently. SDLC framework comprises the following steps:

a. Planning

Project leaders assess the project's terms during the planning phase. Calculating labor and material expenses, setting a schedule with specific deadlines, and forming the project's teams and leadership structure are all part of this process. Stakeholder comments may be included into the planning process. Anyone who stands to gain from the application is referred to as a stakeholder. Obtain comments from potential consumers, developers, subject matter experts, and sales representatives. The scope and goal of the application should be clearly defined throughout planning. It charts a route and equips the team to build software efficiently. It also establishes limits to prevent the project from growing or diverging from its initial goal.

b. Requirement analysis

Defining requirements is part of the planning process to figure out what the application is meant to accomplish and what it needs. A social networking program, for example, would need the ability to connect with a buddy. A search function may be required by an inventory program. The resources required to complete the project are also defined in the requirements because it is it is necessary to know what is expected from the system to be developed before implementation. A group may, for example, create software to control a bespoke manufacturing equipment. The machine is required for the process to work.

c. Designing project architecture

The developers are creating the architecture in the second step of the software development life cycle. The specifics of the design are presented to the system stakeholders, alongside the possible drawbacks that may occur (budget, risks, and time constraints) and each of these parameters is reviewed to determine the best design approach to solve the problem. All of the stakeholders, including the client, debate the many technical questions that may arise at this point. In addition, the technologies utilized in the project, team load, limits, time frames, and budget are all described here.

d. Development and programming

Following the approval of the requirements, the process moves on to the next stage: actual development. Programmers begin by writing source code while keeping previously defined requirements in mind. System administrators configure the software environment, while front-end programmers create the program's user interface and the logic that governs its interaction with the server.

e. Testing

The debugging procedure is included in the testing step. All of the code problems that were overlooked during development are found here, documented, and forwarded to the developers to remedy. Each component of the system is brought together to form a single system after which it is deployed into the testing environment before it is delivered to the client. The testing procedure is repeated until all major issues have been addressed and the software workflow has been stabilized. Black box and white box testing are the two types of testing methodologies used to evaluate software systems.

f. Deployment

When the software is complete and free of major flaws, it is time to make it available to end users. The tech support staff joins when the new software version is released. This section offers user feedback, as well as consultation and assistance during the exploitation period.

g. Operations and maintenance

The development cycle is practically complete at this stage. The application has been completed and is currently being utilized in the field. However, the period of operation and maintenance is still crucial. Users point out the flaws that were missed during testing at this phase. These issues must be addressed, which may give rise to new development cycles.

2.3.2 Software development life cycle models

Developers may use the software development paradigm to help them choose a software development method. A software development paradigm is a set of tools, processes, and procedures that describe the software development life cycle and are explicitly stated. Some examples of software development paradigms or process models are as follows:

a. Waterfall model

The most widely used design process is the Waterfall SDLC model. When one step of the project is completed, it spills over into the next. It is one of the oldest models, having been used in a variety of large government and organizational initiatives. It can be applied to fit various applications, but it is best fitted when the software requirements are well understood documented, clear, and fixed. It can also be used when the technology to be used is not dynamic when the software requirements are not ambiguousThis is a tried-and-true strategy that works. The Waterfall approach has the benefit of allowing each step to be reviewed for continuity

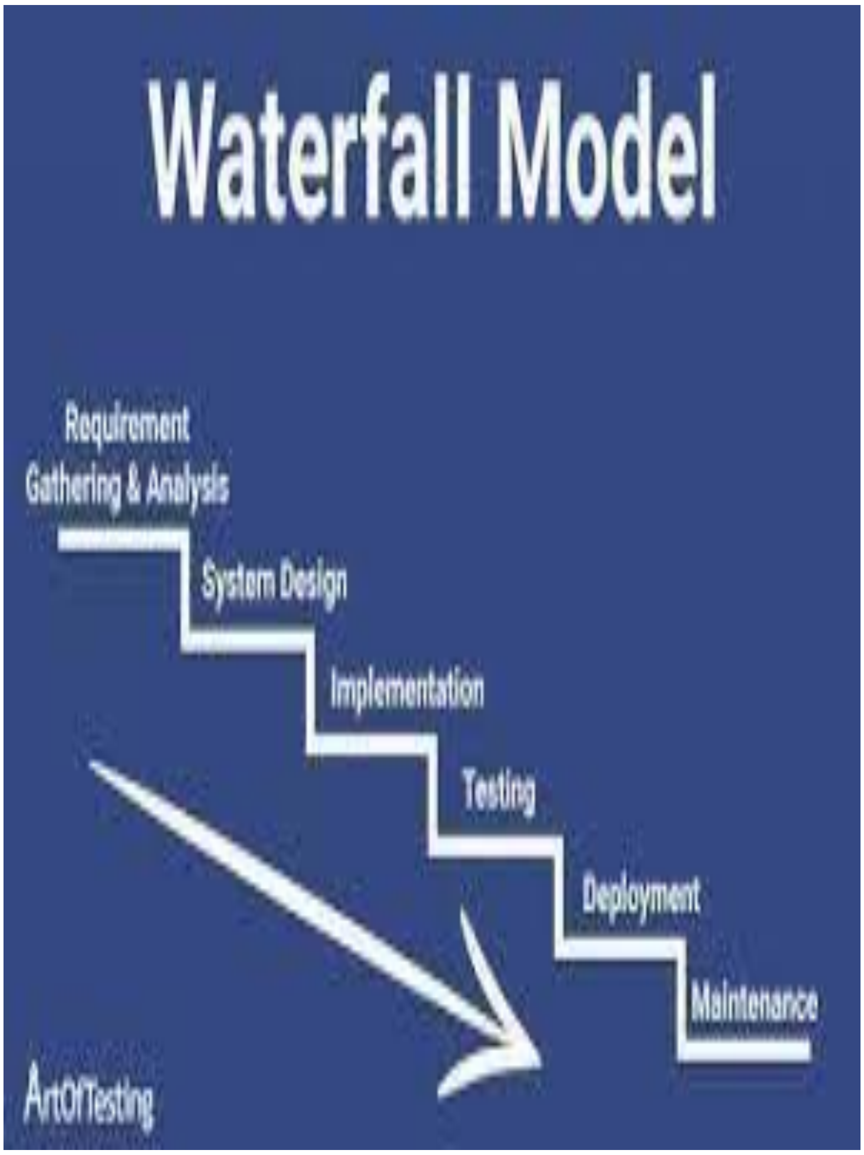


Figure 2.1: Waterfall Model

and feasibility before going on to the next. It is, however, restricted in pace since one phase must end before the next can begin. (jevtic, 2019)

b. Agile model

The agile approach is a hybrid methodology that combines the benefits of both iterative and incremental development by splitting software into components and delivering a functioning model of each component on each cycle or iteration. This approach generates new releases, each of which contains certain incremental modifications. After each iteration, the product is evaluated to see whether it is acceptable or not. As customers, developers, and testers collaborate throughout the project, the agile approach promotes collaboration. The Agile paradigm has the advantage of quickly delivering a functional product and is a very practical development strategy. . One disadvantage of this approach is that, because it is so reliant on client communication, the project might move in the wrong direction if the client is unsure of his or her needs or the direction in which he or she wants to go. (Barjtya, 2017)

c. Iterative model

Repetition is crucial to the Iterative model. Project teams execute a set of software requirements before testing, analyzing, and finding further needs, rather than starting with entirely stated requirements. A new version of the software is developed after each phase, or iteration. Repeat this process until the entire system is operational.

In an iterative approach, the demands are not fully specified; rather, the process begins with a restricted set of criteria, with each iteration producing a smaller form of the product or system, and so on until the final version is developed. In comparison to other common SDLC methodologies, the iterative model has the benefit of providing a feasible solution.

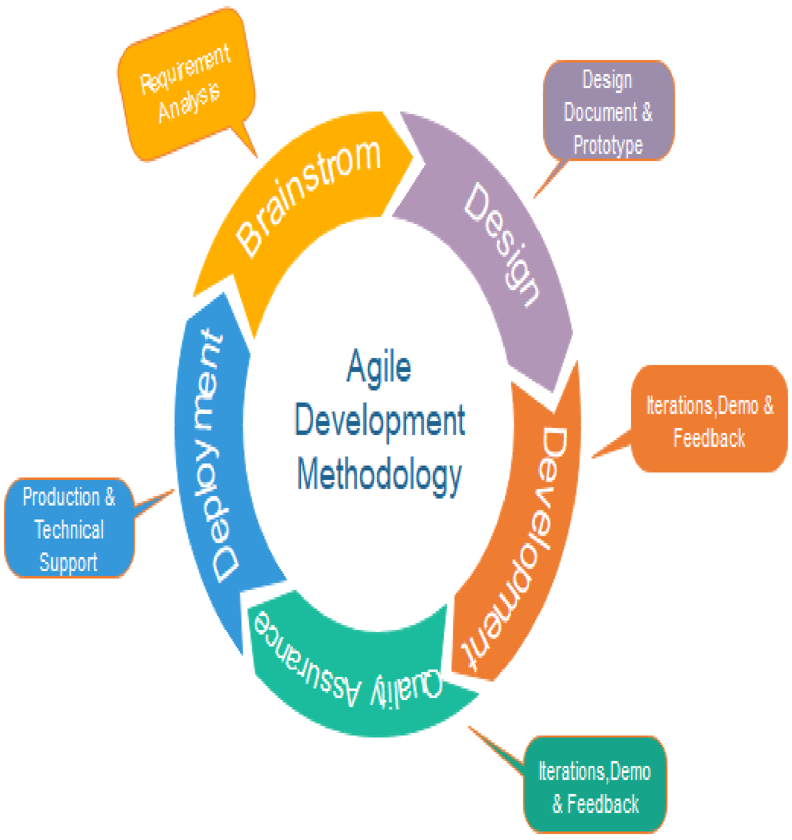

Requirement Analysis
Brainstrom
Design Document & Prototype
Design
Agile Development Methodology
Iterations,Demo & Feedback
Development
Production & Technical Support
Deployment
Quality Assurance
Iterations,Demo & Feedback

Figure 2.2: Agile Model

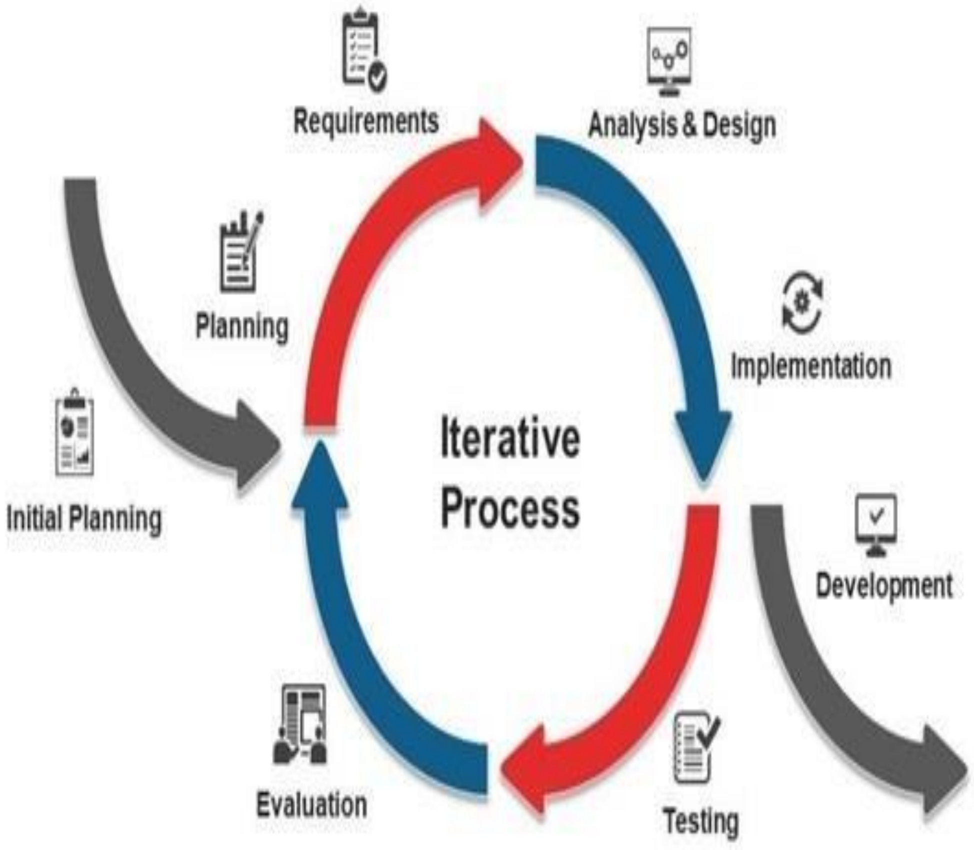


Figure 2.3: Iterative Model

edition of the project early in the cycle, which reduces the cost of change implementation. One drawback is that repetitive procedures can quickly deplete resources. The rational unified process (RUP), developed by IBM's Rational Software group, is an illustration of an iterative approach. (Half, 2019)

d. Spiral Model

Spiral model is a mix of systematic and organized development that takes the benefits of iterative development and combines them with the simplicity of the waterfall model, as well as extra risk analysis elements. The Spiral model's operation is separated into four phases (identification, design, construction, evaluation, and risk analysis), which are repeated until the project is completed. This paradigm allows for gradual updates to software product releases. This paradigm allows for gradual updates to software product releases. The spiral approach is most suited for highly customized software products since user involvement and assessment begin early in the development process. However, you run the danger of constructing a never-ending spiral for a project that never ends. (Barjtya, 2017)

e. V-Shaped Model

The V-Shaped life cycle, like the water fall model, executes a number of operations in a sequential route. Each phase in this paradigm must be accomplished before on towards the next. Testing methods are created early in the life cycle, before any coding is done, in each of the phases before implementation. This life cycle model has the same requirements as the waterfall model. Before beginning development, a system test strategy is established. The test strategy is focused on ensuring that the functionality provided in the requirements is met. The high-level design phase is focused on system architectural design, whereas the low-level conceptual design is

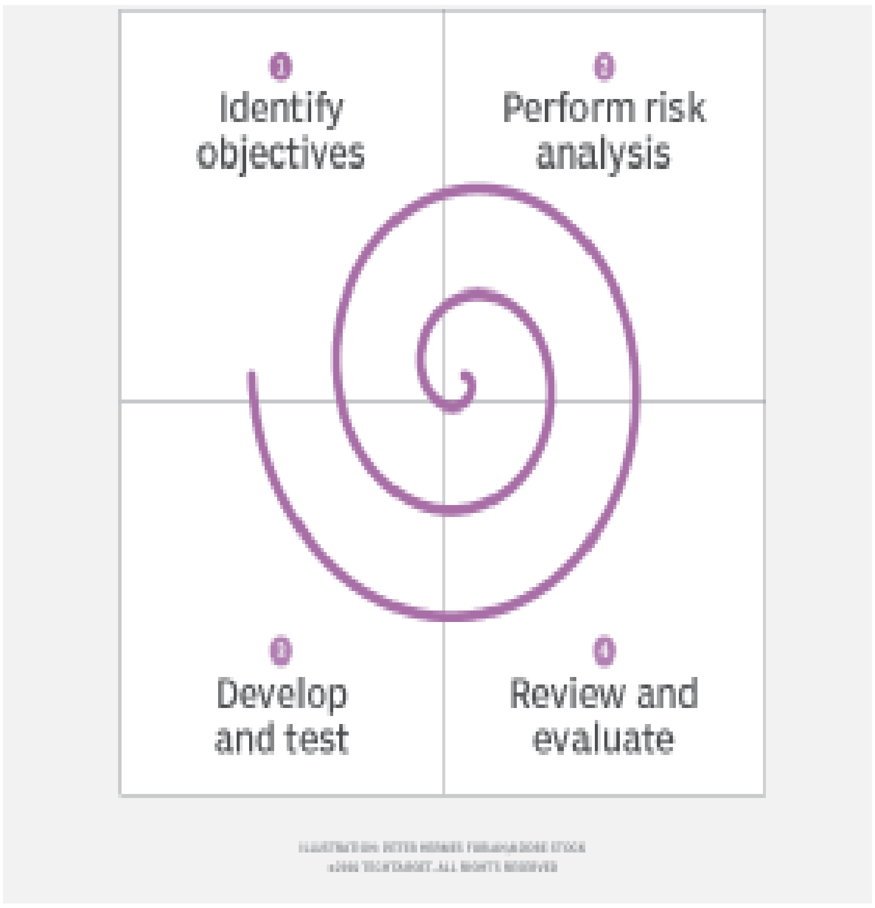


Figure 2.4: Spiral Model

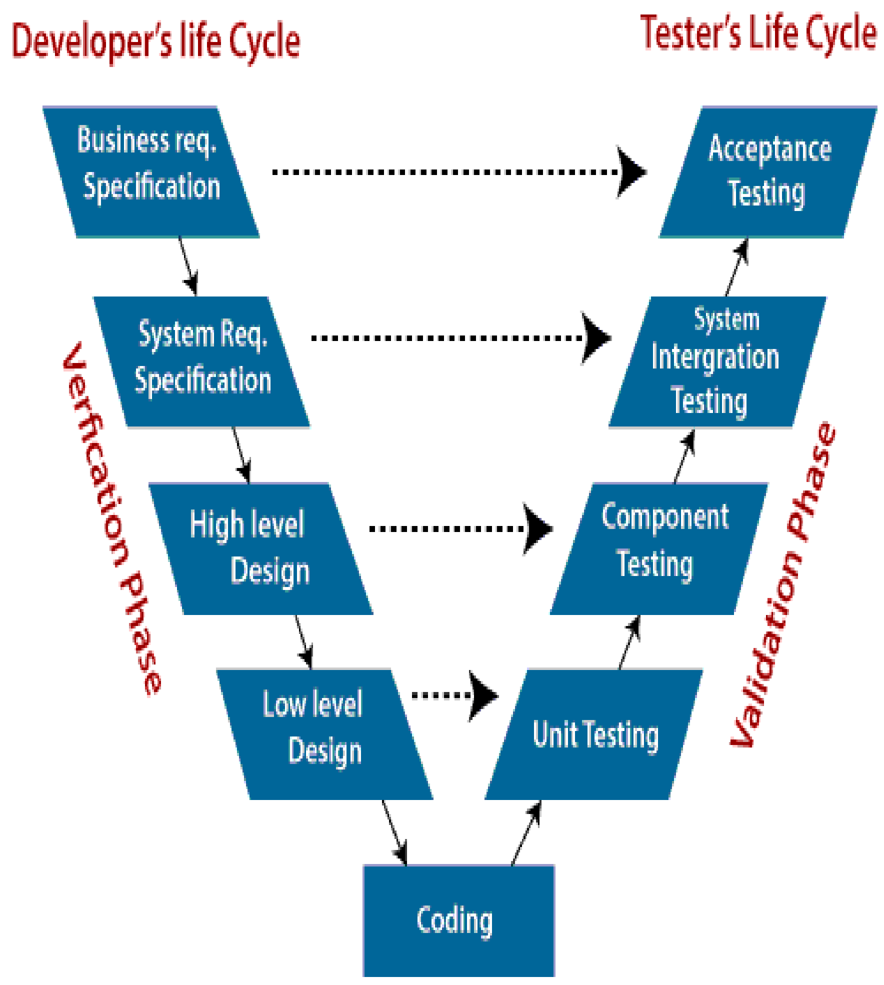


Figure 2.5: V-shaped Model

focused on the application system components and unit testing. (Khurana gourav, 2012)

f. **Rad Model**

The RAD paradigm stands for Rapid Application Development. It's an incremental model of some sort. The components or functions of the RAD paradigm are created in simultaneously, as if they were minor tasks. The projects are timed, delivered, and then put together into a functional prototype. This may immediately offer the client with something to see and use, in addition to an opportunity to provide feedback on the delivery and their needs**.**

g. Incremental Model

The incremental model splits the product into builds, with individual portions of the project being developed and evaluated. Because user feedback is sought at each stage and code is tested sooner after it is developed, this technique is likely to discover flaws in user requirements fast. The overall needs of the end system or product are understood from the outset of development in incremental models, just as they are in sequential models. However, with incremental models, each increment is assigned a restricted set of criteria, and with each subsequent (internal) release, additional requirements are addressed until the final (external) release meets all requirements**.**

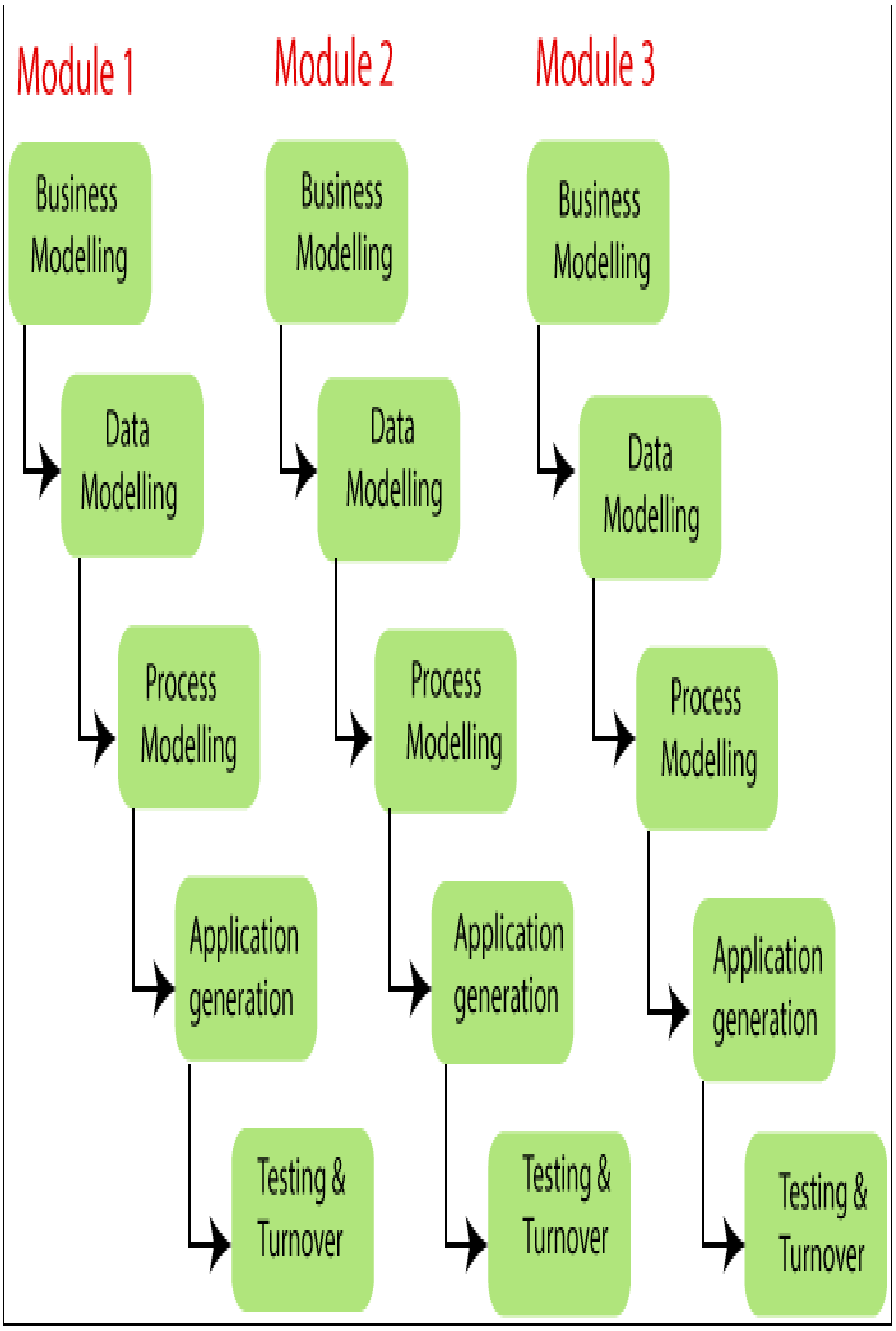


Figure 2.6: Rad Model

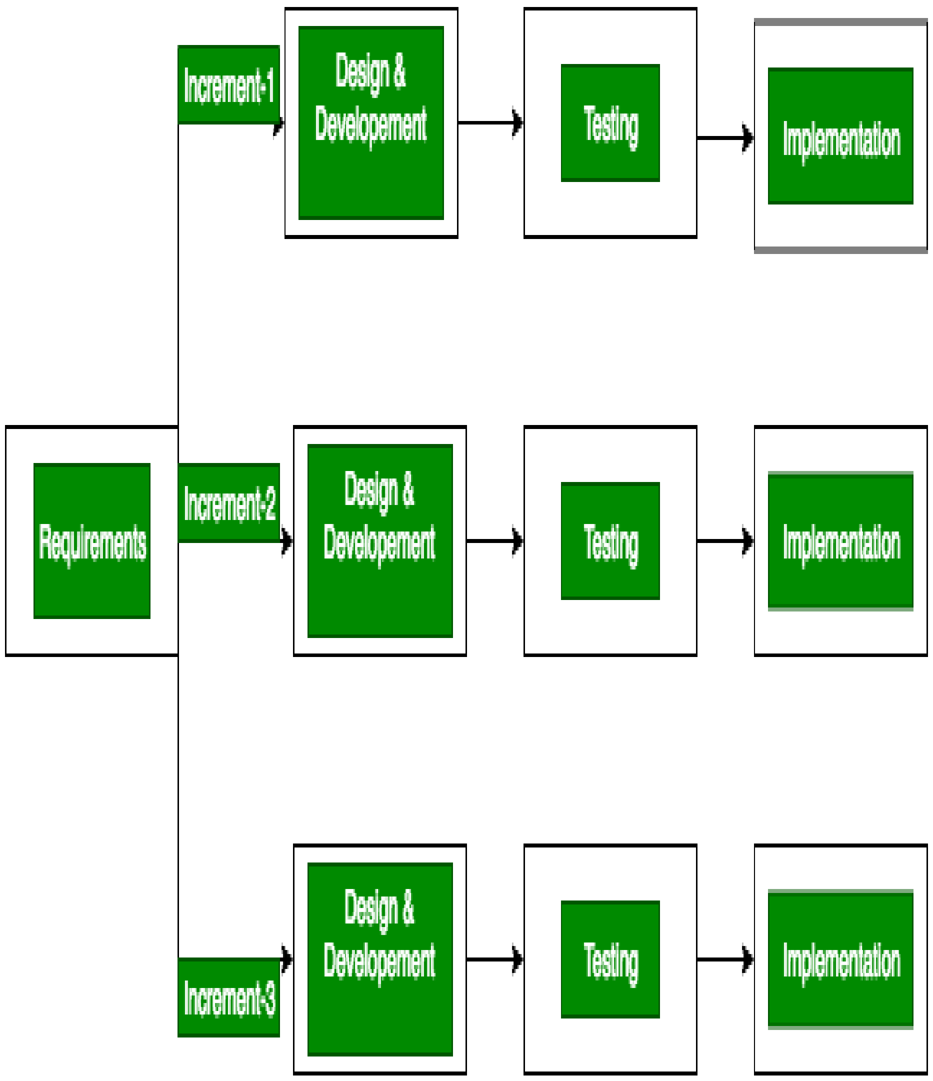


Figure 2.7: Incremental Model

2.3.3 Model Adopted

Extreme programming is a software building style that stresses product quality improvement and responsiveness to changing client requirements over all other factors. (e.g., cost). According to the definition, it is a type of agile development model. As a technique of enhancing productivity, it encourages frequent releases with short development cycles. These releases are designed to boost the program's efficiency while also increasing its overall quality by implementing certain methods aimed at setting certain checkpoints at which client needs can be approved and met. Extreme Programming is a technique that emphasizes the usage of lightweight procedures. Aspects of the XP lifecycle that must be considered are: planning; designing; coding; and testing Given that this is an iterative methodology, the system is created by breaking down the main project into smaller sub-projects. The entire development cycle, from the design phase to the testing phase, is carried out for a single function. After completing the execution of one function and thoroughly debugging it, the developers move on to the next one. XP is centered on rapid release cycles and continuous communication between developers and stakeholders, i.e., customers and other interested parties. In addition to oral communication, frequent testing, code review, and designing are all essential. Communication is seen as a critical criterion in the XP environment. Communication between all parties involved, including developers, customers, and management, should take place on a regular basis. Extreme programming has been demonstrated to produce superior results in software development when used as a software development approach. (Yasvi, 2019)

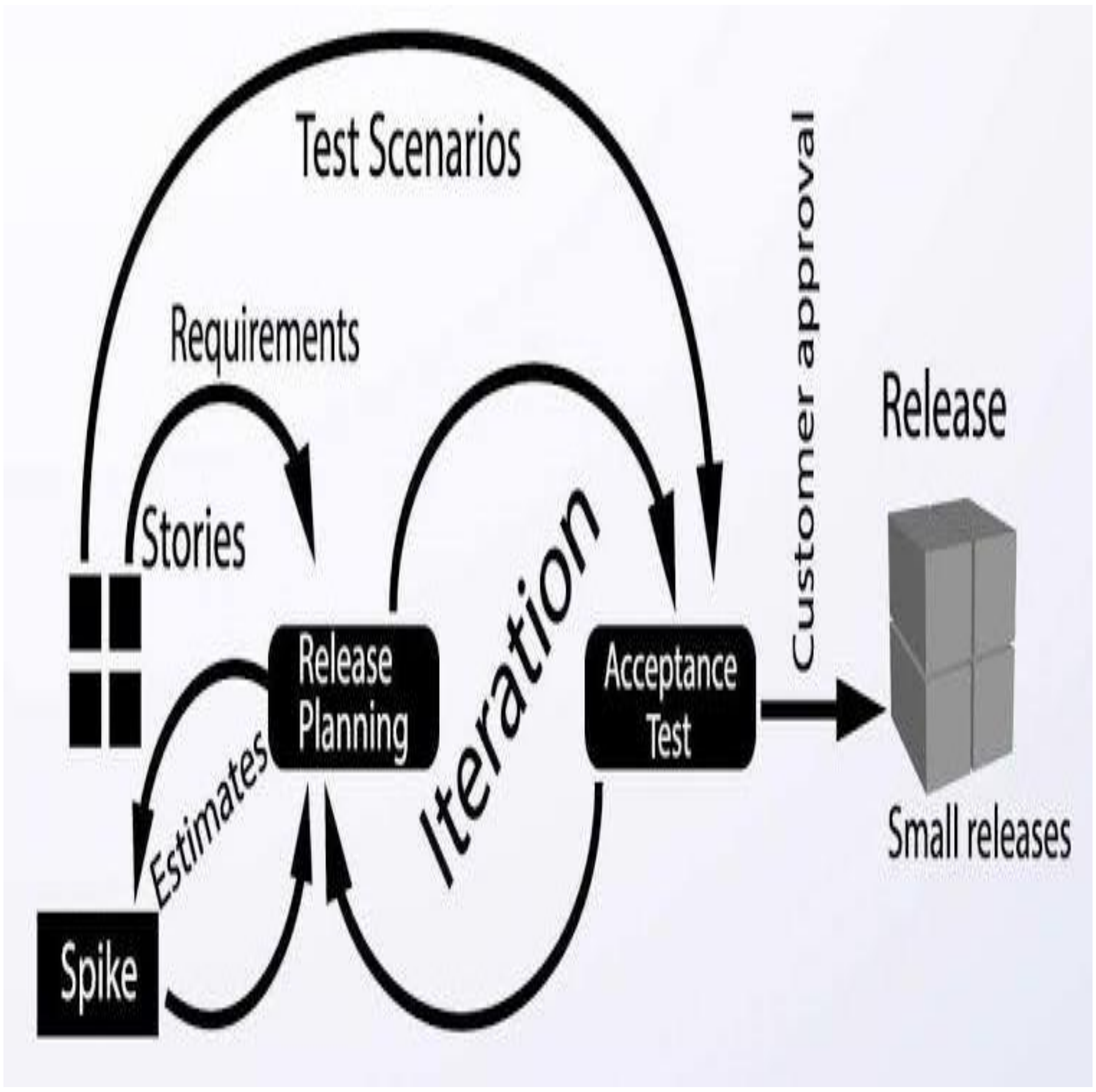


Figure 2.8: Extreme programming

2.4 Unified modeling language (UML)

It was first introduced in the late 1990s, as an evolution of the object-oriented programming language. Its origin is an intersection of the three popular methodologies of the 1980s and 1990s: the Booch method, Rum-baugh’s Object-Modeling Technique (OMT), and Jacobson’s Object-Oriented Software Engineering (OOSE). In the discipline of object-oriented software engineering, the Unified Modeling Language (UML) is a recognized general-purpose modeling language. UML is a set of methodologies for creating visual models of object-oriented software systems using a set of graphic notation techniques. UML is a modeling language that incorporates data modeling, business modeling, object modeling, and component modeling methodologies. It may be used across the software development life cycle and with a variety of implementation technologies. (Padmanabhan, 2012)

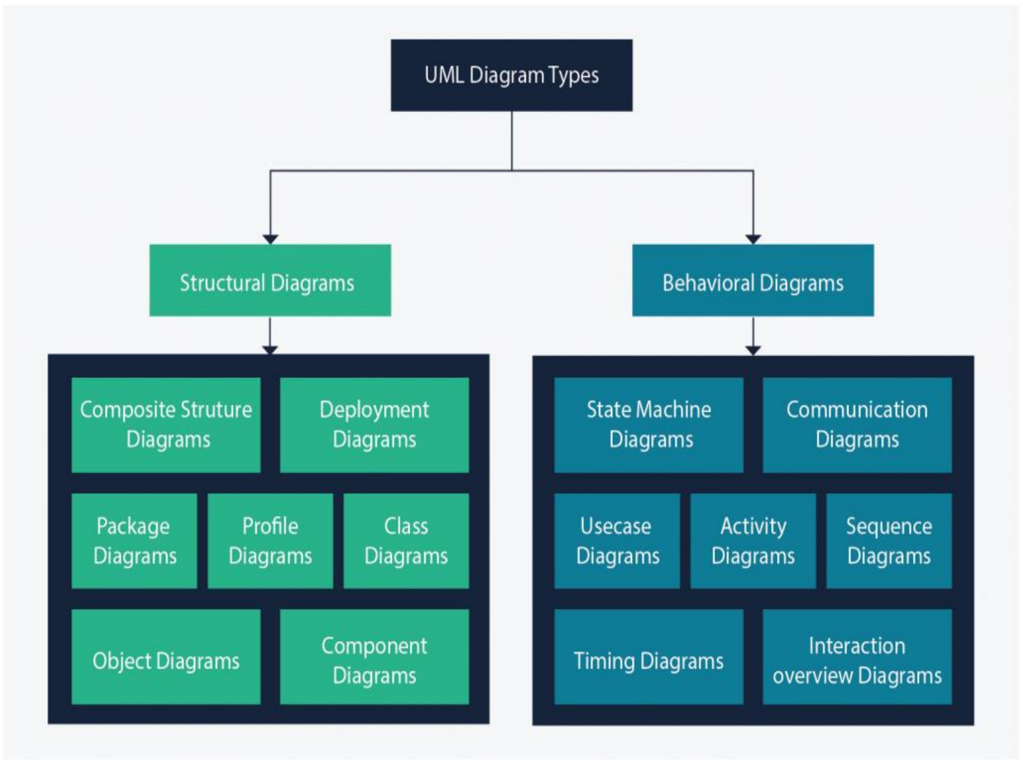


Figure 2.9: Uml diagram

Two views of a system model are represented by UML diagrams:

a. **Static (or structural) view**

Structure view show the things in the modeled system. In a more technical term, they show different objects in a system. Using objects, characteristics, actions, and relationships, this viewpoint emphasizes the system's static structure. Ex: Class diagram, Composite Structure diagram.

i. **Class diagram**

In any object-oriented solution, class diagrams serve as the fundamental building element. The attributes and actions of each class, as well as the relationships between them, are used to depict system classes. A class is usually broken into three portions in most modeling systems. The name is at the top of the page, the characteristics are in the middle, and the operations or procedures are at the bottom. Creating class diagrams is useful in large systems with numerous related classes since it allows for easier grouping of classes. Different types of arrows are used to depict different types of links between classes.

ii. **Components diagram**

A component diagram shows how the structural connections between the components of a software system are made. When dealing with large, complex systems with several components, these are commonly used. Components communicate with one another through interfaces. Connectors are used to link the different interfaces together.

iii. **Deployment diagram**

The physical components of your system, as well as the software that runs on them, are depicted in a deployment diagram. When your software solution is distributed across multiple workstations, each with a different configuration than the others, deployment diagrams are essential.

iv. **Objects diagram**

Object diagrams, also known as Instance diagrams, show the relationships between objects in a similar way to class diagrams. They're similar to class diagrams in that they show how items relate to one another, but they employ real-world examples. They show how a system will appear at a given moment in time. Because the objects include data, they can be used to depict complicated relationships between different entities.

v. **Package diagram**

A package diagram, as the name suggests, depicts the relationships that exist between distinct packages in a given system.

vi. **Profile diagram**

Profile diagrams are a new sort of diagram that was introduced in UML version. In any standard, this is a diagram type that is only very infrequently used.

vii. **Composite structure diagram**

In order to demonstrate the internal structure of a class, composite structure diagrams are utilized. Some of the most often encountered composite structure diagrams.

b. **Dynamic view (or behavioral) perspective**

The behavioral view depicts what should take place in a system. In order to establish a functional system, they define how the objects interact with one another. This view draws attention to the system's dynamic activity by presenting inter-object cooperation as well as changes in the internal states of individual objects. Sequence diagrams, activity diagrams, and state machine diagrams are examples of diagrams.

i. **Use case diagrams**

In addition to being the most well-known diagram type among the behavioral UML kinds, use case diagrams provide a visual representation of the players involved in a system, the various functions required by those actors, and the way these various functions interact. It is an excellent beginning point for any project conversation since it allows you to quickly identify the primary actors engaged as well as the primary processes of the system under consideration.

ii. State machine diagrams

With the exception of the notations and vocabulary, state machine diagrams are very similar to activity diagrams. State diagrams, state chart diagrams, and state chart diagrams are all terms used to describe them. These are crucial for defining the behavior of objects that act differently depending on their current state. The essential states and actions are depicted in the state machine figure on the right.

iii. **Sequence diagram**

Sequence diagrams in the Unified Modeling Language (UML) depict how objects interact with one another and the sequence in which those interactions occur. It's vital to notice that they only display the interactions that occur in a specific scenario. Processes are shown vertically, and interactions are depicted as arrows to indicate their existence.

iv. Activity diagrams

Activity diagrams and interaction summary diagrams are extremely similar in appearance. However, whereas process flow diagrams depict a sequence of processes, interaction overview diagrams depict a sequence of interaction diagrams. They are a collection of interaction diagrams that are displayed in the sequence in which they occur.

v. Timing diagrams

The appearance of timing diagrams and sequence diagrams is very similar. They're used to show how items behave over time. The diagram is simple to understand as long as there is only one thing to portray. When more than one object is involved, however, a Timing diagram is utilized to represent the interactions between the objects across time.

2.5 System Development Tools

This study shows the technologies which were used to implement the system which are:

a. React native

React Native is a JavaScript framework for building real-time, natively rendered iOS and Android mobile apps. It's centered on React, Facebook's JavaScript framework for developing user interfaces, although it's optimized for mobile rather than web use. To put it another way, online developers may now use a JavaScript editor to create mobile apps that look and feel totally "native."

b. Node j.s

Ryan Dahl created Node.js in 2009 as a cross-platform runtime environment for building server-side applications. It's a type of JavaScript that's run on the server. It was intended to address the problems that platforms sometimes have with network communication performance, such as spending too much time processing web requests and responses. Node.js is a framework for creating fast, scalable network applications that is built on Chrome's JavaScript runtime.

Node.js is lightweight and efficient because it uses an event-driven, non-blocking I/O architecture. It's ideal for data-intensive real-time applications that operate across several devices. Node.js is a server environment that is available for download for free. Node.js is a server-side programming language that may be used with a wide range of platforms. (Windows, Linux, UNIX, Mac OS X, and so on) (Refsnes Data, 1999).

c. Firebase

Firebase was previously known as Envolve, a company that was bought by Google in 2012. Developers may add online chat features into their own websites using an API offered by the company, which was then known as Envolve. It's amazing to observe that people were utilizing Envolve to pass application data that was more involved than just chat messages. Developers were using Envolve to sync application data, such as a game state, across their consumers' devices in real time.

As a result, Envolve's founders, James Tamplin and Andrew Lee, were compelled to remove the chat system from the real-time architecture. Firebase was spun off from Google in April 2012 to provide real-time Backend-as-a-Service for web apps and websites. Firebase is a mobile and web application development platform that provides a plethora of tools and services to assist developers create high-quality apps, expand their user base, and increase their profits. Firebase is an online and mobile application development platform.

Firebase grew into the multifunctional monster of a mobile and online platform that it is today as a result of its acquisition by Google in 2014. (geekyants, 2017)

2.6 Related Works

Over time, different forms of publish/subscribe systems have been developed. The majority of research on publish/subscribe systems has focused on heterogeneous database integration, network centric integration, or both.

According to (Babur et al., 2018), a pub-sub system was used to develop a university notification subscription system. The research implement Amazon web service’s Simple Notification Service (SNS) to allow creation and sending of messages from publisher to subscriber on logical access points and topics of communication channels. It also achieved extensibility of the system by distributing topic sets in a number of message providers while the pub-sub middleware forwards publisher's event to relevant subscribers.

According to (Wentao et al., 2018), a distributed pub-sub communication framework for building management systems over Named Data Network (NDN) called "NDN-PS" that supports data subscriptions from consumer applications running on different platforms and with interests in different data as well as different data consumption semantics. Their design is based on the NDN's Interest-Data primitive.

SIENA is one of the most well-known examples of the publish/subscribe system to be developed in this area. SIENA uses a P2P model of interaction among servers (super-peers terminology) and adopts a language based on attribute-value pairs in order to express notifications, subscriptions and advertisement. SIENA adopted a conventional network based algorithm on shortest paths and minimum weight spanning trees for routing messages. Another hash based publish subscriber model adopts publish/subscribe communication paradigm to WMNs (Wireless Mesh Network). This system automatically selects brokers to store and forward messages from publisher to subscriber. When published events/services match the subscriptions, subscribers receive notifications.

According to (Rahimian et al., 2019) proposed a gossip-based hybrid P2P structure for pub/sub systems, called Vitis. In Vitis, peers are arranged in a ring structure and use a gossip-based peer sampling mechanism to find subscriptions and form connections, resulting in clusters of peers interested in similar topics. Despite the fact that Vitis is able to lower the number of relay nodes, peers with a high social degree have a significant traffic overhead since the rest of the peers want to connect with the social users who share the most social friends.

Another distributed content-based publish/subscribe system was approached with the advent of distributed hash-tables. . The schema, which is a set of principles for picking topics, is used in this approach to automatically generate themes from the content of subscriptions and publications. After some statistical analysis, the application designer can supply the schema, which is application-specific. The schemas that have been used are very much similar to the schemas of RDBMS (Relational Database Management System). With this approach, they have increased the expressiveness of subscriptions compared to the purely topic-based systems.

CHAPTER THREE

METHODOLOGY OF THE STUDY

3.1 Method of Identification of User and System Requirement

The system and user requirements were sourced using secondary sources. These sources include online articles, journals, review of other similar publish/subscribe platforms etc. The analysis of these sources led to the elicitation of the valid user and system requirements. This chapter would also talk about the method of system implementation, database implementation, and front-end implementation.

3.1.1 Identification of system requirements

The requirements of the system were used to design the program. These are what the system was based on during the development stages. Some of these requirements are functional while some others are non-functional in nature.

a. Functional Requirements

The following functional requirements should be met by a publish/subscribe system:

i. Create post: The publisher must be able to create post for the subscriber to view

ii. Delete post: The publisher must be able to delete post from the application

iii. Comment and like a publisher post: The subscriber must be able to comment on and like the post made by the publisher that

iv. Subscribe to a publisher page: The subscriber must be able to subscribe to the publisher's page.

v. Get notification of post: The subscriber must receive notifications whenever a new post is made by the subscriber he subscribed to.

vi. Comment and like a publisher post: The subscriber must be allowed to comment on and like the post made by the publisher that piques his or her interest.

b. Non-functional Requirements

The non-functional requirements include:

i. Security and validation: Access to the system is to be controlled through the use of a unique login, password, and authentication procedure each time. Modules will be designed dependent on the type of user that will be logging into the system.

ii. Authentication and authorization: When a user logs into the system, it should be able to verify that they are who they say they are and grant them authorization to access a particular function.

iii. Usability: There must be ease of navigation for the user.

iv. Reliability: The system should be trustworthy and performing consistently well.

v. Response time: A request made to the system should be responded to immediately

c. Software Requirement

In order for the scheme to function properly, it is necessary that certain hardware and software components be available on the system. The scheme specification is made up of the software and hardware components that make it possible to create the scheme in an efficient manner. The following software was installed and utilized in order to ensure the successful and efficient deployment of the system:

Front-end technologies: ` React Native

Backend technologies:	JavaScript
Database Management System:	Firebase
Local Server:	Node j.s
Web Browser:	Google Chrome, Mozilla Firefox

d. Hardware Requirements

One of the most common sets of requirements defined by any operating system or software application is the physical computer resources, also known as hardware resources. A hardware requirements list is frequently supplemented by a hardware compatibility list (HCL), particularly in the case of operating systems and software applications. An android device is needed with camera and image support and a reliable internet connection. (Android 7.0 and above)

3.1.2 Identification of user requirement

The User Requirements Specification outlines the business requirements for what users expect from the system and how it will be implemented.

a. System Admin requirements

In order to establish accounts for publishers and subscribers, as well as to delete people when necessary, the admin must have the maximum level of privilege and can view the user that has registered on the system and their information.

b. Basic user requirements

There are two basic users on the publish/subscribe system which are:

i. Publisher (educators): They log in using the credentials provided by the administrator and have the ability to create and delete posts.

ii. Subscriber (student): They log in using the credentials provided by the administrator and are allowed to like, follow, and comment on the publisher's page that they are interested in.

3.2 System design methods

The system was modelled using extreme programming which is an agile methodology. It allows for quick iteration and testing of ideas due to its lean structure. At each phase of the development process, an increment of the software is developed with all the specified features planned out for the release cycle implemented. The implemented system architectural components were designed with the aid of UML diagrams which showed the interactions between the system users and the system components.

3.2.1 Architectural diagram

The architectural diagram is a system diagram that encapsulates the overall outline of the software system as well as the relationships, restrictions, and boundaries between its many components. The architectural design of the system, the components that make up the system, and their interactions are depicted in this figure below.

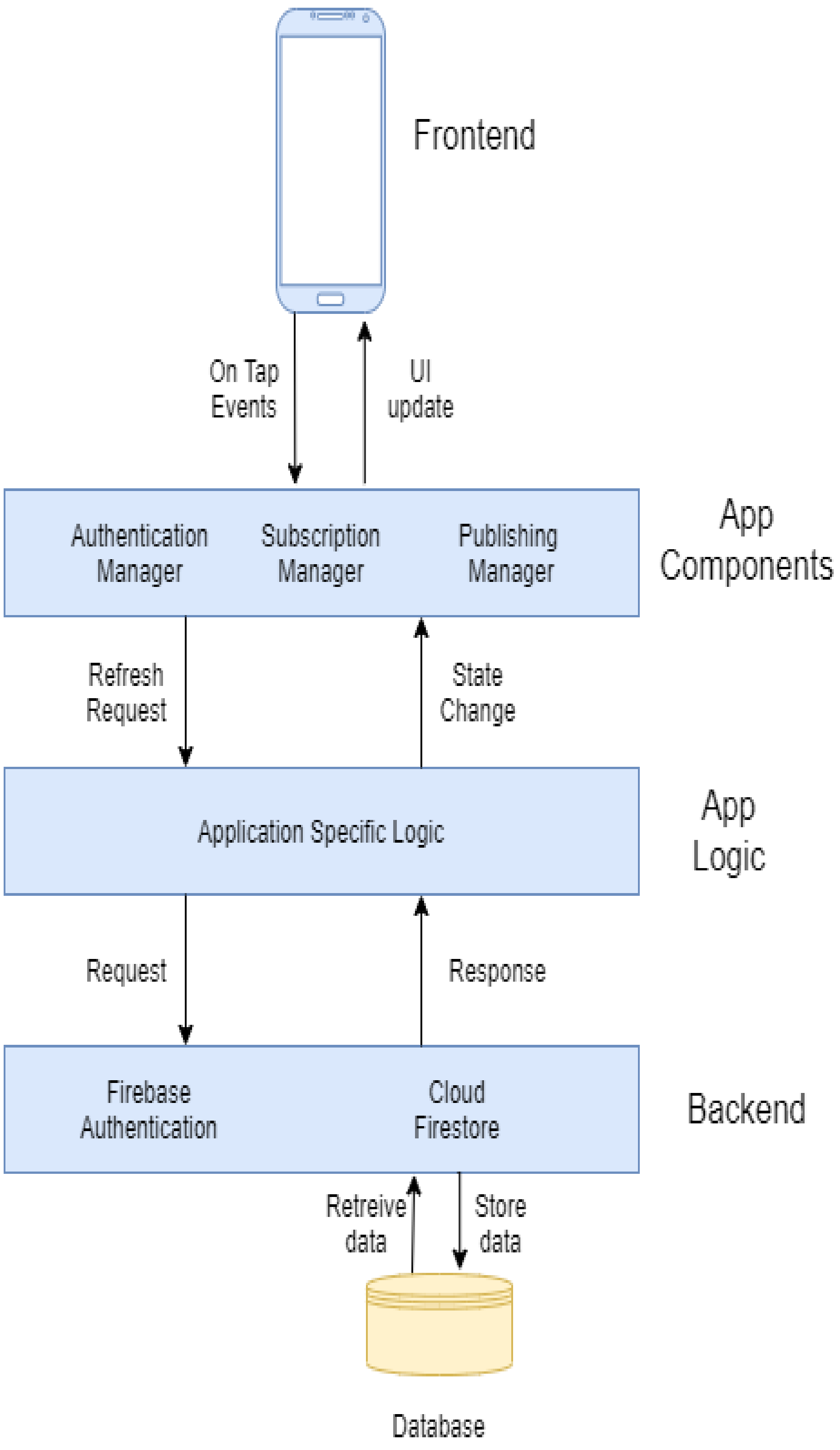


Figure 3.0: Systems architectural diagram

3.2.2 Use case diagrams

This section illustrates the functional requirements of the identified users via the UML use case diagrams.

a. Admin use case diagram

Figure 3.1 below describes the admin use case diagram of the system. The admin is in charge of overseeing the interactions between the publisher and subscriber, two other users. Below is their use case diagram showing the expected functionality from the system.

b. Publisher use case diagram

Figure 3.2 below describes the publisher use diagram of the system. The publisher user type is responsible for churning out content in which the subscribers subscribe to. They are responsible for moderating their content on the platform.

c. Subscriber use case diagram

Figure 3.3 below describes the subscriber use case diagram of the system. The subscriber class of users are responsible for subscribing to content posted by the publishers and reacting to said content via comments and likes.

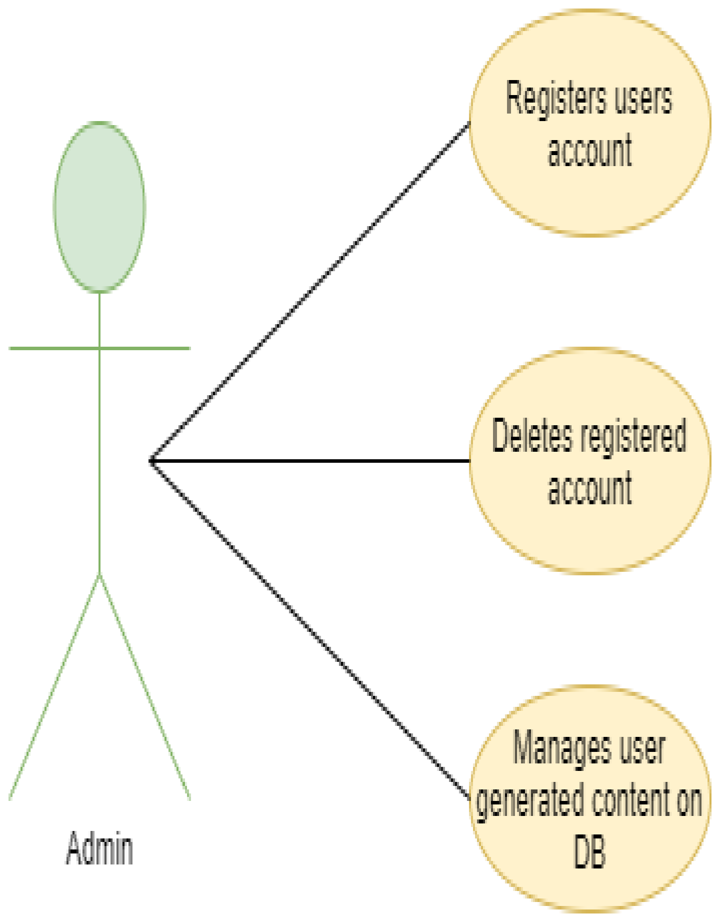

Registers users account
Deletes registered account
Manages user generated content on DB
Admin

