UNITED TECHNICAL COLLEGE

(Affiliated to Pokhara University)

Bharatpur-11, Bhojad, Chitwan

[Subject Code: CMP-290]

A MINOR PROJECT FINAL REPORT ON

"DAILY CLASS SCHEDULES PROVISIONING CHATBOT"

Submitted in Partial Fulfillment of the Requirements for

the **Bachelor of Engineering in Computer Engineering**

under **Pokhara University**, Nepal

Submitted by:

Amit Baral [Roll No.: 3]

Binit Ghimire [Roll No.: 7]

Narayan Pokhrel [Roll No.: 12]

Pratima Bhandari [Roll No.: 15]

Submitted to:

Department of Computer Engineering

June, 2021

ABSTRACT

In the recent world of technological advancement, chatbots have played a huge role being a type of software with the ability to communicate with people using artificial intelligence, natural language processing, big data and other similar technologies. These chatbots are mainly used in performing different activities including sending quick response to users, generating automated response for users during company's off-hours, helping users in getting information about different things, and providing better services to people. This paper proposes an approach to create a chatbot based on Natural Language Processing (NLP) with serverless deployment in the cloud to enable students in getting their daily class schedules in a very short time just with a little number of clicks. Schools, colleges, universities and other educational institutions can make use of this approach to build a chatbot that makes daily class schedules provisioning easier for students and faculty members. We also make these things possible with super-fast response rate making use of cloud and serverless computing.

Key Words:

Chatbot, Natural Language Processing, Serverless Deployment, Amazon API Gateway, AWS Lambda

TABLE OF CONTENTS

ABBREVIATIONS

Abbreviation	Definition
API	Application Programming Interface
AWS	Amazon Web Services
CI/CD	Continuous Integration and Continuous Delivery
CSS	Cascading Style Sheets
DAST	Dynamic Application Security Testing
DB	Database
DevOps	Development and Operations
DevSecOps	Development, Security and Operations
DFD	Data Flow Diagram
DL	Deep Learning
ECMA	European Computer Manufacturer's Association
ER Diagram	Entity Relationship Diagram
HTML	HyperText Markup Language
IaaC	Infrastructure-as-a-Code
IaaS	Infrastructure-as-a-Service
IAST	Interactive Application Security Testing
JS	JavaScript
ML	Machine Learning
NLG	Natural Language Generation
NLP	Natural Language Processing
NLU	Natural Language Understanding
NN	Neural Network
NoSQL	Not only SQL
PaaS	Platform-as-a-Service
QA	Quality Assurance
RDBMS	Relational Database Management System
SaaS	Software-as-a-Service
SAST	Static Application Security Testing
SDLC	Software Development Life Cycle
SFD	System Flow Diagram
SMS	Short Message Service
SQL	Structured Query Language
UI	User Interface
UX	User Experience
Verbot	Virtual-Robot

LIST OF FIGURES

LIST OF EQUATIONS

CHAPTER 1: INTRODUCTION

1.1. Background

"Chatbot", also known as "Chatterbot", is a term first coined by Michael Mauldin, who is also the creator of the first Verbot, "Julia". He coined the term in 1994 for explaining about programs which can offer conversational experience to the end-users [1]. These conversational programs offer the ability to design different messaging applications such as Facebook Messenger, WhatsApp, Signal, etc., as well as add amazing conversational functionalities in them with the inclusion of automated response systems.

Besides these, the conversational programs can also be used to assist with the responsiveness and availability, plus reduce the dependence on human beings in the modern world of Technology powered by automation. Responsiveness, which simply refers to the ability to respond, offers the functionality of quick and positive reactions during the time of concurrent conversations with multiple people at a given time. Taking this in concern, a human being mayn't be able to provide a quick and immediate response concurrently. For eliminating this restriction, chatbots have been introduced in the world to improve the response rate, and also to enable concurrent communication between a single organization and multiple end-users or customers [1], [2].

During our research, we were able to observe that students in different schools, colleges, universities and educational institutions are facing problems with their daily class schedules being provisioned to them. Sometimes they have to request their classmates or faculty members in person, whereas the other times they must be requesting so through major communication platforms like Facebook Messenger or Viber. This way, it isn't always sure that they would be provisioned in time, and sometimes they might even waste their time waiting for response and at the end, finding out that the contacted people don't possess the daily class schedules as well.

This is where our project on Daily Class Schedules Provisioning Chatbot comes in. In this project, we would be developing a chatbot that provisions daily class schedules to students upon their requests for the specific day, and one special thing is, we would be deploying the chatbot with the power of serverless computing in the cloud to make it faster and more efficient in provisioning the requirements to the end-users, i.e., students in this case. With the implementation of NLP, our standardized chatbot would be able to match the queries from the end-users and respond back with the pre-fed data that has been built into our platform.

We have seen that the interested candidates (generally students, and sometimes faculty members) are always trying to communicate with the institution representatives to get the daily class schedules, but every time it isn't possible for the institution representatives to respond immediately due to not having access to course materials at any specific period of time excluding the academic hours. With the implementation of our project, the problems that can be caused as a result of traditional form of daily class schedules provisioning (in person or through messaging platforms) can be resolved easily since our end-users would be able to access the daily class schedules within their fingertips in just under a minute.

1.2. Problem Statement

In every education institution, there are classes, sessions or lectures of different subjects or courses on a daily basis during a certain period of time for a certain duration. A question shall arise among students and faculty members in this regard, about which subjects would happen in a certain day. The only option for them to get the daily class schedules is to directly contact the respective department in the institution. However, contacting the department and getting the class schedules mayn't be feasible for many students and faculty members, due to limited number of staffs and inaccessibility of the department documents at any period of time during off-hours.

Also, students might request their classmates to provide daily class schedules, but it isn't always sure that other classmates would have the class schedules, plus getting response from them at any period of time of their own will isn't possible, because everyone has a different schedule of using different messaging platforms for communication. This creates a problem among students, due to not being able to get the daily class schedules whenever they want as per their will.

As per our research, we were able to find out that students are always trying to contact their respective department's representatives to get the daily class schedules for a specific day, but every time it mayn't be possible for the representatives to respond back immediately due to not having access to the academic documents or course materials at any specific period of time during non-academic hours. To mitigate all of these problems, there has to be a platform through which students can be provisioned with their daily class schedules whenever they want at any specific period of time, whether it be in the morning or in late-midnight when everyone is asleep.

There are already several digital platforms available on the Internet, which are useful in assisting students with their academics, answering queries about different topics or even about different educational institutions and much more. Also, there are different chatbots for the same kind of purposes. But there is no existing platform out there which is specifically built or designed with the core goal of provisioning students with their daily class schedules.

Listed below are some of the major ways on how different educational institutions are using chatbots to improve the experience of students in different academic matters as of the recent date:

- Admission Chatbots: Different universities have built their own chatbots to assist students about their admission process. The kind of assistance these chatbots offer include answering different queries about the requirements, fees, and many more. Plus, some of these chatbots also enable students to get enlisted in the institution’s further contact lists so that they can get notified whenever new admission applications are opened.
- Digital Coach: Many educational institutions have developed chatbots for assisting students with their academic queries and providing answers to several common questions from students. These chatbots are meant to be informative in a way that students can get help regarding their doubts in just little period of time. Some of these chatbots also offer the ability to reach out to active mentors who can assist students in different things.

- Event Schedulers: Some of the educational institutions have also brought event scheduling chatbots to help students and faculty members know about different activities or events that are going to happen at the institution's premises in the upcoming days. These chatbots also offer timetables regarding different events being held in the institution.

From these examples of different kind of chatbots that educational institutions have implemented to improve the accessibility of students, we can certainly see that chatbots are playing a huge role in assisting students with different activities, assignments, events, admissions and many more. However, we can't still see any of these chatbots that are helping students in getting quick access to their daily class schedules.

As seen above, event scheduling chatbots are somehow near to using a similar approach as ours, due to the fact that they are providing timetables to students about different events, however the problem that students are facing isn't solved with these chatbots. Taking all of these matters in concern, we came up with this idea to build a chatbot that can help students in being away from all of these problems regarding the provisioning of daily class schedules within their fingertips.

1.3. Objectives

Our project on Daily Class Schedules Provisioning Chatbot is being developed to resolve all of the problems as discussed earlier by implementing the power of NLP with serverless deployment over the cloud. The following are the objectives of this project:

- To offer quick response to queries from the end-users making use of serverless computing in the cloud,
- To create a conversational system that can respond back to queries at any specific period of time,
- To enable students and faculty members to obtain their daily class schedules just within their fingertips.

1.4. Applications

Our project can be implemented in several application areas for different purposes. Some of the major applications of using this project are listed below:

- An educational institution can implement this project to make students able to get their daily class schedules at any period of time without having to keep asking the staffs.
- This project can also be beneficial for different organizations to offer a fresh working schedule to employees and prevent them from having queries on what to work on further.
- Our project can also be used during several events for provisioning the attendees and visitors with the event schedules and timelines.
- Normal individuals can implement this project to establish a habit of following daily routines, therefore increasing their productivity.

1.5. Features

Until the current stage of development of our project, the following features have been made available:

- Students can interact with the chatbot through our Facebook page, Telegram handle, and even on our website.
- The power of serverless computing in the cloud has helped in improving the performance and scalability of our chatbot, therefore concurrent conversations can be handled in an efficient manner.
- The official website of our project offers users the ability to get proper information about everything related to the chatbot, and also get the opportunity to try out the chatbot in real-time without getting out of the website.

1.6. Research Questions and Hypothesis

To propose the idea of our project for daily class schedules provisioning, we asked several questions to ourselves, and researched on several topics to figure out the existing independent and dependent variables in the academic chatbot technologies, which are being implemented by different institutions. Based on our research, we came across the following null hypothesis:

- Educational institutions mostly develop or implement chatbots only for generalized things like admission procedures, event scheduling, academic questionnaires, etc., but they don't have a similar implementation for daily class schedules or examination schedules.

Regarding this null hypothesis, we asked ourselves the following question:

- What could be a good way to ensure none of the students in an educational institution face problem in uncovering their class schedules for every specific day?

While researching on different ideas to figure out the solution to this problem, we came across an idea of building a natural language processing chatbot for daily class schedules provisioning to provide responses to students quickly and in more efficient manner making use of serverless computing over the cloud. After performing an intense feasibility analysis on this topic, we decided to propose the plan for a project based on this idea.

CHAPTER 2: REVIEW OF LITERATURE

2.1. Introduction

The state-of-the-art or literature review section for this minor project final report primarily focuses on the observations and conclusions made as per the several research papers that we went through related with chatbots, natural language processing and other things related with our approach. We also performed three different case studies on popular educational chatbots that are working on eliminating different problems related with academics, along with how we are trying to cope up with other problems that students are still facing.

