

# Exploring AI Chatbot Technology as an Information Support System for Water Resources Management

Sifiso Patience Ncube  
MSc Thesis Identifier **WSE-HI.22-13**  
May 2022

Updated version, May 2022



# **Exploring AI Chatbot technology as an Information Support System for Water Resources Management**

Master of Science Thesis  
by  
**Sifiso Patience Ncube**

Supervisor  
Prof. Dimitri Solomatine

Mentors  
Dr. Gerald Corzo  
Mr. Santiago Duarte  
Ms. Ha Do

Examination Committee  
Prof. Dimitri Solomatine  
Dr. Gerald Corzo  
Dr. Vsevolod Moreydo

This research is done for the partial fulfillment of requirements for the Master of Science degree at the  
IHE Delft Institute for Water Education, Delft, the Netherlands.

**Delft**  
09/05/2022

Although the author and IHE Delft Institute for Water Education have made every effort to ensure that the information in this thesis was correct at press time, the author and IHE Delft do not assume and hereby disclaim any liability to any party for any loss, damage, or disruption caused by errors or omissions, whether such errors or omissions result from negligence, accident, or any other cause.

© Sifiso Patience Ncube May 2022.

This work is licensed under a [Creative Commons Attribution-Non Commercial 4.0 International License](#)





# Abstract

The advent of Artificial Intelligence (AI) brought with it improved efficiency in the execution of various computational tasks. One area of application of this technology is in the development of conversational applications otherwise known as chatbots. The chatbot technology employs the Natural Language Processing (NLP) and Machine Learning (ML) components of AI to develop models that enable computers to better understand natural human languages. This human-computer interaction has allowed for easier and faster access to information. This research, therefore, employs this chatbot technology to develop a prototype conversational application that extracts flooding information from EPA SWMM and relays this information to citizens in flood-prone areas. SWMM simulations for Do Lo, an area in Hanoi, Vietnam, that is severely impacted by floods annually were used as the study case. A database of intents with different terminologies that users may use in conversation with the chatbot was created. These terminologies then served as training phrases for the NLP and ML algorithms in the Google Dialogflow platform which was used in developing the chatbot. Then the Pyswmm library was used in developing Python functions to extract the data from the SWMM model. For integrating the Dialogflow chatbot with the SWMM model the Flask web framework hosted on the Google App Engine was used. Also, to ensure that computations were accurately done for each user based on their input location, a database was created to store the user's session ID and location. For the user interface a webpage was designed and integrated into the application using Flask. The final developed chatbot prototype was capable of responding to the user's questions on the flooding situation at their location. To test the performance of the chatbot, a pilot test was done where a majority of the respondents reported that the application provided accurate results and they also found the application easy to use and worth recommending to others. In conclusion, a prototype chatbot application was developed that could respond to the user's questions providing, otherwise inaccessible flooding information obtained from SWMM. If deployed, the chatbot would assist in flood awareness creation by allowing the users to make informed decisions in case of flooding. Finally, the study recommended among other issues that further research could incorporate rainfall forecast data in the SWMM simulation to provide citizens with real-time information for timely response in times of flooding.

**Keywords:** Flood-risk management, flood preparedness, AI chatbot, decision-support, Google Dialogflow, webhooks, Flask, EPA SWMM, Pyswmm Natural Language Processing (NLP)

# Acknowledgements

Firstly, my gratitude goes to my supervisory team: Dr. Gerald Corzo, Prof. Dimitri Solomatine, Mr. Santiago Duarte and Ms. Ha Do for the priceless support you gave me. To my mentor Dr. Corzo, thank you for the time and effort you put into making this research a success, you always pushed me to do better. To Prof. Dimitri, thank you for your guidance and contribution to the research. To my mentors Mr. Santiago Duarte and Ms. Ha Do, you never tired of making time to help me, thank you.

I am grateful to the Dutch government for awarding me the opportunity to study under OKP Scholarship.

My appreciation also goes to all my lecturers in the Hydro-Informatics department, you taught those difficult modules with so much grace and patience. Thank you for your guidance.

To my fellow HI class of 2022, you guys were the best. My fellow Zimboz, you rock!

To my children, Zoe, Anodiwanashe and Akudzweishe thank you for always wishing me well and loving me still. My siblings, my mother and my magogo, and the clan, thank you for your prayers and for making the journey a bit lighter. Tafadzwa thank you for carrying the load.

Special mention to Geert Martin and Sukoluhle, words fail me. I cannot begin to describe how grateful I am to have you two in my life. Your support throughout this period is beyond priceless. May the Lord God reward you for all your efforts and love. Thank you also for opening up your home to me. Thanks again to the clan in Germany.

Most importantly, to the Almighty God, the one who sustains me, my friend, my everything, if You had not been on my side, the torrent would have engulfed me. Thank You, Father.

# Table of Contents

<b>Abstract .....</b>	i
<b>Acknowledgements .....</b>	ii
<b>Table of Contents .....</b>	iii
<b>List of Figures .....</b>	vi
<b>Abbreviations.....</b>	viii
<b>Chapter 1    Introduction.....</b>	1
1.1    Background.....	1
1.2    Research Gaps .....	3
1.3    Statement of the Problem .....	3
1.3.1    Technological Problem .....	3
1.3.2    Water Resources Problem (Urban flooding) .....	4
1.4    Objectives of the Study.....	4
1.5    Research Questions.....	5
1.6    Innovation .....	5
1.7    Practical Value.....	5
<b>Chapter 2    Literature Review.....</b>	7
2.1    Pluvial Floods .....	7
2.2    Decision Support Systems for Flood Management .....	10
2.3    Chatbot Technology .....	11
2.3.1    The History of Chatbots .....	12
2.3.2    Basic Functions in Chatbots .....	12
2.3.3    Chatbot Applications in Water Resources Management.....	13
<b>Chapter 3    Technological Tools Used in the Research.....</b>	15
3.1    Natural Language Processing Algorithms .....	15
3.1.1    Pre-processing Techniques in NLP .....	15
3.1.2    Word Embeddings in NLP .....	21
3.2    Dialogflow .....	23
3.3    Webhooks .....	25
3.3.1    Flask .....	25
3.4    EPA SWMM.....	25

<b>Chapter 4 Study Area.....</b>	<b>29</b>
4.1 General Description .....	29
4.2 Flooding Trends and Awareness .....	31
4.3 The Urban Drainage Network in Ha Dong .....	32
<b>Chapter 5 Methodology: Chatbot Development .....</b>	<b>34</b>
5.1 Setting the Scenario .....	34
5.2 Chatbot Development Using Dialogflow .....	34
5.2.1 Comprehending NLP Algorithms .....	37
5.2.2 Dialogflow: SWMMBot.....	41
5.2.3 EPA SWMM Data Extraction .....	51
5.2.4 Geocoding .....	54
5.2.5 Creating a Database.....	55
5.2.6 Designing the Webpage .....	56
5.3 Chatbot Integration Using Webhooks .....	57
<b>Chapter 6 Results and Discussion .....</b>	<b>60</b>
6.1 The Developed SWMMBot Conversational App.....	60
6.1.1 Scenario 1: Flooding Situation .....	60
6.1.2 Scenario 2: Flooding situation for Navigation .....	63
6.1.3 Scenario 3: Flooding Situation for a Flood with a Ten Year Return Period .....	65
6.1.4 Scenario 4: Actions to Take During Flooding .....	67
6.1.5 Scenario 4: Flood Proofing a House or Neighbourhood .....	69
6.1.6 Not Flooded or Location Outside Do Lo .....	69
6.1.7 Discussion SWMMBot .....	71
6.1.8 Limitations of the SWMMBot Application .....	72
6.2 The Graphical User Interface: Website .....	73
6.3 SWMMBot Pilot Test.....	78
6.3.1 Ease of use.....	78
6.3.2 Accurate and understandable responses .....	79
6.3.3 I could easily access my location on the map and other locations .....	79
<b>Chapter 7 Conclusions and Recommendations.....</b>	<b>84</b>
7.1 Conclusions .....	84
7.2 Recommendations .....	85
<b>References .....</b>	<b>88</b>
<b>Appendices .....</b>	<b>94</b>
Appendix A. - Research Ethics Declaration Form .....	94

Appendix B. -	Vectorization Output.....	96
Appendix C. -	Python Functions .....	97
Appendix D. -	Default Fall Back Intent Responses.....	99
Appendix E. -	The Other Developed Intents .....	100
Appendix F. -	EPA SWMM .....	102
Appendix G. -	Python Script to Trigger Actions on the Webhook .....	104
Appendix H. -	Sample HTML and CSS .....	105
Appendix I. -	Questionnaire for the Pilot Test .....	106

# List of Figures

Figure 1: Flood displacement risk by region.....	8
Figure 2: Synthesis assessment of observed change in heavy precipitation and confidence in human contribution to the observed changes in the world's regions. Source:(IPCC, 2021) .....	9
Figure 3: Major and minor flow interactions during an extreme event Adapted from (Schmitt, et al., 2004, Ellis, et al., 2009)	9
Figure 4: A flooded manhole .....	10
Figure 5: Natural Language Processing Pipeline Source: Geitgey ( 2018).....	16
Figure 6: Tokenization .....	16
Figure 7: An Illustration of Word-Based Tokenization .....	17
Figure 8: Character-Based Tokenization.....	18
Figure 9: Subword Tokenization.....	18
Figure 10: Stop Words Removal .....	19
Figure 11: Lemmatization vs Stemming .....	20
Figure 12: Continous Bag of Words .....	22
Figure 13: Skip-Gram Method .....	23
Figure 14: Node(manhole) Flooding Adapted from (Rossman and Huber, 2017).....	27
Figure 15: Do Lo, Hanoi, Vietnam .....	30
Figure 16: Landcover in Do Lo, Yen Nghia, Ha Dong Source:(Do, 2021) .....	31
Figure 17: SWMM Drainage system in Do Lo, Yen Nghia, Ha Noi. Source: Do (2021)... .	32
Figure 18: Chatbot Flow Processes .....	35
Figure 19: Webhook request .....	36
Figure 20: Webhook response.....	37
Figure 21: Tokenization .....	38
Figure 22: Stemming .....	39
Figure 23: Lemmatization .....	39
Figure 24: Vectorization.....	40
Figure 25: Word2Vec .....	41
Figure 26: Sample JSON object for 1 training phrase of the Maximum flooding depth intent	42
Figure 27: Flooded point intent .....	43
Figure 28: Setting the actions and parameters .....	43
Figure 29: Welcome and Flooded point intents .....	44
Figure 30: Checking if an address is in Do Lo.....	44
Figure 31: Checking whether a location is flooded.....	45
Figure 32: Flooding duration intent .....	46
Figure 33: Maximum ponded depth intent .....	47
Figure 34: Time of maximum flooding intent.....	48
Figure 35: What to do intent .....	48
Figure 36: Thank you and Goodbye Intents .....	49
Figure 37: JSON file showing intent detection confidence for a matched intent .....	50
Figure 38: Python code for running the model .....	51
Figure 39: Do Lo- Precipitation for 14-17 October 2020 .....	52
Figure 40: Precipitation for a flood with a 10-year return period .....	52
Figure 41: SWMM data extraction function .....	53
Figure 42: Flooded Nodes .....	53
Figure 43: Flow rate for Node LN31-5 extracted using Python .....	54
Figure 44: Flow rate plot for Node LN31-5 from SWMM .....	54
Figure 45: Geocoding.....	55

Figure 46: Different location names stored in the Database .....	56
Figure 47: Inserting values into the database table and retrieving the user's location.....	56
Figure 48: The Conversational flow Adapted from: (Raturi, 2020).....	58
Figure 49: Creating the Webhook .....	59
Figure 50: Integrating the HTML webpage into the Webhook.....	59
Figure 51: Scenario 1: Flooding situation .....	62
Figure 52: Scenario2: Flooding Situation for Navigation .....	63
Figure 53: Scenario 2: Conversation to assess the flooding situation in different parts of the city .....	65
Figure 54: Scenario 3: Conversation to compare the present flooding situation and that for a flood with a 10 year return period.....	67
Figure 55: Conversation on what to do during flooding .....	68
Figure 56: Flood barriers Source: <a href="https://floodcontrolinternational.com/flood-barriers/">https://floodcontrolinternational.com/flood-barriers/</a> .....	69
Figure 57: Chatbot conversation if the location is not flooded .....	70
Figure 58: Chatbot conversation for an area outside Do Lo .....	70
Figure 59: When Chatbot fails to connect to the server.....	72
Figure 60: The Developed Website.....	77
Figure 61: Ease of use .....	78
Figure 62: Accurate and easy to understand responses.....	79
Figure 63: Accessing locations from maps .....	80
Figure 64: Satisfaction with the chatbot.....	81

# Abbreviations

AI	Artificial Intelligence
API	Application Programming Interface
CSS	Combined Sewer Systems
DSS	Decision Support System
EPA SWMM	Environmental Protection Agency's Storm Water Management Model
HTTP	Hypertext Transfer Protocol
IPCC	Intergovernmental Panel on Climate Change
JSON	JavaScript Object Notation
NLP	Natural Language Processing
NLTK	Natural Language Toolkit
NLU	Natural Language Understanding
TF-IDF	Term Frequency- Inverse Document Frequency

The purpose of research has mostly been to build on available knowledge to enhance the way people live. This chapter, therefore, provides the context that motivates this research thesis mainly focusing on issues pertaining to urban flood management, challenges faced in abstracting data from modelling systems and possible solutions to these challenges. The set objectives to address these challenges are also discussed together with the research questions that arise. In addition, innovative aspects of the study are outlined together with the practical value of the research in water resources management.

## **1.1 Background**

Urban floods, like numerous other water-related disasters, are on the increase globally, due to an increase in extreme precipitation as a result of variations and changes in climate as reported by the Intergovernmental Panel on Climate Change IPCC (2021). On the other hand, as of 2007, global trends show that 54% of the population is now based in urban areas (Sørensen, et al., 2016). Consequently, urban development will continue to rise in response to this migration to provide housing and other necessary infrastructure. The resulting outcome is more impervious surfaces, and less infiltration leading to shorter times of concentration and higher peak flows when it rains. The resulting increased volumes of stormwater consequently put a strain on the drainage system which has to convey these volumes within a shorter time frame. Moreover, in cities with Combined Sewer Systems (CSS), a continued increase in the urban population results in more wastewater being produced, subsequently increasing the strain on the drainage systems whose capacity would thus have been exceeded. The drainage systems may then in turn surcharge and eventually overflow, causing flooding which results in the devastation of properties, disruption of economic activities, and even loss of lives (Najibi and Devineni, 2017).

To better understand these flood disasters, decision support and modelling technologies are used (Ahmad and Simonovic, 2006, Jung, et al., 2020). Thus they provide answers to issues at a cheaper cost and the use of computers allows for multiple tasks to be performed at a given time, for information to be processed at faster speeds and for the delivery of accurate and reliable results. To harness the benefits that come with computer use, in most urban areas, simulation models like the Environmental Protection Agency's Storm Water Management Model (EPA SWMM), MIKE-Urban, and SOBEK-Urban, among others, have been used widely for decision support. They allow for the design, planning and management of urban drainage systems.

One particularly important feature, in view of flooding, is the ability of these modelling systems to show locations in the drainage network where manholes are flooded, the flooded area and the depth of flooding. This information is critical for flood awareness creation since if disseminated on time, it can assist the affected population to make decisions on the proper course of action to take in times of flooding. The main challenge, however, is that there is no direct access to these modelling results by the general population. Chatbots offer a solution to this gap in information dissemination.

Chatbots are computer programs (agents) that use Artificial Intelligence AI-trained natural language to simulate human-computer interaction (Ranavare and Kamath, 2020) as one would with another human being. They make it possible for people to access modelling results without having to run the simulation model themselves. The widespread use of chatbots results, firstly from the fact that they allow for the transfer of information through normal conversation at the user's own pace and at a rate at which they can understand and assimilate it. Secondly, chatbots provide instant answers and support. In addition to this, they provide ubiquitous (round-the-clock) access to information which would be of great advantage for flood awareness creation amongst the general public.

Chatbots fulfill the user's request, which can either be a text or voice message, by matching the user's intent to a predefined response (Dahiya, 2017). The intent is the intention/objective of the user, that is, what they want to extract from the system or what they want to know as they type their question(s). As various users would ask the same question in different ways, a 'bank' of possible questions is created and then the Chatbot is trained using artificial intelligence to match the queries to their respective responses (Ranavare and Kamath, 2020, Setiawan, et al., 2020). For it to be deemed successful, the Chatbot has to understand the user's request to a point where it can match it with the required response, a process called 'parsing' the intent. To be able to simulate situations that change with time, i.e. dynamic content, the chatbot has to connect to a webhook, that is, "an HTTP (Hypertext Transfer Protocol) request by which real-time information is provided to other applications" (Google, 2021, 25 Oct). Due to the numerous benefits brought about by chatbots their use is evident in various fields such as banking, marketing, agriculture, flood awareness and water management, among others.

The popularity of chatbots has led to their adoption in a variety of fields to accomplish different tasks. Firstly, chatbots have successfully been used in the healthcare sector (Divya, et al., 2018, Kavitha and Murthy, 2019, Nivedhitha, et al., 2021). These chatbots are used to communicate with patients and provide a diagnosis from the symptoms that the patient will have provided to the chatbot. The patient can then be prescribed over-the-counter medication to take and prompted to set an appointment to visit a general practitioner where necessary. Secondly, application in agriculture has involved the development of, for example, the AgronomoBot, a chatbot that could access information from a network of sensors in a vineyard and display this to the user (Mostaco, et al., 2018). Also, chatbots have been used in education to enhance online

learning and for creating student enrolment platforms at universities (Reyes, et al., 2019, Deveci Topal, et al., 2021). Finally, other sectors where chatbots are being used extensively include retail, human resources management, banking, manufacturing, insurance logistics and marketing (Quah and Chua, 2019, Cheng and Jiang, 2021), among others. The benefit of chatbots in these sectors is that they enable users to gain instant access to the organization or company's products at any time, and be able to query the chatbot if they have any questions.

## 1.2 Research Gaps

From the reviewed literature, presented in detail in Chapter 2, the following gaps can be mentioned:

- Okon (2021) explored offline static flood forecasting information which would then need to be updated when new forecasts are received making the whole process rigorous. He analyzed scenarios of citizens using a mobile app to know more about the flood forecast reports of National Agencies in Nigeria. Based on his results, he recommends that the Chatbot should be connected to an API so that it can run automatic updates. Setiawan, et al. (2020) also recommends the development of a dynamic system. This gap in Chatbot applications could contribute to larger information systems that are dynamic.
- Another example of technological development in this area is “Ask Diana”, by Tsai, et al. (2019), who uses Fuzzy logic and a keyword mapping table for understanding the user’s requests. This performed poorly when the inputs were long. A more extensive study of the Google Dialogflow platform’s functionalities has been shown to help improve the Chatbot technology, particularly in Natural Language Processing (Tsai, et al., 2019, Setiawan, et al., 2020, Okon, 2021).
- There is also a need to increase the volume of data input into the Chatbot as this will enhance the training phase and ensure certainty of intent matching leading to a successful response generation Okon (2021).

## 1.3 Statement of the Problem

### 1.3.1 Technological Problem

Floods, by nature, bring with them financial, health and psychological impacts most of which are felt even long after the flooding ceases. To help alleviate these impacts, advancements in technology have brought with them improvements in flood risk and water resources management where a variety of modeling systems are now used to assist and inform decision-making. However, information obtained from these modelling systems is not easily accessible to the affected population. There is, therefore, a need to develop tools that will improve access to flooding information in an easy, straightforward and fast way, especially for the general public so that they are better informed about the flooding situation in their respective locations.

Modelling systems, have been, and will continue to be priceless as decision support tools in water and flood risk management. This is because they provide an understanding of how the real physical system works. However, results obtained from these systems are not readily accessible to the public. Nonetheless, advancements in Artificial Intelligence have enabled easier access to information through the development of chatbots. These provide a simplified solution that can assist in bringing modelling system results to the affected population. They allow for the direct extraction of information from the modelling systems, and instant feedback to the user in small portions that are easy to understand. Chatbot information systems have been used in numerous sectors, for example in aquaculture (Al Rasyid, et al., 2020); agriculture (Mostaco, et al., 2018); flood awareness (Tsai, et al., 2019, Okon, 2021), water supply (Setiawan, et al., 2020) and in the banking and marketing sector. Nevertheless, while there has been some adoption of the chatbot technology in flood risk management, this is still comparatively minimal and therefore needs improvement.

### **1.3.2 Water Resources Problem (Urban flooding)**

Most urban areas with combined sewer systems are prone to flooding during extreme rainfall events. This is because traditional grey infrastructure designs are now incapable of coping with increased stormwater flows resulting from high-intensity rainfall events (Salinas-Rodriguez, et al., 2018). High flows then lead to surcharging of the drainage pipes as the flow in the network becomes pressurized (Schmitt, et al., 2004). Extended surcharging may lead to the bursting of manholes and flooding as water outflows from the system. Furthermore, surcharging of the sewer network also hinders the surface water from entering the sewer network (Schmitt, et al., 2004) leading to increased ponding of the water above the ground surface. Flooding resulting from this failure of the drainage network leads to damages to property, disruption of transport services and environmental and health impacts which in turn lead to other indirect socio-economic effects (Cembrano, 2004). The objectives of the study are therefore outlined in the following sections.

## **1.4 Objectives of the Study**

The **main** aim of the study is to explore the technological tools and develop a prototype of a Chatbot urban flood information system.

To achieve the main goal, the **specific** objectives are:

- To analyze and learn the chatbot AI algorithms.
- To design a database of intents and actions of the information system. Develop a word bank of possible terminologies used by users in dialogue when asking questions related to flooding in Do Lo district (Vietnam).

- To analyze the flood information system in SWMM (variables, runtimes, etc.) and how to connect these using Python.
- To develop algorithms (build a library of functions) that will connect(couple) the Chatbot and SWWM.
- To test and validate the computational functionality of the connection results of the model runs.

## 1.5 Research Questions

- How can we connect (couple) the Chatbot and SWWM using Python? How can these technologies be integrated so as to have a water resources information system?
- Is the chatbot capable of extracting information from a modelling system, running the model and providing answers from the model run?
- Which properties (variables) are important for an AI chatbot flood information system?
- Is the AI chatbot be able to match the users' requests with the respective responses?
- What are the limitations and advantages of this technology?

## 1.6 Innovation

The innovations of the research include:

- Developing an AI-trained Chatbot to provide comprehensible information and support as a tool for creating flood awareness in urban areas.
- Developing a Chatbot coupled with an urban drainage modelling system to assess flooding.
- Application of AI chatbots in the management of urban drainage systems. This will enhance the speed with which information is extracted for more efficient flood management.

## 1.7 Practical Value

- Development of an intelligent system to provide direct sources of answers for flood management awareness in urban areas.
- Easy extraction of model results through asking the Chatbot questions to determine low and high flows in urban drainage systems.

- Development of a decision support information system for use by citizens, and to a lesser extent, decision-makers and water managers.

Chapter 1 introduced the basic concepts of Chatbots. The processes involved from the time a user types their request until they receive a response from the Chatbot were discussed. Then, a discussion of flooding in urban areas in light of extreme rainfall events resulting from changes in climate and growth in the urban population was done. Together with this, a description of how flooding occurs resulting from interactions between major and minor flows was included. Following that, a statement problem was defined together with the objectives and research questions to address these problems. Finally, the innovation of the study and the practical value were expounded on. In Chapter 2 a review of related literature will be done.

## **Chapter 2      Literature Review**

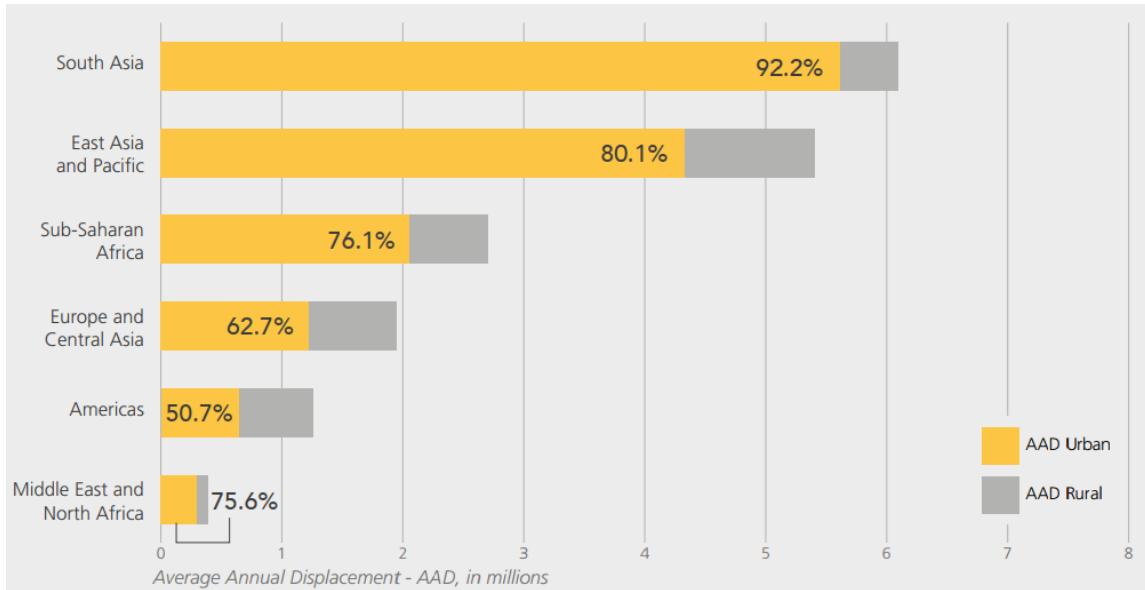
---

For every study, a review of literature is necessary to learn from other researchers' successes and failures and thus identify any gaps that need further research. This chapter, therefore, outlines the literature reviewed to better understand urban flooding, decision support systems and finally, the chatbot technology. The chapter expounds on the ideas introduced in Chapter 1 and sets the foundation for the methodology and results outlined in the proceeding chapters.

### **2.1 Pluvial Floods**

The catastrophic impacts of floods are felt annually in various areas across the globe. The Dartmouth Flood Observatory reports the death of more than 500000 people and the displacement of more than 650 million during the period 1985-2014. Economic damages during the same period are estimated to be around US\$800 billion making floods one of the costliest of natural disasters on the globe (Kocornik-Mina, et al., 2020). In Europe alone, direct flood damages between 2000 and 2012 were estimated to have been around 4.9 billion Euros per annum (Sørensen, et al., 2016). In the year 2021, European flood damages are estimated to have been around 10 billion Euros. Focusing on flood impacts on humans, the Internal Displacement Monitoring Centre IDMC (2019) 2019, reports that annually, on average, floods leave around 17 million people at risk of displacement and 80% of these are urban or peri-urban dwellers. Figure 1 shows the proportion of the number of people at risk of being displaced by floods annually per region, illustrating that a majority of these live in urban areas.

The occurrence of floods in urban areas is exacerbated mainly, by the increase in paved surfaces as construction increases to cater for the continuous increase in urban migration. Sørensen, et al. (2016) highlight that a greater percentage (54%) of the world population now lives in urban areas and this value is expected to continue to rise. Consequently, urban development continues to rise in response to this increase in urban migration (García, et al., 2015). This increased development leads to more paved surfaces which hinder infiltration, leading to reduced times of concentration and greater peak flows when it rains (Eckart, et al., 2018) and thereby resulting in more flooding.



*Figure 1: Flood displacement risk by region*

Furthermore, increased flooding results from the intensification and frequency of extreme precipitation (Salinas-Rodriguez, et al., 2018, IPCC, 2021). The IPCC report confirms the increase in heavy precipitation in most countries where data is available, as illustrated in Figure 2. This increase in the frequency of heavy precipitation will consequently translate to more recurrent flooding, the impact of which will be felt in urban areas which are usually developed on flood plains. The Global Disaster Assessment Report also concurs with these other findings stating a 33% increase in the frequency of flood events in 2020 compared to the mean for the previous 10 years. The resulting increased volumes of stormwater flowing at rapid speed consequently put a strain on the drainage system which has to convey these volumes within a shorter time frame. Moreover, in cities with combined sewer systems, a continued increase in the urban population implies that more wastewater is produced, subsequently increasing the strain on the drainage systems as their capacity is exceeded (García, et al., 2015). As a result, these systems, which normally flow under gravity, may surcharge and eventually overflow, causing flooding of streets and buildings. A layout of such a system is illustrated in Figure 3 and Figure 4, showing an overflow from the manhole resulting in flooding. These floods usually result in the devastation of properties, disruption of economic activities, and even loss of lives as discussed above (Najibi and Devineni, 2017).

The Global Disaster Assessment Report of 2020 reports that in 2020 flood-related deaths were 7% less than the historical average (AoDREM, 2020). This may be attributed to advancements in technology that allow for better forecasting and information dissemination through early warning systems and improved flood risk management. To enhance flood risk management modelling technologies and decision support systems are used to better understand the flooding phenomena. These add value to raw data by providing additional vital, synthesized information.

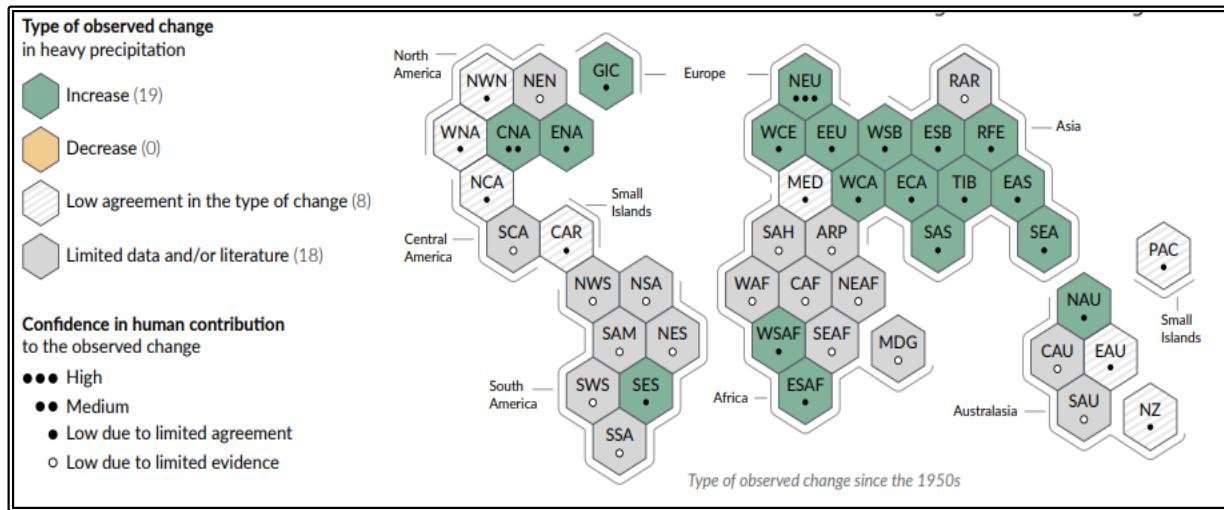


Figure 2: Synthesis assessment of observed change in heavy precipitation and confidence in human contribution to the observed changes in the world's regions. Source: (IPCC, 2021)

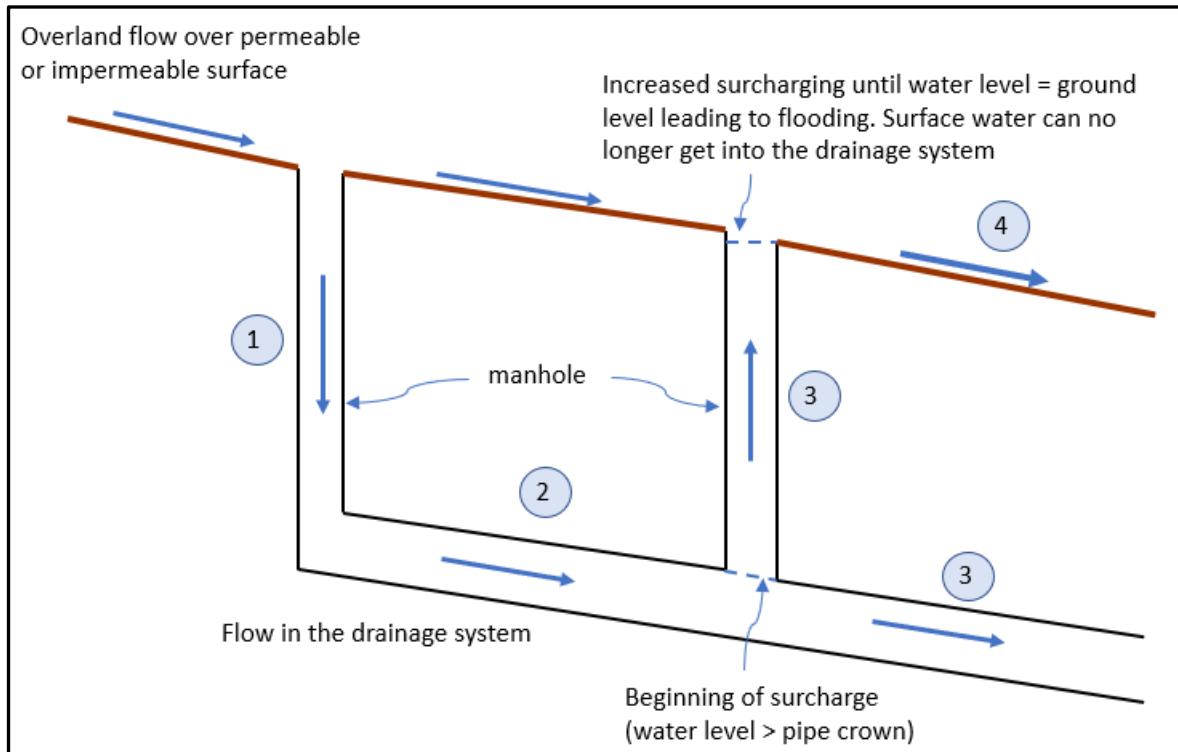


Figure 3: Major and minor flow interactions during an extreme event Adapted from (Schmitt, et al., 2004, Ellis, et al., 2009)



*Figure 4: A flooded manhole*

## **2.2 Decision Support Systems for Flood Management**

As a way of reducing the impacts of flooding, information from computer-based simulations has been combined with human judgement for more informed decision-making. Abebe and Price (2005) define a Decision Support System (DSS) as one “that assists a decision-maker or a relatively small group of decision-makers to make a decision regarding a problem by the use of readily available resources”. They developed a DSS where they used real-time water level data to help authorities decide on what actions to take and information to be disseminated to the public in case of an expected flood.

Ahmad and Simonovic (2006) also developed the Decision Support for Management of Floods (DESMOF), a decision support system that could do flood forecasts, choose the best option for reducing flood damage and use future developmental and population growth projections together with existing data to assess different options for managing floods. The development of this DSS incorporated expert knowledge on reducing damages from floods together with the use of Artificial Neural Networks for flood forecasting and for hydraulic analysis, the HEC-RAS model was used.

Other decision support tools aimed at monitoring and controlling floods include the use of sensors for monitoring floods and water quality in urban drainage networks (Ewing and Demir, 2021). Linked to these sensors are automated and non-automated control structures which enable operators to manage the drainage networks. This has the advantage of allowing for more real-time control of any possible flooding situation in both space and time. This combination of different techniques helps in providing a holistic approach to flood management.

Another example of a flood mitigation DSS was introduced by Mattos, et al. (2022) who developed a web application for warning residents of pending floods. In developing the system, precipitation forecasts were input into the HEC-HMS hydrological model for the rainfall-runoff modelling. The generated runoff forecasts were then input into the HEC-RAS model and flood maps showing the stages of the river and results of simulations of the floodplain were created. These were then used as input into the web application for flood alerts.

Furthermore, in most urban areas, simulation models like the EPA's Storm Water Management Model (EPA SWMM), MIKE-Urban, and SOBEK-Urban, among others, have been used extensively for decision support (Riaño-Briceño, et al., 2016). These aid in the designing, and analysis of drainage(gray) infrastructure and its control systems (Gironás, et al., 2010). While the use of these modelling systems is invaluable in water management and particularly in flood management, access or comprehension of modelling results is a benefit enjoyed only by those skilled in using these systems. Generally, non-technical citizens would therefore require some training to understand the modelling system(s) or results obtained from these. To bridge this gap, computer programs, called chatbots, that can comprehend users' requests, and in turn, provide a response understandable to the user have been developed and these are discussed in the proceeding section. The adoption of these chatbot systems would allow citizens to ask any question about flooding in the city and obtain model simulation results in a form that is easily understandable to them. The following section discusses chatbots in a bit more detail.

## **2.3 Chatbot Technology**

As society becomes more and more technologically advanced, Artificial Intelligence (AI) has proven to play a very significant role in everyday life, from smart assistants like Siri, Alexa, or Google assistant to computer games, internet searches engines, smartphone functionalities, chatbots, self-driving cars and even autonomous weaponry. AI allows us to solve complicated issues by imitating how humans learn, understand and use their intellectual abilities to solve problems (Chen, et al., 2008). These advances in artificial intelligence have strongly influenced the development of chatbot technology through advances in natural language processing and machine learning.

While different definitions of Chatbots have been brought forward, the underlying concept of chatbots is that they are computer programs(agents) that can hold conversations with users in a friendly way as one would with a person (Przegalinska, et al., 2019, Setiawan, et al., 2020). They use what is called natural language processing (NLP) which translates human language into computer language and vice versa, responding in a language understandable to the user (Dahiya, 2017).

### **2.3.1 The History of Chatbots**

The history of chatbots strongly hinges on Alan Turing, who in the early 1950s questioned whether a machine could think. It was only a decade and a half later that the first chatbot was developed. The first chatbots were created to imitate human behavior by recognizing keywords from text input and matching them against some rules to provide a response. ELIZA, was the first to be created in 1966 by Joseph Weizenbaum (Cahn, 2017) at MIT. This was a basic chatbot that used a pattern-matching algorithm to provide answers in a conversation. It however could not grasp the context of the conversation. After ELIZA, PARRY was developed by Kenneth Colby in the 1970s at Stanford to impersonate someone with paranoid schizophrenia. PARRY managed to convince many psychiatrists into believing that it was a real person. In 1995 Richard Wallace started developing A.L.I.C.E (Artificial Linguistic Internet Computer Entity) which was a natural language processing chatbot (Cahn, 2017, Shum, et al., 2018). All of these chatbots performed well, but under constrained situations (Shum, et al., 2018). Further enhancements were therefore done on chatbots such as the incorporation of the ability to fulfill a task, such as providing tourists with flight data (Shum, et al., 2018). Owing to advances in AI, the process of retrieving information improved to a point where responses are obtained from analysis of web search results and the user input is now verbal as is the case with Virtual Personal Assistants (VPA) like Cortana, Siri, Alexa, and Google Assistant (Cahn, 2017). Presently, chatbots are being created to address various issues as alluded to in previous sections.

### **2.3.2 Basic Functions in Chatbots**

Cahn (2017) quoting Sansonnet et al. (2006) proposes three basic functions that should be fulfilled by every chatbot. Firstly, the chatbot must be able to comprehend what the user is requesting and provide appropriate responses(Cahn, 2017). This is regardless of the type of input that the user uses, which could either be textual, visual or oral depending on the developer's preference and the purpose of the chatbot. If the chatbot is designed to accept for example both textual and visual input, it should, therefore, under no means fail to support these types of input and provide responses accordingly. For instance, a chatbot designed to serve blind patients would have to incorporate an option for oral communication (Nurdiana and Wijaya, nd) so as to adequately address the users' needs. Secondly, the chatbot must be integrated with an external database from where it can extract the information needed by the user. These can be static(unchanging) or dynamic(Boné, et al., 2020) getting updated in real-time. Finally, chatbots should “provide the function of presence” or a persona. This is the aspect of chatbots that ensures that they are relatable so that the users trust them (Cahn, 2017 ) and the responses that they provide. The essence of chatbots hinges on the users getting a feeling of communicating with someone who understands their needs. Their popularity emanates from the fact that they are accessible to everyone as there is no need for the user to have knowledge about computers, they just have to ask questions and answers will be provided. After it analyses and understands (parses) the question the chatbot then provides the appropriate response.

### **2.3.3 Chatbot Applications in Water Resources Management**

As outlined in previous sections, the Chatbots technology has been applied in numerous fields a few of which will be discussed in more detail in this section. To start with, Okon (2021) developed a chatbot for flood awareness creation in Nigeria. In the study, NLP was used to convert any text the user input into a format understandable by the computer using the Natural language Tool Kit (NLTK), TensorFlow and TFLearn. Intents were then developed in the Google Dialogflow platform to cover different phrases users would use to ask different questions. The possible responses were also set up on the platform based on flood forecast information for different areas in the country. The Flutter framework was used in the development of the mobile application for the front-end ensuring access to anyone with a smartphone.

Tsai, et al. (2019) also developed a chatbot called Ask Diana which managed great amounts of data to provide just the critical information to decision-makers. This chatbot was meant to lessen the amount of data that decision-makers had to handle. They made data accessible by providing easy-click buttons on the chatbot. The chatbot provided different kinds of information including weather data, reports on past disasters and other information related to the prevention of droughts and floods. They created a database to store both static and dynamic information from government reports and expert advice. In using the chatbot if the user utilized the menu buttons then intent-matching was done using a keyword table. However, if the input was text a Fuzzy search algorithm would be used to match the user's request to an intent. The Line platform was used for integrating the chatbot for use on mobile smart cellular phones.

Once more, Boné, et al. (2020) developed DisBot, a chatbot that speaks Portuguese and was designed to provide flooding information to citizens and first-responders during a disaster. The processes involved in developing the chatbot included using SpaCy to tokenize and extract important features of the messages that the users input. Then through Natural Language Understanding (NLU) the user's intent was classified. The context between different intents was fixed using Dialogue Management and then responses were set to either be predefined or obtained from a Dynamic Knowledge Base (DKB) Boné, et al. (2020).

Finally, a chatbot that allowed real-time monitoring of the water level, pH and dissolved oxygen in a fish pond using sensors and the Internet of Things was developed by Al Rasyid, et al. (2020). The system was developed to allow for aeration of the fish pond using smart aerators when necessary. The user would be able to control the devices via text allowing for remote monitoring of the ponds.

The chapter discussed some literature reviewed on the prevailing flood situation in urban areas in light of increased rainfall intensities and increased development to provide housing and services for a larger urban population. After that, different decision support

systems developed for water management were also reviewed. Finally, the chatbot technology was expounded on starting with the history of chatbots. After this, basic functions that should be fulfilled by chatbots were outlined. Finally, the chapter discusses different studies that have developed chatbots for the management of water resources and particularly mitigating the impacts of floods. The next chapter outlines some of the technological tools used in the research.

# **Chapter 3      Technological Tools Used in the Research**

---

This chapter will set the foundation for the methodology highlighting some technological tools that were used in the research starting with a discussion of NLP algorithms that are used by computers to understand the user's intent. Here, the different pre-processing techniques together with various word embedding methods will be discussed. Then the Dialogflow platform will be explained together with other platforms that are also used for chatbot development. In addition to this webhooks will be described highlighting the Flask platform for developing web applications. Finally, a description of the EPA SWMM model will be provided.

## **3.1 Natural Language Processing Algorithms**

The development of chatbots primarily centers on advancements in NLP which provides the foundation for humans- computer communication. NLP is a branch of AI that involves using several techniques to analyze normal text (Liddy, 2001). Machine learning and deep learning techniques are used in this analysis to improve the understanding of human language by computers so that communication between humans and computers is made possible. Furthermore, NLP incorporates both text and voice analysis. It is categorized into two main branches: Natural Language Understanding (NLU) the process by which computers can understand user input and Natural Language Generation (NLG) the process by which the computer will respond to the user (Ghosh, 2019). NLP is a field of study on its own, which is rapidly changing with advancements in AI. Numerous techniques are used in NLP, and these are described in the following section starting with the different pre-processing techniques.

### **3.1.1 Pre-processing Techniques in NLP**

Our natural language as humans is the way we communicate with each other, usually via text or speech. So as we evolve, so does our language, and with this transition comes the need to constantly change or upgrade the techniques we use to understand language. The advent of computers enabled various techniques to be performed on our natural language to come up with different ways of understanding it. Firstly, however, the computer needs to understand the text input by a user and to adequately process it and provide a response where necessary. Since the whole concept of chatbot design hinges on NLP, this section is dedicated to discussing the various techniques used in natural language processing.

NLP incorporates both text and voice analysis and is categorized into Natural Language Understanding (NLU), the process by which computers can understand user input and Natural

Language Generation(NLG) the process by it which will then respond to the user (Ghosh, 2019). A variety of techniques are used in NLP to understand the users' input and provide as output, a response that is understandable by the users. As illustrated in Figure 5, shows the NLP text pre-processing pipeline.



*Figure 5: Natural Language Processing Pipeline*

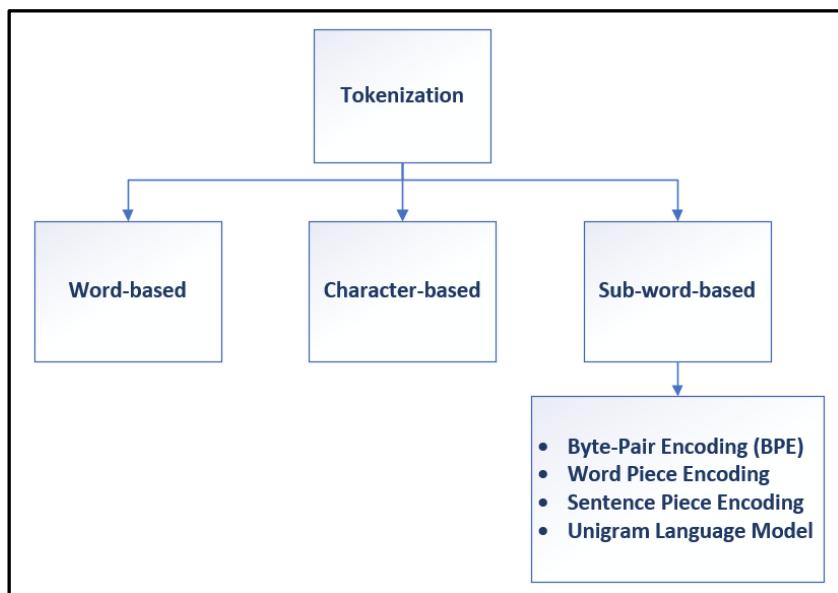
Source: Geitgey ( 2018)

#### i. Tokenization

The NLP pipeline comprises of various techniques, the first of which is tokenization. Tokenization involves the splitting of text into smaller units or tokens that can be understood by the computer. Depending on the tokenization algorithm used, text can be split so that the different words in the sentence are used (word-based), or for each word the different letters(characters) of the words are used (character-based) or the words are split into subwords sub-word based as illustrated in Figure 6. When created, these tokens are then mapped with numbers (encoding) so that they can further feed the next stages of the NLP model.

##### a) Word-based Tokenization

Tokenization can be word-based, where the text is split on white spaces, punctuation or delimiters (e.g. comma, brackets) depending on the task at hand and the nature of the input data.



*Figure 6: Tokenization*

Figure 7 shows an example of how word-based tokenization happens. A pass is made by the computer over the text and firstly, the text is split based on white spaces. After this, it is then split based on punctuation. In this case, care should be taken so that shortened words like “isn’t” should not be split as “isn/t”, and to prevent this, tokenization exception rules are applied. These then ensure that the word is split accurately as “is/not” so that the meaning of the word is

preserved after tokenization. Prefixes, suffixes and infixes (e.g. hyphens) are also used to tokenize text. When all these have been used on the given text whatever forms that remain are considered as a single token. If every single word is extracted individually then the tokens are called unigrams, however, if it is two tokens at a time then they are called bigrams and so forth. The decision on the number of tokens ‘n-grams’ extracted depends on the task to be fulfilled.

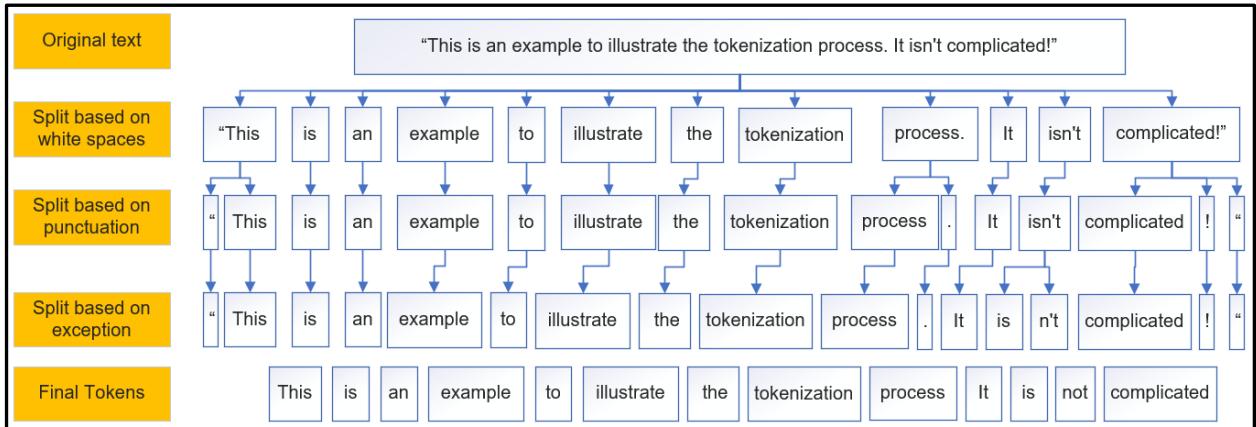


Figure 7: An Illustration of Word-Based Tokenization

This method of tokenizing text has several setbacks including the fact that there is the creation of a large vocabulary to cater for different variants of words e.g. “*let*” and “*lets*” which would then need their own tokens. The second setback comes in handling hyphenated words like “*in-depth*”, “*ice-cold*” where a set of heuristic rules then have to be developed to enable proper tokenization. A third limitation of the method is that names like ‘*South Africa*’ will also be tokenized into ‘*South*’ ‘*Africa*’ thereby losing their meaning (Ghosh, 2019). Finally, word-based tokenization would not work on languages that do not use white spaces as separators like Chinese. It is interesting to note that word embeddings like Word2Vec and GloVe which are used in most NLP software fall under word-based tokenization

### b) Character-based Tokenization

In addition to word-based tokenization, we have character-based tokenization where word characters are considered as tokens as illustrated in Figure 8. Since it is based on characters, encoding of the characters (numbers assigned to characters in computers) can be done using ASCII or Unicode for the embedding of any word. Furthermore, character-based tokenization brings the advantage of increasing the vocabulary that the NLP models can handle since the words are split into letters which can therefore be used to identify any other word with similar letters. It is computationally faster than the other methods. This tokenization technique can accommodate abbreviations, slang and emojis since it considers the character itself. The main setback of character-based tokenization is that the semantics of the word is lost as the tokens (characters) carry no inherent meaning.

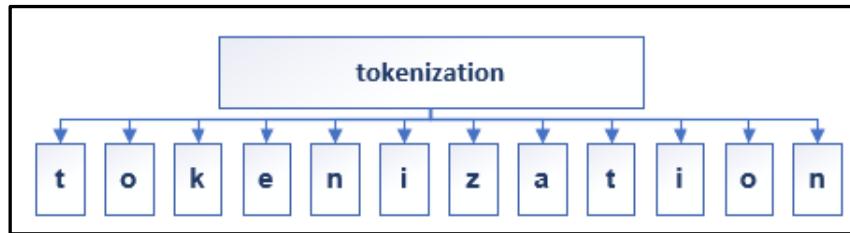


Figure 8: Character-Based Tokenization

### c) Sub-word-based Tokenization

The main aim of sub-word-based tokenization was to capitalize on the pros of both the word-based and character-based tokenization i.e. retaining the meaning of the tokens and reducing the vocabulary size. The splitting of words is done using prefixes and suffixes and the resulting sub-words can be put together to create other words. Figure 9 illustrates sub-word tokenization showing how the different sub-words are used in the creation of other words. In the figure, the word *informatics* is split into ‘*in-form*’ adapted from Latin *in-forma* (*informare*) meaning to form/shape/ describe to someone and ‘*matics*’ from the Greek meaning to act on its own or thought. The subwords can then be used to form other words as prefixes e.g. ‘*inform-ality*’, ‘*inform-ation*’ or as suffixes ‘*auto-matic*’ and ‘*schematic*’.

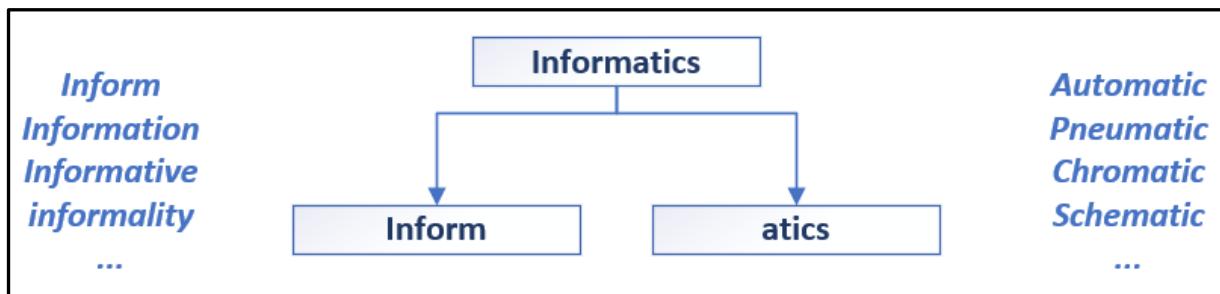


Figure 9: Subword Tokenization

## ii. Parts-of-Speech Tagging

This refers to when words in a sentence are tagged into or assigned their parts in speech, i.e. whether it is a noun, verb or adjective and so forth in relation to other adjacent words. Tagging is therefore dependent on the context in which the word is used. This technique can assist in distinguishing whether the word is being used for example as a verb or a noun. An example is the word ‘*help*’ which can be used as a verb (‘*May I help you carry that?*’) or as a noun( ‘*Their help was greatly appreciated.*’).

Before the advent of computers, Parts of speech tagging was done manually and this was a cumbersome process. While computers have lessened the process, automatic PoS tagging is made difficult by the ambiguity in most English words. Manual tags are therefore sometimes used as a checking mechanism to the automatic process. PoS tagging is done in different ways, such as following a set of context rules to determine the appropriate tag for the given word

(rule-based tagging) or using frequencies to assign the most likely tag(statistical tagging) or a combination of both (transformation-based tagging).

### iii. Stop Words Removal

In most text corpus, there are a lot of frequently occurring words that do not add much meaning to the sentence and these are called Stop words. These are prepositions and pronouns such as ‘a’, ‘the’, ‘am’ ‘of’, ‘at’, ‘to’ and so forth which are eliminated because their absence does not affect the sense or meaning of the sentence. An example of stop words removal is shown in the diagram below.



Figure 10: Stop Words Removal

The process of removing stop words involves performing a look-up for a set of stop words in a list. While there are numerous stop words lists, a comprehensive list covering all stopwords does not exist and thus one can adopt any list and add any words they may deem necessary. It is worth noting that the removal of some of these words may completely alter the meaning of the sentence. An example is the word ‘not’ which negates the sentence and its removal, therefore, transforms the sentence into having a positive meaning.

### iv. Normalizing Text

Text normalization is a process that involves converting text into standard form e.g. ‘*doing*’ and ‘*does*’ are normalized back to ‘*do*’. Also, words with the same meaning like ‘SA’ and ‘South Africa’ are normalized into one form. Lemmatization, stemming and correcting spellings are varied ways of normalizing text (Ghosh, 2019). Lemmatization and stemming are both useful algorithms in search engines and other platform where relationships and similarities in words are of importance.

#### a) Stemming

This process involves trimming words so that only their stem remains and is done through the removal of affixes that is prefixes and suffixes. The resulting stem does not necessarily have to be a whole word. Some stemming algorithms use simply rules like removal of suffices like ‘*ed*’, ‘*ing*’, ‘*ly*’ when the algorithm comes across words with such suffices. An example of suffix stemming is ‘*coming*’ becomes ‘*come*’ while ‘*firing*’ becomes ‘*fire*’. This can, however, be a challenge since some suffices are used to create different words in the English language, for example, the ‘*ist*’ in ‘*hydrologist*’, and ‘*scientist*’ create totally new words which if stemmed lose their intended meaning.

In addition, since stemming does not consider the meaning of the stem produced, there are errors encountered where words with different roots are stemmed to the same stem. This error is called over-stemming. On the other hand, under-stemming is another error where words with the same

root are stemmed to different stems e.g. ‘alumnus’ and ‘alumnae’ being stemmed into different stems while they both have the same root. Regardless of the errors that are sometimes incurred, stemming has the advantage of being fast and straightforward. Lemmatization algorithms have been noted to produce better results since it considers the dictionary meaning of the words. This algorithm is discussed in the following section.

### b) Lemmatization

As stated in the section under stemming, the main difference between stemming and lemmatization is that the latter considers the dictionary form of the words while stemming does not. Therefore, lemmatization gives the dictionary form of a set of words which is called the lemma. Figure 11 illustrates the difference between the two techniques, and from the figure, it can be noted that Lemmatization takes cognisance of the meaning of the word regardless of its tense while stemming only removes the inflection of the word i.e. ‘ing’ or ‘en’ in this case. Another example by Ghosh (2019) shows how the lemma of the word ‘products’ is ‘product’ while ‘production’ will remain ‘production’. If however, we use stemming, the stem of both would be ‘product’.

Since dictionary forms are required, lemmatizers are more difficult to build as they require more knowledge about the language structure. In this case, it is therefore also important to assign the right parts of speech for the given word in order to obtain its correct meaning. As a result, lemmatization takes a longer time compared to stemming. In terms of its application, lemmatization can be used in chatbot development where we require an understanding of the users’ questions to provide the appropriate response.



Figure 11: Lemmatization vs Stemming

## Word Sense Disambiguation and Sentence Boundary Detection

Other techniques used in NLP pre-processing, usually together with other methods discussed in the preceding sections, include Word sense disambiguation and Sentence boundary detection. Word sense disambiguation refers to when words are matched to their intended meaning for example ‘bank’ could mean a river bank or a savings bank. The disambiguation process, therefore, ascribes the true meaning of each word as it is used in the sentence, while Sentence

boundary detection is for determining the end of a sentence and the beginning of another. This process is however complicated by words that are abbreviated for instance ‘Mr.’, ‘Mrs’, ‘Dr.’.

### 3.1.2 Word Embeddings in NLP

After the pre-processing techniques discussed in the preceding sections, the words or phrases are then converted into numerical form so that the computer can understand their meaning. This process is called word embedding or word vectorization. In word vectorization, the words or phrases are mapped into vectors. This helps in finding similarities and meanings of the words. When the words have been converted into vectors, Cosine similarity(the angle between two vectors in the vector space) or Euclidean distance(distance between the two points.) is then used to check the similarity between words. Words with similar meanings are closer together in the vector space i.e. they have similar representations or values than those which are not similar. Word embeddings can either be based on frequency analysis e.g. TF-IDF or prediction-based e.g. word2vec. These are discussed further in this section.

#### i. One-Hot Encoding

This is a method by which a word is assigned the value of ‘1’ in the vector depending on its position in the sentence. For example, for the sentence:

“*This is an example of one hot encoding*”

The one hot-encoded vectors for the words would be:

This = [1,0,0,0,0,0,0]; is = [0,1,0,0,0,0,0], an= [0,0,1,0,0,0,0]; example = [0,0,0,1,0,0,0]; of=[0,0,0,1,0,0,0]; one = [0,0,0,0,1,0,0]; hot = [0,0,0,0,0,1,0]; encoding = [0,0,0,0,0,0,1]

Each word, therefore, has its distinct location in the vector space. This method has the disadvantage of not showing the relationship between the different words.

#### ii. Term Frequency- Inverse Document Frequency (TF-IDF)

This is a method for quantifying the importance of a word (term) in a number of words (document). It is a valuable method in text mining. Term frequency will count the number of times a word is present in a document while for document frequency we consider in how many of the documents the word is present. This is represented by the formula (Scott, 2019):

$$TF - IDF = \left( \frac{\text{count of word in document}}{\text{total no. of words in document}} \right) * \log \left( \frac{\text{total number of documents}}{(\text{occurrence of word in the documents set}+1)} \right)$$

The log is so that we do not get values that are too high and +1 is to ensure that we do not have to divide by 0. The main disadvantage of this method is that it does not maintain the relationship between the words.

### iii. Word2Vec

Developed at Google by Mikolov, et al. (2013), Word2Vec uses an Artificial Neural Network model to learn links or relations between words. Neural networks are used because they can efficiently process large amounts of text. Each word is assigned its own vector in the vector space and the trained model can then be used to distinguish similar words. It is applied in text recognition and is comprised of 2 models, the Continuous Bag of Words and the continuous Skip-gram.

#### a) Continuous Bag of Words (CBOW)

This model predicts the target word using the context or surrounding words (Pilehvar and Camacho-Collados, 2020). For example, using the sentence from section i, the words ‘one’ and ‘encoding’ can be used to predict the word ‘hot’. Firstly, one-hot encoding of the words ‘one’ and ‘encoding’ is done and then these are then fed as input into the neural network with some initial random weights. It is passed to the SoftMax layer where the target word is then predicted. This output is then compared with the actual expected ‘target’ word and the error is used to update the weights. Back-propagation is done each time until the lowest error is obtained and the word with the least error will be the predicted output. Figure 12 illustrates this process. In comparison to Skip-Gram, CBOW is computationally faster.

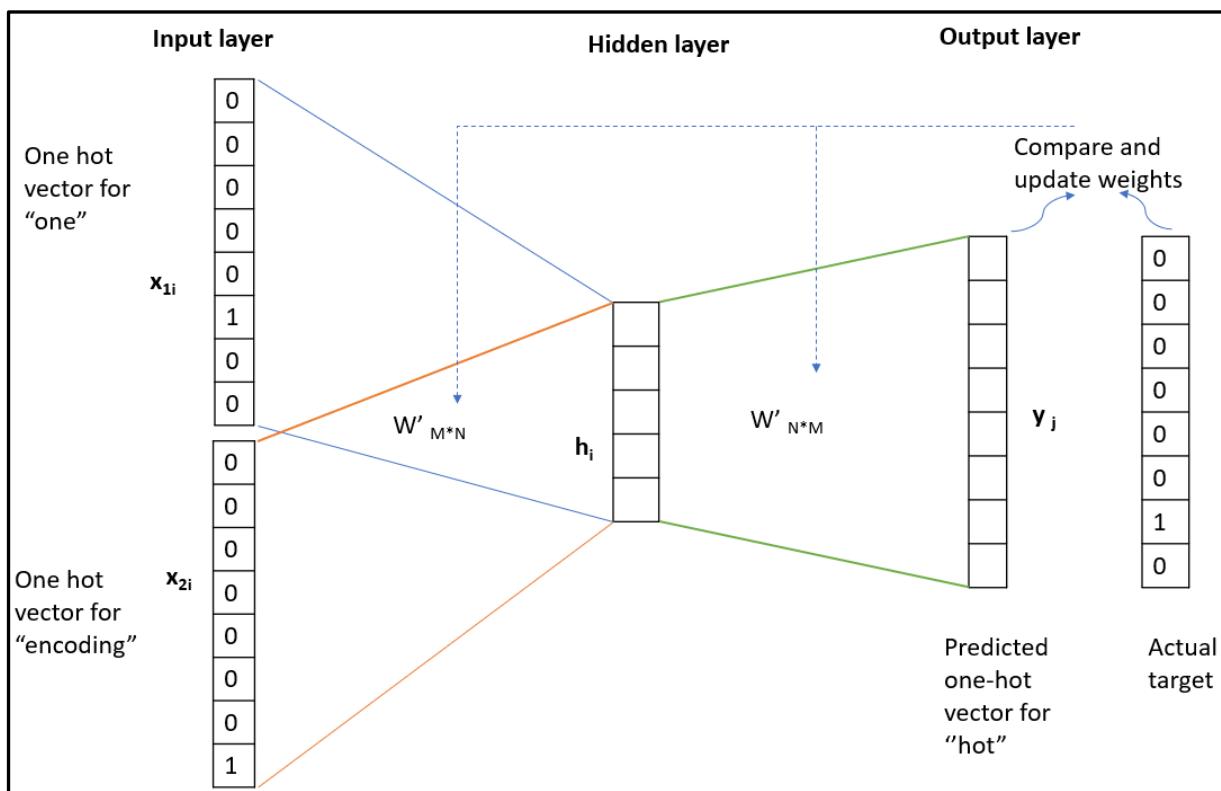


Figure 12: Continuous Bag of Words

#### b) Skip-gram

It is used to predict the surrounding or context words from a given center word. It is sort of the opposite of the CBOW algorithm. The center word is the one that is passed to the hidden layer and then the SoftMax function is used to find the probability of getting any of the words as

outputs. The word that has the highest probability is the result. If, however, errors are obtained between the predicted outputs and the target words, backpropagation is used to update the weights until the word with the highest probability is obtained. Skip-gram is better for predicting less frequent words but it is computationally slower than CBOW. An illustration of the Skip-gram method is shown in Figure 13. From the figure, it can be noted that the context words ‘one’ and ‘encoding’ are obtained from the center word ‘hot’.

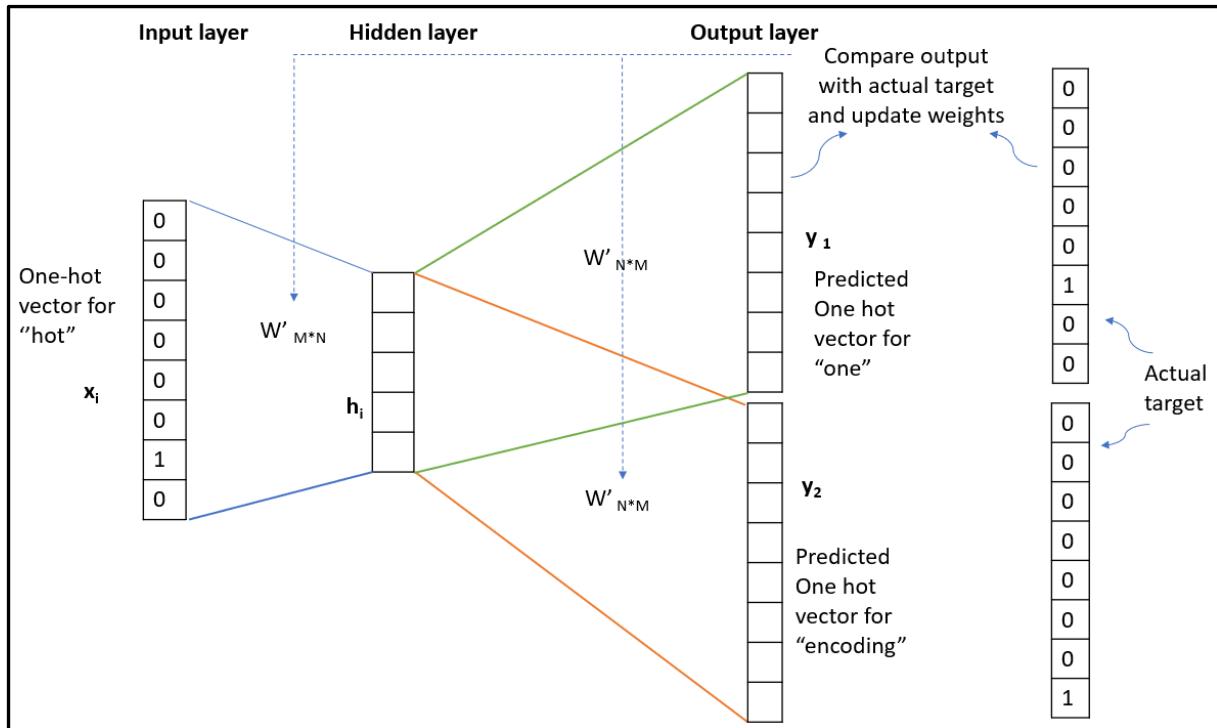


Figure 13: Skip-Gram Method

## 3.2 Dialogflow

As people increasingly spend most of their time on messaging platforms, chatbots are now being developed and integrated on these platforms to reach these people. This has therefore led to an increase in chatbot development platforms such as Google’s Dialogflow, IBM Watson Assistant Framework, Microsoft Bot Framework, Facebook’s Wit.ai, Amazon’s Lex Framework and RASA, among others. All these platforms have their strengths and weaknesses and the selection of which platform to use depends on whether the platform has the features required to fulfill the purpose of the chatbot and also the pricing of the platform versus the expected number of users of the chatbot per month. Another issue to consider is whether the platform allows direct integrations with other platforms e.g. messaging platforms. In addition to this, one should also consider the programming language supported by the platform i.e. whether the developer knows the language. Finally, some platforms are limited in terms of the languages they support.

Some reasons for the use of Dialogflow include, according to Shukla (2019), that it has better ML capabilities compared to others. It allows for integration with other platforms like Facebook Messenger, Twitter and Line among others and it can be developed for use by millions of users since it runs on the Google infrastructure. Also, Dialogflow supports more than twenty languages which adds the advantage of possible use in various locations in the world.

Dialogflow is an AI platform used for developing conversational applications. It does this by understanding natural language using NLP and uses Machine Learning (ML) algorithms to train different phrases so that when the user inputs a question it can be matched to an intent to generate a response. It can analyze both text and oral inputs from users and likewise respond via text or speech (Google, 2021, Oct 25). What happens is that the user's question (query text) is matched to an intent that triggers particular actions at the backend. The output is a JavaScript Object Notation (JSON) file.

There are a few key terms used in chatbot development that one needs to know. To begin with, we have the word *intent* which is the users' aim or motive as they type (or speak), that is, what they wish to extract from the system. Intents are critical components since the interaction between the user and the chatbot is only successful when the chatbot parses (analyzes/deconstructs/explains) the intent. Intent matching, also called intent classification, occurs when Dialogflow matches the user's request to the best intent (Google, 2021, 25 Oct). Another term we have is *contexts* and these allow for a flow in the conversation by layering it ensuring that responses to current questions are linked to previous ones.

Further, we have *entities* which are keywords that may be grouped according to their purpose (Reyes, et al., 2019). The entity type can be customized or predefined system entities such as date, name, and address may be used. Each entity must have a list of possible values and alternative words or expressions that users could input (Ranavare and Kamath, 2020). Again we have *training phrases* are phrases or sentences of what users may typically say. A number of these phrases are defined per intent and these are then trained through machine learning so that if a user's request is similar to these then there's intent matching (Google, 2021, 25 Oct). The greater the number of training phrases, the better the ML model and the better the intent matching and response.

Moreover, we have what is called *fulfillment* which is the code used to communicate with the 'backend', transferring information to and from the business logic. The business logic defines the ways and methods through which documents or data are passed or accessed or updated. By default, when an intent is matched the chatbot responds with a static response. For a dynamic response, a webhook service is called which then calls the service that is defined (Google, 2021, 25 Oct) e.g. an external API or a given database. Dialogflow has inbuilt integration with Google cloud functions but the developer can also provide other HTTPS endpoints for Dialogflow to connect to (Google, 2021, 25 Oct).

Finally, we have *responses* which are the outputs of the intents which can be in text, audio, or graphical format. A response is triggered when there's intent matching. They appear in a variety of formats mainly as text but they can be integrated where possible into lists, cards, and tables among others (Reyes, et al., 2019).

## **3.3 Webhooks**

The same way humans communicate with each other, computer applications also have their own way of sending messages to each other. The way they do this is by ‘pushing’ or ‘pulling’ messages from other computer applications. There are different ways of transferring this information between computers e.g. polling or webhooks. In polling one program has to continuously check the state of the other programs to see if they have anything to communicate. It is as if the program is continuously asking, “Do you have something for me?”. On the other hand, webhooks are ‘hooks’ used by computer applications to automatically send information to other applications whenever something happens. Webhooks must therefore strictly communicate through a web protocol. This process occurs automatically and whenever there is an update on one application, then the other is automatically updated. The message or payload which they carry is sent to a unique Uniform Resource Locator (URL) and the simplest way to send this is through an ‘HTTP GET or POST request’. The GET method is used for viewing something without changing it and the POST is used when something will be changed. An example would be a web search page should GET data while a form that changes e.g. a username or password would POST the information. Also, in terms of security, since each user gets a unique random URL to send information to, webhooks are secure, but a key or signature can be used to further secure them.

### **3.3.1 Flask**

Flask is a web framework for developing web applications using Python (Mufid, et al., 2019). It has a server for developing applications and a debugger built into it and it allows for integration with other APIs and services. It uses Werzeug WSGI (Web Server Gateway Interface) toolkit which is the standard interface used by web servers, that is, a method that allows your developed web application to work with a lot of web servers (Copperwaite and Leifer, 2015). It is a way that is used by web servers to forward requests to web applications in Python.

## **3.4 EPA SWMM**

The EPA SWMM model, developed by the US Environmental Protection Agency, is an open-source software used mostly for rainfall-runoff simulation in urban areas. It can model the hydrological, hydraulic and pollutant loads. The model is a distributed hydrological model where the area under consideration is divided into different subcatchments depending on the drainage patterns. The subcatchments are divided into pervious and impervious portions. The input of the model is precipitation and the generated runoff is routed through pipes, control structures like pumps and also through storage devices (Rossman, 2015). The flow rate, flow depth, quality and quantity of the runoff are some of the parameters that are monitored during a simulation. The user edits the input data for their study area and sets the simulation time steps. The hydraulic component models “backwater, surcharging, reverse flow and surface ponding”(Rossman, 2015). When routing the flows, either a Kinematic wave or a Dynamic

wave method is used. The Kinematic wave, unfortunately, cannot handle “backwater effects, pressurized flow, or flow reversal”(Rossman, 2015) and for such cases, the Dynamic wave is used.

Nodes and links(conduits) are some of the main components of the EPA SWMM model. Nodes or junctions are the points where conduits or pipes are joined. Normally, there is no water stored at the nodes. At each node that is not an outfall node, the user defines the maximum depth or head ( $H_{max}$ ) that can exist at that node and could also assign a surcharge depth allowing the pipes to be pressurized. If the node is a manhole, this maximum depth would be the elevation of the ground surface. During simulation, when a new head ( $H_{new}$ ) is greater than  $H_{max}$  the node becomes flooded (Rossman and Huber, 2017). If however there is no set surcharge depth, a free surface is maintained and water is allowed to pond the node Figure 14 and it will be treated as a normal storage node. The ponded depth will increase when the inflow exceeds the outflow and decrease when there is a deficit. When the water level in the manhole drops then external flows then flow back into the drainage system. The water level in the manhole, therefore, varies with time. For a one-dimensional model, the ponded surface area is fixed by the user and the flooded volume is treated as a large storage tank as shown in Figure 14. Ponding nodes should ideally be located on flatter ground. The resulting report on node flooding provides the hours that the node was flooded, the maximum rate of flooding, the time at which flooding was maximum, the total volume of the flood and the maximum depth or volume of the ponded water (Rossman, 2015)

In terms of application, the EPA SWMM model has been used extensively in the design and management of urban drainage systems. Application in flood management includes: Ma, et al. (2022) applied the model in Zhengzhou, central China to improve accuracy in simulation through real-time calibration of SWMM model parameters. In India, Ahamed and Agarwal (2019) applied the model to assess the runoff generated from extreme rainfall events and the accuracy of the modelled drainage system. Agarwal and Kumar (2019) also used the SWMM model to determine the response of a catchment to peak flows. Finally, Jiang, et al. (2015) used the model to determine which areas are flooded under a forecasted 2, 5 10, and 20-year return flood.

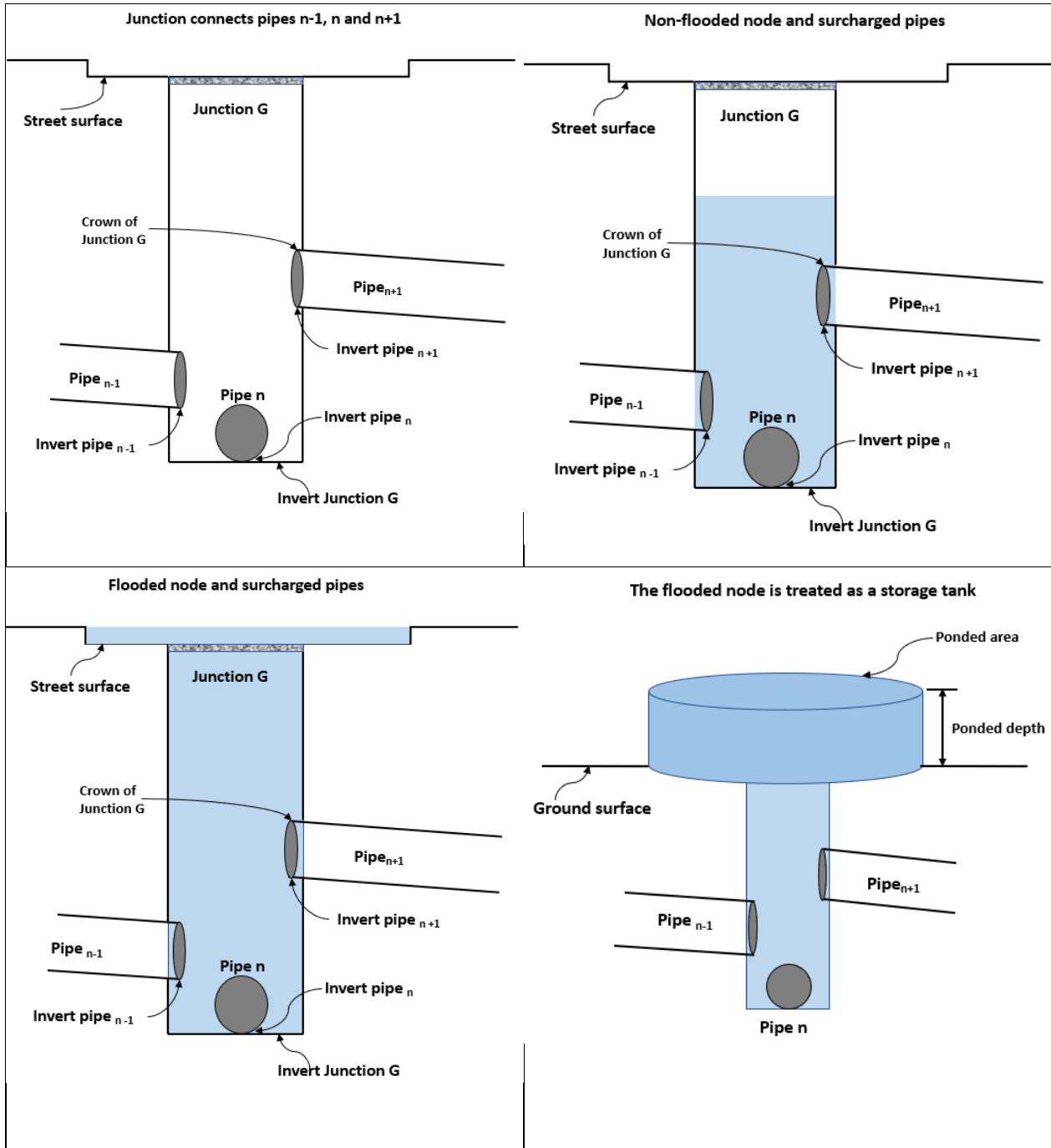


Figure 14: Node(manhole) Flooding

Adapted from (Rossman and Huber, 2017)

This chapter highlighted some technological tools that were used in the research starting with a discussion of NLP algorithms that are used by computers to understand the user's intent. Here, the different pre-processing techniques were discussed. After this, various word embedding methods were also discussed. Then the Dialogflow platform was explained together with other platforms that are also used for chatbot development. In addition to this

webhooks were described highlighting also the Flask platform for developing web applications. Finally, a description of the EPA SWMM model was provided. This chapter provided a foundational description of the methods used in the Methodology. The next chapter describes the study area.

## **Chapter 4      Study Area**

---

Since the research used a model developed for a real study case, this chapter describes the study area. It starts will a general description of Do Lo, showing its location and describing the general climatic conditions in the area. This is followed by a discussion of the flooding trends in this area. Finally, an explanation of the drainage system in the area is provided.

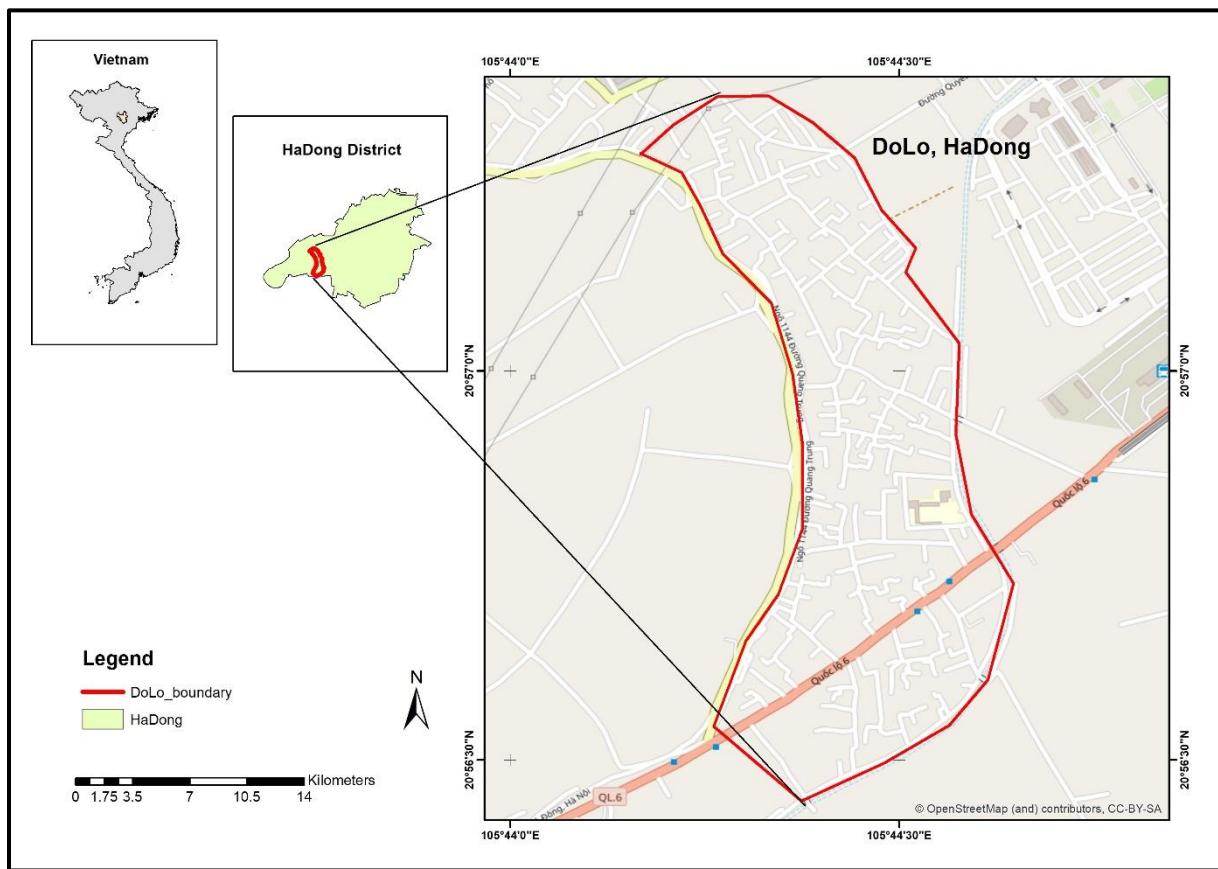
### **4.1 General Description**

The SWMM model used in the study is for Do Lo an area lying to the south of Yen Nghia ward in the Ha Dong District of Hanoi, the capital city of Vietnam. The general climate in Vietnam is tropical in the southern parts of the country and monsoonal in the North (Groen and Jacobs, 2012). Hanoi receives rainfall every month with the driest months (December and January) having on average around 19mm. The average annual rainfall received in the area is generally high at above 800mm with most years being above 1000 mm. In 1994 and 2008, the area received rainfall in excess of 2000mm. Due to the high amounts of rainfall received, flooding is thus prevalent in the area.

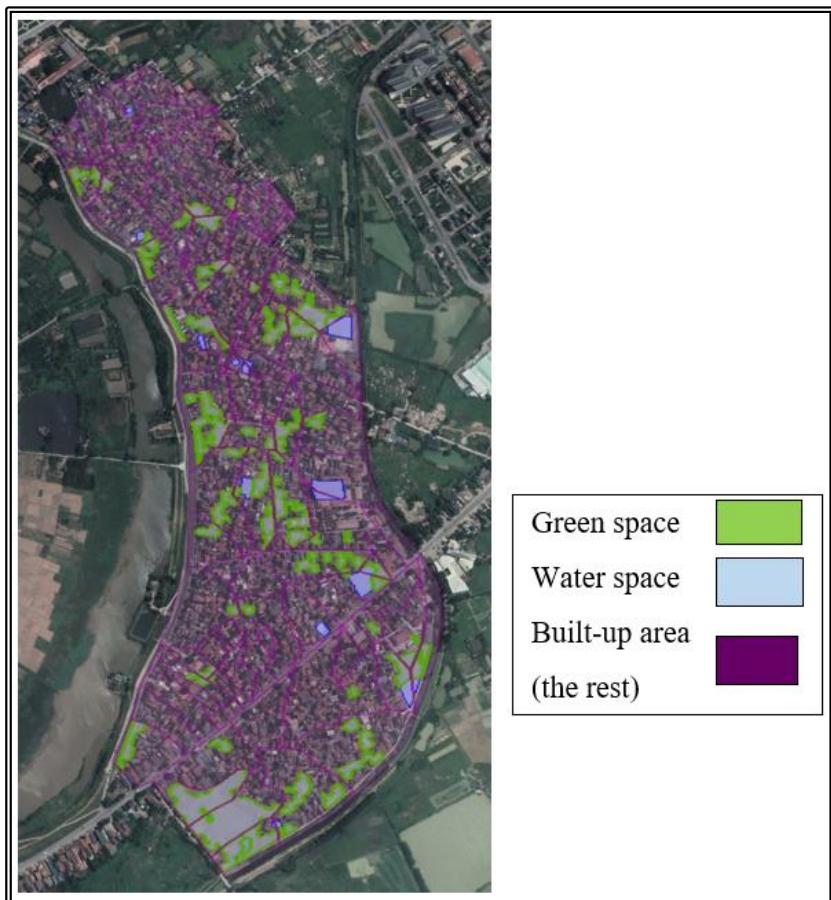
In terms of natural disasters, the country experiences varied natural disasters including sea-level rise, tropical storms, floods, mudslides, droughts and earthquakes as highlighted by Groen and Jacobs (2012). Flooding usually occurs several times in a given year, mostly on the coasts and closer to rivers. To add to this, most of the population resides in these high-risk areas and this compounds the impacts of the floods leads to deaths and annual economic losses. According to Nguyen, et al. (2021), southern and central Vietnam usually experiences river flooding, while the northern mountainous areas are affected by cyclones and typhoons leading to flash floods. Population growth and the resulting development have led to the construction of buildings in flood-prone areas which exacerbates the flooding situation in the area by introducing more impervious surfaces.

Do lo was initially an old village, but as areas around it developed it also grew into being part of Yen Nghia ward when Ha Dong was awarded city status in 2006 (Vietnamese Government, 2006). It is in the Ha Tay province. The location of the study site is shown in Figure 15. Do Lo, Yen Nghia is about 61.5 ha in area and is located next to the Yen Nghia dyke. It is this dyke that protects the area from any flooding from the Day River which lies to the west of the ward. The Co Ban channel, located to the South and the East of Do Lo, is the main output of the drainage system in the study area. The Co Ban channel is connected to the La Khe channel,

which later connects to the Day River through Yen Nghia Pumping Station and also through a gate in the West. The connection to the Nhue River is through the Van Phuc gate. Belonging to the Red River delta, Do Lo, Yen Nghia is part of Hanoi the capital city and has the same geographic, climatic and meteorological characteristics as Hanoi: a flat geographical landscape with an average elevation of 5.00 m amsl. The average temperature is +23°C and the lowest annual temperature is +5°C and the highest annual temperature is +38°C. With over 10373 residents (census survey data 2019 – people's committees of Ha Dong District), Do Lo, Yen Nghia is quite a dense area with a population density of 16862 persons per square kilometer (Do, 2021). Figure 16 displays the land use in the area, showing a fair proportion of green spaces and some water-covered spaces. The larger portion of the area is built up.



*Figure 15: Do Lo, Hanoi, Vietnam*



*Figure 16: Landcover in Do Lo, Yen Nghia, Ha Dong*

*Source:(Do, 2021)*

## 4.2 Flooding Trends and Awareness

The monsoonal rains together with tropical cyclones often lead to annual flooding in Vietnam. According to Nguyen, et al. (2021) 50-80% of the annual water volumes are received during the flooding season (October to December). He also discusses the occurrence of floods in the period between 2012- 2015 in many areas including Hanoi (Nguyen, et al., 2021). In a report for 2021, the IFRC reported the death of 291 people (108 from flooding) with 66 missing and 876 injured from natural disasters. Also according to Luu, et al. (2020), from 1996 to 2015, among all Asian countries, Vietnam was ranked as one of the countries most affected by floods in Asia. Estimates show that 930,000 people in Vietnam were exposed to floods in 2010 and losses due to flooding were around US\$ 2.6 billion. Most of the population settles in low-lying areas which leaves them at risk of flooding. Rapid urbanization as people seek better livelihoods has also exacerbated the situation where urbanization rates increased from 19.6% in 2009 to 36.6% in 2016 (Luu, et al., 2020). This leaves more and more people exposed to floods.

For flood control, the main focus of the Vietnamese government has been the development of structural measures mainly in the form of dykes (Nguyen, et al., 2021). In addition, the government instituted flood protection measures including the National Strategy on Disaster Prevention, Response and Mitigation policy document to address disaster risk management (Groen and Jacobs, 2012). Also, to reduce vulnerability to the impacts of climate change, the

National Target Programme on Climate Change Adaptation was introduced in 2009. Emergency response teams were established and early warning systems are in place. However, at local levels, there is little preparedness and low stakeholder participation in planning for risk reduction(Nguyen, et al., 2021).

### 4.3 The Urban Drainage Network in Ha Dong

The drainage system of the area is shown in Figure 17. It has 3 outlets (shown by the yellow stars on the map) which discharge to Co Ban Channel. The drainage system is almost divided into 2 separate sub-systems, one to the north and one to the south. There are however still some pipes in the middle, therefore, the system is considered as a whole. The Hanoi Sewage and Drainage System Company (HSDC) manages the drainage system in Do Lo.

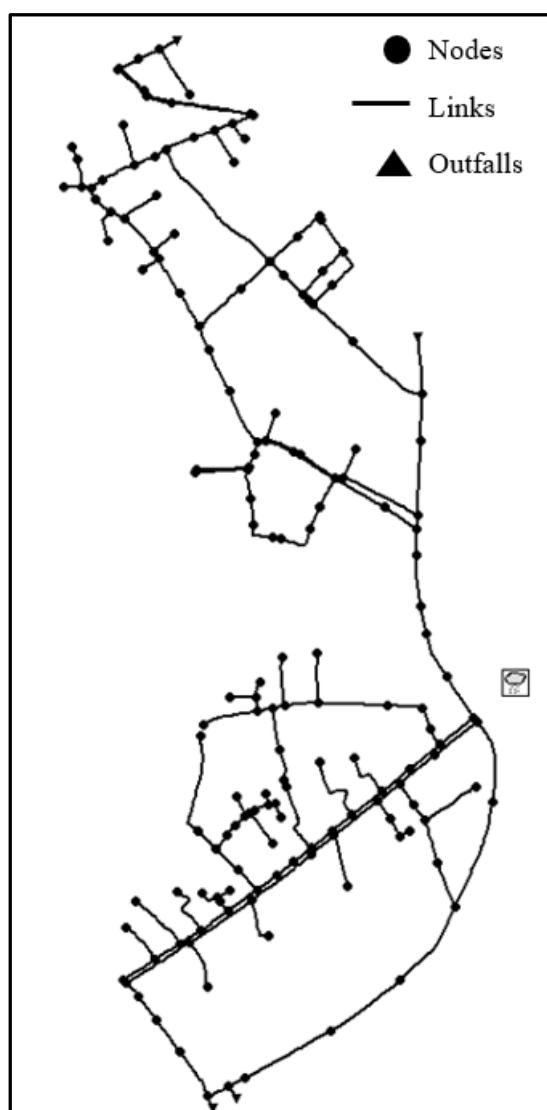


Figure 17: SWMM Drainage system in Do Lo, Yen Nghia, Ha Noi. Source: Do (2021)

In this chapter, the study area was described with a map showing its location and a description of the climatic conditions in the area. In addition to this, the flooding trends were outlined and different measures instituted to protect the area from floods. finally, a display of the drainage area was presented. The next chapter will outline the methodology used in designing the chatbot.

# **Chapter 5      Methodology: Chatbot Development**

---

The methodology is an important aspect of any research as it helps in understanding how the research was carried out. This chapter will therefore present the different methods used and how these were integrated to come up with the chatbot. The chatbot development process in Dialogflow will be discussed. After this, an explanation will be given of how data was extracted from the EPA SWMM model. Then the geocoding exercise will be discussed together with an outline of how the database was created for storing users' locations. In addition to these, the website design process will be described and finally, the chatbot integration process will be presented.

## **5.1 Setting the Scenario**

For one to better understand the methodology used in developing the chatbot, a scenario outlining a scene where one would use the chatbot will be described. From this, the chatbot development process will be detailed showing the processes involved in the development. Imagine it is that time of the year when the heavens are open and it has been raining for some time. You have a job interview at that company you have always dreamt of working at. Obviously, you can't miss this great opportunity. Unfortunately, the company is located on the other side of town. So, you want to know which parts of the city are flooded? How deep are the floodwaters? For how many hours will this flood last? So you can plan your trip, go to the SWMMBot application and ask these questions checking out which areas are flooded and how you can navigate to your destination.

The chatbot was therefore developed to be able to answer these questions based on results obtained from the SWMM model. The steps involved in the design are outlined in the following sections.

## **5.2 Chatbot Development Using Dialogflow**

The chatbot development process involved the steps outlined in the flow chart shown in Figure 18 from the user input through the NLP processes and the creation of different intents in Dialogflow and then integrating this with the backend using webhooks. This allowed for the extraction of SWMM model results to provide responses to the questions asked in the scenario described in the previous section.

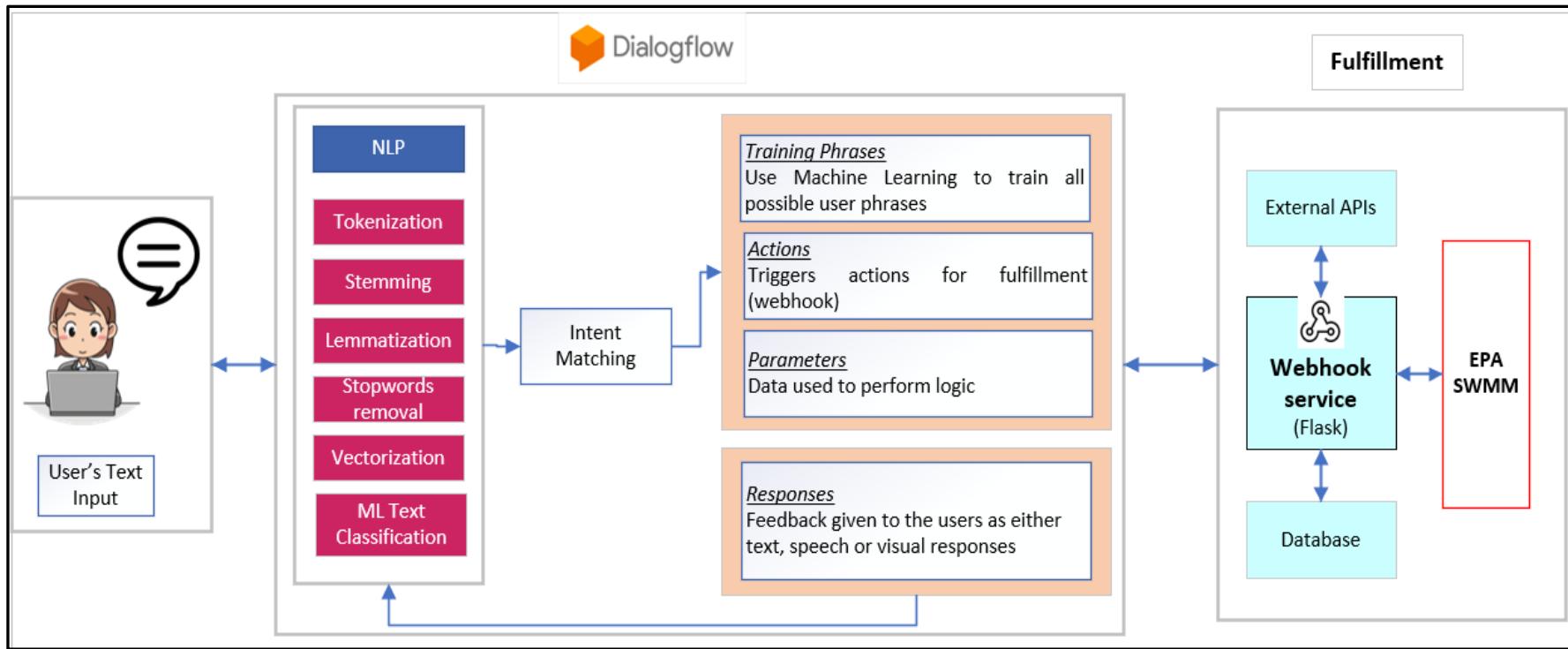


Figure 18: Chatbot Flow Processes

What happens is that the user inputs their question on the application in their device. Then the application transfers the message to the Dialogflow platform. This message is then processed using NLP and classified and matched to the intent with the highest probability match. The intents are manually defined as described in section 3.2. If matching fails, a static default response set out on the Dialogflow platform is provided to the user. For each intent, actions are defined in Dialogflow and when there is successful intent matching, these actions are then triggered in the webhook. Dialogflow sends an HTTPS POST webhook request to the webhook. This is in the form of a JSON structure providing more information about the intent that was matched. Figure 19 shows a sample of such a structure for the flooding duration intent. The user just typed duration and this was matched to the flood duration intent. The actions that will trigger certain actions in the backend are also provided in this JSON structure. The fulfillment text shown in Figure 19 is for the static response in the event that the connection to the server was unsuccessful. When all is working well, upon receiving this request, the webhook then retrieves the data required to answer the user's question using other APIs and databases and the SWMM model. It then sends a fulfillment response to Dialogflow. An example of a fulfillment response from the webhook is presented in Figure 20. Dialogflow then sends the response to the user and it is displayed on the user's device as text or voice.

Diagnostic info			
Raw API response	Fulfillment request	Fulfillment response	Fulfillment status
<pre> 1 { 2   "responseId": "47f52dcf-0e84-4655-abb1-12a97faa740e-7b9a0dae", 3   "queryResult": { 4     "queryText": "duration", 5     "action": "flood.duration", 6     "parameters": { 7       "Flooding_duration": "duration", 8       "Location_name": "" 9     }, 10    "allRequiredParamsPresent": true, 11    "fulfillmentText": "Sorry, I have failed to connect to the server. Please try again in a few minutes.", 12    "fulfillmentMessages": [ 13      { 14        "text": { 15          "text": [ </pre>			

Figure 19: Webhook request

Diagnostic info			
Raw API response	Fulfillment request	Fulfillment response	Fulfillment status
<pre> 1 { 2   "displayText": "25", 3   "fulfillmentText": "Your requested location will be flooded for 7.43 hours. Is there anything 4   else you would like to know about the flooding situation? ", 5   "source": "webhookdata" </pre>			

Figure 20: Webhook response

The following sections discuss the chatbot development process in detail starting from NLP.

### 5.2.1 Comprehending NLP Algorithms

As highlighted in preceding chapters, the development of chatbot hinges greatly on NLP algorithms which are the first processes that determine whether the user's request is understood by the chatbot and that accurate responses are provided. In light of this, the Natural Language Toolkit (NLTK) library was used to explore some of the techniques that are used in Natural Language Processing. It is a Python library that is widely used for NLP. Other libraries include spaCy, Gensim, CoreNLP and TextBlob among others. NLTK can be used to perform tasks like tokenization, stemming, parts of speech tagging and semantic reasoning, and others.

#### i. *Tokenization*

Firstly, for tokenization, a sample sentence that a user might ask the chatbot was input for processing (Figure 21). The necessary libraries were imported and the 'tokenize' module was used which has the `sent_tokenize()` function for the splitting of sentences e.g. from a paragraph or a text corpus and also the `word_tokenize()` function for the splitting of words. When tokenizing, the punctuation marks are also considered as tokens and therefore removed in the following processes.

As seen from Figure 21, the first output shows tokenization of sentences, while the second is that of words i.e. based on white spaces. The decision on which type of tokenization to use depends on the task at hand. In our case word tokenization was what was needed as it feeds into other preprocessing techniques like removal of punctuation, stemming or lemmatization. Word tokenization is also important for the vectorization process where text is converted to a numerical form that the computer can understand.

```

1 import nltk
2 from nltk.stem import WordNetLemmatizer, PorterStemmer
3 from nltk.tokenize import sent_tokenize, word_tokenize
4 from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
5
6 sentence = "Is my location flooded? For how long will it be flooded?"
7 # tokenization
8 senttok = sent_tokenize(sentence)
9 print(senttok)
10 tokens = word_tokenize(sentence)
11 print(tokens)

```

#### **Output :**

```

['Is my location flooded?', 'For how long will it be flooded?']
['Is', 'my', 'location', 'flooded', '?', 'For', 'how', 'long', 'will', 'it', 'be', 'flooded', '?']

```

Figure 21: Tokenization

#### *ii. Stemming*

The stemming process was done using the Porter stemmer as illustrated in Figure 22. As discussed in section 3.1.1, while the root of most of the words is maintained, some words are sliced (stemmed) in such a way that the output is not a meaningful word. An example is the word ‘*location*’ which is stemmed to ‘*locat*’. Stemming helps in finding the root of words. When words with the same stem are used by the user, through stemming the ML algorithm can match the words to the same intent making stemming an important process for intent matching. From Figure 22, the second output shows the root of the three words as ‘*flood*’ showing how stemming doesn’t consider the tense or meaning of the word, contrary to what happens with lemmatization which will be discussed in the next section.

```

13 #Stemming the sentence
14 sentence = "Is my location flooded? For how long will it be flooded?"
15 stemmer = PorterStemmer()
16 for t in tokens:
17     print(t, ":", stemmer.stem(t))
18
19 example_words = ["flooded", "flooding", "floods"]
20 for w in example_words:
21     print(w, ":", stemmer.stem(w))

```

#### **Output :**

```

Is : is
my : my
location : locat
flooded : flood
? : ?
For : for
how : how
long : long
will : will
it : it
be : be
flooded : flood
? : ?

flooded : flood
flooding : flood
floods : flood

```

Figure 22: Stemming

### iii. Lemmatization

Lemmatization is of paramount importance in chatbot development where matching of similar text is fundamental to the subsequent matching of intents and responses. The lemmatization process was done using the WordNet Lemmatizer in NLTK. Since the lemmatization process considers the dictionary meaning of the text, it can be noted from Figure 23 that the output from lemmatization maintains the tense of the word. The lemmatized sentence did not change after the lemmatization process and the meaning of the sentence was maintained. Comparing this output with the one from the stemming process (Figure 22), it can be observed that lemmatization maintains the meaning of the word which can be vital in chatbot development where the goal is to understand the meaning of the user's query.

```

23 print("Lemmatization")
24 # Lemmatization
25 lemmatizer = WordNetLemmatizer()
26 #lemmatize the list of the tokenized words from the sentence and then join to form a sentence
27 lemma_out = ' '.join([lemmatizer.lemmatize(w) for w in tokens])
28 print(lemma_out)
29 # lemmatizing words
30 print("flooding: ", lemmatizer.lemmatize("flooding"))
31 print("flooded: ", lemmatizer.lemmatize("flooded"))
32 print("floods: ", lemmatizer.lemmatize("floods"))

```

### Output :

```

Is my location flooded ? For how long will it be flooded ?
flooding: flooding
flooded: flooded
floods: flood

```

Figure 23: Lemmatization

#### iv. Vectorisation

As discussed in section 3.1.2ii, numerous methods are used in vectorization. The Scikit-learn module was used for TD-IDF vectorization (Pedregosa, et al., 2011). It transforms text data into numerical form. From Figure 24, the output shows a 5\*20 vector matrix for just the first sentence (document). This result is fairly sparse as stop words are assigned a lower weight in the TD-IDF method because they appear more frequently. The words are scored as normalized values, that is between 0 and 1. The complete output of the vectorization process is in Appendix B. - The output of the vectorization process is what then feeds into a machine learning model.

```
37 #Vectorization
38 corpus = [
39     'what is the flooding depth?.',
40     'what is the flooding duration?',
41     'For how long will it be flooded?',
42     'I want to know the flooding rate?',
43     'what am i to do now?']
44 vectorizer = TfidfVectorizer()
45 X = vectorizer.fit_transform(corpus)
46 vectorizer.get_feature_names_out()
47 print(X.shape)
48 print(X.toarray())
Output (for 1st sentence) :
(5, 20)
[[0.          0.          0.57769148  0.          0.          0.
  0.38688672  0.          0.          0.46607785  0.          0.
  0.          0.          0.          0.38688672  0.          0.
  0.38688672  0.          ]]
```

Figure 24: Vectorization

Using the sample sentences in Figure 24, a word2vec model was developed using the Gensim library based on an article by Eric (2022). The minimum count was set to 1 so that the model can include all the words in the sentences. For a corpus of over e.g. millions or billions of words, a higher value is used to eliminate those words that appear fewer times as these may be typos. The default value is 5, so all words that appear less than 5 times in such a large corpus would be eliminated. The vector size is the space that word2vec maps the words onto. The default vector size is 100, but a bigger size would lead to more accurate models. From the results shown in Figure 25, it can be observed that a much denser vector matrix was obtained compared to the one obtained with the TF-IDF. This implies that better text analysis such as similarities between text can be performed. The text used in Figure 25 is too small for any meaningful training to take place. In most NLP exercises, a model pre-trained on a larger corpus with more than millions of words is used to improve the accuracy of the model.

```

71 # WORD2VEC
72 sentences = [['what', 'is', 'the', 'flooding', 'depth'],
73             ['what', 'is', 'the', 'flooding', 'duration'],
74             ['for', 'how', 'long', 'will', 'it', 'be', 'flooded'],
75             ['I', 'want', 'to', 'know', 'the', 'flooding', 'rate'],
76             ['what', 'am', 'i', 'to', 'do', 'now']]
77
78 # train word2vec model
79 w2vmodel = Word2Vec(sentences, min_count=1, vector_size=5)
#it will include all words that occur ≥ one time and generate a vector with a fixed length of five
80 print(w2vmodel)
81
82
83 vocab_len = len(w2vmodel.wv)
84 print(vocab_len)
85
86 # access vector for one word
87 normed_vector = w2vmodel.wv.get_vector("flooding", norm=True)
88 #print("vector 4 flooding : ", normed_vector)
89
90 # access vectors for all words
91 all_normed_vectors = w2vmodel.wv.get_normed_vectors()
92 print(all_normed_vectors)

```

#### Output:

```

Word2Vec(vocab=22, vector_size=5, alpha=0.025)
22
[[-0.038489  0.01697034  0.36630523  0.6466623 -0.66774166]
 [-0.48855633  0.44339007  0.6159798 -0.34430027 -0.25834888]
 [ 0.6150826 -0.12779781 -0.37807617  0.54620695 -0.4050401 ]
 [-0.13908541  0.2203119   0.07596576 -0.6345492 -0.72366744]
 [ 0.56778425  0.39369524  0.5247549   0.05921042  0.49371186]
 [-0.31360865 -0.0870167   0.53157    -0.6930861 -0.36213627]
 [-0.52173674 -0.06459854  0.6624952  -0.508372  -0.16209812]

```

Figure 25: Word2Vec

Google Dialogflow has an inbuilt ML algorithm that automatically trains your model as you enter the different possible phrases that users can use when asking questions. For its NLP, Google uses **BERT(Bidirectional Encoder Representations from Transformers)**, which can encode the word and its position in the text. It can identify unknown words and to distinguish meanings of words that have many meanings such as '*bank*'. BERT has the advantage of being trained on the Wikipedia corpus and the corpus of a collection of unpublished books which improves its accuracy. It enables the analysis of entities, syntax, and sentiment and also the classification of text. The chatbot development in Dialogflow is described in the next section.

### 5.2.2 Dialogflow: SWMMBot

The first step in developing the chatbot involved defining the intents of the users in line with the flooding questions outlined in the scenario in section 5.1 and based on flooding data for Do Lo obtained from the EPA SWMM model. The chatbot was designed with the intention of it being integrated for audio communication where the chatbot listens and waits for the user to pass a greeting in order to start a conversation.

### i. Setting the Intents

Dialogflow intents are in the form of a JSON object as shown for the Maximum ponded depth intent in Figure 26. For every intent, numerous phrases that resemble how different users might ask the same question are input as **training phrases** for the Dialogflow machine learning algorithm. The algorithm will then be trained on these phrases so that it recognizes and matches any other questions similar to the training phrases. In addition to the training phrases, **parameters** and **entities** were also developed. The entities which determine how different parameter values are extracted for processing in the backend. **Actions** were also defined to determine which actions are triggered at the backend. The dynamic **responses** were mostly obtained from the SWMM model through the webhook or set on the Dialogflow platform for static responses. This section will expound on how different intents were developed.

```

1300  {
1301    "isTemplate": false,
1302    "data": [
1303      {
1304        "text": "I would like to know the ", ← training phrase
1305        "userDefined": false
1306      },
1307      {
1308        "text": "maximum depth",
1309        "userDefined": false,
1310        "alias": "max_ponded_depth", ← parameter
1311        "meta": "@Max_ponded_depth"
1312      },
1313      {
1314        "text": " for ",
1315        "userDefined": false
1316      },
1317      {
1318        "text": "N45-1",
1319        "userDefined": true,
1320        "alias": "Location_name", ← parameter
1321        "meta": "@Location_name"
1322      }
1323    ],
1324    "count": 1,
1325    "id": "336a25f6-63b6-4882-9338-560207aa294a",
1326    "updated": null
1327  },
1328  ],
1329  "webhookForSlotFilling": false, ← webhook call
1330  "webhookUsed": true ← enabled
1331 }
```

Figure 26: Sample JSON object for 1 training phrase of the Maximum flooding depth intent

When the user greets the chatbot, the Default welcome intent is triggered. This intent has, as training phrases, different greetings that users' could use such as “*Hi*”, “*Hey*”, “*Hallo*” and so forth. The user's greeting then triggers a webhook response where the SWMMBot chat agent introduces itself, states what it can do, and also asks the user to input their location. This location is then matched with the **Flooded point intent** which has training phrases like the ones shown in Figure 27. A total of ninety training phrases were developed for this intent to cover the different ways users would input their addresses and pin locations.

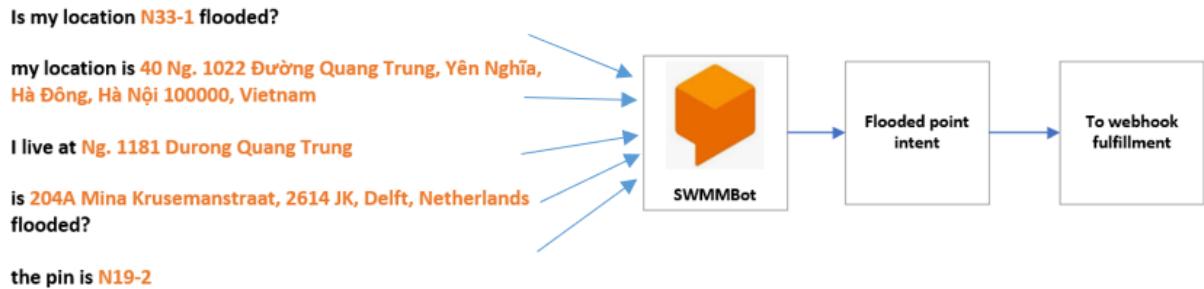


Figure 27: Flooded point intent

In addition to the training phrases, actions were set on the Dialogflow platform, that would trigger a specific action for this intent at the backend (Figure 28). This action name is what Dialogflow would provide to the fulfillment webhook request. Therefore, the action for the Flooded point intent was set as ‘*node.location*’ and the parameter ‘*Location\_name*’ was used. This parameter value is what Dialogflow extracts from the user input. In this case the parameter value would be the address that the user inputs. It is worth mentioning that the ‘*Location\_name*’ parameter was used not just by this intent but by every other intent that extracted values from the SWMM model. **This was done to maintain a context** between intents so that if the user inputs their location then that location has to be maintained and used for computations in all the other intents. For example, if the user enters their address then that same address has to be used when they ask the follow-up question “*what is the flooding depth?*” or “*at what time will the flooding be maximum?*” and so on. For other intents, the context was set in Dialogflow to ensure a flow in conversation is maintained. In addition to the actions and parameters, entities were also defined. These ensure that similar data is matched together e.g. “*location name*”, “*node name*”, and “*place name*” refer to the same thing and so these were defined to fall under one entity type: ‘*Location name*’. These entities determine how data from the user’s input is extracted and matched to the appropriate intent.

Action and parameters					
node.location					
REQUIRED	PARAMETER NAME	ENTITY	VALUE	IS LIST	PROMPTS
<input checked="" type="checkbox"/>	Location_name	@Location_name	\$Location_name	<input type="checkbox"/>	Please enter a n...

Figure 28: Setting the actions and parameters

Additionally, the chatbot was set in such a way that it could take either the user’s address or the name of the node that was closest to the user. The node names are the actual node names on the EPA SWMM model which were added to the Google maps platform. The map was then incorporated into the website to allow the user to easily navigate to their location of interest on the map and then get the pin name closest to them. Then, when the user inputs their address the processes outlined in the flow chart in Figure 29 would take place.

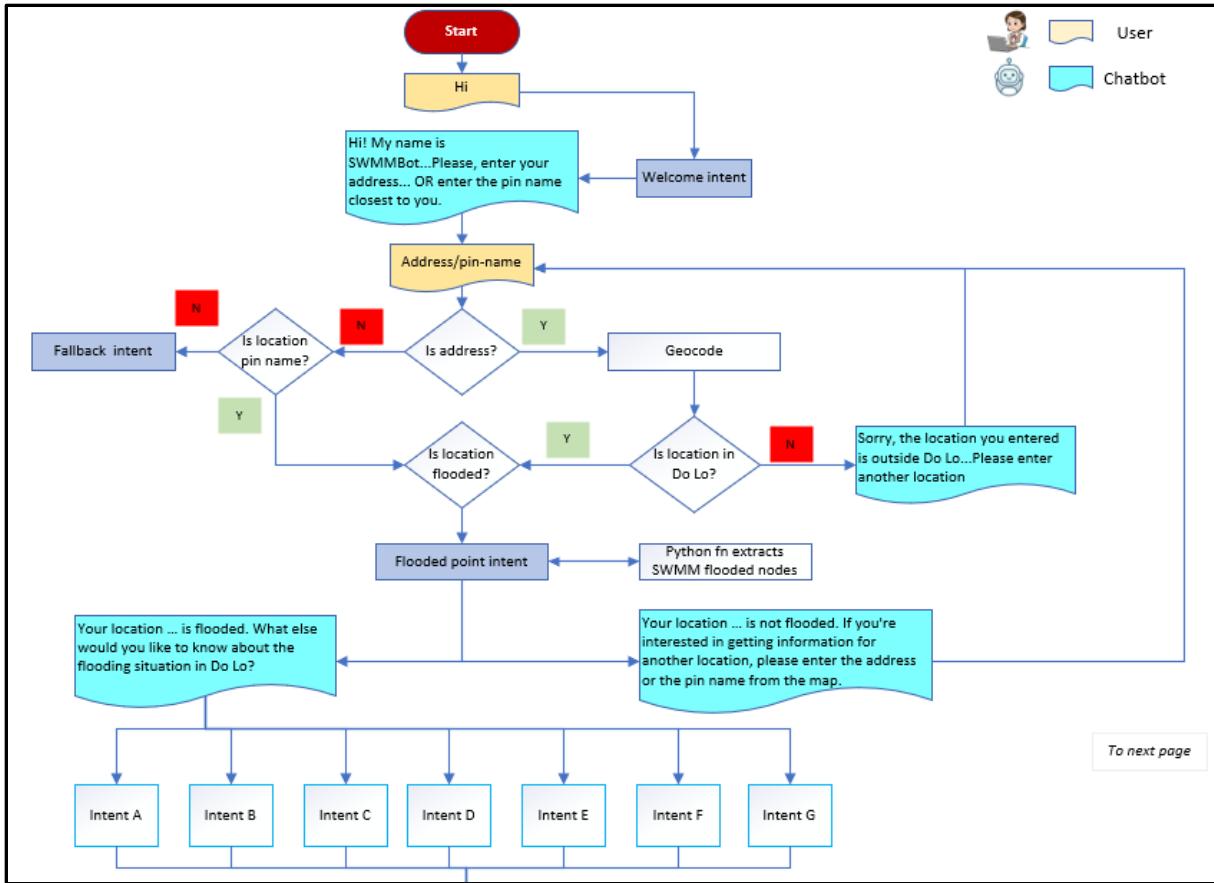


Figure 29: Welcome and Flooded point intents

Firstly, the chatbot checks whether the user inputs an address or a node name and if it is an address then the address was geocoded to obtain its coordinates (Latitude and Longitude). Then a check is done to see if the location was in Do Lo so as to provide only results for the area under study. For this, the shapely library was used to create a polygon using coordinates of different points around Do Lo. If the location is outside Do Lo, then a response was sent to the user that their location is outside Do Lo. The code developed for this process is in Figure 30.

```
# if user input is an address i.e. has more than 8 characters
if node_name is not None and len(node_name) > 8:
    # if the user input is outside Do Lo
    try:
        lat1, long1 = geocode(node_name)
        point = Point(lat1, long1)
        polygon = Polygon([(20.940415, 105.739589), (20.943201, 105.744567), (20.946006, 105.744719), (20.948801, 105.744858)])
        result = polygon.contains(point)

        if result == False:
            fulfillmentText = "Sorry, the location you entered is outside DoLo, for now I can only provide you with information for locations within DoLo. Please enter a location within DoLo or a pin name closest to your location."
```

Figure 30: Checking if an address is in Do Lo.

If the location is in Do Lo, then a function is used to find the node that is nearest to the user's address. Then using this node name a check is done to see if it is flooded or not (Figure 31). For

this, Python functions were created to extract flooded locations from the SWMM model results. The same check is done if the user's location is a pin name. A response is then sent to the user on whether their location is flooded, or not. If the location is flooded, then the user is prompted to ask other questions they may have on the flooding situation in Do Lo. Otherwise, they can enter a different location name and the process is repeated.

```
# IF USER INPUTS AN ADDRESS get the nearest node name
if len(node_name) > 8: # if user input is an address i.e. has more than 8 characters
    node_name = Find_nearest_node(node_name) # here node_name is the address the user inputs and it

node_name = node_name.capitalize()
# Flooding state 1 location
if action == 'node.location': # checking if the action key matches node.location
    if FloodedLoc(node_name) == True:
        fulfillmentText = "Your location " + params.get('Location_name') + " is flooded. What else would you like to know?"
    else :
        fulfillmentText = "Your location " + params.get('Location_name') + " is not flooded. If you would like to know anything about flooding in Do Lo, just ask me a question." 
```

*Figure 31: Checking whether a location is flooded*

When the user's input is not understood by the chatbot at any point the **Fallback intent** is then triggered and static responses like the ones shown below are sent back to the user. The other responses are in Appendix D. -

Sorry, I didn't get that. Can you say it again?

This is embarrassing, I missed what you said. May you kindly rephrase?

Goodness me, I missed your question, could you say that again?

From the Flooded point intent, the user can ask any question on either the flooding duration, the maximum flooding depth, the time at which flooding will be maximum, the maximum flooding rate, the depth of flooding for a flood with a 10 year return period and also the duration of flooding for such a flood and finally what they can do in case of flooding. The intents labelled intent A to intent G in Figure 29 were developed to address all these questions and these will be discussed next.

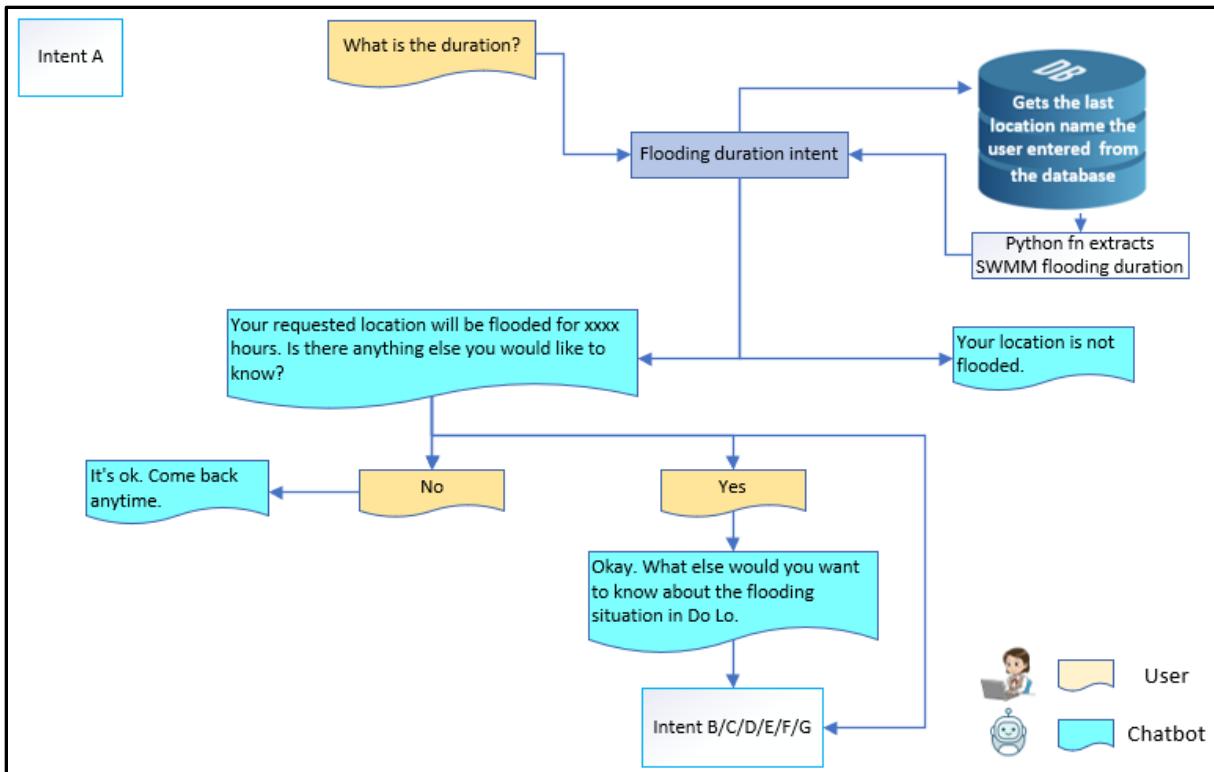


Figure 32: Flooding duration intent

If the user wants to know for how long their area of interest will be flooded, the **Flooding duration intent** will be triggered as illustrated in Figure 32. Since it has to remember the user's location, this intent takes the last location the user entered from a database and then checks the flooding duration through functions that extract this information from the SWMM model. The response will then be returned to the user through Dialogflow. If the location the user entered is not flooded a response will be sent to the user stating that. After responding every intent was set to ask the user if there is anything else they would like to know about the flooding situation in Do Lo. To cater for this, **follow-up intents** for the 'Yes' and 'No' responses were defined. If the user wants to know more the chatbot will ask them what it is they would like to know about the flooding situation. Then the conversation will continue triggering any of the other intents which are discussed below. If, however, the user is not interested in anything else then the conversation ends with the 'Thank you' or 'Bye' intents.

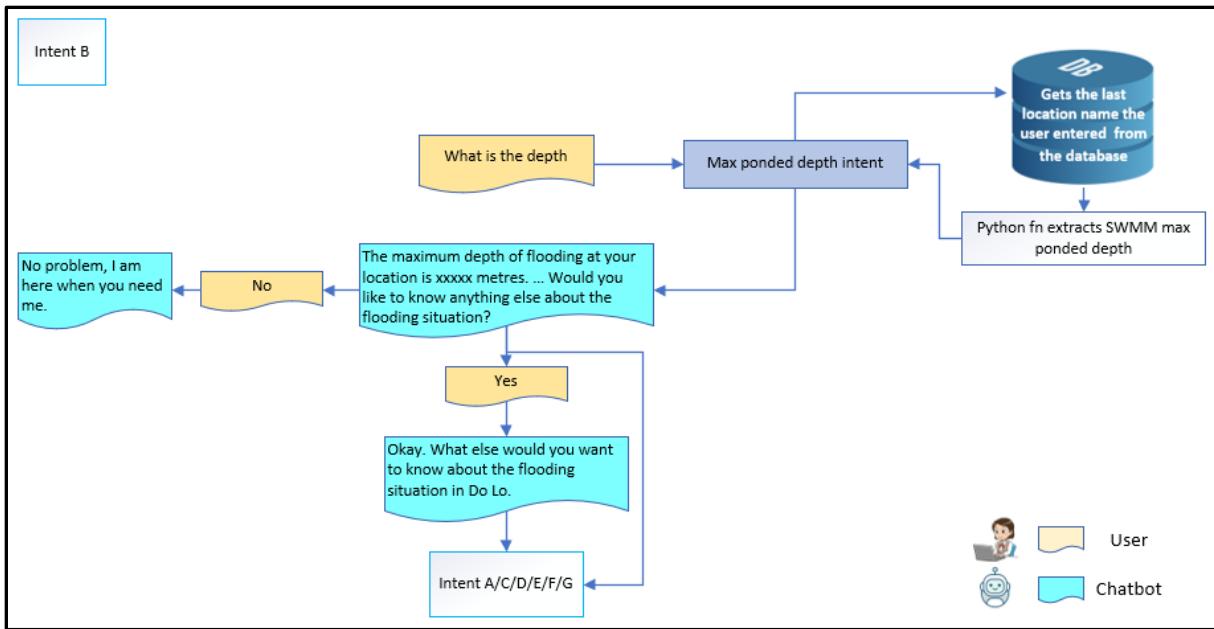


Figure 33: Maximum ponded depth intent

If the user wants to know the maximum depth to which the flood will rise, this would trigger the **Max ponded depth intent** presented in Figure 33. Just like with the flooding duration intent, the chatbot has to remember the location the user is referring to so the intent retrieves this value from the database and then the max ponded depth for that location will be extracted from EPA SWMM model results. The obtained value is then posted back to Dialogflow as a ‘Fulfilment text’ in the JSON file and Dialogflow then sends the response to the user. Again, after this, to continue the conversation, the chatbot then asks the user if they would like to know anything else about the flooding situation in Do Lo. The user could respond by directly asking a question that would trigger another intent A or Intent C, or D or E or F or G. They could also respond with a ‘No’ response which would then trigger the **Follow-up No intent** while the **Follow-up Yes intent** for the ‘Yes’ response continues the conversation allowing the user to ask about any other intent.

The processes discussed for the Max ponded depth intent and the Flooding duration intent are the same for the other intents, except that the corresponding value for that intent will be extracted from the SWMM model. So when the user asks e.g. the time at which flooding will be maximum (Figure 34) or the maximum flooding rate (Appendix E. - ), or the depth and the duration for a flood with a 10 year return period (Appendix E. - ), the respective intent will be triggered. The user’s location of interest will then be extracted from the database, and then the respective value to answer the question will be extracted from the SWMM model. The answer will then be passed back to the user.

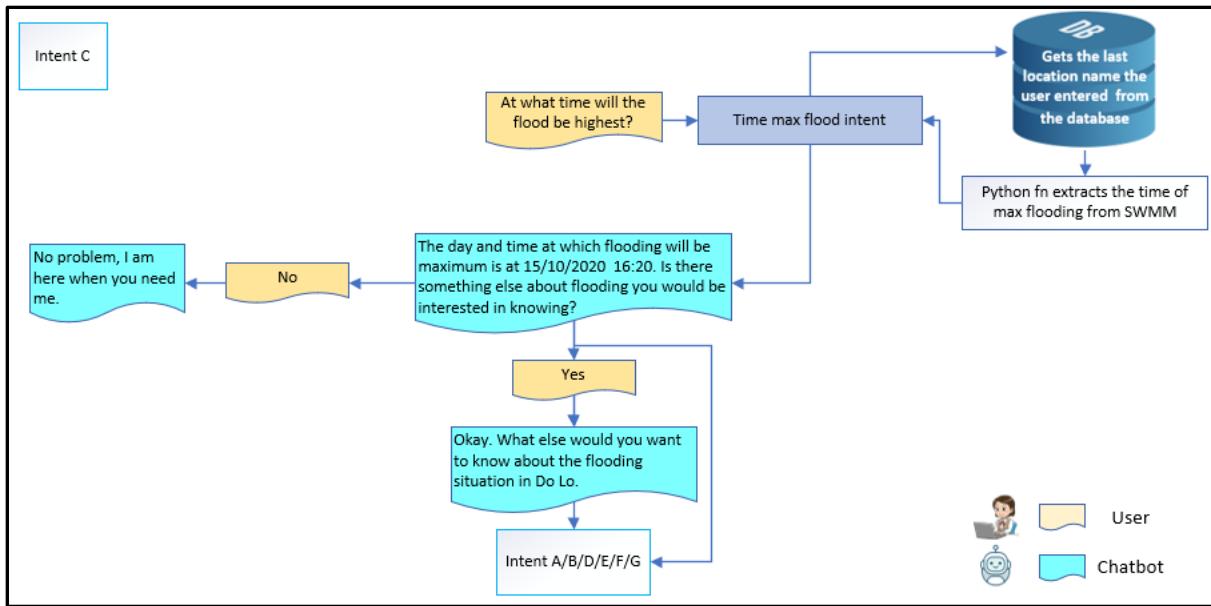


Figure 34: Time of maximum flooding intent

Another intent that addresses what the user should do in case of a flood was also developed (Figure 35). This intent offers some basic guidelines on what the user can do if their house is flooded based on an article by Risicokaart (2022). At the end of the conversation, the user is prompted to enter another location if they are interested in knowing more. This would take the conversation back to the beginning triggering the flooded point intent. If the user wishes to end the conversation, they could type ‘thank you’ or ‘bye’ which would trigger the respective intents shown in Figure 36.

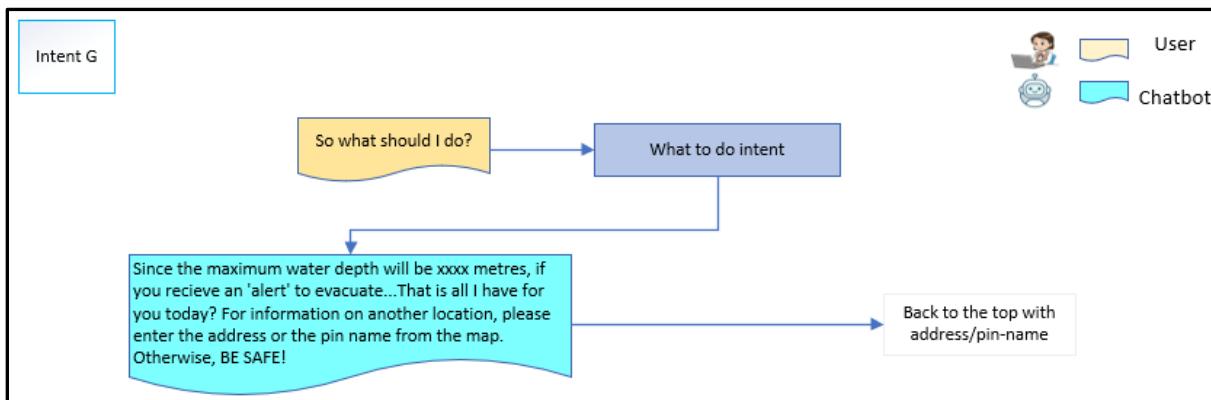


Figure 35: What to do intent

The ‘**Thank you**’ or ‘**Goodbye**’ intents (Figure 36) can be triggered at any point in the conversation if the user types any English language variants of ‘Thanks’ or ‘Goodbye’. These are static response intents developed in the Dialogflow Console.

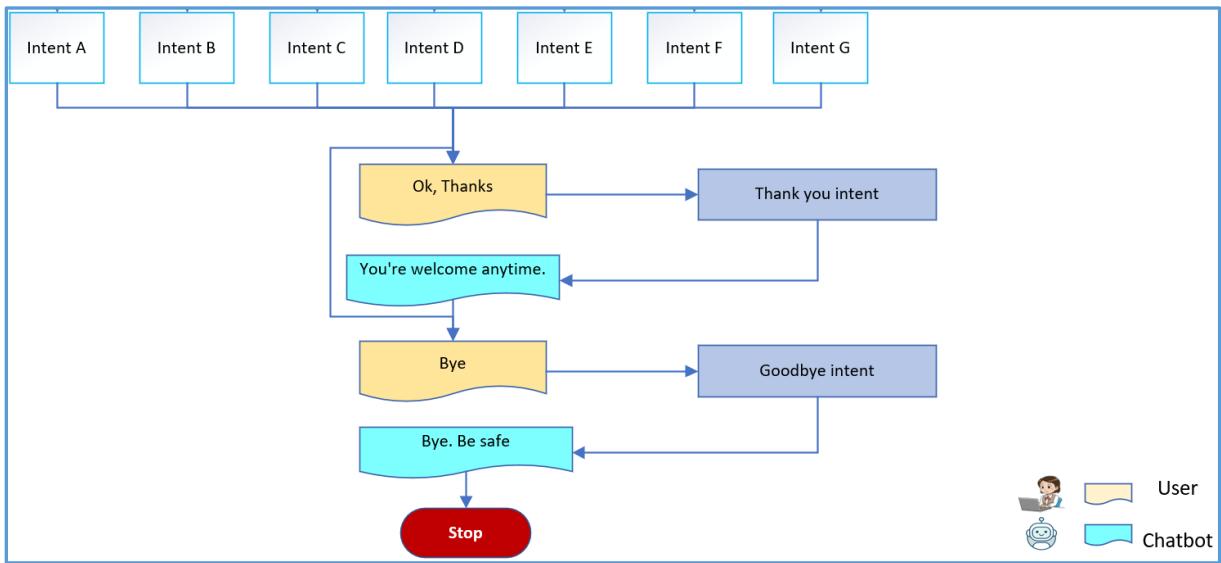


Figure 36: Thank you and Goodbye Intents

### *ii. Intent Detection and Matching*

The matching of intents is the first step towards the provision of accurate results. When the user asks a question, Dialogflow uses algorithms to check if it matches with any of the training phrases. These algorithms are: “Rule-based grammar matching and machine learning matching” (Google, 2021), which are used simultaneously to select the best match. A ML classification threshold eliminates false-positive results. An “intent detection confidence score” between 0 and 1 is allocated to potential matching intents. The default threshold is 0.3. The matched intent will be the one with the highest confidence score that is higher than the threshold. If all the intents score below the threshold, then a fallback intent will be matched. Figure 37 shows an intent successfully matched with an intent detection confidence of 1.

### Diagnostic info

Raw API response	Fulfillment request	Fulfillment response	Fulfillment status
<pre> 81      } 82    } 83  ], 84  "intent": { 85    "name": "projects/swmmobot-isvg/agent/intents/1d5ef012-4959-4551-8ad7-15c802202158", 86    "displayName": "Flooding_duration" 87  }, 88  "intentDetectionConfidence": 1, 89  "languageCode": "en" 90 }, 91  "originalDetectIntentRequest": { 92    "source": "DIALOGFLOW_CONSOLE", 93    "payload": {} 94 }, 95  "session": "projects/swmmobot-isvg/agent/sessions/7a5693cb-e32a-982c-99bd-cda8e3b7ac0d" 96 }</pre>	-	-	-

Figure 37: JSON file showing intent detection confidence for a matched intent

### iii. Running the Simulation

To get the chatbot to run the simulation, two challenges were faced. One is that Dialogflow only allows the webhook 5 seconds to generate a response before the request is timed out (Google, 2022b). The simulation on the other hand takes around 4 mins. The second challenge was that, while simulating EPA SWMM generates a report file and an output file but Google App Engine does not allow for the writing of files(Google, 2022c). So the generated files had to be saved in a temporary folder during the simulation and then saved to the database so that they are available for data extraction in the next intent. Therefore, two intents were developed, one to start the simulation and send a response telling the user to wait, and another one to then extract information after the simulation (Figure 38). The simulation function uses the Pyswmm ‘Simulate’ class.

```

def simulate():
    sim = Simulation("DoLo.inp")
    simul= sim.execute()
    return simul

if action == 'model.run':
    sim = Simulation("DoLo.inp", "/tmp/DoLo.rpt", "/tmp/DoLo.out") # to write the report n output file int
    # because App Engine doesnt allow writing of files in the normal folders hence the temp
    sim.execute() # executing take 4 mins but Dialogflow only allows 5 sec to generate a reply https://clou
    #Dialogflow will then show the response set in Dialogflow for this intent. So the webhook continues run
    #the output file are temporary so when the next action is called they will have been deleted so we have
    with open("/tmp/DoLo.rpt", "r") as f:
        report_lines = f.read()
        with pool.connect() as con:
            con.execute(text(
                "INSERT INTO swmm_reportfile(session_id, content_data) values(:session_id, :content_data)")
                session_id=sessionID, content_data=report_lines))
```

```

if action == "model.finish": # this intent uses the values from the table in the database
    with pool.connect() as con: # its a way to connect to the db faster.pool is so that we can have many connections
        resultrows = (con.execute(text("SELECT content_data from swmm_reportfile WHERE session_id =:sessionID"))
                      # we filter the users based on session ID.
                      sessionID=sessionID)

        for row in resultrows:
            report = row[0] # we are getting the whole report from the table under the heading content_data and

            with open("/tmp/final_report", "w") as file: #we have to write the file because the Ex_NodeFloodHours
                file.write(report[report.find("10^6 ltr      Meters"):]] # a way of extracting string from string

            hours = Ex_NodeFloodHrs(node_name, "/tmp/final_report", 3, 114) # the function takes 2 parameters because
            fulfillmentText = "Yes, I am done running the model.Your requested location will be flooded for "

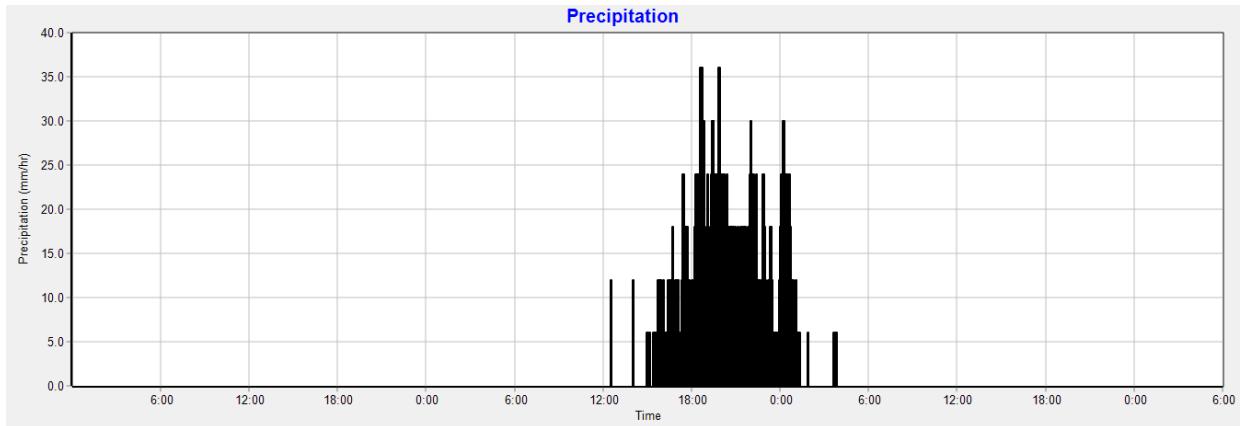
```

Figure 38: Python code for running the model

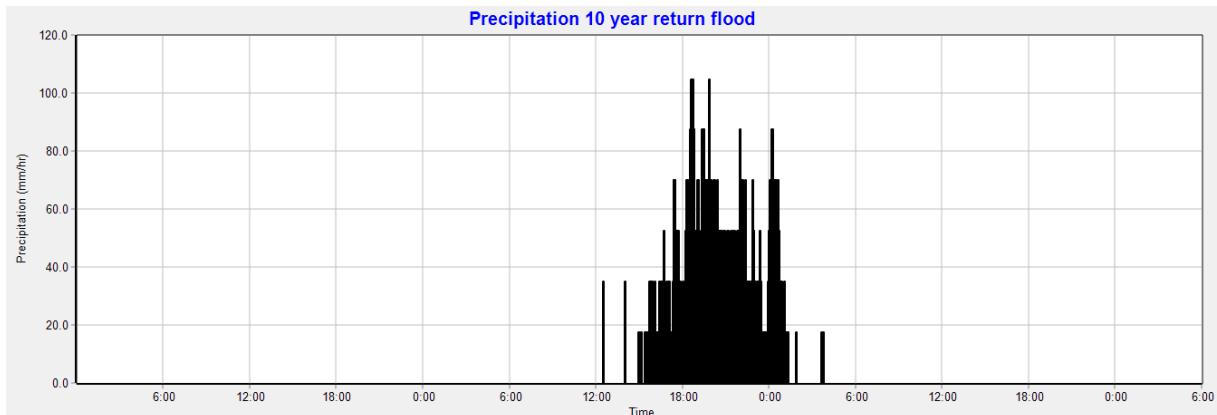
### 5.2.3 EPA SWMM Data Extraction

The intents discussed in the previous sections extracted the results from a EPA SWMM model. This section will outline the data extraction process. There are several libraries available to extract data from the EPASWMM model like PySWMM, SWMM-Toolkit, PCSWMM and MatSWMM, among others. These are wrappers or decorators i.e. functions within other functions. allowing access to SWMM model data even during simulation without altering the original model. For this research, the PySWMM library was used because it allows direct simulation of the model and extraction of model results using Python. It is also an ongoing open-source project that gets improved and updated, with the latest update in January 2022. Pyswmm has been used in other research (Sadler, et al., 2018, Li, et al., 2019, Gao, et al., 2020) to mention just a few.

The EPA SWMM model used in the research is part of an ongoing project and the setup of the model was done by Do (2021). The simulation period was for a rainfall event that occurred between 14 October 2020 and 17 October 2020 shown in Figure 39. The results provided by the chatbot are therefore for that rainfall event. The model has one rain gauge with a total rainfall of 84.8 mm. Also, it has 163 junction nodes and 4 outfall nodes; the conduit links are 175; the subcatchments are 114 and the Horton infiltration model was used. It is also important to note that the SWMM model used is a one-dimensional model with a fixed ponded area of 40m<sup>2</sup>. Therefore all computations are based on the flooding occurring over this fixed area. Another simulation was done for an event with a probability of occurring once in 10 years. The frequency-duration curve for this analysis was obtained from historical rainfall data for the period 1975-2018. The rainfall used in the model is displayed in Figure 40.



*Figure 39: Do Lo- Precipitation for 14-17 October 2020*



*Figure 40: Precipitation for a flood with a 10-year return period*

The required data was then read from the report file. The ‘linecache’ module was used to read the particular lines of interest from the report. The code used for extracting the number of hours the nodes are flooded is shown in Figure 41. The extraction function takes the location name that the user will input and returns, the day and time at which flooding will be maximum at that particular location. Other functions used to extract data from the SWMM model are in Appendix F. - Figure 42 shows locations that are flooded at 18:39 on 15 October 2020.

### Extracting the time of maximum flooding

```
startingDate = date(2020, 10, 14)
def Ex_Node_Time_maxflood(node_name):
    nodes= []
    T_maxflood = []
    for num in range(758, 869):
        filteredList = []
        result = linecache.getline("swmm.rpt", num)
        list = result.split(" ")
        for element in list:
            if len(element) > 0:
                filteredList.append(element) # append every string with greater than 0 characters this else
        nodes.append(filteredList[0])
        T_maxflood.append([str(startingDate + (timedelta(days = int(filteredList[3])))), filteredList[4]])
    index = nodes.index(node_name) # location of node_name
    return T_maxflood[index]
```

Figure 41: SWMM data extraction function

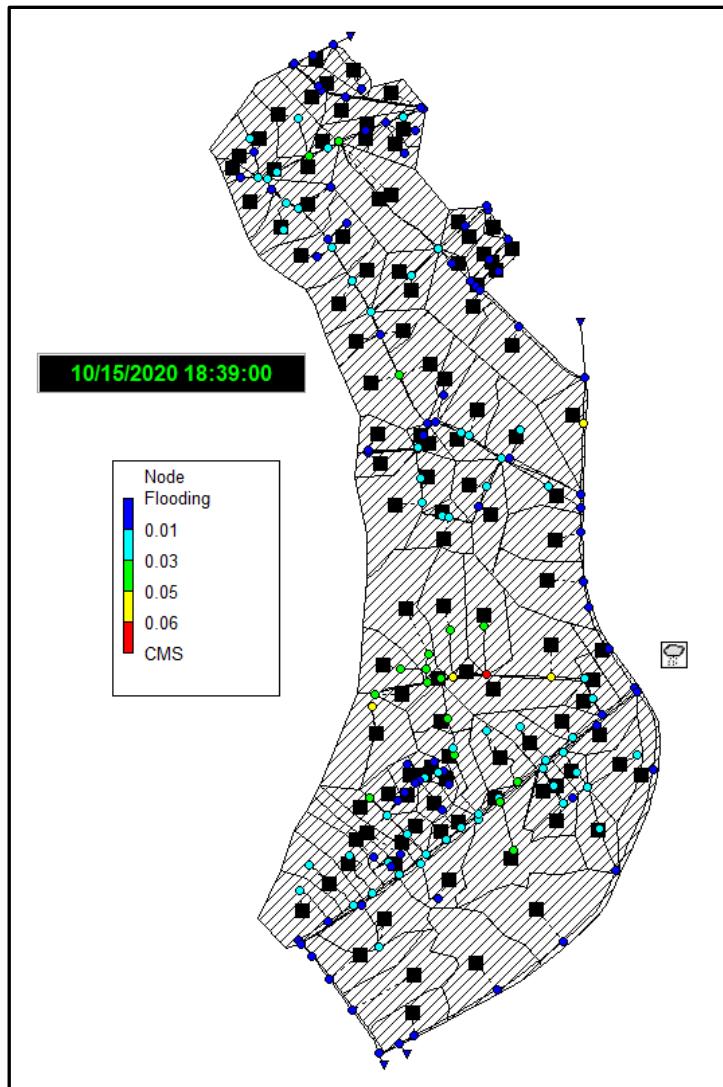


Figure 42: Flooded Nodes

Figure 43 and Figure 44 shows plots for the flow rate at one node generated using Pyswmm and the one plotted directly from the model. The next section discusses the geocoding exercise.

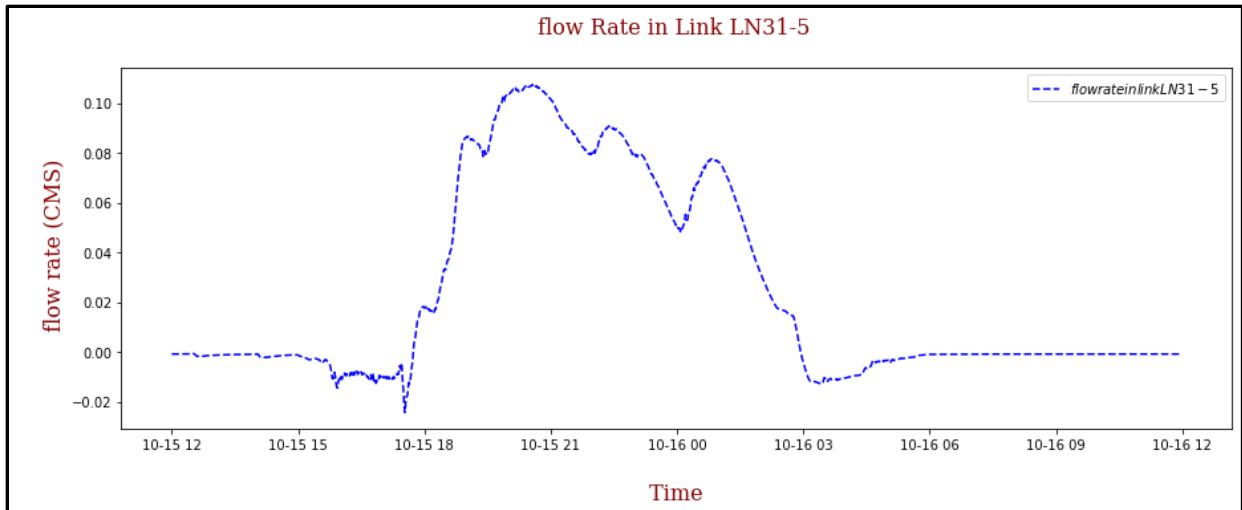


Figure 43: Flow rate for Node LN31-5 extracted using Python

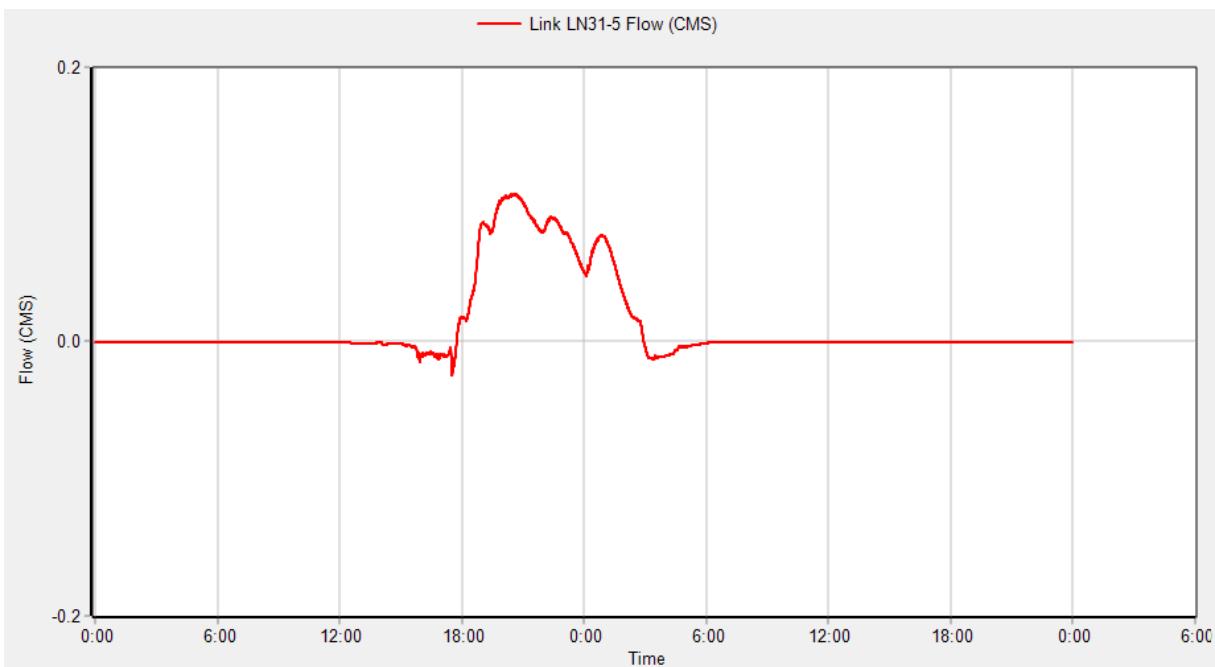


Figure 44: Flow rate plot for Node LN31-5 from SWMM

#### 5.2.4 Geocoding

As discussed in section 5.2.2, since in using the chatbot the user could input their location of interest in the form of a physical address or use the map on the website to choose a node name this had to be provided for in developing the chatbot. The goal was that if the user inputs their address then this can be matched to the node located nearest to it and then assess whether it was flooded or not. Initially, the Geopy library to get addresses from the Open Street Map API was

used because it is open-source and therefore free. However, Do Lo is a small town that has few named streets on the Open street map, so the Google maps API was then used. Google maps also has the advantage of being more accurate than Open Street Map.

The process involved creating a geocoding function to convert the users' addresses to geographical coordinates. For this an API key was obtained from the Google Cloud account and this was used to link with the Geocoding API of Google maps. The developed geocoding python function shown in Figure 45 returns the latitude and longitude of the given address.

```

30 #GEOCODING
31 gmaps_key = googlemaps.Client(key=os.environ["maps_key"])
32 def geocode(node_name):
33     g = gmaps_key.geocode(node_name)
34     lat = g[0]["geometry"]["location"]["lat"]
35     long = g[0]["geometry"]["location"]["lng"]
36     #print('Latitude: ' + str(lat) + ', