CHAPTER ONE

INTRODUCTION

This chapter describes the project and provides some background information. Furthermore, this chapter follows the project's problem description to provide a clear understanding of the project's scope and objectives. This chapter will serve as a blueprint for all subsequent phases of development.

1.1 Background of The Study

Messenger is one of the most useful functions of mobile smartphones. It becomes the primary means of communication, information sharing, and interaction. People generally utilize messenger applications such as WhatsApp, Blackberry Messenger (BBM), and Telegram to do direct communication as Short Message Service (SMS) becomes obsolete. (Setiaji, & Paputungan, 2018)

BBM was popular in the late 2000s, but after a decade, it began to lose momentum to WhatsApp. WhatsApp, which was created in 2009, has evolved to become the most popular messaging program in 109 countries. Facebook bought WhatsApp in 2014 in order to increase its footprint in the rapidly booming messaging market, in addition to its own messenger platform. Telegram, a recent software that behaves similarly to WhatsApp employs end-to-end encryption to safeguard shared information in. Setiaji, & Paputungan (2018), as referenced in Pinto 2014,

The computer science department at Kaduna polytechnic grants admission to a large number of students at the start of every new session, as a new student there are a lot of questions to be asked, due to the number of continuous registrations the department may find almost impossible to respond to the question of the student even after orientation. Students may want to make inquiries ahead of time about the courses that are to offer that very semester and semesters to come, the course credit loads are also important to make inquiries about, the lecturers, and what courses or courses they take. Furthermore, a prospective student would want information about the current development in the department. The need for a department inquiry system arises due to the in-existence nature of the departmental website, an outsider would not know where to search for a particular piece of information or to have knowledge of the kind of information that is available, so, therefore, it is difficult for the person outside school domain to extract information.

A telegram bot is a software program that uses artificial intelligence to conduct an automated online chat discussion with a user in a natural language (human language such as English) through text. Telegram bots provide a solution to this problem by providing all the important information that students need to get comfortable in school. It also helps students navigate through the usual interface used by people every day on their smartphones. (Rinke, 2022)

Telegram bots recognize user input through the use of special commands and patterns called "triggers." When a user types a message that matches one of the triggers that have been programmed into the bot, the bot recognizes the input and responds in a predetermined way. In addition to recognizing specific triggers, bots can also use machine learning algorithms to understand the meaning of more complex user inputs and respond appropriately. (Rinke, 2022)

1.2 Statement of the Problem

Applicants are buzzing with inquiries as the new academic year approaches, and it's surely a busy time for the department to be responding to an overflow of inquiries demanding quick responses by the student such as the courses that they are to offer each semester and semesters to come, the course credit loads are also important to make inquiries about, the lecturers, and what courses or courses they take. In previous years, students and parents had to visit the institution to inquire about specifics and other departmental information, which was a tedious and time-consuming procedure. This is also a time-consuming and resource-intensive procedure for the departmental offices. This may now be done via the internet with telegram bots to save time, energy, and resources.

1.3 Aim and Objectives of the Study

To develop an application based on a telegram bot, that aids students with academic-related information and requests regarding the department.

Objectives

The objectives of this research work are as follows:

i. A responsive GUI that replies to users will be implemented to stimulate a real person conversing.

ii. To implement a system where vital testing will be carried out in ensuring the efficacy of the research work

1.4 Scope of the Study

This research work is centered on the development of a telegram-based application that aids students with academic-related information and requests regarding the department through the use of a telegram bot. The telegram bot will only provide information regarding the courses offered by the department, a list of all the lecturers and their courses, courses credit load, links to lecture materials for download, and the location of all computer lecture halls as the classes are not in one location.

1.5 Limitations of the Study

This study's scope has been constrained by several core issues, including:

Time - The researcher's everyday busy academic pursuits limited the time allotted for research for this study.

Access to literature – Access to some material was restricted, although the available material was optimized.

1.6 Significance of Study

This study will have a potential impact on the students who will be using the application, as well as on the department. By providing students with easy access to academic-related information and resources, the application could potentially improve their engagement with course content and their overall academic performance. Additionally, the telegram bot can manage several queries at once without compromising interaction quality which could help to streamline processes within the department and make it easier for students to get the help they need. By demonstrating the effectiveness of the application, the researchers could potentially inspire other educators to adopt similar approaches and improve the quality of education for students around the world.

1.7 Project Organization

The project is divided into five chapters. The outlines are presented below:

Chapter One: Introduction

Chapter one introduces this project work, the study's background, the problem statement, the purpose and objectives, the scope of the study, the constraints of the study, the relevance of the study, the project organization, and the definition of terms.

Chapter Two: Literature review

This chapter focuses on the literature review, and the contributions of other scholars on the subject matter being discussed.

Chapter Three: Methodology and Design

This chapter is concerned with the presentation of the results of system analysis and design. It presents the research methodology used in the development of the system to facilitate an understanding and effective future implementation of the system.

Chapter Four: System Implementation Evaluation

This chapter describes the system implementation and documentation, analysis of modules, and system requirements for implementation.

Chapter Five: Summary, Conclusion, and Recommendation

The chapter provides a summary of major findings, conclusions, and recommendations based on the study conducted.

1.8 Definition of Terms

i. **Telegram**: A messaging app that allows users to send messages, photos, and other media to each other and to groups of people.

ii. **API** (Application Programming Interface): A set of rules and protocols that allow different software systems to communicate with each other.

iii. **Bot**: A piece of software that performs automated tasks, such as responding to user input or collecting data.

iv. **User interface (UI)**: The part of a software application that the user interacts with, including the layout, buttons, and other elements.

v. **User experience (UX)**: The overall experience of a user interacting with a product or service, including their emotions, perceptions, and behaviours.

vi. **Academic-related information**: Information that is relevant to a student's academic pursuits, such as course materials, homework assignments, and departmental policies.

vii. **Information and communication technology (ICT)**: The use of technology to transmit, process, and store information, including the use of computers, the internet, and other digital devices.

CHAPTER TWO

LITERATURE REVIEW

2.1 Introduction

This chapter seeks to explain how the topic under research relates to prior research, current practice, or other areas of knowledge by citing relevant works by other scholars that have addressed a related issue. Furthermore, this chapter will present a synthesis of current research on the topic, highlighting areas of agreement, disagreement, and gaps in the literature, to demonstrate the project topic's relevance in the field and to recommend opportunities for future research.

2.2 Literature Review

Hari and Irving (2018). Design of Telegram Bots for Campus Information Sharing.

Several problems can arise when using the manual method for campus information sharing such as inaccurate or outdated information, limited accessibility, lack of organization, limited reach, and time-consuming.

The proposed Telegram bot will be implemented at TF UII to improve campus services. The bot uses MySQL as the database system, Python as the programming language, and a security mechanism to distribute information via Telegram. The Telegram bot must supply information such as the TF UII profile, upcoming events, and class or mentoring schedule. The bot will send out information to the registered accounts regularly.

Webhooks and Long polling are the two communication methods available within the Telegram bot. In general, the Webhooks method is easier to develop, webhooks utilize HTTP and respond quickly to requests. However, it requires a dedicated server with a public IP address, whereas long polling does not. The long polling method lowers the cost of developing home automation. It can support a large number of updates without requiring an API call for each one. With the addition of additional devices, such home automation can be made more secure. Webhooks are used in this implementation because the requested information must be provided quickly and continuously.

This article presents the design and prototype of a Telegram bot. The bot can do basic and complex tasks repeatedly, such as invoking and checking database information. The Webhooks approach is

used to communicate with the bot, which employs relatively basic commands. A visual interface may likewise be created using this way. The development of a visual interface for the chatbot becomes the future work.

Ahmadi et al. (2020) Design of Academic Information System Based on Bot Telegram in Smart Campus Concept.

With the present rapid growth of technology, it is conceivable to establish an academic information system that is integrated with the telegram application, given the widespread usage of the telegram application in STTAL educational environments by distributing information to students via a bot. To deliver notifications to the Android application with fresh information provided by lecturers and study program staff, a telegram bot is required, so that students may receive real-time updates.

Several tools were used in the development of the Telegram Bot-Based Academic Information System, including firebase SDK, Telegram and Telegram Bot token, CSS, Dream Weaver, PHP, and Xampp.

The establishment of the Smart Campus idea is an aim for many educational institutions. Smart Campus is the notion of a smart campus gives optimum service to the entire academic community by effectively and efficiently monitoring and managing current resources. Smart Campus can also deliver relevant information to students or campus institutions at all times, including during unforeseen situations. It is recommended in this study to optimize the academic information system that is connected with the Telegram communication program. This Telegram application is free, lightweight, and multiplatform, with a somewhat extensive and better-developed Bot API. Students receive real-time updates and may speak with the Telegram Bot, which is meant to offer all STTAL information. The admin can post course material to the Telegram Bot, which students can subsequently download. As a prototype, 11 commands are constructed in this study. This study was put to the test by executing all of the commands supplied. This Bot Telegram application is quite useful for study program personnel and instructors to communicate information to students, especially because utilizing it is relatively simple.

Rianto et al. (2019) Telegram Bot for Automation of Academic Information Services with The Forward Chaining Method.

The issues in the process of acquiring lecture information that is still not organized in a single application system may be considered the root of the problem, thus the application design is extremely important to optimize the process of obtaining lecture information. This is meant to make it easier to meet lecture information demands by utilizing Telegram bots.

This study employs the Rational Unified Process (RUP) method, which was created by Rational Software Corporation. RUP employs the object-oriented concept, with activities centered on developing models using the Unified Model Language Forward Chaining method and using Python Telepot Framework for Telegram Bot API for applications to run via telegram instant message. Facilitate communication and dissemination of academic information to professors, students, and the academic community by developing this application.

The findings of the White Box testing reveal that the forward chaining approach implemented on the Telegram bot system works effectively and that the search using the inline keyboard works as intended. The results of the Black Box testing demonstrate that the application constructed matched the functional criteria and produced the expected output, particularly in the search feature that employs the inline keyboard, which has been functioning as predicted.

According to the findings of the research, the forward chaining approach is effectively applied to telegram bot lecture information services, and the method can process information quickly and easily with the use of online search boards. In terms of future research, it is vital to provide push notifications when the bot receives new information updates, and bots created can be coupled with other systems.

Selomon (2022). The Role of Telegram Application for Information Sharing in the Case of Online Ge’ez Language Learning

Currently, the Ge'ez language is becoming more well-known among the local and worldwide communities; as a result, the number of Ge'ez language speakers is increasing from time to time (Tadesse, 2018). Learning a second language (or more languages in general) takes a long time and requires a lot of input and interaction (Blake, 2008). Despite this, the Telegram app helps to improve learning and teaching by providing a quick and easy way to create and share knowledge.

The research was carried out on Telegram channels that were launched between 2017 and 2021. They were given online Ge'ez language training. The quantitative research strategy was adopted. To achieve the study's goal, both primary and secondary data were collected. The information was gathered by a search of telegram channels in the internet world, namely those made in Ge'ez language learning user names. To achieve this study goal, the following fundamental information and shared material data were gathered. Both quantitative and qualitative data were examined.

The Telegram channel facilitates Ge'ez learning by direct text posting and sharing of resources such as photographs, movies, files, audio, shared links, voice messages, and GIFs. Basic to advanced Ge'ez language learning was accessible online through the Telegram channel. The channels taught the Ge'ez Alphabet, Ge'ez Numerical, Vocabularies, and the main norms of Ge'ez language writing and reading. It is preferable to infer that the telegram application plays an important part in the growth of Ge'ez language learning in the online system by exchanging large amounts of resources; they are necessary for Ge'ez language learning. As a result, anyone interested in learning the Ge'ez language at any time and from any location might follow the telegram channels investigated in this study.

Sajad et al. (2019). Telegram: An instant messaging application to assist distance language learning.

The usage of various technical gadgets, such as Telegram, distinguishes e-learning platforms. This dynamic environment necessitates the presence of a solid stage for language learners. Many language instructors and curriculum designers, for example, are interested in the ability to exhibit many file types concurrently, such as PowerPoint files, drawings, audio/video files, Macromedia, and animated files. At the same time, because of the interactive character of this online environment, the system of review and continuing assessment may be made much more convenient by employing Telegram.

Telegram is a free application that may be used for online language learning programs and has several features to help with the learning process. It is one of the most downloaded messaging applications that is continually updated and has new features introduced daily. Elekaei (2018) demonstrated Telegram's educational potential as a tool for pursuing online language learning programs by demonstrating statistically significant outcomes in L2 learners' listening progress, vocabulary growth, vocabulary retention, autonomy, and learning strategy training.

Telegram, a free online program, offers it all: a vast cloud-based storage system for data, a venue for arranging collaborative online classes, hundreds of robot helpers, and the possibility to create one's customized robot for any function. As a result, it is a little gadget that eliminates the need for further applications. Its adaptability and user-friendliness make it particularly popular among teachers and students of all levels. However, several unexplored potentials need to be investigated further. Future research might focus on testing students' ability in other skills like speaking, reading, and writing, as well as automated assessments in Telegram.

2.3 Summary of Related Literature Reviews

Author & Year	**Title & Description**	**Merit and Demerits**
Hari and Irving (2018).	Design of Telegram Bots for Campus Information Sharing This article describes the design and prototype of a Telegram bot. The bot may do simple and sophisticated activities repeatedly, such as invoking and validating database information.	An ideal set of programming approaches was used in implementing the system. The bot employed relatively few basic commands.
Ahmadi et al. (2020)	Design of Academic Information System Based on Bot Telegram in Smart Campus Concept. Many educational institutions want to implement the Smart Campus concept. The concept of a smart campus is that it provides optimal service to the whole academic community	Useful for study program personnel and instructors to communicate information to students. Constant revisions, upkeep, and improvement of their knowledge base and how they communicate with users.

	by effectively and efficiently monitoring and managing current resources.	
Rianto et al. (2019)	Telegram Bot for Automation of Academic Information Services with The Forward Chaining Method. The purpose of this research will be to use the forward chaining method as a search for information and knowledge facts on the telegram bot of Siliwangi University academic information services.	The implemented method can process information quickly and easily with the use of online search boards. The bot lacks push notifications for receiving new information.
Selomon (2022).	The Role of Telegram Application for Information Sharing in the Case of Online Ge’ez Language Learning. The purpose of the study was to look into how online learning resources enable knowledge exchange for Ge'ez language learners through the use of telegram.	the telegram application plays an important part in the growth of Ge'ez language learning in the online system. Internet connectivity is required.

Sajad et al. (2019).	Telegram: An instant messaging application to assist distance language learning. Telegram is a free application that may be used for online language learning programs and has several features to help with the learning process	Its adaptability and user-friendliness make it particularly popular among teachers and students of all levels. Unable to test students' ability and other skills like speaking, reading, and writing, as well as automated assessments.

2.4 Analysis of the Current System

The current system of providing students with academic-related information is during the departmental orientation when a student is far gone with lectures and course work, moreover not all the relevant information is passed across, the information passed across focuses on examination and examination malpractice leaving out the courses credit loads and other important information, some students get to find out some course credit load after an examination. Some of the information placed on the notice boards are not recent and some student is working with the information. The manual method of accessing academic-related information can be time-consuming and inconvenient for students, especially if the information they are seeking is not readily available at their campus library or resource center. It may also be difficult for students to locate specific materials or information, or to find assistance from librarians or other staff members if they are not available at the time of the search.

One potential solution for these issues is for the department to offer additional resources or services to help students access academic-related information more easily. For example, the department could provide access to additional online databases or resources that students can use from any location. The department could also offer training or workshops to help students learn how to effectively search for and locate information using various resources.

2.4.1 Problem Inherent in the Current System

There are several problems inherent in the current system of accessing academic-related information by students from the department

i. **Inefficiency**: Manual allocation of students to supervisors and student assessment can be time-consuming and labour-intensive. It may also be prone to errors and omissions.
ii. **Outdate information**: Because the information is placed on the notice board there is no way to find out how recent the information is unless dates are attached to the document.
iii. **Limited scalability:** as the number of students increases, it can become difficult for departments to keep up with the volume of requests and maintain good communication with all students.

2.5 Analysis of the Proposed System

Keeping in mind the aforementioned shortcoming, the suggested approach efficiently addresses the aforementioned issues. A Telegram bot could potentially be a useful tool for providing students with academic-related information and handling requests to their department. One benefit of using a Telegram bot is that it can be accessed quickly and easily from any device with an internet connection, making it more convenient for students to get the information they need. Additionally, the Telegram bot can be programmed with consistent policies and procedures for handling requests, ensuring that all students receive the same level of service. This can help to address some of the issues of inconsistency and lack of transparency that can arise in the manual process.

2.5.1 Advantages of the New Proposed System

i. **Efficiency**: students can quickly and easily access information and make requests through the Telegram bot, without having to go to the department in search of the information.
ii. **Accessibility**: The telegram bot can be accessed from any device with an internet connection, making it easier for students to get the information they need no matter where they are.
iii. **Consistency**: The telegram bot can be programmed with consistent policies and procedures for handling requests, ensuring that all students receive the same level of service.
iv. **Scalability**: The telegram bot can handle a large volume of requests and inquiries, making it easier for departments to keep up with demand as the number of students increases.

CHAPTER THREE

METHODOLOGY AND DESIGN

3.1 Introduction

A methodology is an approach to rigorous study or investigation, particularly to uncover new facts or information; hence, research methodology should be good enough to make the attainment of the established objectives attainable with certain components, such as methods of data collecting and design. This chapter includes the input/output specifications and system requirements for the development of a telegram bot that provides students with academic-related information, as well as the system modeling (use case, activity, and class diagrams).

3.2 Methods of Data Collection

It is crucial to acquire data and facts about the current system before implementing any system since one has to understand what is happening. Three techniques were used to conduct this study.

i. Observation of the Work Environment
ii. Documentation
iii. Interview

3.2.1 Observation of the Work Environment

This approach was used to collect information/data for this study by examining how the manual system was carried out, the method provides varying degrees of control over the context in which they are used, and the careful inspection revealed the obvious flaws in the present system.

3.2.2 Documentation

A secondary form of data acquisition is documentation. Journals, manuals, previous projects, publications, and other sources are used in this approach. This type of data collecting is employed because it provides a foundation for comparison with previous research. This includes the internet, a tool for gathering data. The internet was utilized to find information on topics that seemed challenging or unclear.

3.2.3 Interview

The primary goal of utilizing interviews as a data-gathering strategy is to collect data in a comprehensive and intensive manner. The researcher met with the project coordinators from the department and obtained trustworthy information based on the questions provided by the researcher.

3.3 System Modeling

A system model is a conceptual model that describes and represents a system. Visual models of the object-oriented software-intensive systems can be made using a set of graphic notation techniques that are part of the Unified Modeling Language, which is employed in this modern system design. Use case diagrams, class diagrams, and activity diagrams are among the UML diagrams used in this new design.

3.3.1 Use Case Diagrams

Use cases are collections of interactions between systems and users. Use case diagrams are used to visually summarize a system's functionality in terms of its actors, its goals (represented as use cases), and any dependencies between those use cases.

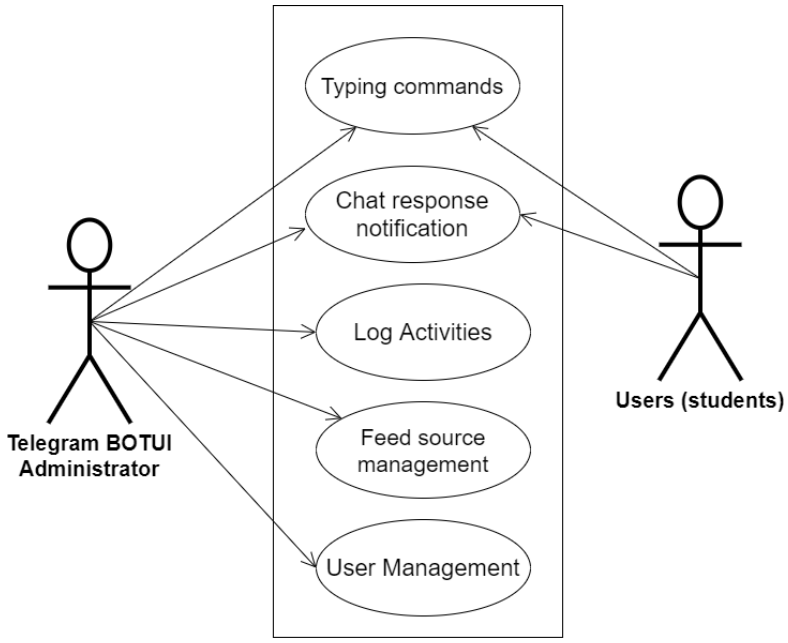

Fig 3.1 System Use Case Diagram

3.3.2 Class Diagrams

The Unified Modeling Language (UML) class diagram is an implementation of an independent view of how the system interface will be, with each class having its own properties and illustrating how they interact with one another. Class diagrams use the rules established by the Unified Modeling Language to visually depict the static structure and composition of a specific system (UML).

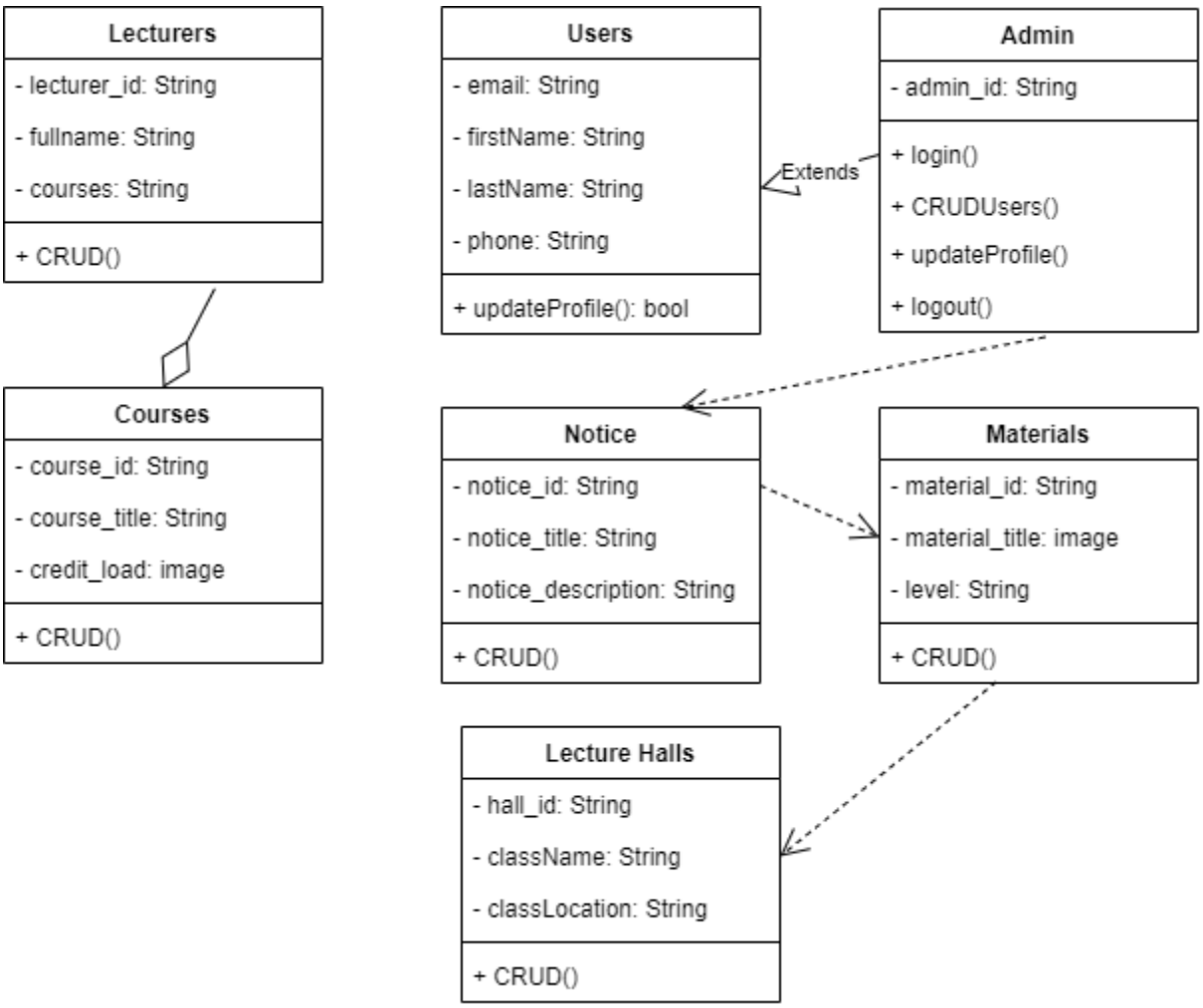

Fig 3.2 System Class Diagram

3.3.3 Activity Diagrams

Similar to a flowchart or a data flow diagram, an activity diagram visually depicts a sequence of events or the flow of control in a system, but it functions more like an advanced version of both.

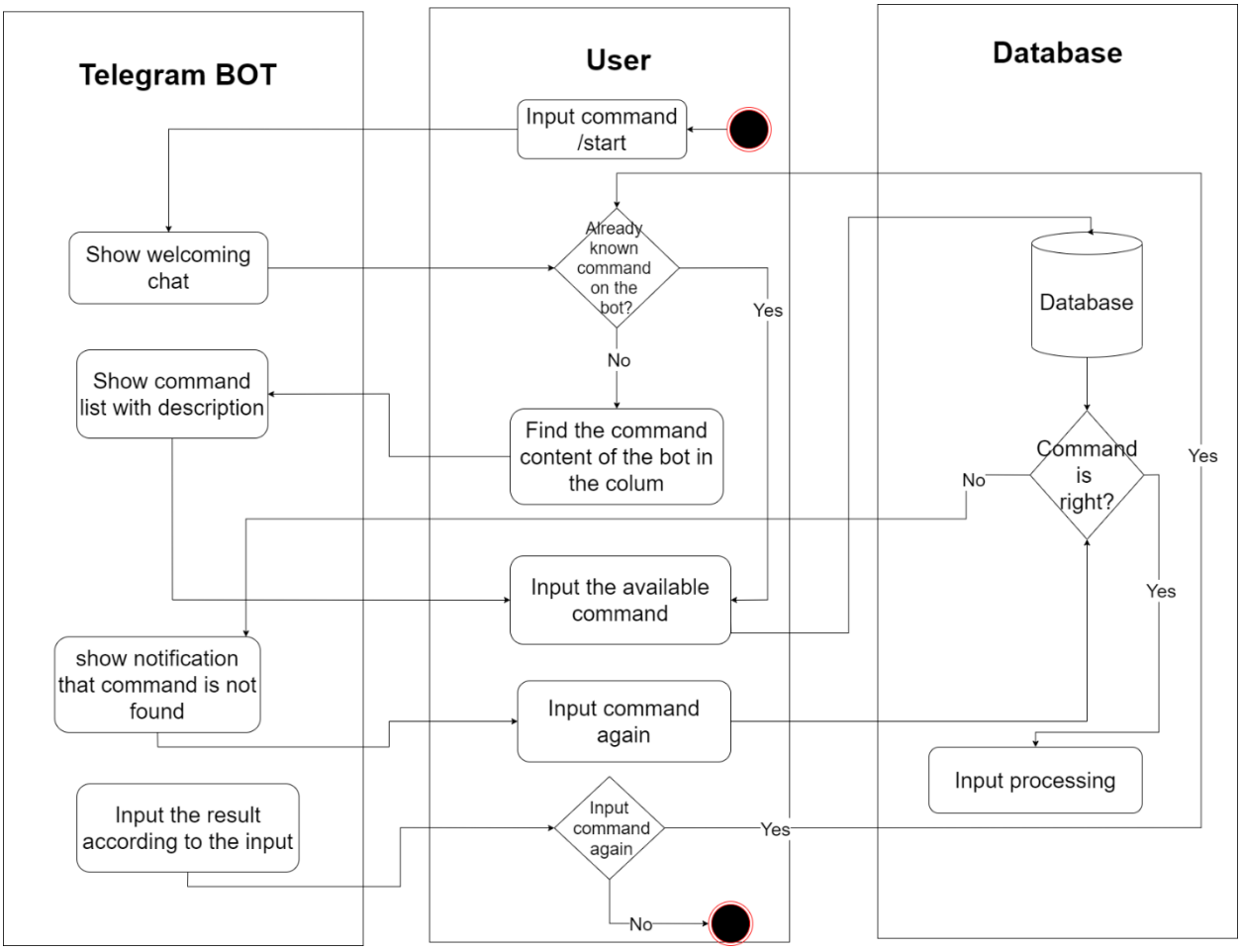

Fig 3.3 Application Activity Diagram

3.4 Database Design

The logical explanation of how data is kept in the computer's memory is called input specification. The freedom experienced in using the system and the convenience of retrieving and reading the data and assuring applicability across the internet make SQL standards essential for ensuring that structured data is uniform and independent of applications. Some of the input specifications employed in this project work are presented below.

i. Users Table: contains basic information about all system users.

ii. Courses Table: contains department course information.

Table 3.1 Users Table input specification table

FIELD NAME	DATA TYPE	LENGTH	DESCRIPTION
Email	String	150	User email address
Firstname	String	150	User first name
Lastname	String	150	User last name
Phone	String	150	User phone number
user_id	String	64	A unique string for identifying users

Primary key: user_id

Table 3.2 Courses Table input specification table

FIELD NAME	DATA TYPE	LENGTH	DESCRIPTION
Course_title	String	150	Course title
Course_load	String	150	Course credit load
Course_desc	String	150	Course description
Course_level	String	150	Level offering the course
course_id	String	64	A unique string for identifying courses

Primary key: course_id

3.5 Output Design

This declares and displays the outcome of the given input. The automated system's output is dependent on its input. The output specification is listed below.

Table 3.3 Users Output Design Table

Email	Firstname	Lastname	Phone	User_id
XXXX	XXXX	XXXX	XXXX	XXXX
XXXX	XXXX	XXXX	XXXX	XXXX
XXXX	XXXX	XXXX	XXXX	XXXX
XXXX	XXXX	XXXX	XXXX	XXXX

Primary key: user_id

Table 3.4 Courses Output Design Table

Course_title	Course_load	Course_level	Course_desc	Course_id
XXXX	XXXX	XXXX	XXXX	XXXX
XXXX	XXXX	XXXX	XXXX	XXXX
XXXX	XXXX	XXXX	XXXX	XXXX
XXXX	XXXX	XXXX	XXXX	XXXX

Primary key: course_id

3.6 Input & User Interface Design

This shows a visual representation of the system interface; it will be made to be intuitive to use, quick to respond to, and visually appealing. Additionally, it will be properly protected, so signing in will be necessary to view some levels of the contents. A mid-fidelity wireframing application named Draw.io is used to assist with the designs.

Fig 3.4 User Login Screen

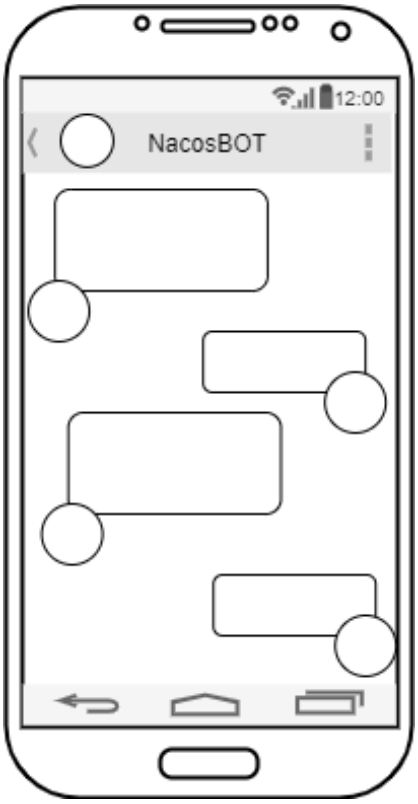

Fig 3.5 Chat Screen

3.7 System Requirement

Every piece of software that is created has preset system requirements that it must meet in order to run at its best. However, the system requirements are the bare minimum hardware and software needed for the system's intended operation.

3.7.1 Hardware Requirement

System Hardware Requirement Include:

a. Minimum of 8 GB of RAM (Random Access Memory) installed.

b. Minimum of intel core i3 processor.

c. Minimum of 250GB HDD (Hard Disk Drive).

3.7.2 Software Requirement

The software requirements include:

a. At least Windows 10 OS (Operating System).

b. Python Installation.

c. Vs. Code installation.

d. Browsers include Chrome and Firefox.

3.8 Choice of Programming Language

This research work will be a mobile-based application and will be implemented on a relational database system (SQLite). The student telegram Bot will be developed using the python request library, BotFather API, while Python will serve as the backend these are the modern language used in developing the system.

CHAPTER FOUR

SYSTEM IMPLEMENTATION EVALUATION

4.1 Introduction

This section describes in concise detail how the new system is implemented for effective operation. It shows samples of the working (new) system designed and how the system is to be installed.

4.2 System Testing and Evaluation

There are several reasons to test the produced system. For example, only via testing will we be able to assess any faults in the new system and propose answers to these problems. This project used both unit and integration testing to confirm the efficacy and efficiency of the design, as well as to ensure the new system satisfies its functional requirements and is error-free.

Unit Testing

specific units or single components of the system are examined individually in this part to confirm that specific phases function properly and without problems.

Integration Testing

The software was tested utilizing integration testing, in which all parts were assembled and functioned as one. The connectivity between the various components was checked to ensure that they are properly integrated and that the units can work together as one.

4.3 System Installation

In order to use the proposed application on any computer system, the following steps need to be taken:

1. Regardless of the device platform (Ios, Android, Desktop, etc.) ensure Telegram is installed on the device.
2. Ensure pip, pipenv, and Python3 or greater is installed on the System.
3. Copy the project folder to any location of your choice.
4. Open the project folder in Visual Studio Code.
5. From the terminal install create a virtual environment and install all dependencies in the Pipfile.

6. Locate the nacosbot.py file and run the file in debug mode.

4.4 Security Measures

Since the scope of the application is public, literally all the information is made available to any user, but some functionalities are restricted to the admin, functionalities that have to do with updating the bot information, adding content, and making modifications are restricted from the general public. The restriction is carried out by using passwords when the admin webpage is accessed.

4.5 Sample Outputs

These describe and give the pictorial representation of the program or software; it shows and gives clear understanding of the design, and displays all the interfaces.

Fig 4.5.1: Chat List

Fig 4.5.1 **Chat List:** This depicts that the bot appears just like every regular contact ready to be conversated with.

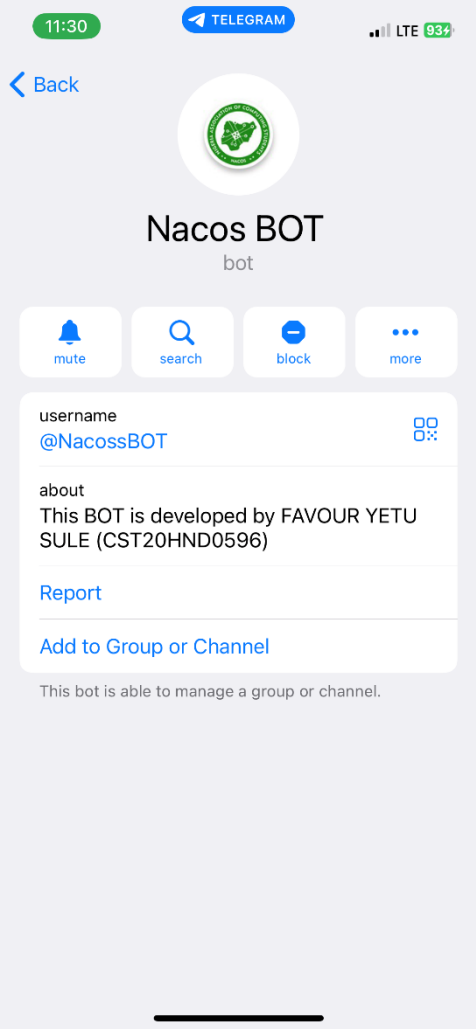

Fig 4.5.2: Bot Details

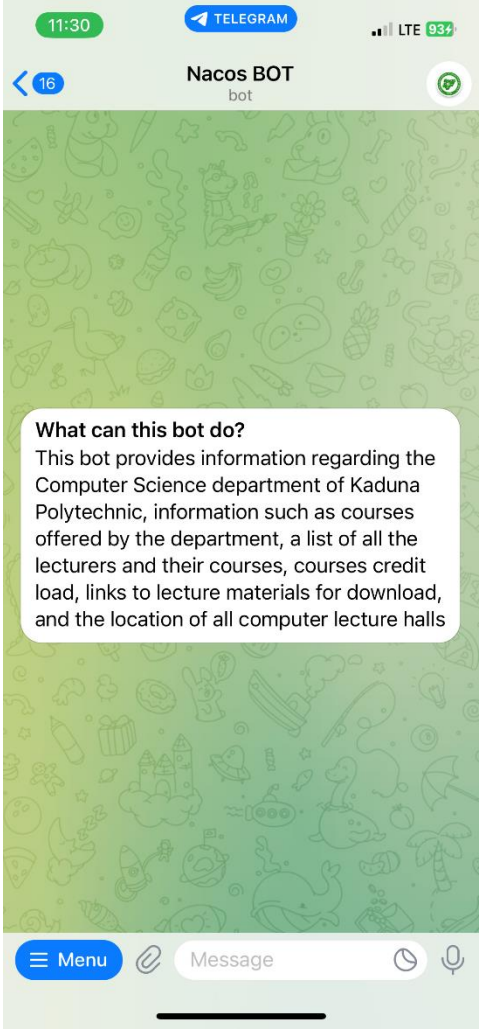

Fig 4.5.3: welcome Screen

Fig 4.5.2 **Bot Details**: The screen displays the information regarding the bot such as the username, about the bot, the bot name, and the bot profile image.

Fig 4.5.3 **Welcome Screen**: This is the first screen displayed to every user that wishes to interact with the bot.

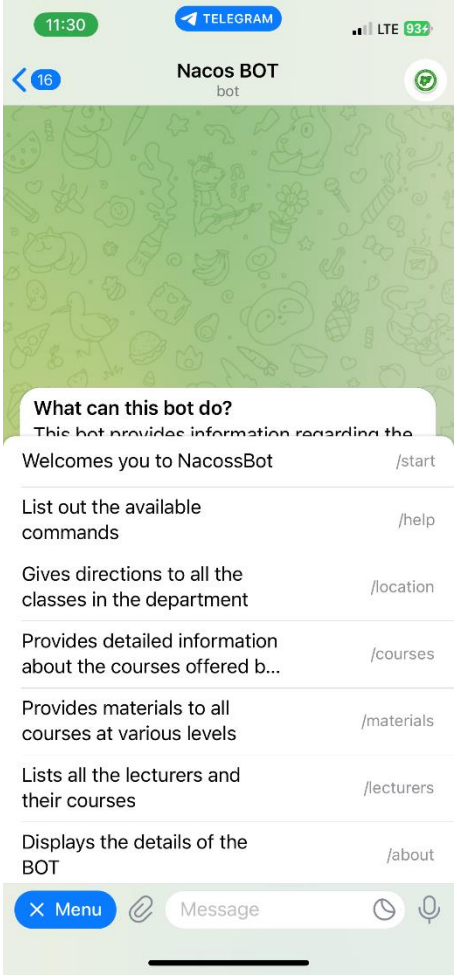

Fig 4.5.4: Bot Command List

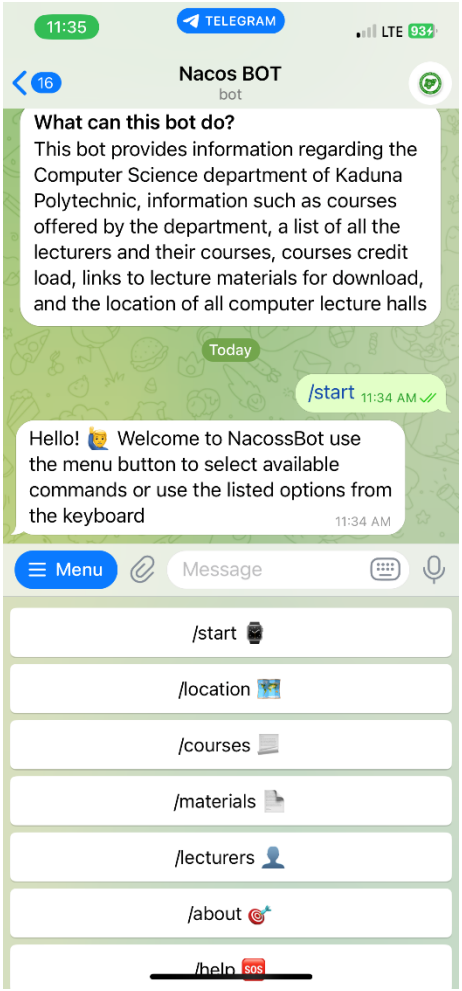

Fig 4.5.5: Start Command

Fig 4.5.4 **Bot Command List**: The screen shows the available commands that a user can interact with. By clicking of the menu the bot command list will be displayed.

Fig 4.5.5 **Start Command**: on clicking or by typing the command "/start" the bot will respond with some texts and would display the bot command list.

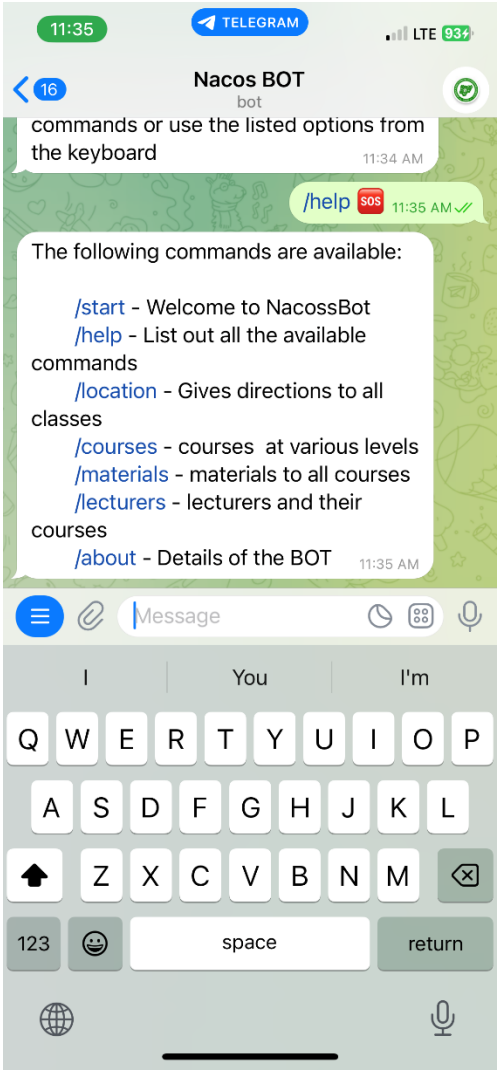

Fig 4.5.6: Help Command

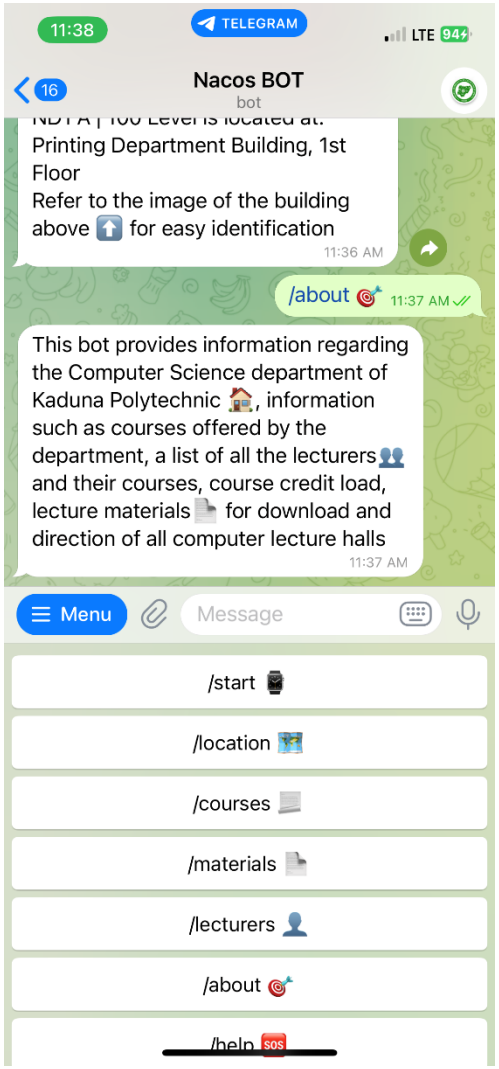

Fig 4.5.7: About Command

Fig 4.5.6 **Help Command**: By clicking or by typing the command "/help" the bot will respond with all the available command in a text base format, and users can interact with the text base command

Fig 4.5.7 **About Command**: By clicking or by typing the command "/about" the bot will respond with a brief description of the bot

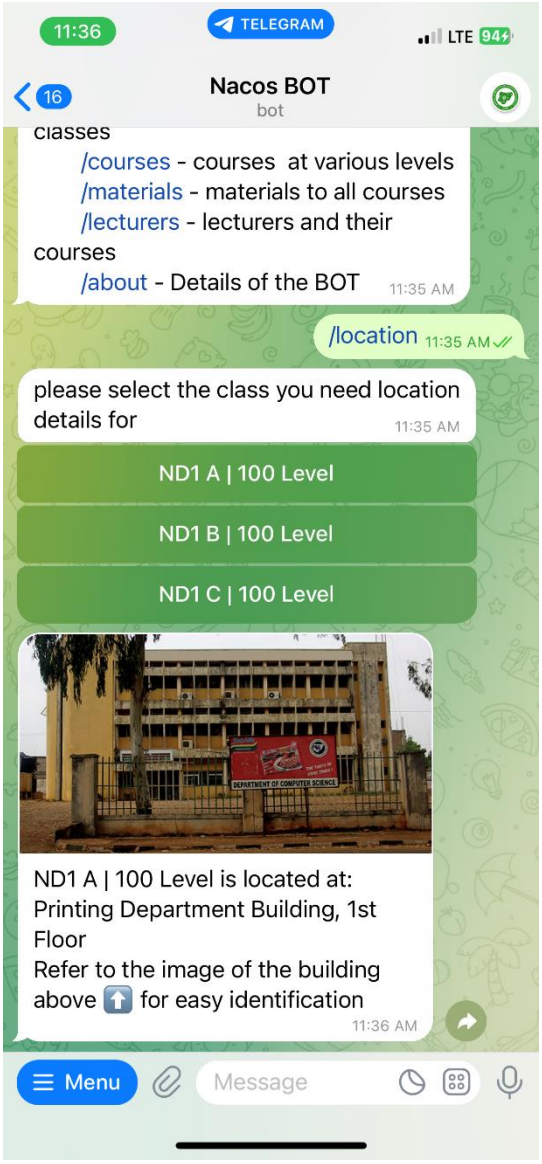

Fig 4.5.8: Location Command

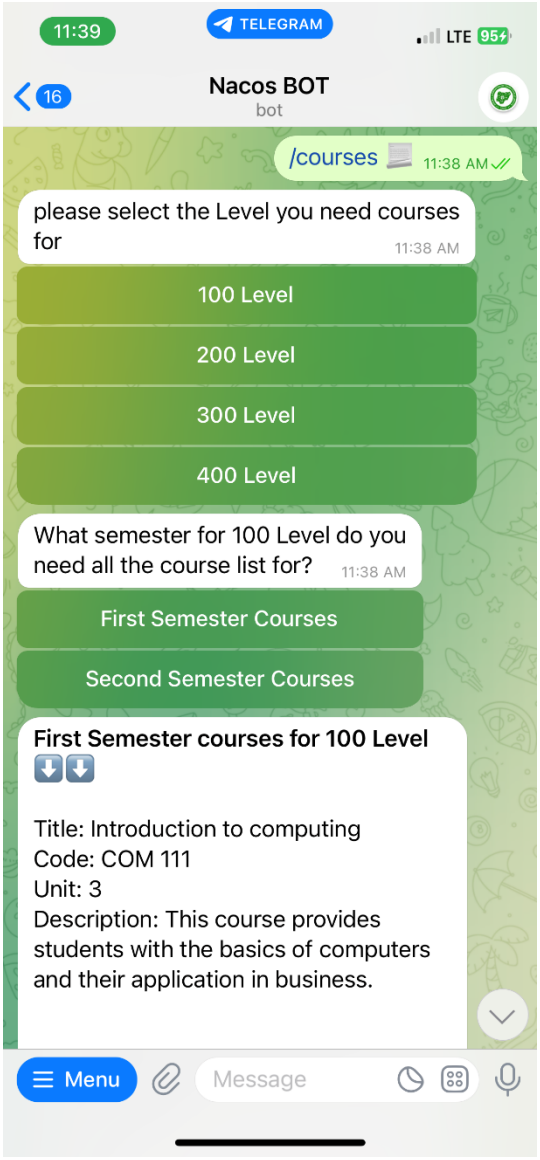

Fig 4.5.9: Courses Command

Fig 4.5.8 **Location Command**: By clicking or by typing the command "/location" the bot will respond with some sub-buttons containing the various classes that are available in the system. When one of the buttons is interacted with, the detailed description of the class will be displayed.

Fig 4.5.9 **Courses Command**: By clicking or by typing the command "/courses" the bot will respond with some sub-buttons containing the various levels that are available in the system. When one of the buttons (level) is interacted with to completion, the detailed description of the courses for the selected semester will be displayed.

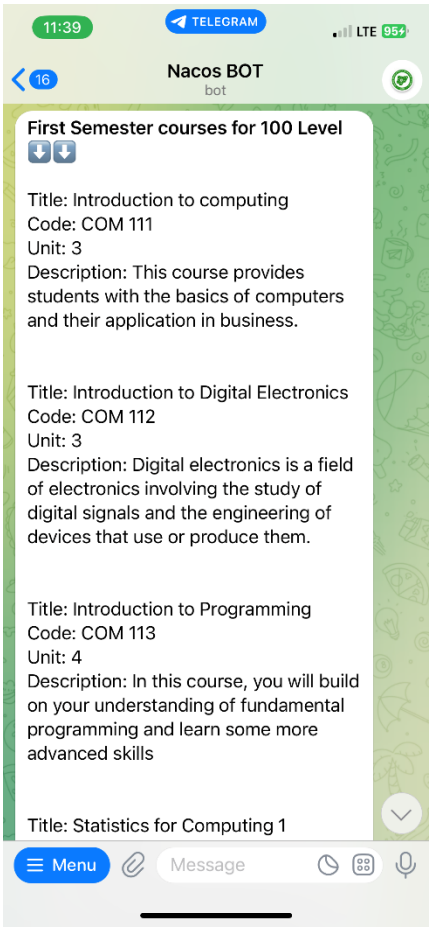

Fig 4.5.10: Bot Course Response

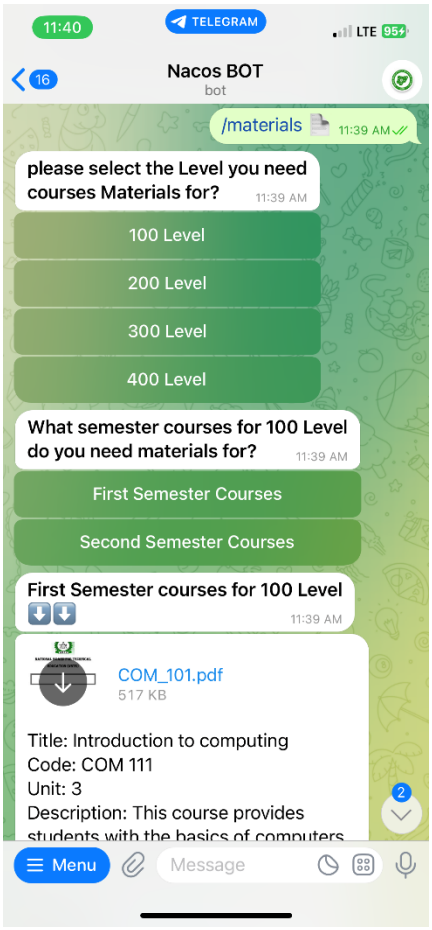

Fig 4.5.11: Command Materials

Fig 4.5.9.1 Bot Course Response: It shows the response gotten from the bot when the courses command was sent and interacted with to completion.

Fig 4.5.9.2 Command Materials: By clicking or by typing the command "/materials" the bot will respond with some sub-buttons containing the various levels that are available in the system. When one of the buttons (level) is interacted with to completion, the detailed description of the e material for the selected semester will be displayed and ready for download.

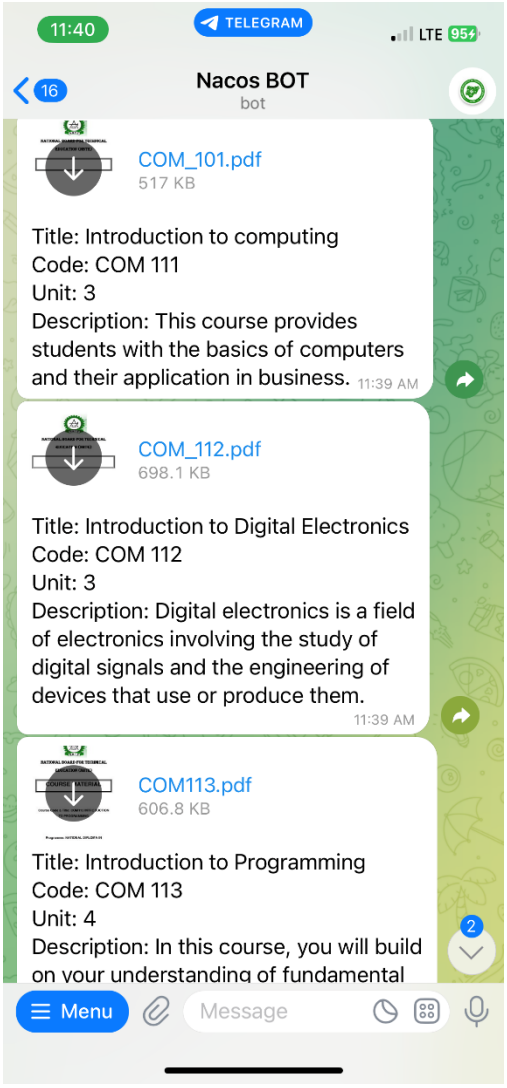

Fig 4.5.12: Bot Materials Response

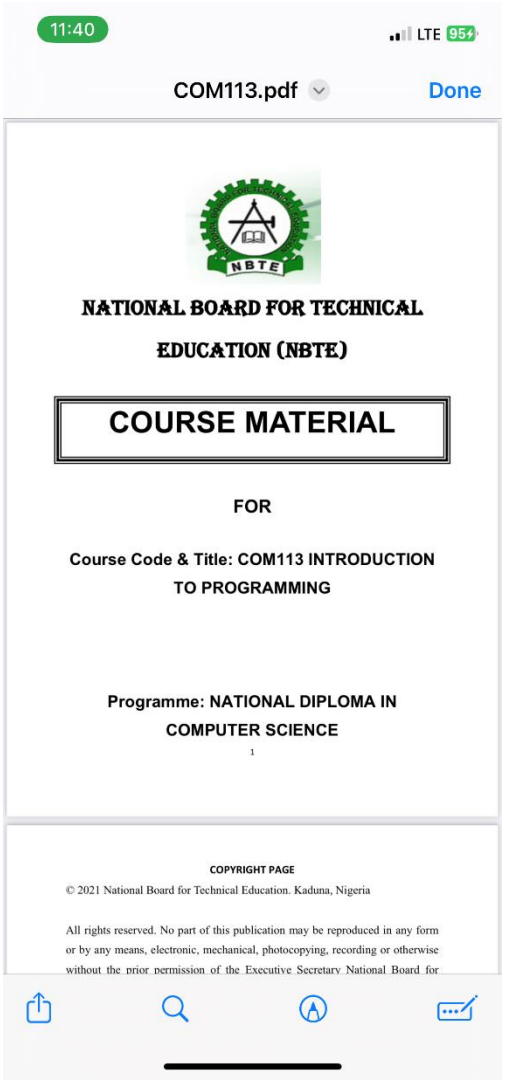

Fig 4.5.13: Material Preview

Fig 4.5.9.3 **Bot Material Response**: The image depicts the response gotten from the bot when the material command was sent and interacted with to completion.

Fig 4.5.9.4 **Material Preview**: The image depicts that the material can be downloaded and launched.

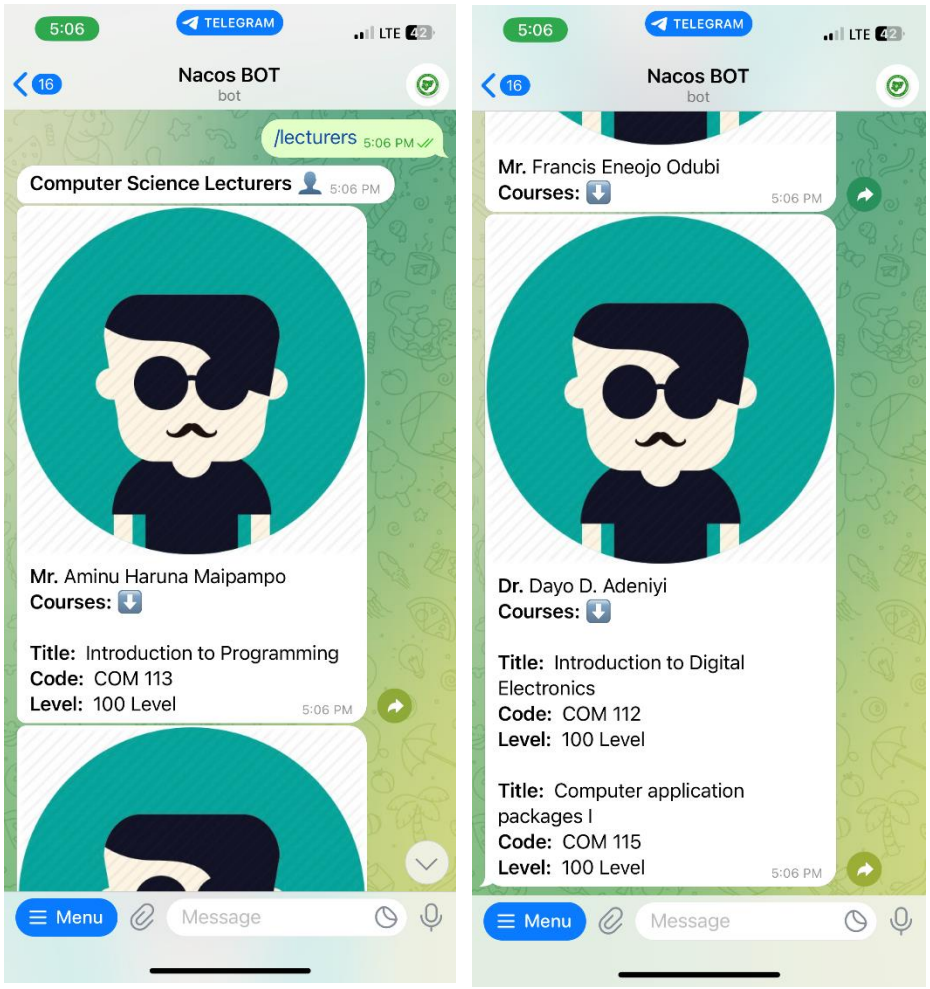

Fig 4.5.14: Command Lecturers

Fig 4.5.9.5 **Command Lecturer**: By clicking or by typing the command "/lecturers" the bot will respond with details of all lecturers profile available in the system, the profile contains the lecture image, the list of courses and level they are taken.

CHAPTER FIVE

Summary Conclusion and Recommendation

5.1 Summary

The project titled "Telegram Bot that Provides Students with Academic-Related Information in the Computer Science Department" aims to develop a telegram-based application that assists students in obtaining academic-related information and making inquiries about the department. The application will utilize a telegram bot to provide students with details about the courses offered, lecturers and their courses, course credit loads, links to lecture materials, and the locations of computer lecture halls. The project aims to streamline the process of accessing departmental information, saving time and resources for both students and the department. By providing a responsive GUI and ensuring the efficacy of the application through testing, the project intends to create an interactive experience for users. The success of this project could inspire similar approaches in other educational institutions, ultimately enhancing the quality of education globally.

5.2 Conclusion

In conclusion, the research project has successfully achieved its objectives and made significant advancements in improving access to departmental information for students. Through the development of a telegram-based application, the project has provided a user-friendly platform that allows students to easily obtain academic-related information and make inquiries. The scope of the study covered essential details such as course offerings, lecturers and their courses, course credit loads, lecture material access, and computer lecture hall locations. Despite the limitations in terms of time and literature access, this project holds great significance for both students and the department. The application's potential impact includes enhancing student engagement, and academic performance, and streamlining departmental processes.

5.3 Recommendation

Based on the findings and outcomes of the research project, the following recommendations are proposed:

1. Implementation of the Telegram Bot: The developed Telegram bot has proven to be an effective tool for providing academic-related information to students. It is recommended to integrate the bot into the official communication channels of the Computer Science Department. This would ensure easy access to the bot and encourage students to utilize its functionalities for obtaining information and making inquiries.

2. Continuous Updates and Maintenance: To ensure the accuracy and relevance of the information provided by the Telegram bot, it is crucial to establish a system for regular updates and maintenance. This includes updating course offerings, lecturer information, and any changes in course credit loads. The department should allocate resources and assign responsible individuals or a team to manage and maintain the bot's database.

3. Expansion to Other Departments: Considering the success and potential benefits of the Telegram bot, it is recommended to explore the possibility of expanding its functionality to other departments within the institution. Each department can develop its own customized version of the bot to cater to the specific academic-related information needs of its students. This would provide a comprehensive and unified platform for students across different departments.

By implementing these recommendations, the Telegram bot can become an indispensable tool for students, effectively addressing their academic-related inquiries and information needs. It would contribute to a more efficient and accessible communication system within the department, enhancing the overall student experience and supporting their academic success.

REFERENCES

Ahmadi, A., Setiawan, D., Suprayitno, S., & Hartoko, P. (2020). "Design Of Academic Information System Based On Bot Telegram In Smart Campus Concept". *Journal asro, 11*(03), 88. https://doi.org/10.37875/asro.v11i03.310.

Alawadhi, S., & Dashti, M. (2021). "The Use of the Telegram Application as an Information-Sharing Tool". *Journal of Information and Knowledge Management, 20*(2). https://doi.org/10.1142/S0219649221500246

Elekaei, A. (2018) Using Vocabulary Podcasts Tasks to Improve Iranian EFL Learners' Vocabulary Gain and Retention in an E-learning Project: Attitude, Autonomy, and Language Learning Strategies in Focus.

Faramarzi, Sajad & Tabrizi, Hossein & Chalak, Azizeh. (2019). Telegram: An instant messaging application to assist distance language learning. 19. 132-147.

Selomon Yenesew. The Role Telegram Application for Information Sharing in the Case of Online Ge'ez Language Learning. https://doi.org/10.21203/rs.3.rs-2251284/v1

Setiaji, H., & Paputungan, I. V. (2018). "Design of Telegram Bots for Campus Information Sharing*". In IOP Conference Series: Materials Science and Engineering (Vol. 325). Institute of Physics Publishing*. https://doi.org/10.1088/1757-899X/325/1/012005

Rianto, R., Rahmatulloh, A., & Firmansah, T. A. (2019). "Telegram Bot Implementation in Academic Information Services with The Forward Chaining Method". *Sinkron, 3*(2), 73–78. https://doi.org/10.33395/sinkron.v3i2.10023.

Rinke, A. (2022, June 3). *What is a Telegram Bot? Explanation with an industrial focus.* System Integration With the OPC Router. https://www.opc-router.com/what-is-a-telegram-bot/

APPENDIX

```
import django, pprint
import os
os.environ['DJANGO_SETTINGS_MODULE'] = 'Nacosbot.settings'
django.setup()
from telegram.ext import Updater, CommandHandler, CallbackQueryHandler
from telegram import Bot
from telegram import KeyboardButton, ReplyKeyboardMarkup, InlineKeyboardButton,
InlineKeyboardMarkup
from decouple import config
from Nacosbot_API.models import *

TOKEN = config('TOKEN')
updater = Updater(TOKEN, use_context=True)
dispatcher = updater.dispatcher

def help(update, context):
    update.message.reply_text(
        """
        The following commands are available:

        /start - Welcome to NacossBot
        /help - List out all the available commands
        /location - Gives directions to all classes
        /courses - courses  at various levels
        /materials - materials to all courses
        /lecturers - lecturers and their courses
        /about - Details of the BOT
        """
    )
```

```
def start(update, context):
    reply_keyboard = [
        ["/start ⏱"],
        ["/location 🗺"],
        ["/courses 📃"],
        ["/materials 📄"],
        ["/lecturers 👤"],
        ["/about 🎯"],
        ["/help 🆘"],
    ]
    update.message.reply_text("Hello! 🤗 Welcome to NacossBot use the menu button to select available commands or use the listed options from the keyboard",
reply_markup=ReplyKeyboardMarkup(reply_keyboard,))

def location(update, context):
    keyboard = [[InlineKeyboardButton(str(pro), callback_data=f'{pro.pk}-location')] for pro in Classes.objects.all()]
    reply_markup = InlineKeyboardMarkup(keyboard)
    update.message.reply_text('please select the class you need location details for',
reply_markup=reply_markup)

def courses(update, context):
    keyboard = [[InlineKeyboardButton(f'{pro} Level', callback_data=f'{pro.pk}-course')] for pro in Level.objects.all()]
    reply_markup = InlineKeyboardMarkup(keyboard)
    update.message.reply_text('please select the Level you need courses for',
reply_markup=reply_markup)

def materials(update, context):
```

```
    keyboard = [[InlineKeyboardButton(f'{pro} Level', callback_data=f'{pro.pk}-material')] for pro in Level.objects.all()]
    reply_markup = InlineKeyboardMarkup(keyboard)
    update.message.reply_text('<b>please select the Level you need courses Materials for?</b>', reply_markup=reply_markup, parse_mode='HTML')

def about(update, context):
    chat_id = update.message.chat.id
    context.bot.send_message(chat_id=chat_id, text='This bot provides information regarding the Computer Science department of Kaduna Polytechnic 🏠, information such as courses offered by the department, a list of all the lecturers 👥 and their courses, course credit load, lecture materials 📄 for download and direction of all computer lecture halls', parse_mode='HTML')

def lecturers(update, context):
    chat_id = update.message.chat.id
    context.bot.send_message(chat_id=chat_id, text='<b>Computer Science Lecturers 👤 </b>', parse_mode='HTML')
    for pro in Lecturer.objects.all():
        details = f'<b>{pro.title}</b> {pro.name}\n<b>Courses: ⬇\n\n</b>'
        for lec in CoursesToLecturer.objects.filter(lecturer=pro):
            details += f'<b>Title: </b> {lec.course.title}\n<b>Code: </b> {lec.course.code}\n<b>Level: </b> {lec.course.level} Level\n\n'
        context.bot.send_photo(chat_id=chat_id, caption=details, photo= open(pro.pics.path, 'rb'), parse_mode='HTML')

def button(update, context):
    chat_id = update.callback_query.message.chat.id
    query_data = update.callback_query.data.split('-')

    if query_data[1] == 'location':
```

```
        try:
            loc = Location.objects.get(province_id=query_data[0])
            context.bot.send_photo(chat_id=chat_id, caption=str(f'{loc.province} is located at:
\n{loc}\nRefer to the image of the building above ⬆ for easy identification'),
photo=open(loc.image.path, 'rb'))
        except Location.DoesNotExist:
            keyboard = [[InlineKeyboardButton(str(pro), callback_data=f'{pro.pk}-location')] for pro
in Classes.objects.all()]
            reply_markup = InlineKeyboardMarkup(keyboard)

            context.bot.send_message(chat_id=chat_id, text=f'Unable to get Location Details for
<b>{Classes.objects.get(id=query_data[0])}</b>\nplease select another class you need location
details for', parse_mode='HTML', reply_markup=reply_markup)

    if query_data[1] == 'course':
        keyboard = [[InlineKeyboardButton(f'{pro} Courses', callback_data=f'{pro.pk}-
{query_data[0]}-scourse')] for pro in Semester.objects.all()]
        reply_markup = InlineKeyboardMarkup(keyboard)
        context.bot.send_message(chat_id=chat_id, text=f'What semester for
{Level.objects.get(pk=query_data[0])} Level do you need all the course list for?',
reply_markup=reply_markup)

    try:
        if query_data[2] == 'scourse':
            qs = Course.objects.filter(semester=query_data[0], level=query_data[1])
            if qs:
                course_list = f'<b>{qs[0].semester} courses for {qs[0].level} Level</b> ⬇⬇\n'
                for pro in qs:
                    course_list += f'\nTitle: {pro.title}\nCode: {pro.code}\nUnit:
{pro.unit}\nDescription: {pro.desc}\n\n'
                context.bot.send_message(chat_id=chat_id, text=course_list, parse_mode='HTML')
```

```
        else:
            context.bot.send_message(chat_id=chat_id, text=f'Unable to fetch course list for {Semester.objects.get(pk=query_data[0])} {Level.objects.get(pk=query_data[1])} level 😢')
    except: pass

    if query_data[1] == 'material':
        keyboard = [[InlineKeyboardButton(f'{pro} Courses', callback_data=f'{pro.pk}-{query_data[0]}-smaterial')] for pro in Semester.objects.all()]
        reply_markup = InlineKeyboardMarkup(keyboard)
        context.bot.send_message(chat_id=chat_id, text=f'<b>What semester courses for {Level.objects.get(pk=query_data[0])} Level  do you need materials for?</b>', reply_markup=reply_markup, parse_mode='HTML')

    try:
        if query_data[2] == 'smaterial':
            qs = Material.objects.filter(semester=query_data[0], level=query_data[1])
            if qs:
                material_list = f'<b>{qs[0].semester} courses for {qs[0].level} Level</b> ⬇⬇'
                context.bot.send_message(chat_id=chat_id, text=material_list, parse_mode='HTML')
                for pro in qs:
                    context.bot.send_document(chat_id=chat_id, filename=pro.file.path, document=open(pro.file.path, 'rb'), caption=f'\nTitle: {pro.course.title}\nCode: {pro.course.code}\nUnit: {pro.course.unit}\nDescription: {pro.course.desc}\n\n')
            else:
                context.bot.send_message(chat_id=chat_id, text=f'Unable to fetch {Semester.objects.get(pk=query_data[0])} courses for {Level.objects.get(pk=query_data[1])} Level 😢')
    except: pass


# Register Commands
```

```
dispatcher.add_handler(CommandHandler('help', help))
dispatcher.add_handler(CommandHandler('start', start))
dispatcher.add_handler(CommandHandler('location', location))
dispatcher.add_handler(CommandHandler('courses', courses))
dispatcher.add_handler(CommandHandler('materials', materials))
dispatcher.add_handler(CommandHandler('lecturers', lecturers))
dispatcher.add_handler(CommandHandler('about', about))

button_handler = CallbackQueryHandler(button)
dispatcher.add_handler(button_handler)

updater.start_polling()
updater.idle()
```