Figure 3.1: Admin use case diagram

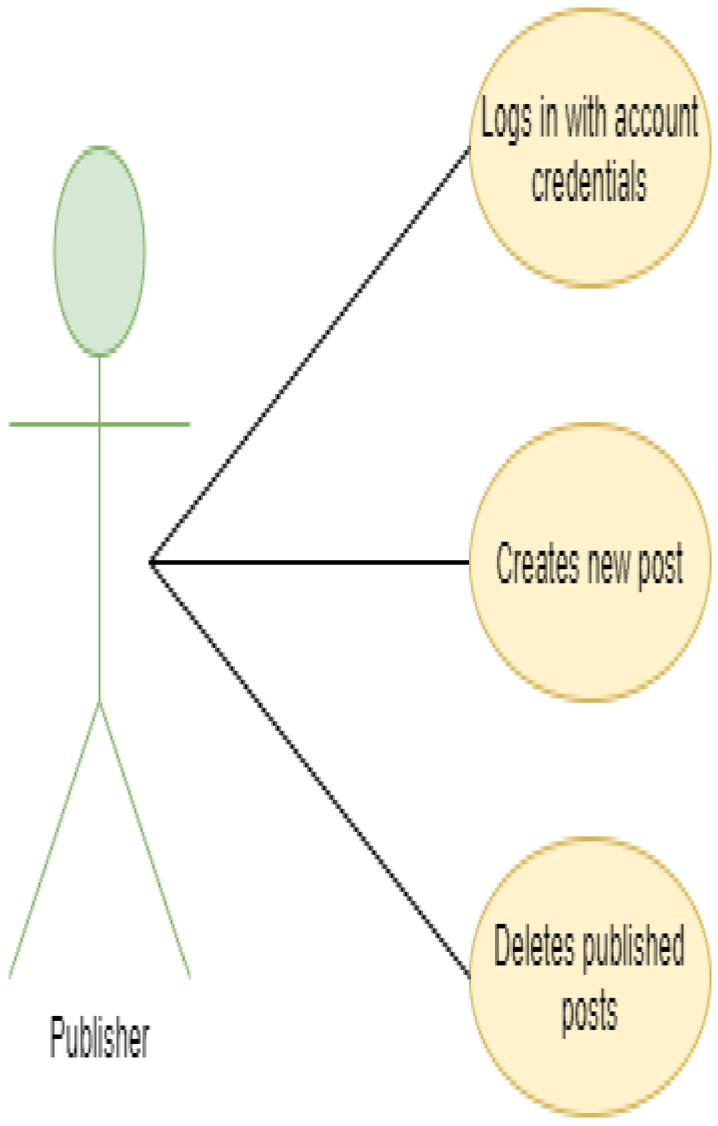


Figure 3.2: Publisher use case diagram

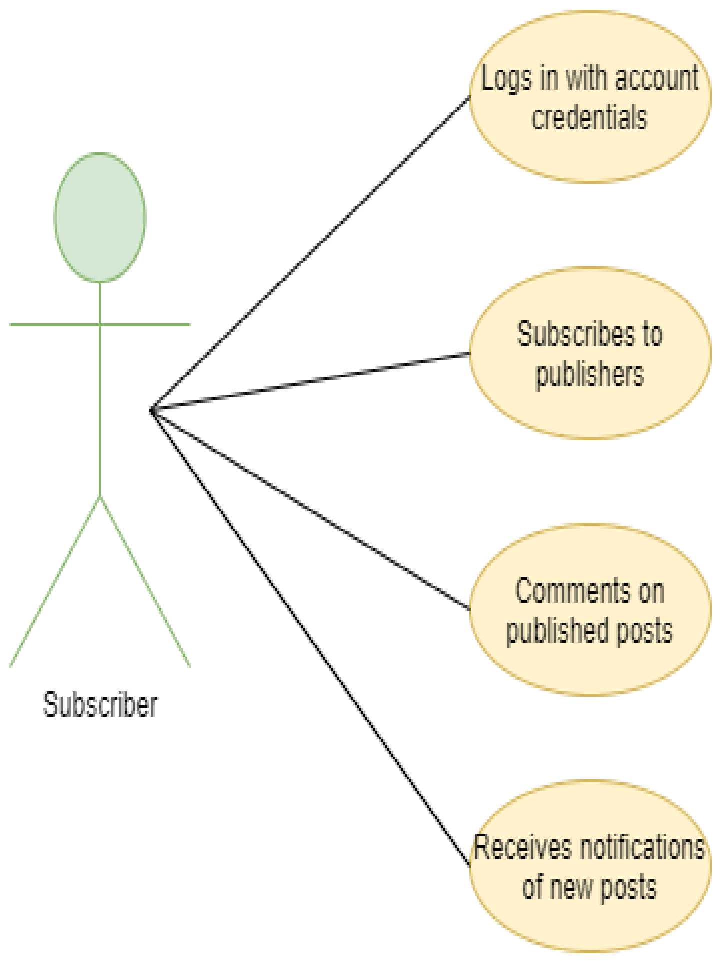

Logs in with account credentials
Subscribes to publishers
Comments on published posts
Receives notifications of new posts
Subscriber

Figure 3.3: Subscriber use case diagram

3.2.3 Sequence diagram

Figure 3.4 below describes the sequence diagram of the system. Because it illustrates how and in what order a group of items interacts, a sequence diagram is a form of interaction diagram. It describes the scenario's objects, as well as the sequence of messages that must be sent between them in order for the scenario's functionality to be carried out. The diagram illustrates the interactions between users of the publish/subscribe platform and the implemented system itself.

3.2.4 Activity diagram

Figure 3.5 below describes the activity diagram of the system. The diagram illustrates the applications workflow from start to finish via the use of an activity diagram. It shows the transition from certain states to the other and the corresponding inputs which take one state to the other.it portrays the control flow from a start point to a finish point showing the various decision paths that exist while the activity is being executed.

3.2.5 Class diagrams

Figure 3.6 and 3.7 below describes the class diagram of the system. It depicts a system's structure by displaying the system's classes, properties, operations, and object relationships. The system's identified objects, as well as their accompanying variables and methods, are depicted in the class diagram below. The user class consists of properties that describe the user of the system. (Registration officer, students and lecturers).

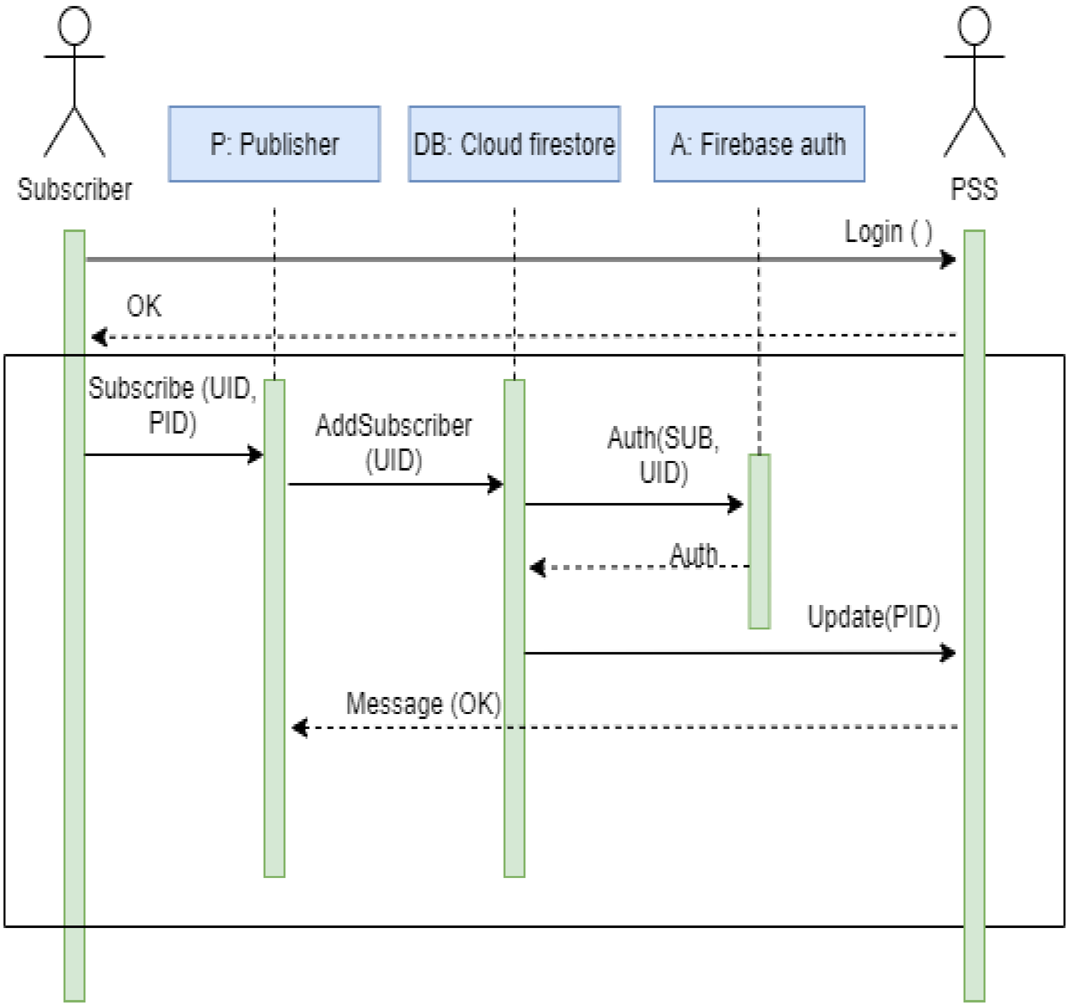


Figure 3.4: Sequence diagram illustrating subscription process

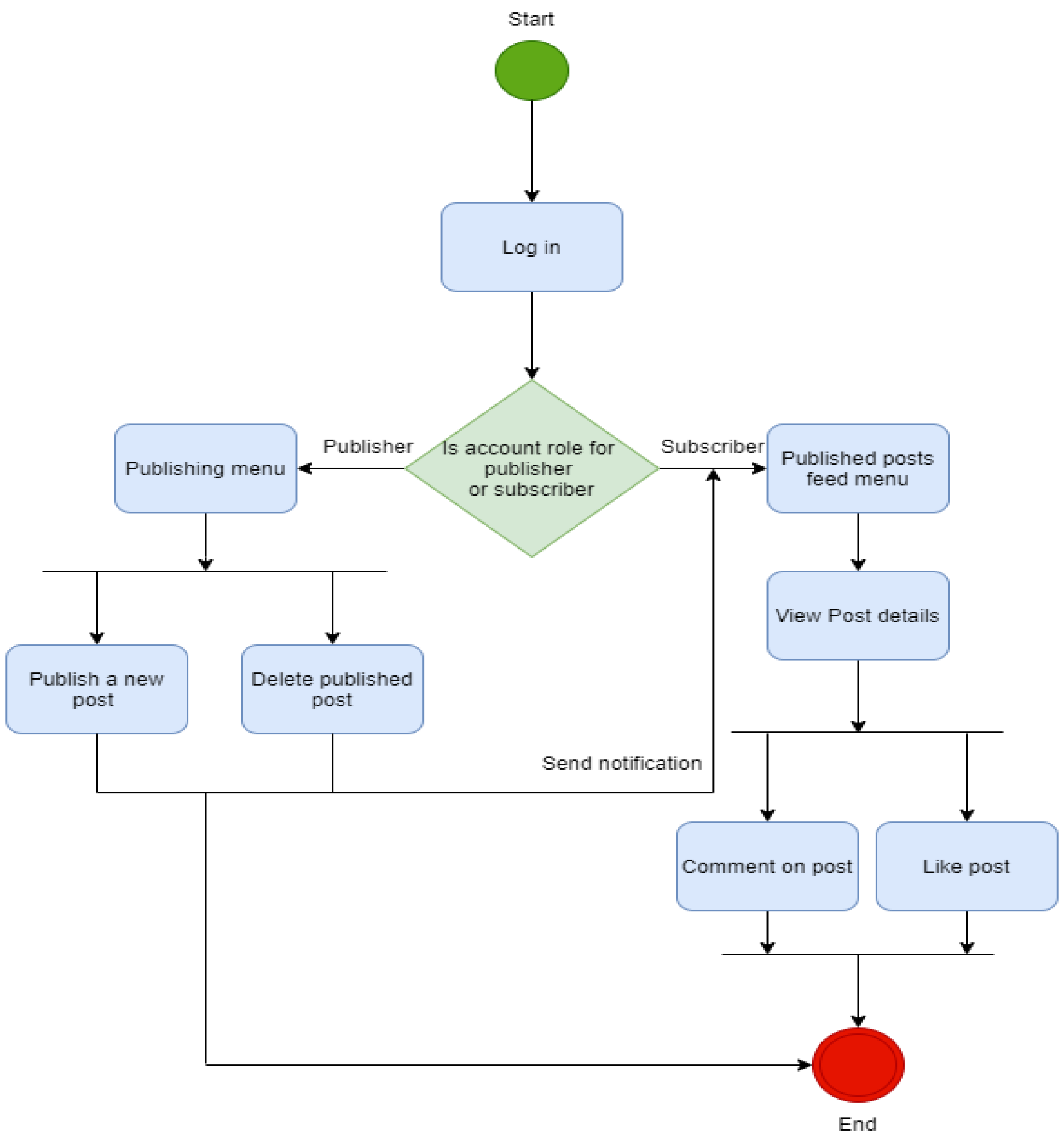


Figure 3.5: Activity diagram illustrating the applications workflow

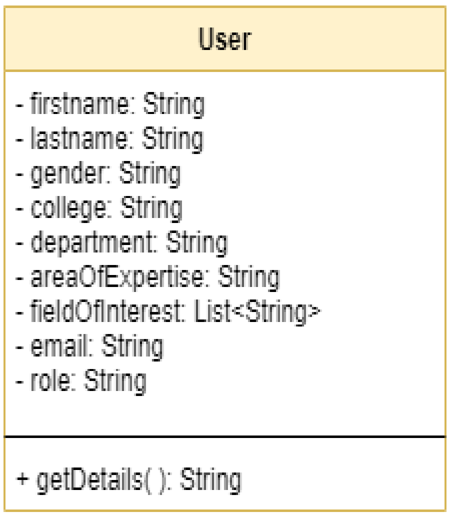


Figure 3.6: User class diagram

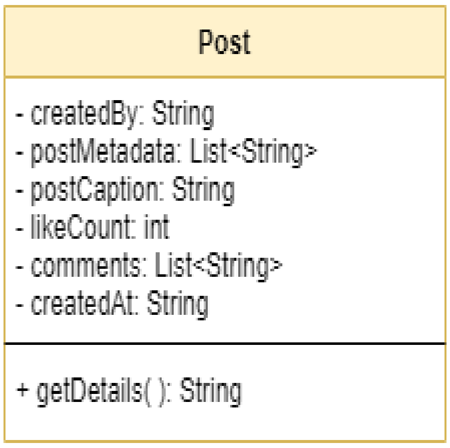


Figure 3.7: Post class diagram

3.3 System implementation

The mobile user interface for the system was developed using the React Native framework. Class components were constructed to control screens, navigation, and other application functionality, such as authorization, for which java script code was written to manage the authorization of users based on the role field in the class component definition. Other libraries were added to provide functions such as picture selection, form validation, and other similar tasks.

3.3.1 Database implementation

Firebase was used to store user personal data, post data, and subscription details in several collections, including: user collection, post collection, subscription collection, subscriber's collection, user registration, authorization, and collection management for users.

3.3.2 Frontend implementation

The frontend and logic of the system got implemented using the React native mobile application development framework. The application screen was generated with class components, and libraries like expo document picker, react redux, and others were deployed for functionality like device directory access and static management.

Node j.s is a runtime server that can be used to execute JavaScript applications. It was utilized in this project to host the react native application, as well as the Node package manager (NPM) to manage the application dependencies.

CHAPTER FOUR

IMPLEMENTATION AND RESULT

4.1.1 Introduction

The results of the publish/subscribe system are discussed in this chapter, along with a description of the outcomes. The data obtained of the database, which was implemented using firebase, and the frontend implementation of the publish/subscribe system, which was designed using react native and JavaScript will be addressed in this chapter.

4.2 Result of database implementation

a. The user collection page

The figure 4.1 indicates the personal data within the system. It includes fields such as college, department, email, field of expertise, first name, gender, interests, last name, and roles, as well as first name, gender, interests, last name, and roles. It keeps adequate records for each user generated by the administrator.

b. The subscription collection page

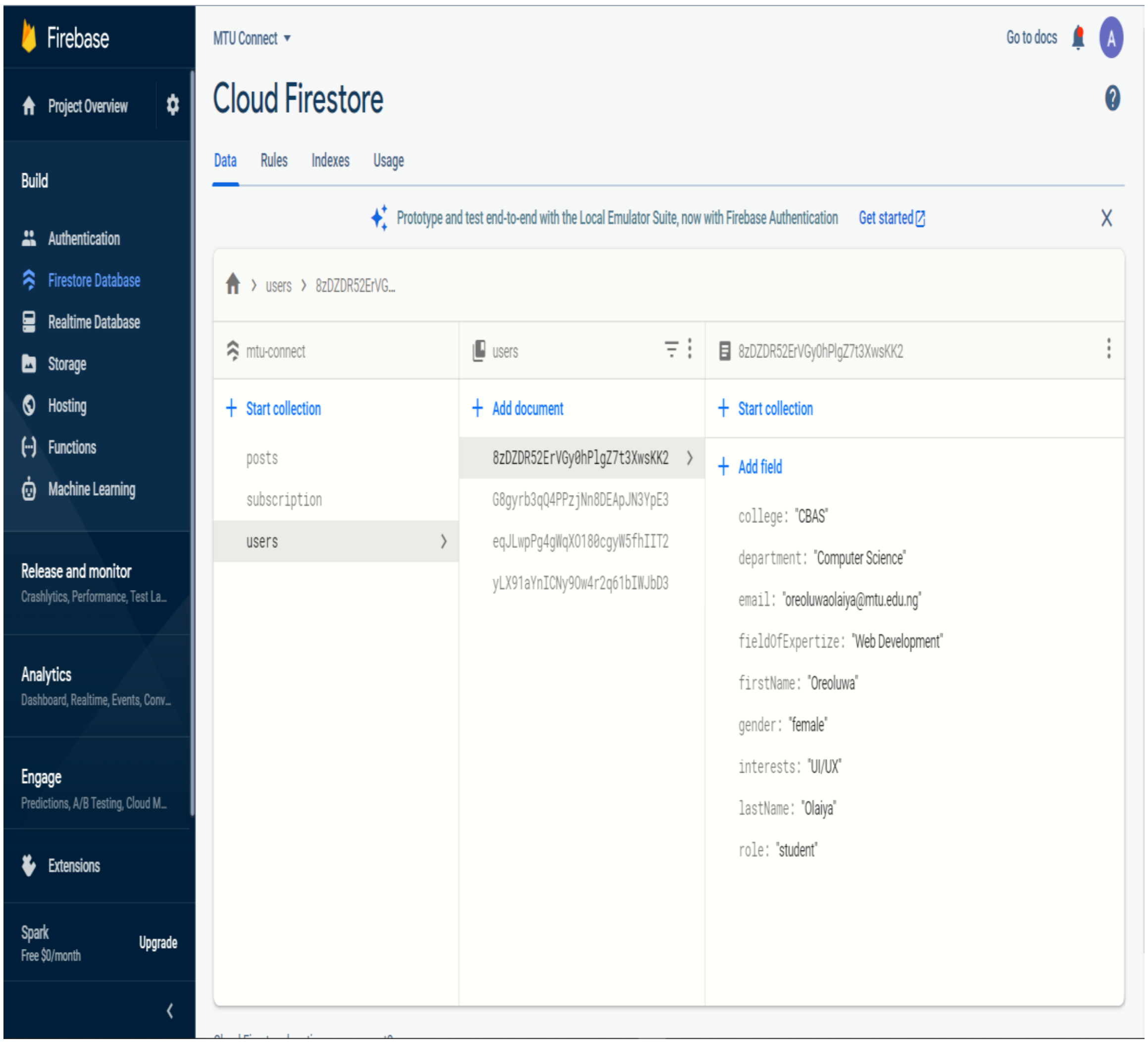
The figure 4.2 gives a list of publisher information. It utilizes the user collection also as main basis and filters by the role field.

c. The subscriber collection page

The figure 4.3 shows the list of subscribers for each publisher in the subscription collection. It also makes use of the user ID to refer to a user collection.

d. The post collection page

Considering educators are publishers in the system, the collection used to keep the data for each publisher and their associated postings is shown in figure 4.4 by referring the individuals table for the user with role "educator."

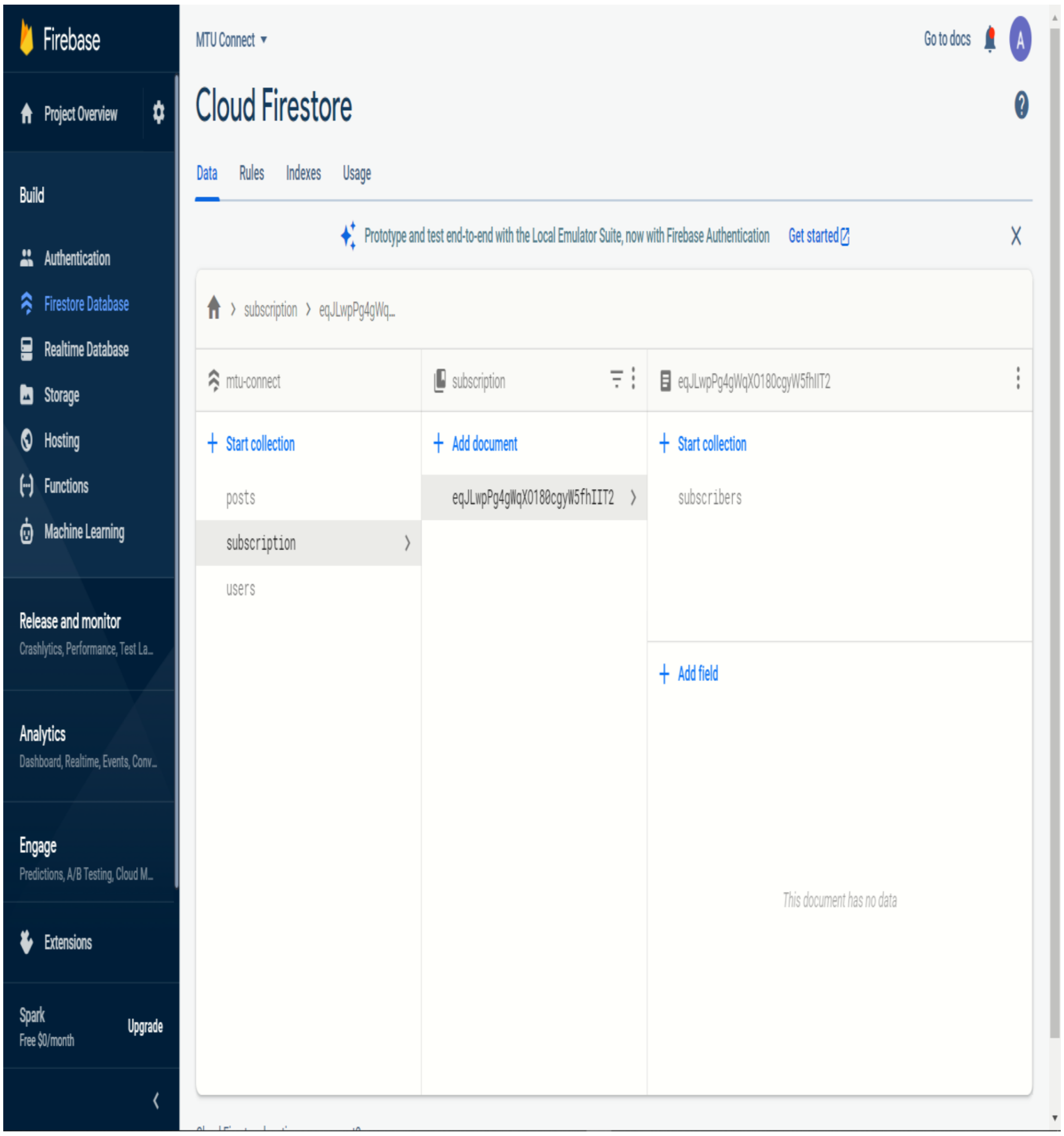


posts
subscription
users

8zDZDR52ErVGy0hPlgZ7t3XwsKK2
G8gyrb3qQ4PPzjNn8DEApJN3YpE3
eqJLwpPg4gWqX0180cgyW5fhIIT2
yLX91aYnICNy90w4r2q61bIWJbD3

college: "CBAS"
department: "Computer Science"
email: "oreoluwaolaiya@mtu.edu.ng"
fieldOfExpertize: "Web Development"
firstName: "Oreoluwa"
gender: "female"
interests: "UI/UX"
lastName: "Olaiya"
role: "student"

Figure 4.1: User collection page

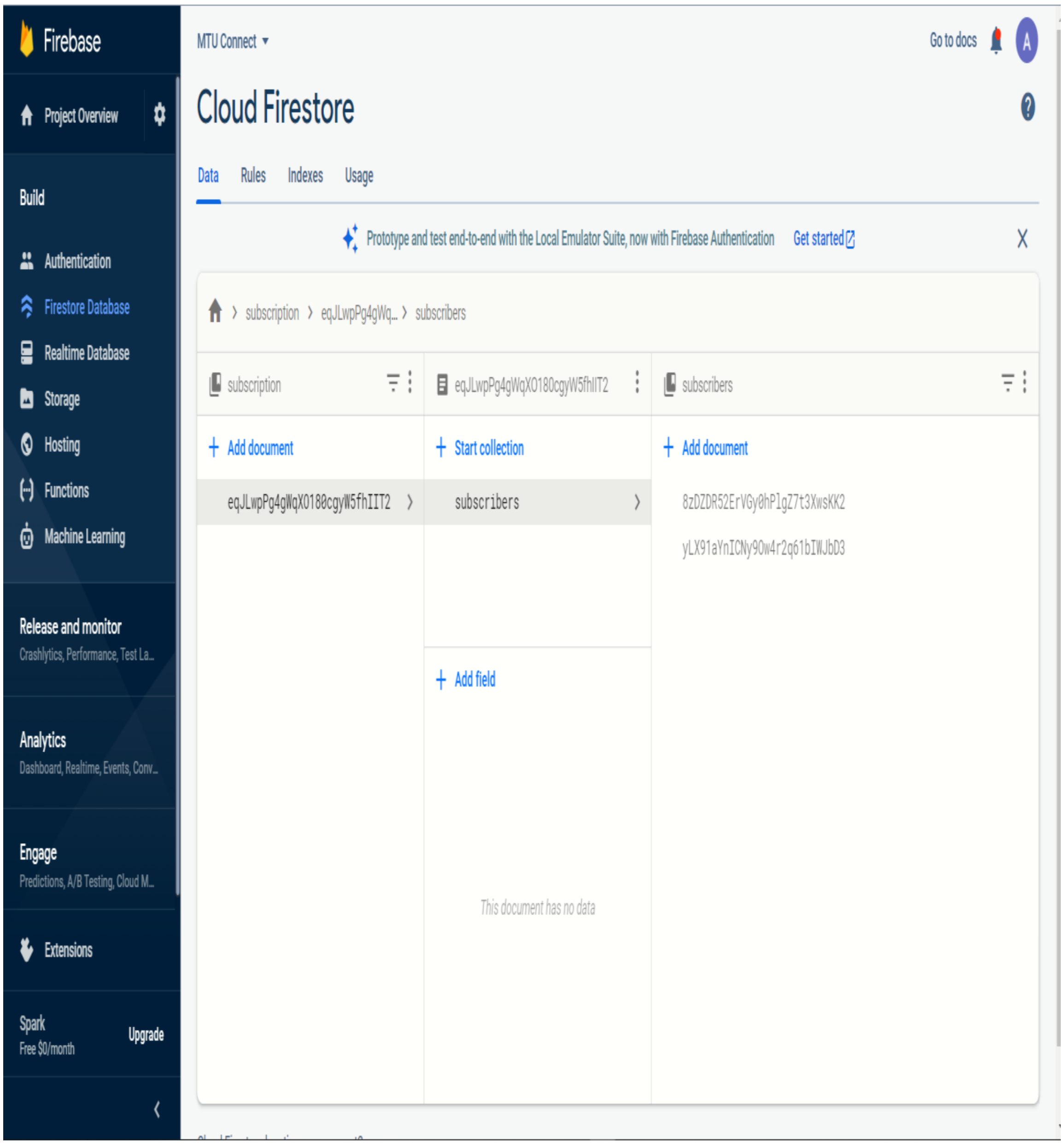


Prototype and test end-to-end with the Local Emulator Suite, now with Firebase Authentication Get started

subscription > eqJLwpPg4gWq...

mtu-connect	subscription	eqJLwpPg4gWqXO180cgyW5fhIIT2
+ Start collection	+ Add document	+ Start collection
posts	eqJLwpPg4gWqXO180cgyW5fhIIT2	subscribers
subscription		+ Add field
users		This document has no data

Figure 4.2: Subscription collection page



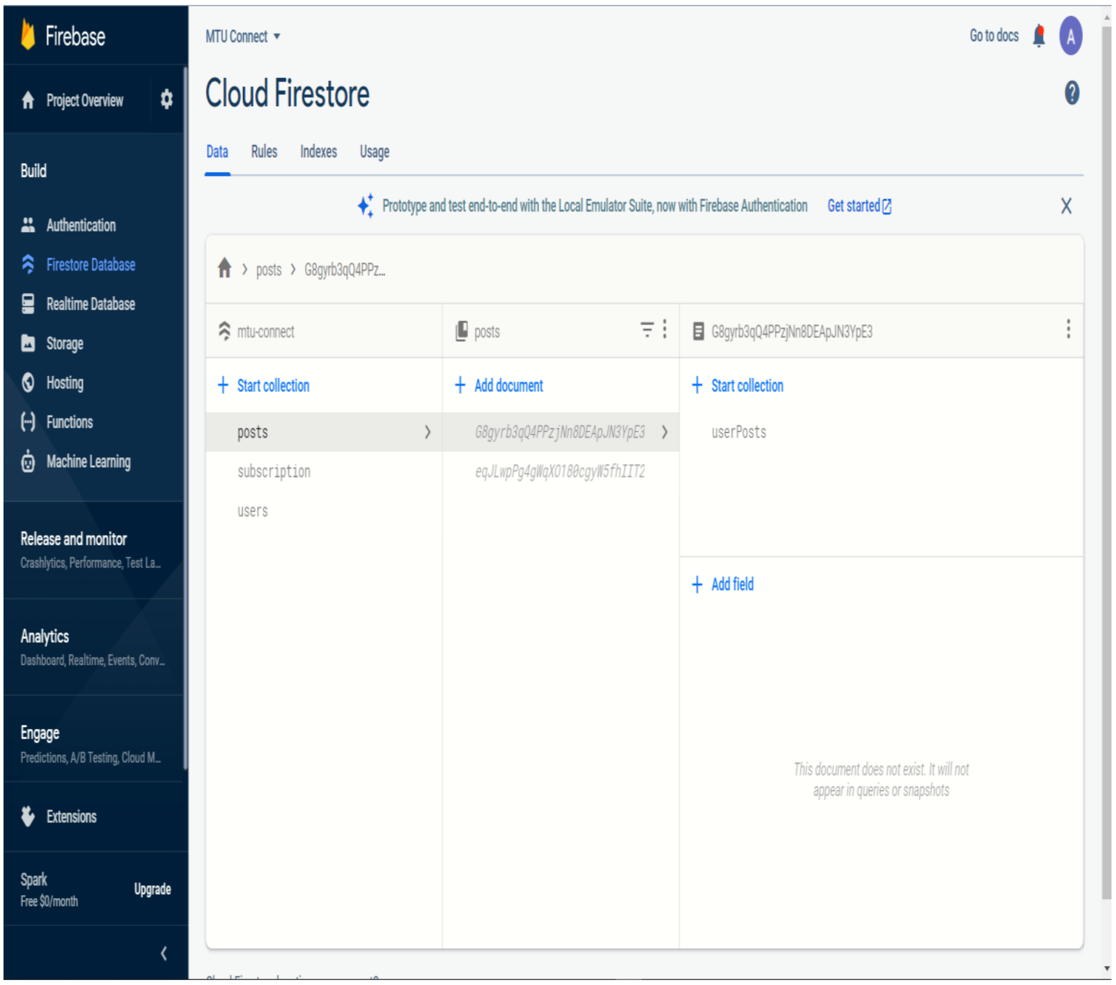
Prototype and test end-to-end with the Local Emulator Suite, now with Firebase Authentication

subscription › eqJLwpPg4gWq... › subscribers

subscription	eqJLwpPg4gWqX0180cgyW5fhIIT2	subscribers
eqJLwpPg4gWqX0180cgyW5fhIIT2	subscribers	8zDZDR52ErVGy0hPlgZ7t3XwsKK2
		yLX91aYnICNy90w4r2q61bIWJbD3

This document has no data

Figure 4.3: Subscribers collection page



This document does not exist. It will not appear in queries or snapshots

Figure 4.4: Post collection page

e. The user post collection page

The figure 4.5 depicts the user post collection page, which contains a list of details for each publisher's entries in the post collection. A post / content that is published by the publisher and exists in the post collection is a documentation in this collections.

4.2 Result of the implementation of the frontend interface

a. The login page

The figure 4.6 shows the login page. Logging in is the procedure by which an individual receives access to a computer system by identifying and authenticating oneself. In this figure below the user is allowed to log in either as a publisher or subscriber.

b. The profile page

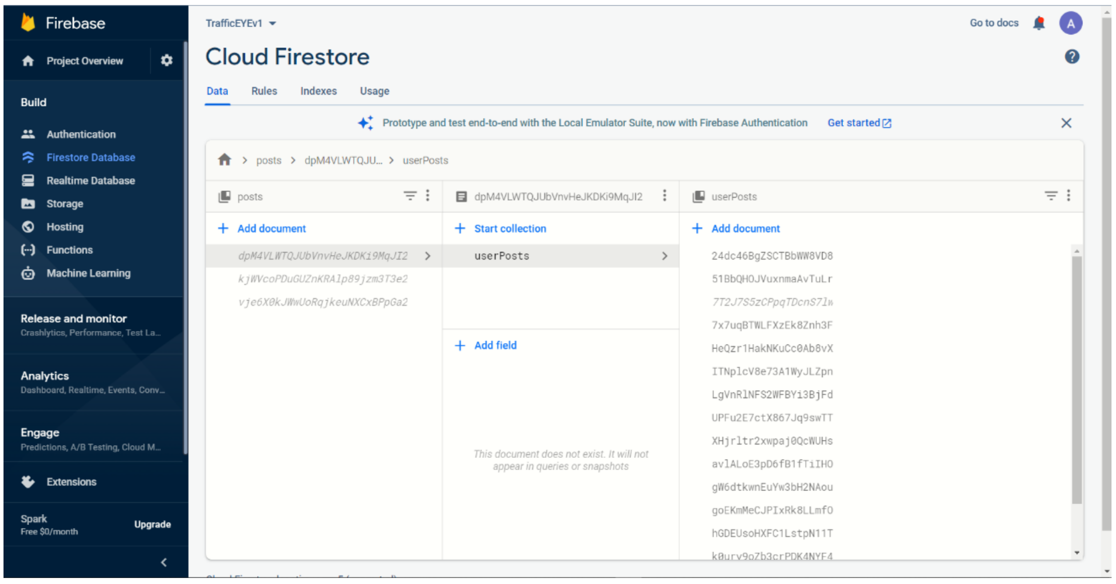
The figure 4.7 shows the profile page. The users' profiles are collections of settings and information that are specific to the individual who created them. The figure is showing the information of the user that logged into the system using the info provided by the administrator. This is a publisher page, he/she is allowed to create post and delete them when necessary.

c. The add post screen

The figure 4.8 shows the add post screen. The image below illustrates that the publisher has the option of posting either by snapping a photo or by accessing files already saved on the device.

d. The search page

The figure 4.9 shows the search page which in order to find a publication page or post that piques the subscriber’s interest, the user is given the option to search.



Cloud Firestore

Prototype and test end-to-end with the Local Emulator Suite, now with Firebase Authentication

posts > dpM4VLWTQJU... > userPosts

posts

dpM4VLWTQJUbVnvHeJKDKi9MqJI2
kjWVcoPDuGUZnKRAlp89jzm3T3e2
vje6X0kJWwUoRqjkeuNXCxBPpGa2

dpM4VLWTQJUbVnvHeJKDKi9MqJI2

userPosts

This document does not exist. It will not appear in queries or snapshots

userPosts

24dc46BgZSCTBbWW8VD8
51BbQHOJVuxnmaAvTuLr
7T2J7S5zCPpqTDcnS7lw
7x7uqBTWLFXzEk8Znh3F
HeQzr1HakNKuCc0Ab8vX
ITNplcV8e73A1WyJLZpn
LgVnRlNFS2WFBYi3BjFd
UPFu2E7ctX867Jq9swTT
XHjrltr2xwpaj0QcWUHs
avlALoE3pD6fB1fTiIHO
gW6dtkwnEuYw3bH2NAou
goEKmMeCJPIxRk8LLmfO
hGDEUsoHXFC1LstpN11T
k0urv9oZb3crPDK4NYF4

Figure 4.5: User post collection page

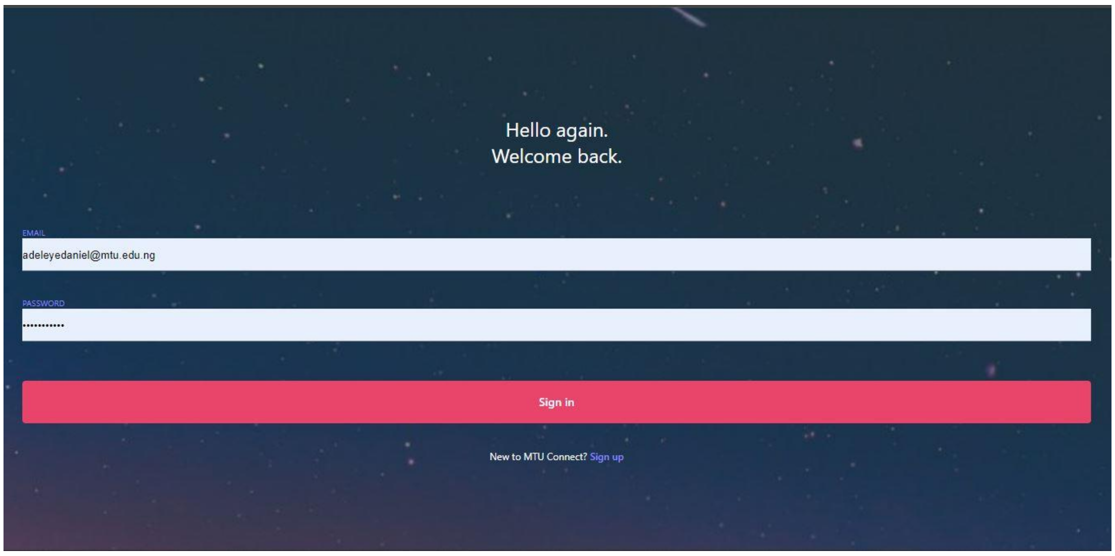


Figure 4.6: Login screen

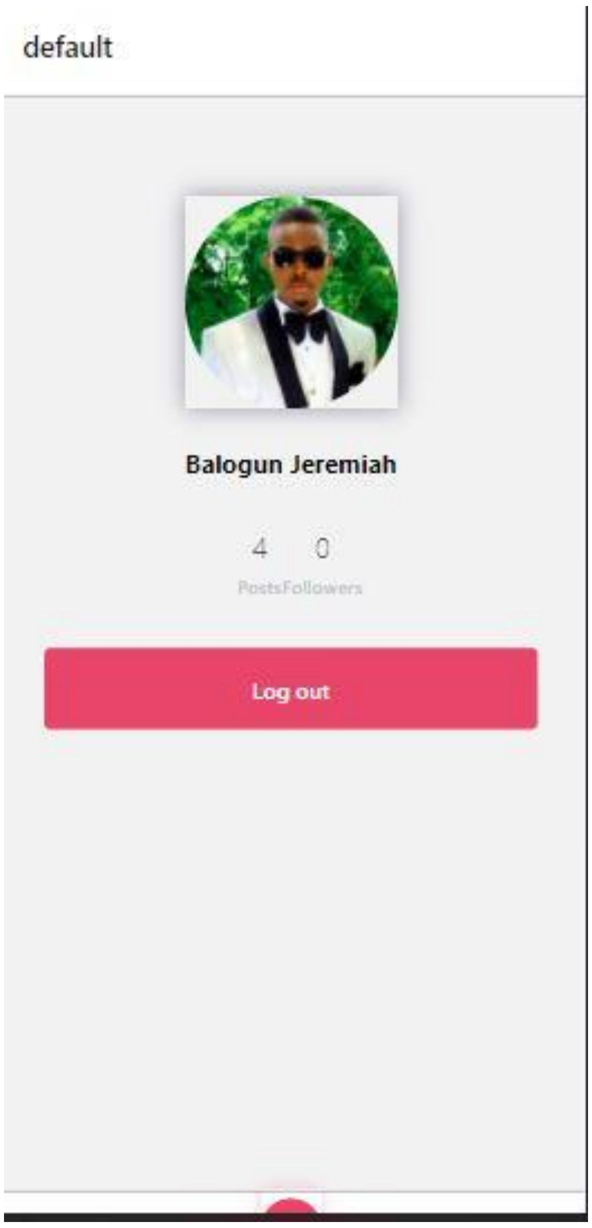


Figure 4.7: Profile screen

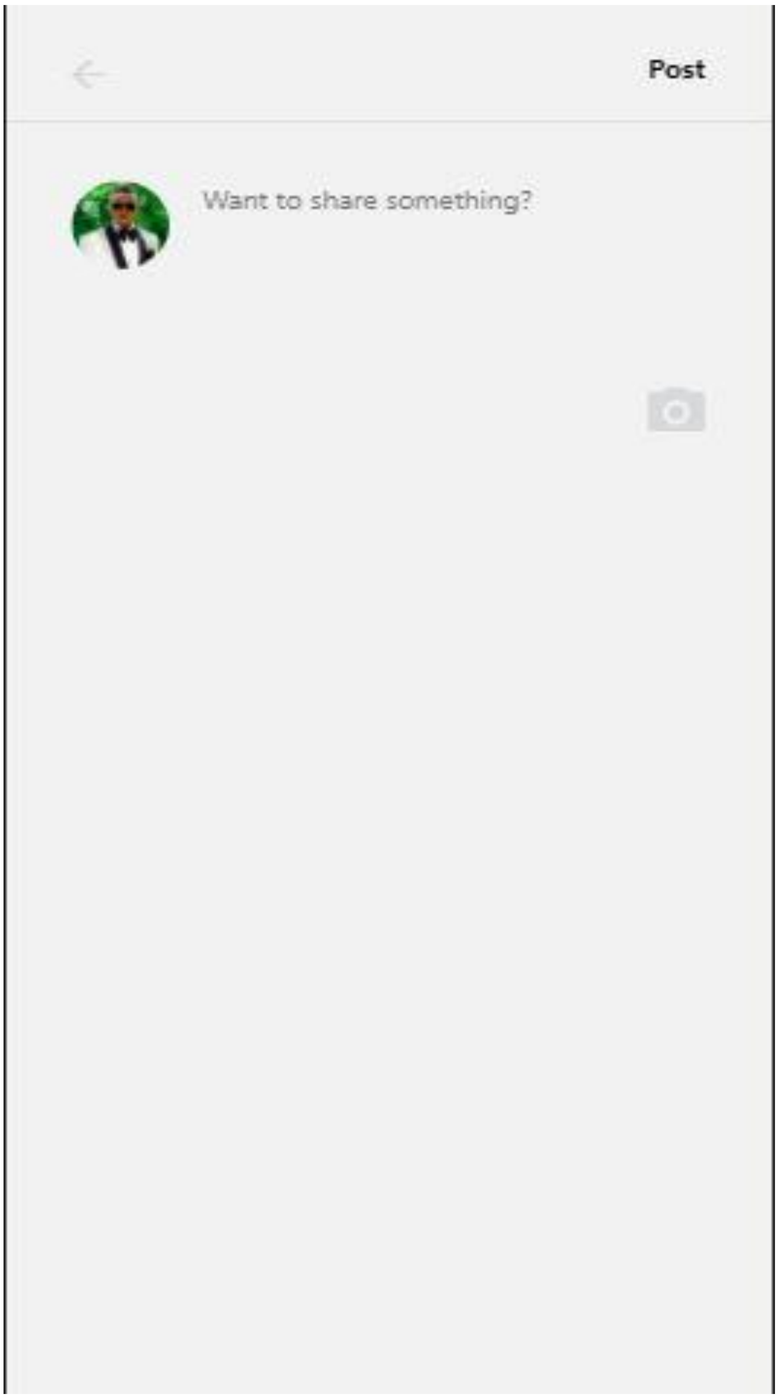


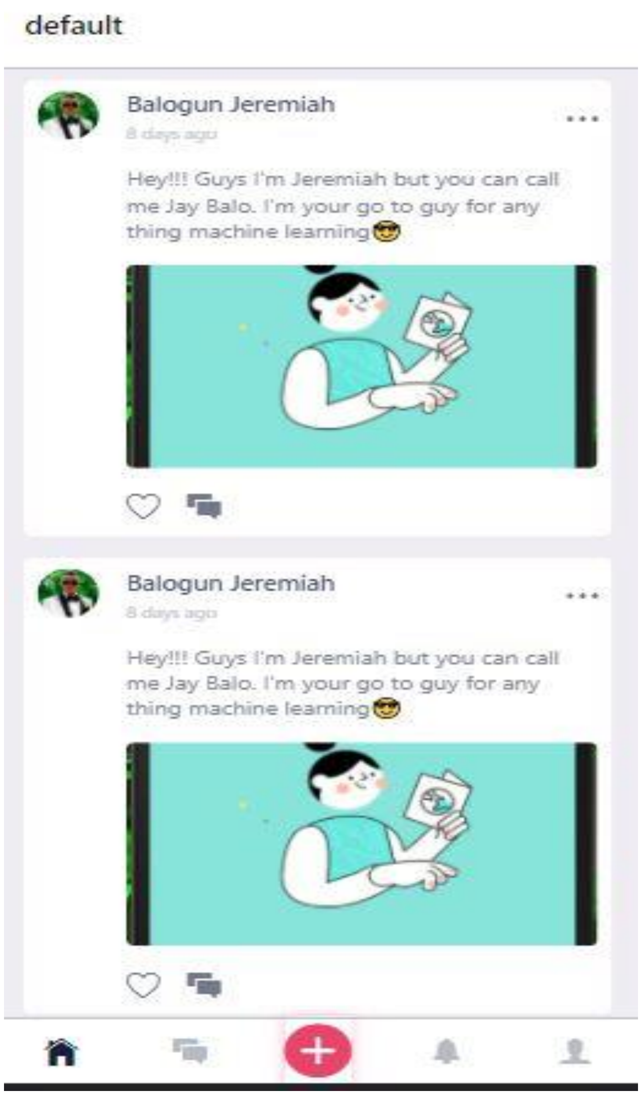
Figure 4.8: The add post screen

default

3 days ago

Hello there!

Figure 4.9: Search page



default

Balogun Jeremiah

8 days ago

Hey!!! Guys I'm Jeremiah but you can call me Jay Balo. I'm your go to guy for any thing machine learning😎

Balogun Jeremiah

8 days ago

Hey!!! Guys I'm Jeremiah but you can call me Jay Balo. I'm your go to guy for any thing machine learning😎

Figure 4.10 the feed screen

4.3 Discussion of Results

The results on the topic of publish/subscribe systems showed desire outcome based on the set goals forth at the beginning. The administrative registration officer, lecturers, and students were identified as system users as a function of the user and system prerequisites having recognized. The administrative officer is in charge of registering students and obtaining their information, according to the results. It implies that the publishers are in responsible for formulating posts for subscribers to engage to. The study demonstrated that a subscriber should subscribe to a publisher page that piques their interest, and thus be informed whenever the publisher submits a post. As a result, it is safe to conclude that the system's outcome deals with the issue of uncertainty or lack of direction in the regarding career path for final academic year or after, as a reaction of the knowledge gap between academic curriculum and practical career criteria.

CHAPTER FIVE

SUMMARY, CONCLUSION AND RECOMMENDATION

5.1 Summary

This research developed a publish/subscribe system that allows professors to post information that piques the interest of students and allows students to subscribe to the publisher of their choice, allowing them to get alerts when that publisher makes a post. During this research, the user and system requirements that the system necessary to meet were identified alongside the software and hardware needs of the system. Unified modeling diagrams such as the use case, activity, sequence, and class diagram were also used to specify the requirements. The react native framework was used to create the frontend, while Firebase was used to handle the database.

5.2 Conclusion

Finally, this study has designed and implemented a publish/subscribe system that solves the problem of indecisiveness in the choice of career path for student in their final year has been constant but the introduction of the publish/subscribe system, which will allow publishers to share articles, pictures, links, and other learning resources and then allow students to decide on the articles that pique his/her interest and is believed to help alleviate the problems. The publish/subscribe system has remarkable features like the subscriber gets notified anytime a new post is posted by the publisher. The app was designed using react native framework, JavaScript, firebase and node j.s.

5.3 Recommendation

It is encouraged that the application be deployed at other universities because it fosters the communication between professors and students and facilitates students to develop a future career through indirect mentorship from the system's publishers.

Reference

Al-Jenaibi, A. a. (2016). Journal of r uoJ. *Social Network and Privacy*, 8.

amazon. (2021). *What is Pub/Sub Messaging*. Retrieved from aws: https://aws.amazon.com/pub-sub-messaging/#:~:text=Publish%2FSubscribe%20(Pub%2FSub,parts%20of%20a%20system%20asynchronously.&text=To%20broadcast%20a%20message%2C%20a,a%20message%20to%20the%20topic.

Barjtya, S. (2017, july 7). International Journal Of Engineering And Computer Science. *A detailed study of Software Development Life Cycle (SDLC) Models*,

Babur, H. M., Zaheer, M. D., Sabah, M. K., Mahnoor, D., Muhammad, H. S., Imran, K., Fatima, M., Hamna, Z., & Asad, A. (2018). University Notification Subscription System using Amazon Web Service. *International Journal of Advanced Computer Science and Applications, 9*(5), 349–354. https://www.ijacsa.thesai.org

Chenxi, H., Peter, R. H., Gareth, A. T., & Kyberd, P. (2007). A Study of Publish/Subscribe Systems for Real-Time Grid Monitoring. *Proceedings, 26–30*, 1–8. https://doi.org/10.1109/IPDPS.2007.370550

Christiansen, L. (2021, january 06). *The 6 Main Types of Information Systems*. Retrieved from altametrics: https://altametrics.com/en/information-systems/information-system-types.html

Eisner, M. (2020, july 24). *processmaker/blog*. Retrieved from processmaker: https://www.processmaker.com/blog/implementing-an-office-automation-system/

Fengyun, C., & Jaswinderpal, S. (2006). *Architecture design for distributed content-based publish-subscribe systems* (No. AAI3214554). Princeton UniversityComputer Science Dept. Engineering Quadrangle Princeton, NJUnited States. https://dl.acm.org/doi/book/10.5555/1195489

geekyants. (2017, december 27). *introduction to firebase*. Retrieved from hackernoon: https://hackernoon.com/introduction-to-firebase-218a23186cd7

Gibb, R. (2019, July 12). *What is pub/sub messaging? A simple explainer*. Retrieved from dev.

Half, R. (2019, may 24). *6 Basic SDLC Methodologies: Which One is Best?* Retrieved from robert half talent solutions: https://www.roberthalf.com/blog/salaries-and-skills/6-basic-sdlc-methodologies-which-one-is-best

Jahan, F. (2013). *SEMANTIC-BASED PUBLISH/SUBSCRIBE SYSTEM IN SOCIAL NETWORK*. North Dakota: North Dakota State University.

jevtic, G. (2019, may 15). *software-development-life-cycle*. Retrieved from phoenixnap: https://phoenixnap.com/blog/software-development-life-cycle

Jackie, G. (n.d.). *The Role of the Educator as a Maker Educator*. ASCD. Retrieved April 5, 2021, from http://www.ascd.org/publications/books/119025/chapters/The-Role-of-the-Educator-as-a-Maker-Educator.aspx

Juneja, P. (2015). MANAGEMENT INFORMATION SYSTEM. *Types of Information Systems - Components and Classification of Information Systems*,

Liebenberg, I. (2017, March 22). *Teacher vs Educator – Which One Are You?* The Eduvation Network. https://eduvationnet.co.za/news_article/teacher-vs-educator-which-one-are-you/

Oracle. (2002, September 1). *Using the Publish-Subscribe Model for Applications*. Retrieved from https://docs.oracle.com/cd/B10501_01/appdev.920/a96590/adg15pub.htm.

Proceedings of 1st IEEE International Conference on Internet of Things Design and Implementation (IoTDI)

Using the Publish-Subscribe Model for Applications. (n.d.). Oracle. Retrieved April 5, 2021, from https://docs.oracle.com/cd/B10501_01/appdev.920/a96590/adg15pub.htms

Padmanabhan, B. (2012). UNIFIED MODELING LANGUAGE. 12.

Radack, S. (2009, april). NIST Special Publication. *THE SYSTEM DEVELOPMENT LIFE CYCLE (SDLC)* , 7.

Qiu, X. (2010). *A Publish-Subscribe System for Data Replication and Synchronization Among Integrated Person-Centric Information Systems* (No. 620). All Graduate Theses and Dissertations.

Shang, W., Bannis, A., Liang, T., Wang, Z., Yu, Y., Afanasyev, A., Thompson, J., Burke, J., Zhang, B., & Zhang, L. (2016). "Named Data Networking of Things,"

Wentao, S., Ashlesh, G., Minsheng, Z., Alexander, A., Jeffrey, B., Lan, W., & Lixia, Z. (2018). *Publish-Subscribe Communication in Building Management Systems over Named Data Networking* (NDN-0066). University of Memphis. http://named-data.net/techreports.html

Yasvi, M. (2019). *Review On Extreme Programming-XP*. delhi: researchgate.

Zwass, V. (2020). Encyclopedia Britannica. *information system*, 10.