2.2. Review of Literature

As of today, there exists an epoch of Artificial Intelligence. The scientific reasons behind human thinking as per different philosophers and psychologists, have been briefly introduced by the experts in the field of Technology with robust sophistication, which has led to the development of the science behind AI [3]. For this research, the natural language conversational programs, referred to as "chatbots", are the major focus.

Chatbots, also known as "Chatterbots", are existing in the world since the mid-1990's with the creation of the first Verbot, Julia, by Michael Mauldin in 1994. The terminology "chatbot" features two terms; "chat" and "bot", which simply refer to conversation and robots, respectively [4]. This kind of conversational programs are designed to mimic the conversations with human beings. Majority of the chatbots are built with the power of AI and Natural Language Processing, so that they can understand human-spoken languages. In the modern world, chatbots have been really helpful to people, all thanks to some of the popular names including Alexa from Amazon, Siri from Apple and Google Assistant. Besides these popular ones, the other chatbots are mostly designed such that they can be easily accessed through third-party messaging applications like Facebook Messenger, Telegram, Slack, etc.

For most of the existing chatbots, the biggest question arises when they have to understand human languages, and generate responses suitable with human emotions. Making chatbots able to support human emotions is one of the most favored things, which can lead chatbots to becoming popular among the end-users. The reason behind this is that such ability offers the users the opportunity to feel satisfactory towards the chatbots. Pattern Matching and Parsing algorithms for Natural Language Processing are the most common techniques being used, however these algorithms aren't capable of simulating satisfactory natural conversation for the end-users [4]. Since the responses are scripted, chatbot developers and psychologists have been working together to insert new features and implement new approaches in existing manners in chatbots, with the use of ML, DL and NN. With these approaches, chatbots become able to remember facts from earlier conversations, identify user intents and simulate similar intents in further conversations [5]. Also, the efficiency of chatbots can be improved in a lot more fascinating manner, with the development of variations in workflow with technical approaches, accompanied by multiple solutions in regards to the same problem.

A research paper published in 2020 International Symposium on Educational Technology (ISET) presents the design of a chatbot which instantly answers questions from students on different social media platforms, and the chatbot is able to answer questions using natural language and commands. When the lecturers upload course materials to the university's

database, the chatbot is able to process the questions from students and answer them based on the pre-fed data in the database as per the course logistics. The researchers also conducted a survey on multiple undergraduate computer science students, and realized that the chatbot has been acting effectively as an online tutor to reduce the workloads of lecturers [6]. There are different other academic chatbots with the approach to assist both the professors and the students in asking questions of any topic, as well as comprehend rationale behind the topics, with the emphasis on accuracy to determine the chatbot system. However, majority of the chatbots don't seem like they would turn out to be economical after a certain period of time without the existence of any grounded business method for reasoning and providing better beneficial results [7].

Besides these, there are different other chatbots and similar conversational programs which are assisting students with different things ranging from admission to examination, plus notifying about different events happening within their institutions. However, none of the chatbots till date are specifically designed in order to solve the problem of students regarding the provisioning of their daily class schedules. Therefore, after going through several research papers related with chatbots based on NLP as well as the existing educational chatbots, we came up with a conclusion to bridge the gap responsible for creating hassles among students in handling their academics as per proper schedule by initiating an approach to design and implement a chatbot useful in assisting students with getting their daily class schedules whenever they want.

2.3. Case Studies

2.3.1. Case I: Admission Chatbots

There are several educational institutions in the world, that have started using admission chatbots in order to answer different queries asked by students including those related with admission form, registration deadlines, academics and course-related questions, location of course buildings, as well as for booking appointment with different professors and admission centers regarding the admission procedure.

Makerobos Innovation Labs, a technology company in India, is also offering admission chatbot templates for different colleges, through which colleges can deploy chatbots in their official websites without any hassle [8]. For this, colleges just have to provide different information to the system and integrate the chatbot with their Facebook page and college website. The chatbots from Makerobos as well as other alternatives tend to be highly competent in the world of conversational programs.

2.3.2. Case II: Digital Coach

Staffordshire University has introduced a new digital friend for students, who will support them regarding their academics and studies, and the digital friend is in the form of a chatbot, known as "Beacon". This chatbot recognizes students when they authenticate into the system, and then the students can prefer to interact through text or voice conversation [9].

This chatbot from the university makes it possible to build positive relationships among the professors and students, plus provide additional support to students who require it, in order to better cater for their requirements. Similarly, several other

universities in the world have launched similar chatbots to assist students with their studies, as well as to create a virtual appointment with their professors.

2.3.3. Case III: Event Schedulers

e2m.live, a leading video-first virtual event platform for some of the biggest companies in the world including those enlisted in the Fortune 500 list, has developed the industry-first chatbot event applications. Their chatbot can be integrated with Facebook Messenger, company websites, as well as SMS Messaging. The chatbot offers an amazing conversational interface and experience for easier accessibility of application features [10].

With the chatbot from e2m.live, the users can put forward their queries, schedule meetings, plus get directions and a lot more. The chatbot also has amazing menus and super-active alerts in order to provision people with frequent updates. They have also provided voice support through Amazon Alexa to enlighten the experience of the end-users, and customize their overall experience. Similarly, other companies have also developed different chatbots for event scheduling, as well as to ask questions about upcoming events.

2.4. Conclusion

With the case studies of three different types of existing chatbot platforms, we can see that the world has already reached a height in the advancement of chatbots. There are no issues regarding performance and scalability in majority of the popular chatbots. However, none of the chatbots or chatbot developers have ever worked on mitigating the problems of students regarding the provisioning of their daily class schedules. As mentioned in our review of literature along with the deep case studies of different platforms, we can conclude that we are working on a project that can remediate the problems of students in getting their daily class schedules, without affecting the performance and scalability of the chatbot.

CHAPTER 3: REQUIREMENTS

3.1. Functional Requirements

For the successful development of a project, it is necessary to plan about the functional requirements of the final product. Functional requirements simply refer to different functionalities that must be implemented during the overall phases of Software Development Life Cycle (SDLC).

When these requirements are fulfilled, the end-users would be able to accomplish their tasks with the help of the final product. Therefore, it is essential to make the functional requirements clear for both the team working behind different phases of SDLC and the stakeholders. In general, functional requirements are used to define the system behavior during specific situations.

The following are the functional requirements of our project:

- Provisioning of daily class schedules without any cases of disruption,
- Ability to handle multiple or concurrent conversations at once,
- Proper integration over popular digital communication platforms to make the services available in those platforms,
- Ability to continue conversations from previous state, without any hindrance over re-initializing conversations, if required!

3.2. Non-functional Requirements

3.2.1. Safety Requirements

The entire assets for our project would be hosted over the Amazon AWS cloud services, where our data is backed up in several data centers inside multiple Availability Zones in different AWS Regions all over the world. So, we don't need to worry much regarding the safety of the assets and the specifications, but it is always a good idea to have things backed up to our local storage devices, so that in case our cloud service accounts are disrupted in any way, we would be able to deploy our project back elsewhere.

3.2.2. Security Requirements

Since we will be deploying our project in the cloud, we don't need to worry about infrastructure security, but it is necessary to take care of the code quality, and make sure there are no security vulnerabilities present allowing attackers to get into our systems.

Also, when our project is further used by different educational institutions to build their own chatbot, the level of security required would be dependent upon the type of infrastructure they will be using and their code quality.

3.2.3. Software Quality Attributes

- Availability: Since we would be deploying our project in the cloud, our chatbot would be available at any time, with respect to the availability ensured by our cloud service provider; i.e., Amazon AWS.

- Accuracy: It would be required for us to build our chatbot in such a way that it can generate proper response for specific messages from the users. In this case, the implementation of NLP would help us out in ensuring accuracy.
- Maintainability: We need to make sure that our chatbot can further upgraded or maintained, when necessary, without having to bring drastic changes to our business logic.
- Usability: The business logic of our project would be required to be developed in a way that it would be able to match the expectations of the potential users.

3.3. Feasibility Study

3.3.1. Economic Feasibility

While performing an economic feasibility study about our project, we were able to find out that majority of the economic focus has to be given to the cloud deployment which includes continuous integration and continuous delivery (CI/CD) along with serverless deployment, and such focus has to be given on a pay-per-use model rather than pay-as-you-go or subscription-based models.

Also, for the external portion of our project which includes the official website, a subscription-based model is to be taken care of on a yearly basis for the domain registration/renewal and web hosting.

Also, while looking at the business model of our project, we found out that developing this project in such a way that every educational institution would have their own chatbot with the implementation of our project shall be highly beneficial in long-term.

3.3.2. Technical Feasibility

During the technical feasibility study over our project, we were able to find out that we have enough technical manpower to construct our system from scratch and deploy everything in a proper manner to ensure the performance and scalability in the level of existing academic chatbots.

Also, the way we would be developing our project would help us in integrating our final product further with existing systems implemented by different educational institutions.

3.3.3. Operational Feasibility

While performing the operational feasibility study over our project, we found out that the majority of the costs would be covered in operating our final product, which includes things like maintenance, monitoring, scaling and more.

While implementing our project in a way to play a role together with the existing systems being used in different educational institutions, there might be some changes in the strategies used in operating our project in order to adapt with different workspace environment, but those changes wouldn't create more hassles for the implementation, evaluation and adaptability of our project, therefore preventing from exponential growth in our operational costs.

CHAPTER 4: SYSTEMS DESIGN

4.1. Introduction

Systems Design is the process used to define the overall architecture of a system including its core elements, modules and components, along with their interfaces and the data, which are all based on the results of planning, requirements analysis and feasibility study. With the help of systems design, we can define, develop and design the final product in an efficient manner, ensuring the satisfaction of every need and requirement and the fulfillment of the proposed results. With this systemic approach, we would be able to develop a coherent and a well-running system.

4.2. Software Analysis and Design Tools

Software analysis and design is a major procedure to go through for becoming able to transform the requirement specifications of a software project into its implementation. All of the functional and non-functional expectations of the software have to be specified during the phase of designing requirement specifications. These requirement specifications are available in the form of human-readable format or in a diagrammatic form without requiring any technical aspect of the project.

Different software analysis and design tools for our project are provided below.

4.2.1. Data Flow Diagram (DFD)

Figure 1: Level 0 DFD

While looking at the simple overview of how data flows in our system, it is seen that the data-from end-users goes to the messaging platform, and then the messaging platform sends it to the cloud.

Furthermore, the cloud returns the response data back to the messaging platform, and the messaging platform displays it for the end-users.

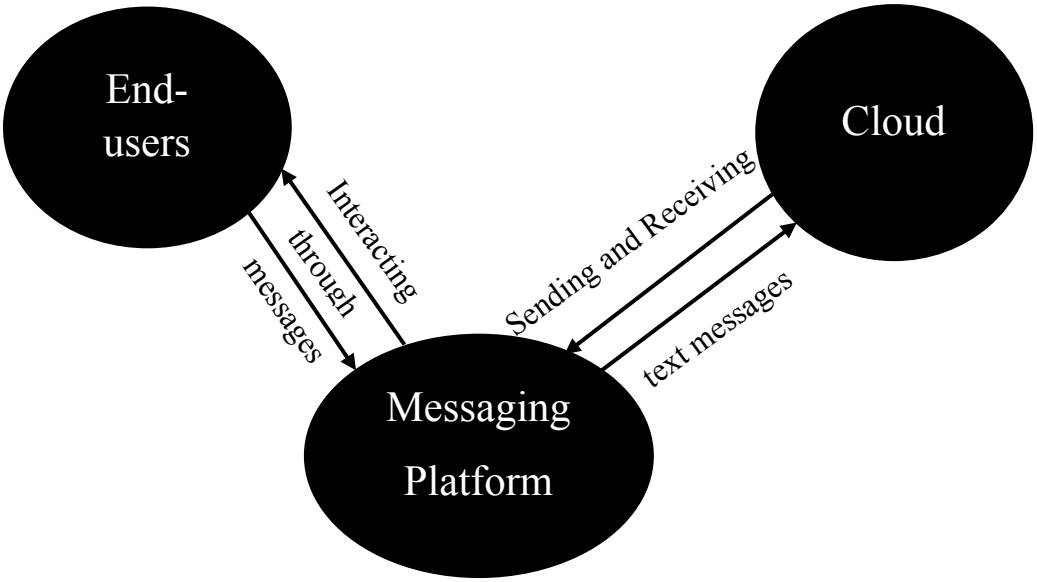

Figure 2: Level 1 DFD

Level 1 DFD is a little more advanced form of its predecessor. In this case, it is seen that the end-users and the messaging platform can interact with the help of messaging platform.

Also, the messaging platform and the cloud go through the process of sending and receiving text messages to respond back to the end-users.

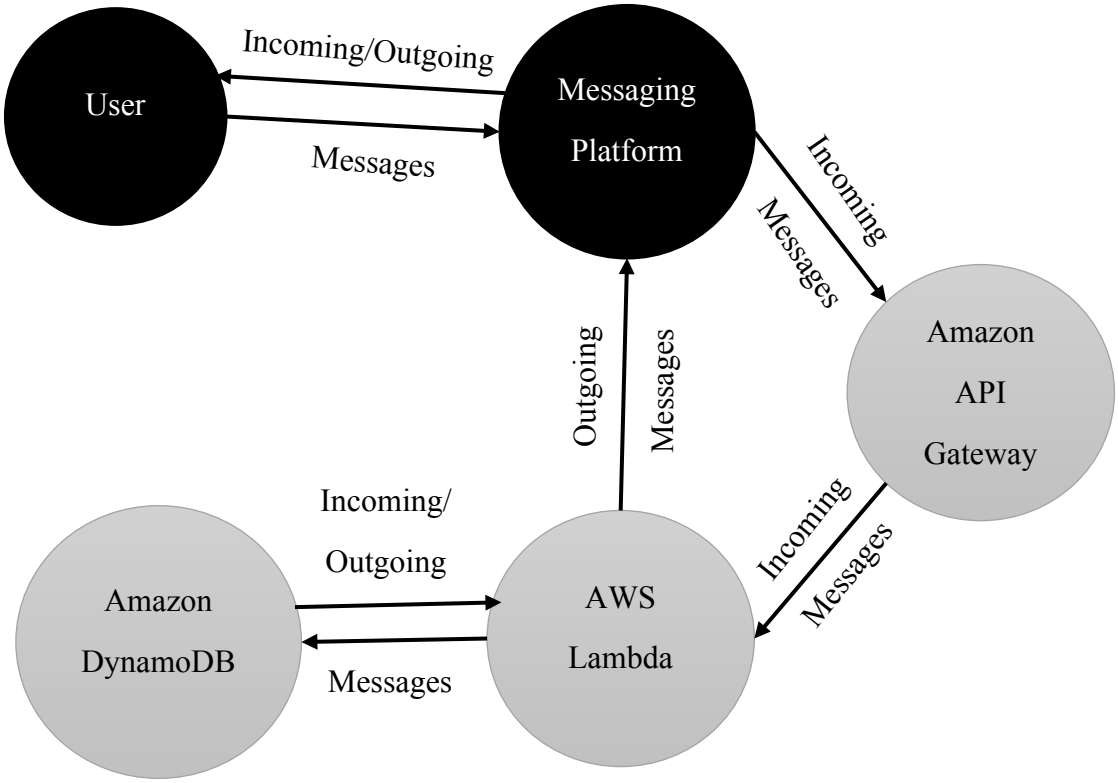

Figure 3: Level 2 DFD

Similarly, in the Level 2 DFD, we can see a clear distinction on how the data flow occurs in overall in our system. Initially, there is a setup of end-users, the messaging platform, Amazon API Gateway, AWS Lambda and Amazon DynamoDB.

The end-users and the messaging platform go through the process of exchanging messages with each other. When the end-users send a message to the messaging platform, the platform sends the messages to the Amazon API Gateway endpoint, and then the Amazon API Gateway makes the communication possible with other services in the AWS cloud, by first sending the messages to AWS Lambda.

Then, AWS Lambda analyzes the messages with the help of the data retrieved from Amazon DynamoDB, to generate a proper response for the end-users. When a response is generated, AWS Lambda sends it directly to the messaging platform, without again having to go through Amazon API Gateway. Finally, the messaging platform distributes the appropriate messages back to the end-users.

4.2.2. System Flow Diagram (SFD)

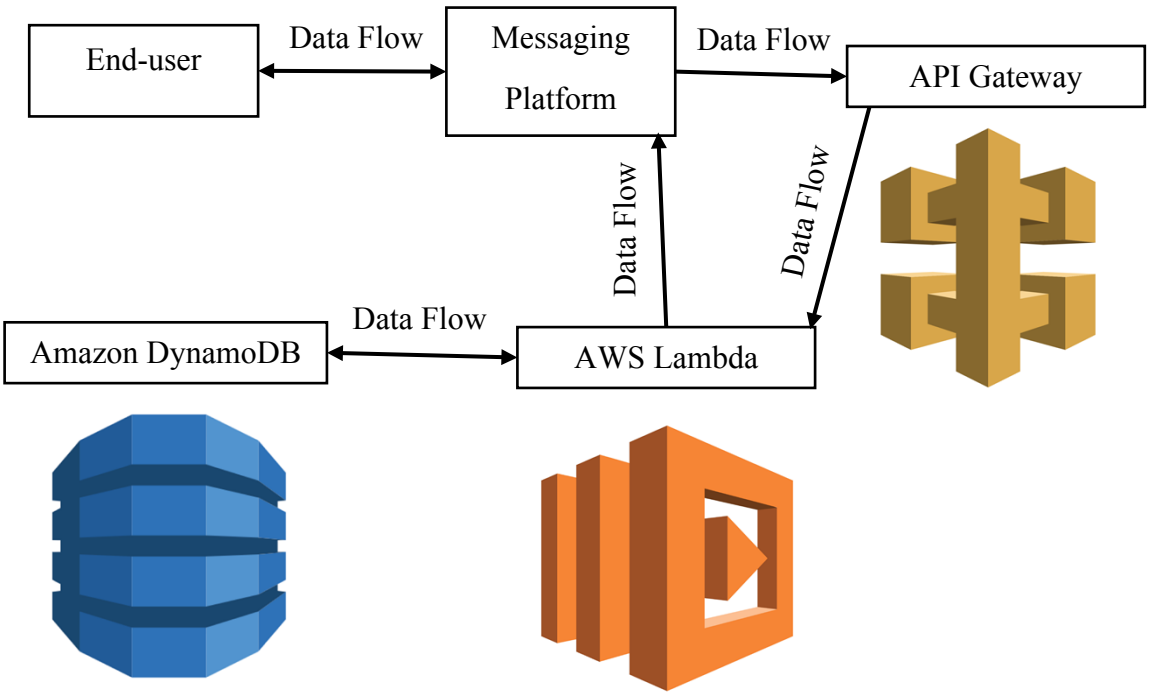

Figure 4: System Flow Diagram (SFD)

The System Flow Diagram is very similar to the Level 2 DFD about which we talked earlier. The only difference is, the SFD shows the overall implementation of the system with how different activities happen with the help of data flow.

It is seen that an end-user communicates with the messaging platform, and then the messaging platform calls an Amazon API Gateway endpoint to send the user's messages, and the API Gateway further flows the data over to AWS Lambda for further analysis and processing. AWS Lambda takes help from Amazon DynamoDB to generate text responses, and sends the text responses back to the messaging platform. Furthermore, the messaging platform displays the messages in front of the end-user.

4.2.3. Use Case Diagram

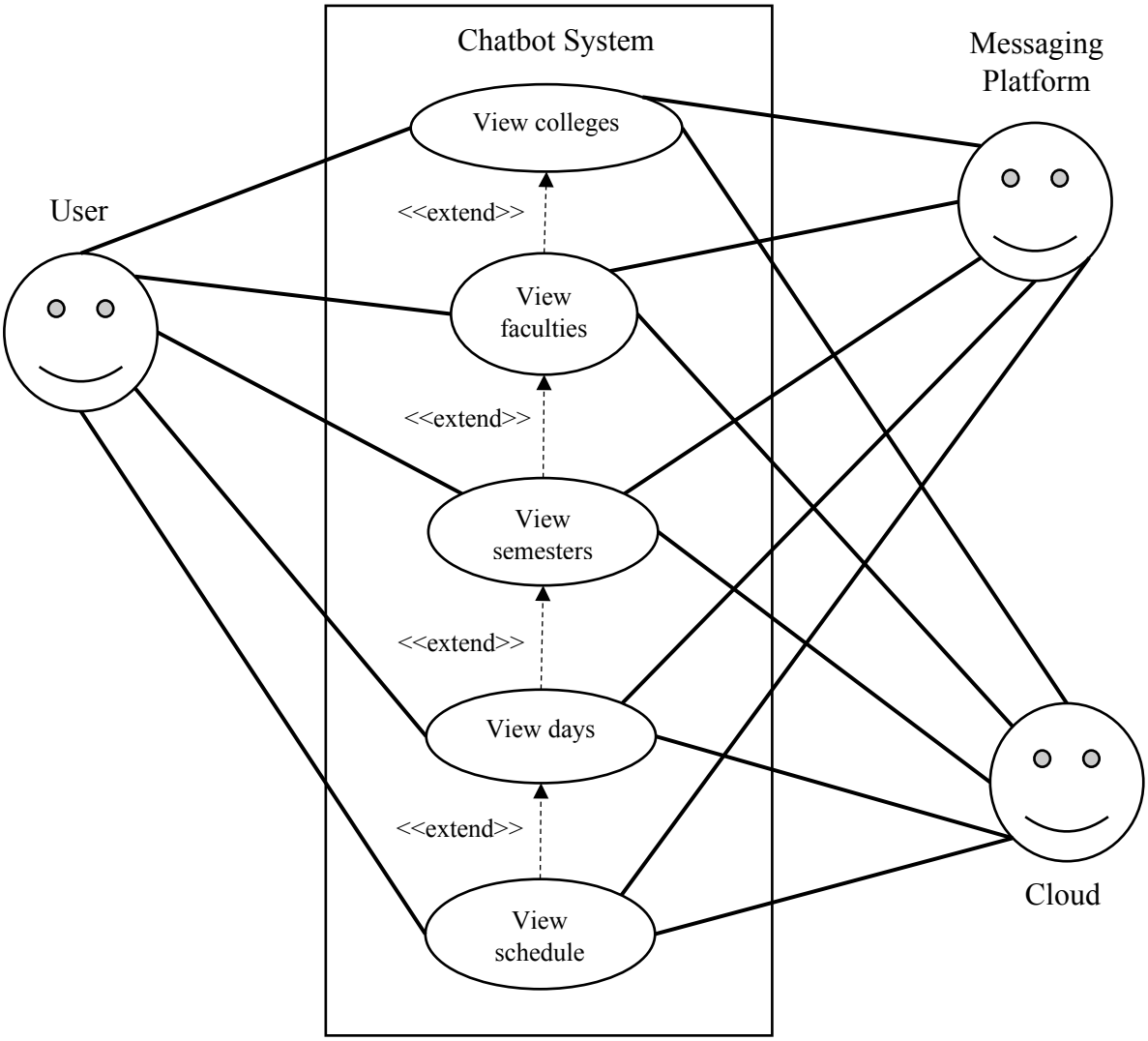

Figure 5: Use Case Diagram

In the use case diagram, we generally demonstrate the different functionalities in the system, with the parameters and actors involved within the system. In our case, the different actors include the end-users, the messaging platform and the cloud services.

The end-users are able to view colleges, then view faculties within the respective college, then view the available semesters in the current academic period, and further view the days of the week when classes run, and finally view the daily class schedules.

To view all of these things, there is a huge involvement of the messaging platform and the cloud. The messaging platform takes the users' requests regarding different actions, and sends them to the cloud, which is further analyzed and processed in the cloud before returning responses back to the messaging platform. The responses are displayed back for the end-users in a proper manner within the messaging platform.

4.2.4. Flowchart

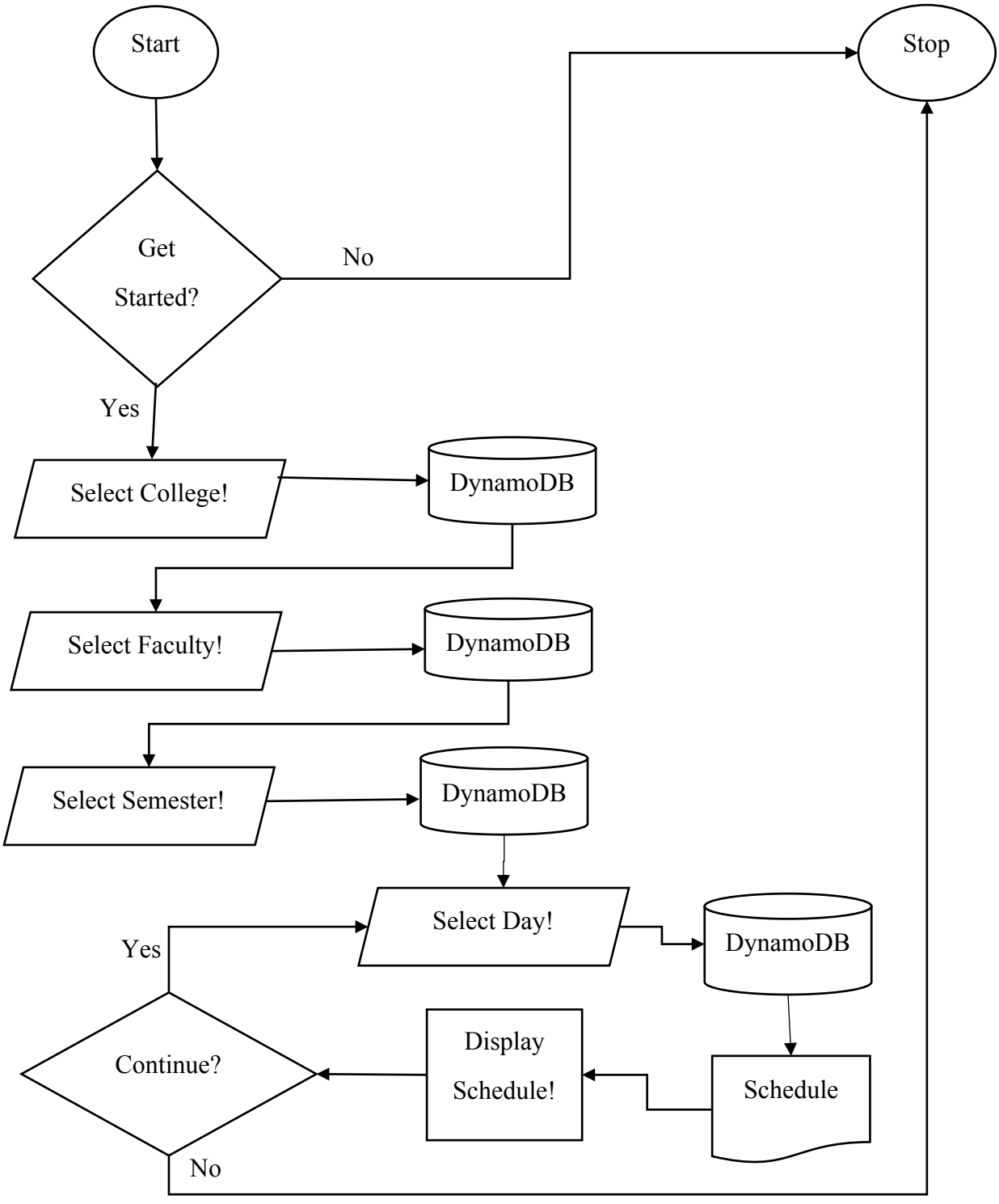

Figure 6: Flowchart

CHAPTER 5: METHODOLOGY

5.1. Introduction

During the time of developing a project proposal for a specific project, a model has to be implemented for the analysis, design, planning, implementation and maintenance of the final output as the output of the project development phase. This model is considered as a "methodology", which is implemented by a project manager or a project lead for achieving different goals in order to fulfill the planned objectives within a pre-defined working schedule and a fixed working budget. The methodology features all of the procedures to be followed during the project development phase, along with different systematic diagrams explaining about the working principles of the project and the technologies that are to be used or implemented throughout the project development phase.

5.2. Software Development Life Cycle (SDLC)

Software Development Life Cycle, also abbreviated as SDLC, is a mechanism consisting of six different phases, ranging from requirement analysis to deployment and maintenance, which helps in producing high quality software in the shortest time period possible with less budget requirement. This mechanism offers the ability of being to create high quality software quickly in a well-structured manner following a series of well-tested phases to make the output ready for use in production environments. There are several models of SDLC, and one of them is the DevSecOps model, which is all about including security in the DevOps SDLC model, which is an upgraded form of the Agile and Lean software development models.

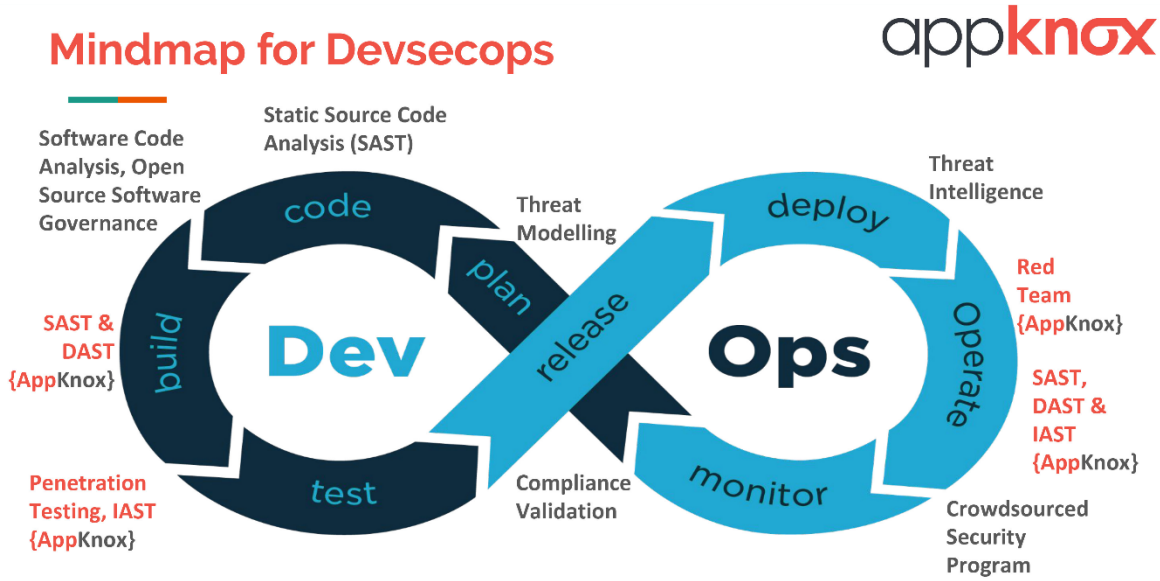

Figure 7: Mindmap for DevSecOps (*Source: appknox*)

We will be using the DevSecOps model for this project, which is all about automating the integration of security at every phase of the SDLC, beginning from the initial planning and analysis through integration, testing, deployment and continuous delivery. This model also integrates application and infrastructure security in a highly flexible way into the existing Agile and DevOps models, processes and technologies addressing security issues that emerge during

the development phase. In short, DevSecOps just refers to the integrated form of development, security and operations, which fulfills the "software, safer, sooner" motto with the automation of secure software delivery without slowing down the software development life cycle.

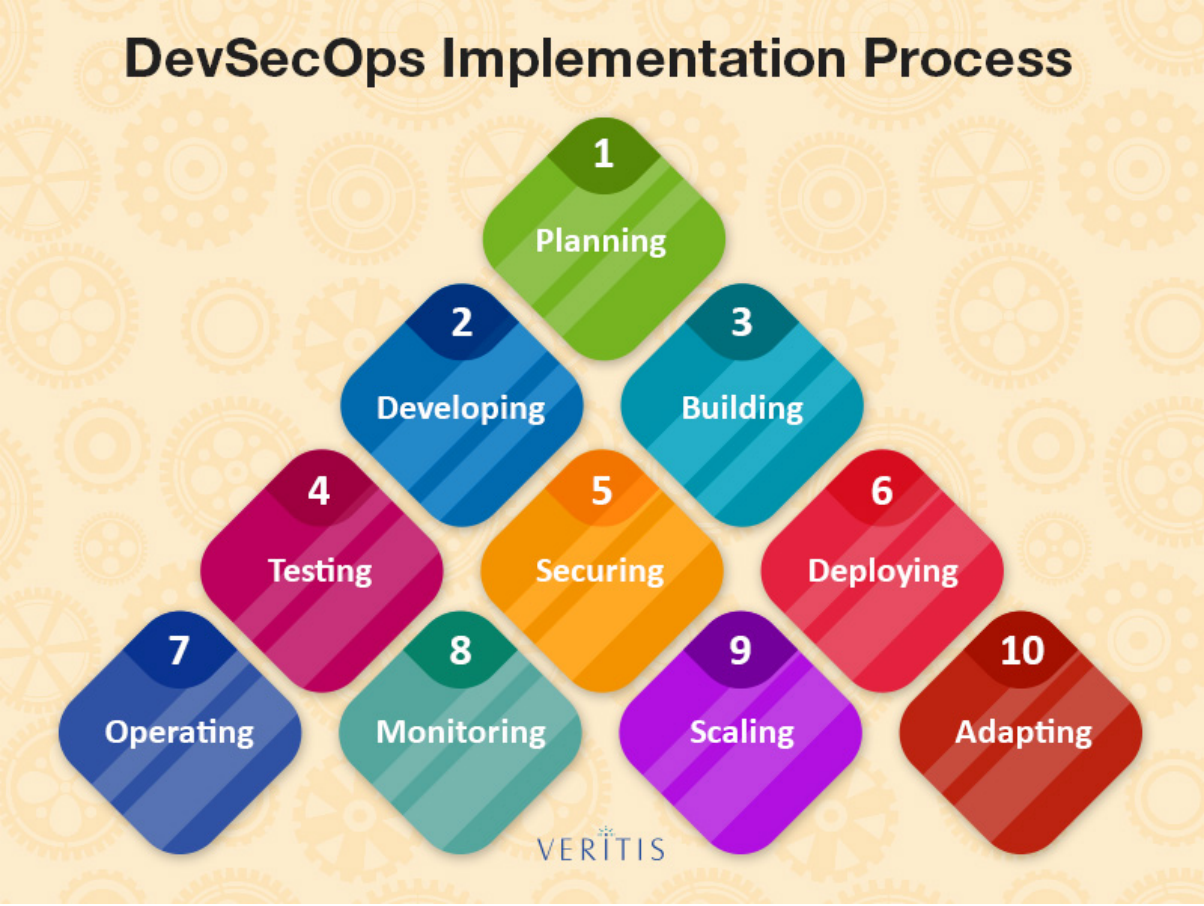

Figure 8: DevSecOps Implementation Process (*Source: Veritis*)

The following are the different phases required for implementing the DevSecOps model:

i. Planning: In the DevSecOps Life Cycle, planning is the first approach that needs to be taken for any software development project, and this is the same phase where the major highlight of this model, i.e., security, has to be focused on. During this stage, the DevSecOps practitioners shouldn't just focus on feature-based project planning, but also should focus on security, performance, scalability, acceptance test criteria, application interface and functionality along with threat defense models.

ii. Developing: In mainstream and traditional SDLC models like Waterfall, Agile, Lean, Scrum and Kanban, developers focus on the "what to do" approach, but in this model, they should rather keep an eye on the "how to do it" approach. Here, the developers have to combine all of the available resources for guidance on implementing reliable coding practices and code review systems for themselves along with other team members to follow up for the next phase of the model.

iii. Building: Since DevSecOps is all about automation, automated build tools are really useful in uplifting the overall implementation process in a large scale.

These build tools enable test-driven development along with the standards for release artifact generation and utilize different tools for ensuring that the design aspect is in parallel with the development team's coding and security standards through static and dynamic code analysis. These automated tools can also be used for the identification of vulnerable libraries which are being used in the application so that they can be replaced with their alternatives or newer version ones.

iv. Testing: DevSecOps professionals need to utilize automated testing capabilities following the strong testing standards including those for the frontend and backend environments, the API and the database. During this phase, passive security testing also plays a huge role as a development stage testing methodology where general security testing isn't conducted on the target application directly.

v. Securing: In DevSecOps, traditional security testing practices are also performed, but there is some sort of tendency in this model for identifying issues towards the end of the development phase. At this phase, advanced security scanning practices can be performed to become more aware of the underlying issues and figure out whether a threat is a critical one or not.

vi. Deploying: The development phase can be boosted up with automated provisioning and deployment for more consistency. There are different Infrastructure-as-a-code (IaaC) tools which can be used to perform the audit properties and configurations as mentioned earlier to ensure that there are secure configurations across the IT infrastructure.

vii. Operating: The Operating phase is all about continuous monitoring and upgrades. DevSecOps practitioners ensure to deploy IaaC tools for updating and securing the organization's infrastructure in an efficient manner with no trace for any human error. The Operations team members have to be careful all the time and keep watching if any zero-day vulnerabilities come out for the tools being used.

viii. Monitoring: This phase of the DevSecOps model is about consistently looking at the irregularities in the application regarding security to prevent from any sort of breach that could occur after deployment in the production side. Therefore, it is highly effective to implement a strong, continuous and consistent monitoring program in real-time for keeping track of the system performance as well as to identify any exploits available in the early stages of the development.

ix. Scaling: In the past, organizations had to spend several hours and money for the maintenance of their large data centers. But, as of now, the world of storage servers has seen a drastic change with the introduction of different IaaS, PaaS, SaaS technologies, virtualization solutions and serverless computing. With these technologies, an organization can scale its IT infrastructure to the fullest as well as replace the infrastructure efficiently in case any threat is detected. Doing all of these things is impossible if anyone had to go with traditional data centers.

x. Adapting: DevSecOps isn't just about development, security and operations, but it is also about a methodology of innovation which substantially keeps adding

value to an organization's infrastructure. Therefore, continuous improvement is a key point for any organization's growth. Being able to adapt with continuous improvement, external changing trends, latest tools and technologies to achieve the desired growth is the major factor required for an organization to improve its DevSecOps practices including security, functionality and performance.

5.3. NLP Life Cycle

Natural Language Processing is a set of different techniques, which can be broken down into five major steps in the lifecycle. The lifecycle provided below is customized in order to be suitable in building a chatbot with NLP.

i. Tokenization: Tokenization is a technique used to separate text into different pieces, known as tokens. This method also throws away certain characters like punctuation, and the resultant tokens are linguistic representative of the text.
ii. Normalization: Normalization is a method of processing the text in order to find out the common spelling mistakes, which could alter the intended meaning of the end-user's request.
iii. Entity Recognition: Entity Recognition is a process which enables the chatbot to identify the context of the conversation. In our case, this process is used in identifying whether the user is mentioning his/her college, faculty, semester or a specific day.
iv. Dependency Parsing: Dependency Parsing is a method used in splitting up the user's message into its constituent objects. This process makes the chatbot able to identify what the user is willing to convey.
v. Generation: Generation is the final stage of the life cycle, where the response is generated to send back to the end-user. The techniques mentioned earlier fall under Natural Language Understanding (NLU), and enable the bot to understand the meaning of the user's request. The technique of generation exists under Natural Language Generation (NLG), and in this stage, the output of above NLU steps is retrieved, and the output of NLG is generated based on what the end-user is willing to request.

5.4. Programming Technologies

5.4.1. JavaScript

JavaScript, also abbreviated as JS, is a lightweight, interpreted (i.e., just-in-time compiled) programming and scripting language with first-class functions which conforms to the specifications defined in ECMAScript. It is a high-level, multi-paradigm language with curly-bracket syntax, dynamic typing along with prototype-based object-orientation.

We have been using JavaScript in our project as a medium to implement the Node.js technology, and we are also using it for the frontend part of our project.

5.4.2. Node.js

Node.js is an open-source, cross-platform runtime environment used to implement JavaScript in the backend. It runs on the V8 engine, and is able to execute JavaScript codes outside the web browser. Using Node.js, the developers can just use JavaScript for

developing different tools and server-side scripts in order to deploy dynamic webpages, i.e., to generate webpages dynamically before sending to the end-user's web browser.

We have been using Node.js as our major programming technology for the development of our chatbot by integrating it with the cloud.

5.4.3. HTML5/CSS3

HTML is a markup language, which is used to present the structure of a webpage in the user's web browser, and the latest major version available for HTML is HTML5, which is also suggested to be used by the World Wide Web Consortium (W3C). Along with HTML, CSS is a stylesheet language, which is mainly used to offer proper design to the structure rendered with HTML in the web browser. It offers the ability to improve content accessibility, while taking care of the flexibility and control of different elements in the webpage.

We have been using HTML5 and CSS3 in our project for the frontend web development to offer users an overview of our chatbot built with Node.js, on top of the Amazon AWS cloud services.

5.4.4. Facebook Developers Messenger Platform

The Messenger Platform isn't the same as Facebook Messenger, but instead it is a Facebook product that offers different APIs and web plugins along with a complete WebView which enables you to build anything that creates amazing experiences among people for Facebook Messenger. This platform can be used by creating an application in the Facebook Developers platform.

We are using the Facebook Developers Messenger Platform in our project to receive messages and other events sent by Facebook users and integrate it with our webhook in order to get responses from the cloud.

5.4.5. Amazon API Gateway

Amazon API Gateway is a fully managed service, which enables developers to build, publish, maintain, monitor and secure APIs at any scale. With the use API Gateway, an individual can build different APIs which offer real-time two-way communicative behavior for different applications. This service can be used with web applications, serverless and containerized workloads.

In our project, we have been using Amazon API Gateway to receive messages from the Facebook Developers Messenger Platform and send it to AWS Lambda for further processing to generate responses for the end-users.

5.4.6. AWS Lambda

AWS Lambda is a serverless compute service from Amazon, which allows you to execute codes without having to provision different servers or create workload-aware cluster scaling logic, plus maintain and manage different event integrations and runtimes. Using Lambda, a developer can run codes for any kind of application or backend service, without requiring any sort of administration.

We are using AWS Lambda in our project to process the messages from end-users retrieved through Amazon API Gateway using Natural Language Processing and respond back to the Facebook Developers Messenger Platform with a proper response to send back to the end-users.

5.4.7. Amazon DynamoDB

Amazon DynamoDB is a NoSQL, key-value and document database service in AWS, which offers single-digit millisecond performance at any scale. This service offers fully managed, multi-active, multi-region and durable database system with in-built security, backup, restore and in-memory caching for any applications on the Internet.

We have been using Amazon DynamoDB alongside AWS Lambda to generate responses for the end-users.

5.5. Implemented Algorithms

5.5.1. Multinomial Naïve Bayes

Naïve Bayes is a simple algorithm used to construct classifiers. These classifiers are simply different models which assign class labels to different problem instances, and they are represented as vectors of feature values. The class levels are designed from a finite set. One of the Naïve Bayes algorithm is the Multinomial Naïve Bayes classifiers. It features a multinomial event model [11].

Within the multinomial event model, there are different samples representing the frequencies with which specific events are generated by a multinomial. There is also a feature vector, often identified as a histogram. This event model is mostly used for classification of texts, based on the representation of events with respect to the occurrence of a word in a single set of text [11].

While being expressed in log-space, this classifier turns into a linear classifier as below:

$$\begin{aligned}\log p(C_k \mid \mathbf{x}) &\propto \log\left(p(C_k)\prod_{i=1}^{n} {p_{ki}}^{x_i}\right)\\ &= \log p(C_k) + \sum_{i=1}^{n} x_i \cdot \log p_{ki}\\ &= b + \mathbf{w}_k^\top \mathbf{x}\end{aligned}$$

Equation 1: Log-space Expression of Multinomial Naïve Bayes classifier

If the given class and the feature value don't occur simultaneously in the training data, then the frequency-based probability estimation would position to the zero value, as the estimation value is directly proportional to the number of occurrences of the feature value. If this happens, then it will be problematic as it can wipe out every information in other cases when multiplied. Therefore, a small-sample correction, known as pseudocount, is generally integrated in every estimate in order to ensure no probability estimate is equal to zero [11].

5.6. Database Design

The database structure for our project is pretty straight-forward. Since there is no requirement of any servers, we don't require any major database to store data. The only thing that we will be requiring to store would be the dynamically generated, pre-fed response sets for the end-users in order to process with AWS Lambda.

For the fulfillment of this database architecture, we would be using Amazon DynamoDB because it is the best one available out there for storing database entries in key-value pairs. The reason why we decided to go with key-value pairs for this purpose is because it appears to be the best method to store the type of data required for our project.

NoSQL database technologies are non-tabular, and the data is stored in the form of key-value pairs, differently than normal relational tables in RDBMS. Therefore, there is no way to build an ER Diagram for such database technologies, and also no way to design table structures.

The structure of our database changes every time we feed new response set to it, since things are stored in key-value pairs. Also, there is no requirement for primary keys, since we can easily retrieve the required data just from the defined key for a certain response set.

CHAPTER 6: RESULTS AND DISCUSSION

6.1. Initial Proposed Output

While developing a proposal plan for our project, we were expecting to have a chatbot as our final output with the completion of our project. The chatbot was supposed to be integrated with Facebook and Telegram, with the ability to be integrated further in other platforms such as WhatsApp, Google Assistant, Hangouts Chat, Slack, Twitter, Line, Viber and InterBot. Every student in an educational institution would be able to get their daily class schedules within their fingertips by communicating with our chatbot through Facebook Page or Telegram handle. The response rate would be dependent upon the bandwidth offered during any specific period of time in our cloud deployed instances. In overall, our project was supposed to be fulfilling a great role in eliminating the problems that students are facing in being provisioned with their daily class schedules, and also offering them the ability to get responses in quick and efficient manner.

6.2. Final Output

As of now, we have fulfilled every part of our expected output, plus we have also designed, developed and deployed a front-end environment, which offers the demonstration of our project in action, along with several guidelines and walkthrough videos explaining the usage of our chatbot in different platforms. Also, the frontend environment consists of a web-based chatbot, which runs with the help of the same assets powering our chatbot on Facebook and Telegram.

6.3. Output Overview

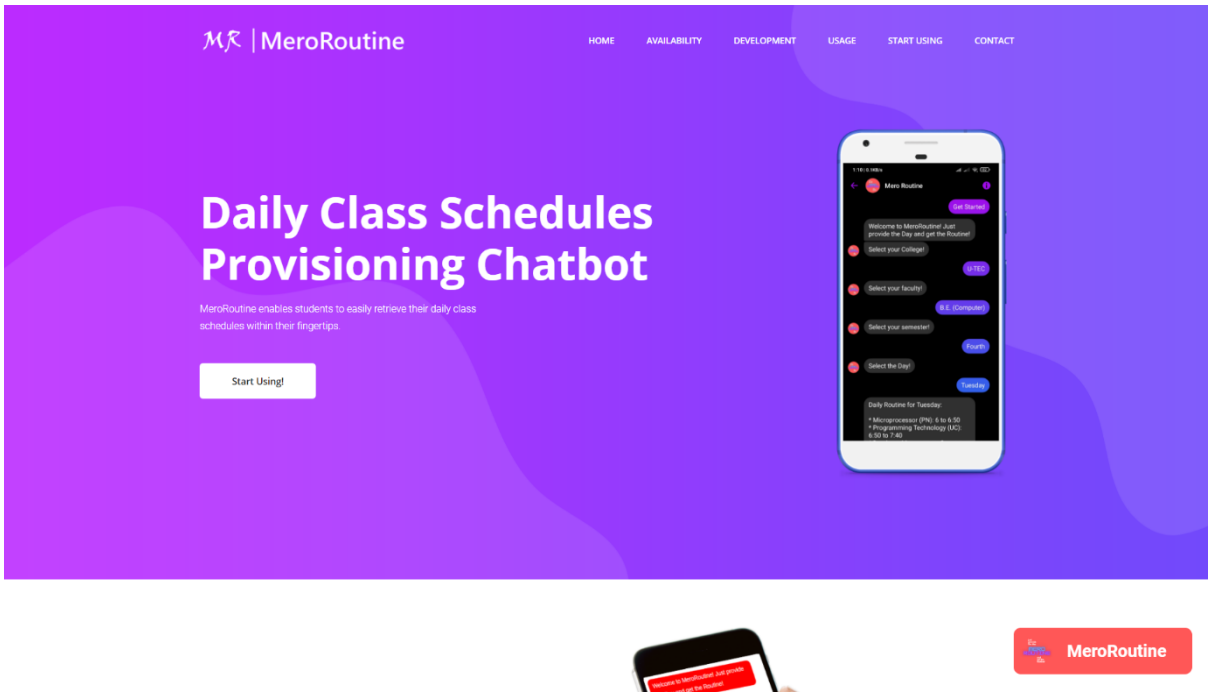

Figure 9: Front-end Environment: Major Highlight

Figure 9. features the upper section of the front-end environment, which features the webpage navigation menu, a UI/UX design mockup of our chatbot on Facebook Messenger and a brief overview of our project.

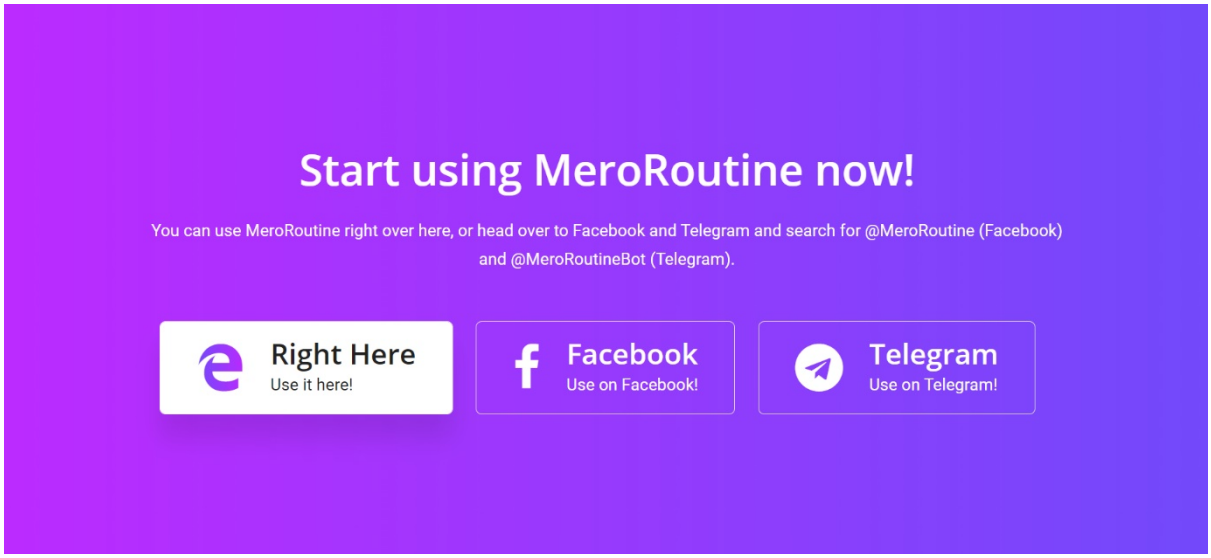

Figure 10: Front-end Environment: "Start Using" section

Figure 10. consists of different internal and external hyperlinks, which can be used to experience our chatbot on the web, use it on Facebook or on Telegram. This is also a part of our front-end environment.

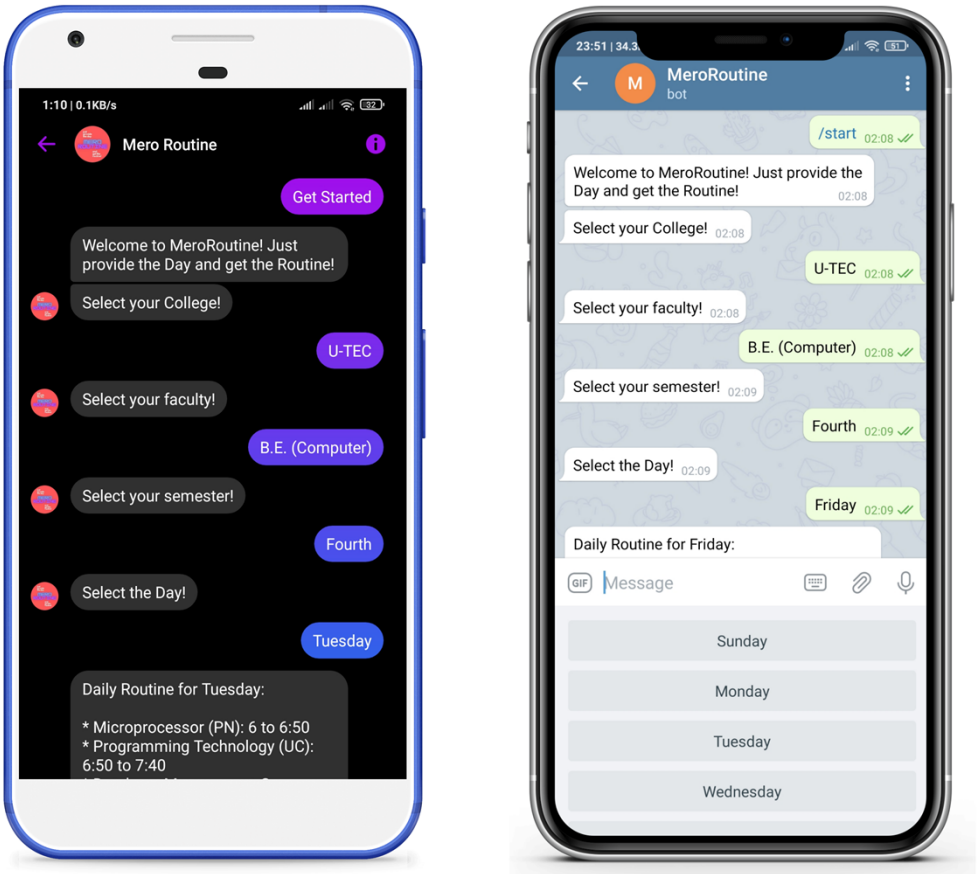

Figure 11: Chatbot Integration on Facebook and Telegram

Figure 11. features the UI/UX design mockup for our chatbot's integration on Facebook and Telegram. The integration of our chatbot on Facebook is one of the major portions where we had to perform a lot of activities during the project development phase.

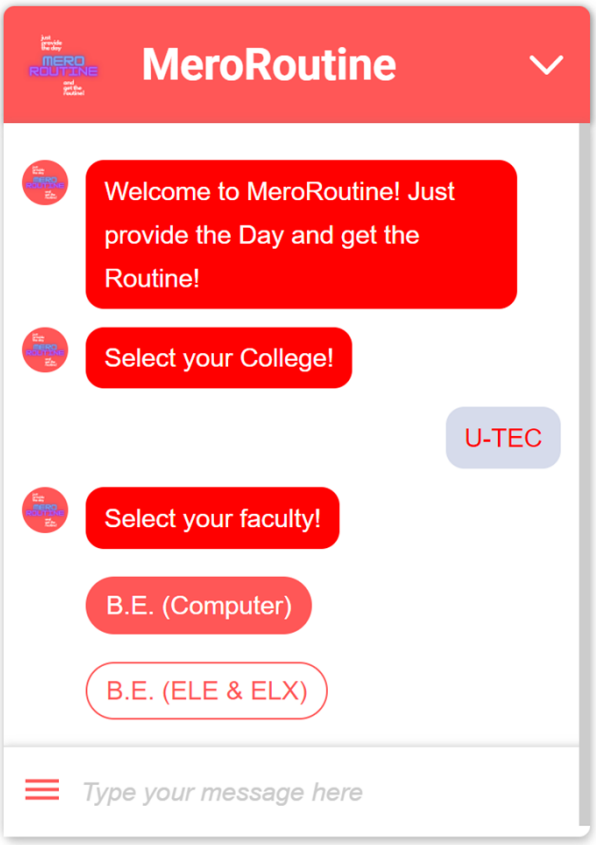

Figure 12: Chatbot on the Web

Figure 12. gives an overview of the workflow of the chatbot on the web; i.e., on our front-end environment. The functionality of our chatbot is same on the web and on the Telegram integration, whereas for Facebook, there are some minor differences in the persistence menu.

6.4. Work Completed

With the final report submission and defense phase coming forward, we are proud to acknowledge that we have successfully completed 100% of our initially proposed project. This includes a chatbot, which has been integrated to Facebook and Telegram, and also a front-end environment, which gives a brief overview of our project and also features the web version of our chatbot.

The chatbot has been built using Node.js, and comprises of the usage of several Amazon AWS cloud services for various activities such as fetching user response, analyzing them, generating response, maintaining the persistence in conversation, and many more. Also, the chatbot is completely deployed over the cloud, making use of Amazon API Gateway, AWS Lambda and Amazon DynamoDB.

Initially, we had only proposed to build the chatbot with the integration on Facebook and Telegram, but in the current phase, we have moved a lot further, and developed a wonderful front-end environment, which has been deployed on GitHub Pages, and also pointed a domain

name to it, therefore making it easier to browse over the web for everyone. As of now, the front-end environment can be accessed through https://meroroutine.whoisbinit.me.

The front-end environment consists of technical information about our project, and also features multiple walkthrough videos explaining and demonstrating the usage of our chatbot on different platforms.

We had proposed to integrate our chatbot on Facebook and Telegram, but as of the current phase, we have included one more platform, where our chatbot can be accessed; i.e., the front-end environment itself. This means, our potential web visitors wouldn't have to leave the webpage in order to start using Facebook or Telegram to try out our chatbot. Therefore, it is easier for users to directly get hands-on experience with our chatbot while browsing the information about it, so that they can rate its usability and think about using it on Facebook or Telegram as returning chatbot users.

6.5. Limitations

Everything has its own limitations, and so does our project. The major limitation of our chatbot is, when we feed a lot of data to the chatbot, for example, adding the daily class schedules of multiple educational institutions, then the chatbot shall be unresponsive sometimes.

However, this limitation can be stripped away when we implement this project in real world, because every institution would have their own version of our project; i.e., a chatbot per institution. Besides this, there is probably no other limitations that we need to worry about regarding our project.

6.6. Problems Faced

While developing our project, we didn't face much problems, because we were already sure about all of the methodologies to be implemented in our project. However, one time when we had to go through a lot of hassles is when we weren't able to generate a persistent access token for our Facebook page to be used in our chatbot, in order to make the chatbot accessible while messaging our Facebook page.

The problem was that we were only able to use a temporary page access token, which would expire on its own few days after its generation. Due to this, we would have to update the Facebook page access token every once in a while, to make our chatbot accessible over Facebook. After trying for several days looking up for every page in the Facebook Developers documentation and several queries on StackOverflow, we were able to mitigate this problem, and continue further in developing our project.

We also had to go through some minor hassles while ensuring everything fed to the chatbot were structured properly, but since these issues were very simple, we were able to solve them quickly.

6.7. Testing and Results

Testing is an essential phase of the Software Development Life Cycle, in which various types of tests are performed to figure out errors in different systems. Since every system has errors and bugs in it, it is necessary to perform different tests with the intention of finding out the flaws during the runtime of the system. We carried out the testing phase following the DevSecOps model, mainly during the development and after the deployment of our project.

Before testing, multiple test cases are defined in order to feed data in some way to the system. When these test cases are checked, the results are gathered, evaluated and then a qualitative indication of the system quality and reliability is determined. These results act as the basis in case any modification in the business logic is required.

6.7.1. Unit Testing

We carried out the unit testing phase after completing the building phase of the DevSecOps model. The major purpose of performing this test was to figure out issues in the developed modules, without taking care of their relationships with other modules.

For any issues discovered, some changes were made in our project, and when every module were tested properly, we concluded the unit testing phase.

6.7.2. Black-box Testing

After completing the unit tests and resolving the underlying issues, we deployed our project for black-box testing. This phase of software testing includes the testing of the business logic of the system, rather than testing the internal technical workflow as in white-box testing.

For black-box testing, we built few test cases with respect to our requirements, and carried out the tests. Some of the tests were functional, whereas the others were non-functional. Multiple valid and invalid inputs are defined, and then tests are carried out to determine whether a correct output is generated or not during the black-box testing phase.

6.7.3. White-box Testing

White-box testing features the testing of internal workflow of the system, rather than the business logic as in black-box testing. During this phase of software testing, we selected different inputs to pass throughout our codes, and then we created breakpoints and started debugging the data transfer workflow to determine functional outputs.

For any issues discovered, we worked on resolving them, and carrying out further unit tests to ensure the changed modules are running fine.

6.7.4. Integration Testing

The integration testing phase was carried out after the unit tests. During this phase, we checked the interfaces in our systems, and also tested their integration with multiple platforms to discover any issues. This kind of tests also offered us the ability to determine that the performance of specific modules wasn’t degraded.

6.7.5. Validation Testing

When the integration tests were carried out and the results were obtained, we carried out validation tests to determine interfacing issues and resolved them.

6.7.6. Acceptance Testing

While performing acceptance tests, we created multiple test cases and carried out the tests in order to make sure the results of our project are properly usable by the end-users. In case any requirement definition problems were identified, we had to consider them, and make our system error-free.

6.8. Work Schedule

Before getting started with any project, we have to prepare a working schedule consisting of several topics that we would be working on throughout the project development phase. For the same reason, the following is the Gantt Chart representing our work schedule in a total span of 3 months, i.e., 12 weeks ranging from the phase after proposal defense to final project report submission and defense:

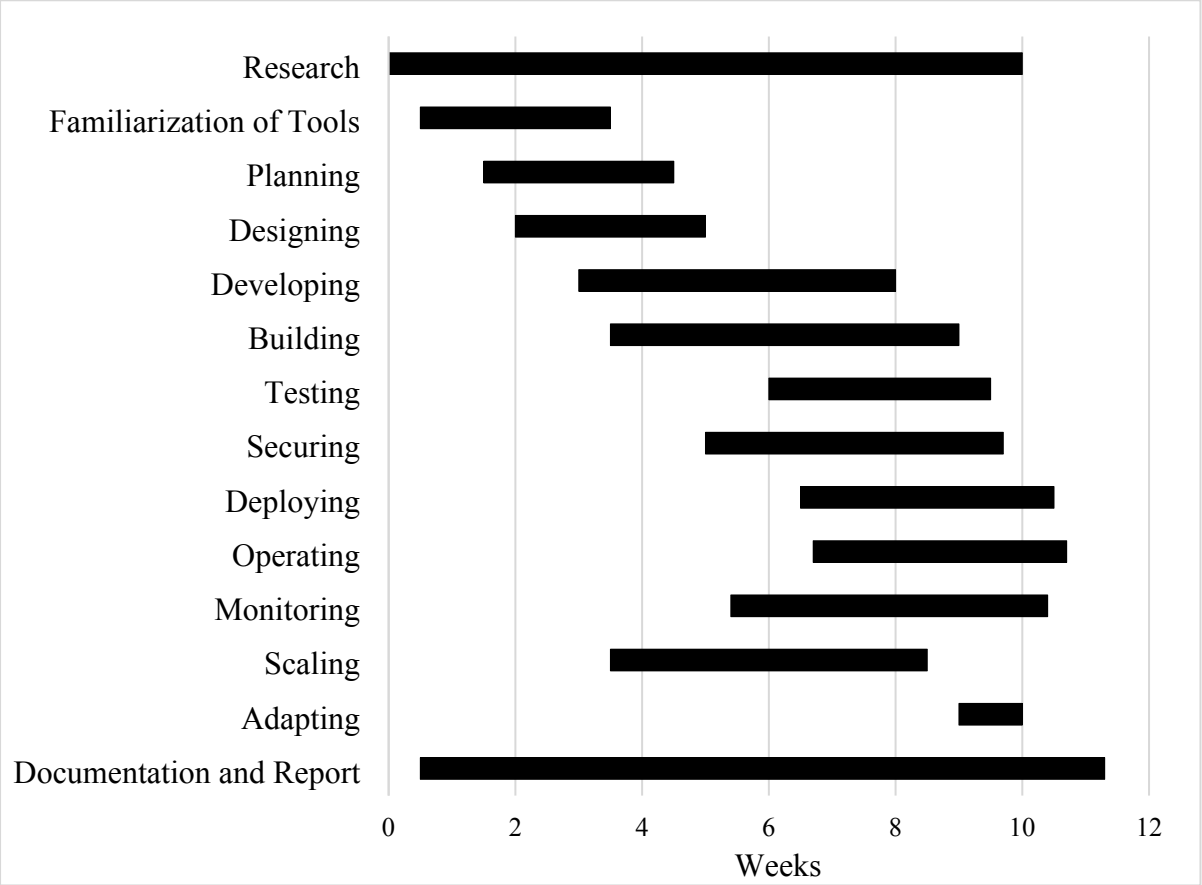

Figure 13: Gantt Chart for Work Schedule with DevSecOps Model

- Week 1-3: At the beginning of the project development phase, we started out with a research on different programming tools that we would be using throughout the project, and we also started getting familiar with these tools, and finally launch the planning and designing phase at the beginning of second week. We also started preparing documentations during this time and kept on updating information in it followed by our progress in the upcoming weeks for the project.
- Week 4-6: Near the starting phase of the fourth week, we started the development and building process, and in the meantime, we also kept focusing on the security and scalability capabilities of our project. We also kept testing and monitoring our application during the beginning of the sixth week for any issues related with scalability and performance.
- Week 7-9: When the seventh week is about to start, we started focusing on the deployment, integration and operations part along with continuous development phase.

The testing phase went on till the end of the ninth week along with deployment, operations and monitoring.

- Week 10-12: The final quarter of our project development life cycle is all about adapting with our application, along with some final tests related with deployment and operations, accompanied with continuous monitoring and status checks of our application. Since we had concluded the proposed plans for our project before approaching the mid-term defense, we were focusing mainly on the preparation of final project report and presentation in this phase, along with all the documentations started during the beginning of the project development phase. Finally, at the end of the twelfth week, we have completed our final project report and are submitting it to the respective department for further review and evaluation.

CHAPTER 7: CONCLUSION AND FUTURE ENHANCEMENT

7.1. Conclusion

With the successful completion of all of our proposed plans before approaching the final defense, we can say that the application of our project, MeroRoutine, is now available to be used by the general public, especially the engineering students studying at United Technical College, Bharatpur-11, Chitwan. MeroRoutine is the name given to the final output of our project, and the name was given in order to deploy our project in an efficient manner for making it accessible for everyone to use.

Our project on Daily Class Schedules Provisioning System offers a chatbot that allows students to get their daily class schedules for every specific day within their fingertips. As of now, the chatbot is available on Facebook, Telegram and the web version. The Facebook integration of our project can be experienced by messaging the @MeroRoutine page on Facebook, whereas for the Telegram integration, an individual can simply start a conversation with a user with the username, @MeroRoutineBot. Also, the front-end environment of our project features the web-based version of the chatbot, which can simply be used by clicking on the chatbot widget in the bottom-right corner of the webpage.

Along with these, all of the objectives and features are fulfilled in a proper manner, following the DevSecOps model of Software Development Lifecycle, which enables people to develop their proposed projects quickly and effectively, since development, security and operations are all covered in a single model, therefore removing the requirement to have multiple skilled manpower for all three categories.

7.2. Scope for Future Enhancement

There are several ways our project can be further enhanced in the future. A major feature that we are interested in adding for our project is a mobile application, that offers a chatbot for students to communicate with, just in the same way as in the front-end assets of our project.

Also, an advanced method of enhancing our project would be to develop a business model, in which we would be building a system that would allow educational institutions to build their own chatbots and integrate them with their Facebook page or embed them in their official websites, allowing students to get their daily class schedules. For this, the only thing the institutions would have to work on would be to feed our system with the daily class schedules of every student division for every day of the week, and integrate their Facebook pages with our chatbot. This way, every institution would have their own daily class scheduling chatbot.

REFERENCES

[1] M. J. Pereira, L. Coheur, P. Fialho, and R. Ribeiro, “Chatbots’ greetings to human-computer communication,” *arXiv preprint arXiv:1609.06479*, 2016.

[2] O. Deryugina, “Chatterbots,” *Scientific and Technical Information Processing*, vol. 37, no. 2, pp. 143–147, 201.

[3] A. S. Ahmad, "Brain inspired cognitive artificial intelligence for knowledge extraction and intelligent instrumentation system," *2017 International Symposium on Electronics and Smart Devices (ISESD)*, Yogyakarta, Indonesia, 2017, pp. 352-356.

[4] S. Singh and H. K. Thakur, "Survey of Various AI Chatbots Based on Technology Used," *2020 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, Noida, India, 2020, pp. 1074-1079.

[5] Bradeško, Luka & Mladenić, Dunja, "A Survey of Chatbot Systems through a Loebner Prize Competition," *2012 Slovenian Language Technologies Society Eighth Conference of Language Technologies*, Ljubljana, Slovenia, C. 34.

[6] L. -K. Lee, Y. -C. Fung, Y. -W. Pun, K. -K. Wong, M. T. -Y. Yu and N. -I. Wu, "Using a Multiplatform Chatbot as an Online Tutor in a University Course," *2020 International Symposium on Educational Technology (ISET)*, Bangkok, Thailand, 2020, pp. 53-56.

[7] S. Das and E. Kumar, "Determining Accuracy of Chatbot by applying Algorithm Design and Defined process," *2018 4th International Conference on Computing Communication and Automation (ICCCA)*, Greater Noida, India, 2018, pp. 1-6.

[8] R. Gauba, “Makerobos - Enterprise Chatbots,” *2019 Makerobos Innovation Labs*. Accessed on: April 4, 2021. [online] Available: https://www.makerobos.com/

[9] A. Proctor, “Beacon - Your digital coach,” *2019 Staffordshire University*. Accessed on: April 4, 2021. [online] Available: https://www.staffs.ac.uk/students/digital-services/beacon

[10] S. Jhunjhunwala, “The Most Definitive Virtual Event Platform for Medium-scale and Fortune 500 companies | e2m.live,” *2020 e2m.live, WS Group*. Accessed on: April 4, 2021. [online] Available: https://e2m.live/

[11] Jason D. M. Rennie, Lawrence Shih, Jaime Teevan, and David R. Karger. 2003. “Tackling the poor assumptions of naive bayes text classifiers.” *In Proceedings of the Twentieth International Conference on International Conference on Machine Learning (ICML'03)*. AAAI Press, 616–623.

BIBLIOGRAPHY

- V. A. Prasad and R. Ranjith, "Intelligent Chatbot for Lab Security and Automation," *2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, Kharagpur, India, 2020, pp. 1-4.
- V. Dutt, S. M. Sasubilli and A. E. Yerrapati, "Dynamic Information Retrieval With Chatbots: A Review of Artificial Intelligence Methodology," *2020 4th International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, Coimbatore, India, 2020, pp. 1299-1303, doi: 10.1109/ICECA49313.2020.9297533.
- S. E. Ahmady and O. Uchida, "Telegram-Based Chatbot Application for Foreign People in Japan to Share Disaster-Related Information in Real-Time," *2020 5th International Conference on Computer and Communication Systems (ICCCS)*, Shanghai, China, 2020, pp. 177-181, doi: 10.1109/ICCCS49078.2020.9118510.
- N. Albayrak, A. Özdemir and E. Zeydan, "An overview of artificial intelligence based chatbots and an example chatbot application," *2018 26th Signal Processing and Communications Applications Conference (SIU)*, Izmir, Turkey, 2018, pp. 1-4, doi: 10.1109/SIU.2018.8404430.
- H. K. K., A. K. Palakurthi, V. Putnala and A. Kumar K., "Smart College Chatbot using ML and Python," *2020 International Conference on System, Computation, Automation and Networking (ICSCAN)*, Pondicherry, India, 2020, pp. 1-5, doi: 10.1109/ICSCAN49426.2020.9262426.
- L. Tommy, C. Kirana and L. Riska, "The Combination of Natural Language Processing and Entity Extraction for Academic Chatbot," *2020 8th International Conference on Cyber and IT Service Management (CITSM)*, Pangkal, Indonesia, 2020, pp. 1-6, doi: 10.1109/CITSM50537.2020.9268851.
- S. N. M. S. Pi and M. A. Majid, "Components of Smart Chatbot Academic Model for a University Website," *2020 Emerging Technology in Computing, Communication and Electronics (ETCCE)*, Bangladesh, 2020, pp. 1-6.

APPENDICES

- Figure 1: Level 0 DFD, Figure 2: Level 1 DFD and Figure 3: Level 2 DFD are three different levels of Data Flow Diagram to demonstrate the systems design architecture for our project.
- Figure 4: System Flow Diagram (SFD) consists of a System Flow Diagram, which explains how data flows within our system, and how different decisions are defined in order to control different events.
- In Figure 5: Use Case Diagram, the end-users and other entities in our system are represented in the way they interact with different use cases in which the end-users are involved.
- Figure 6: Flowchart demonstrates the overall workflow of the system, from the beginning to the end.
- *Figure 7: Mindmap for DevSecOps* is retrieved from "*Why You Need DevSecOps in Mobile Apps?*" (https://www.appknox.com/blog/devsecops-in-mobile-apps/).
- The original source for *Figure 8: DevSecOps Implementation Process* is available at "*https://devops.com/devsecops-implementation-process-and-road-map-security-at-every-step/*".
- Equation 1: Log-space Expression of Multinomial Naïve Bayes classifier defines the way how the Multinomial Naïve Bayes classifier turns into a linear classifier.
- Figure 9: Front-end Environment: Major Highlight and Figure 10: Front-end Environment: "Start Using" section are two different sections of the front-end environment of our project. The technical part of these figures is developed using HTML5, CSS3 and JavaScript.
- Figure 11: Chatbot Integration on Facebook and Telegram include the UI/UX Design Mock-ups of our chatbot on Facebook and Telegram, made using Figma (a collaborative interface design tool), and they feature the user interface of our chatbot on the two social communication platforms. The integration of our chatbot over to these platforms is made possible with the help of developer documentation and environment offered by these platforms.
- Figure 12: Chatbot on the Web demonstrates the user interface of our chatbot on the web, i.e., accessible through our front-end environment. This web chat widget is made

possible using HTML5, CSS3 and JavaScript, with the integration over to the AWS cloud services through webhooks.

- During the several phases of implementing the DevSecOps model for our project, we are following the time schedule as described in Figure 13: Gantt Chart for Work Schedule with DevSecOps Model.
- To learn more about AWS Lambda for running codes without thinking about servers or clusters, you can go through the official documentation available here: https://aws.amazon.com/lambda/.
- The official documentation for Amazon DynamoDB offers an informative overview of this fast and flexible NoSQL database technology for any scale, and you can find it here: https://aws.amazon.com/dynamodb/.
- A simple Facebook Messenger chatbot can be built on top of serverless computing following the guide here: https://www.serverless.com/blog/building-a-facebook-messenger-chatbot-with-serverless.
- An example code for running a Slack chatbot using AWS Lambda and Amazon API Gateway can found here: https://github.com/aws-samples/aws-serverless-chatbot-sample/.
- You can find the front-end environment of our project demonstrating about different features of the chatbot at https://meroroutine.whoisbinit.me/.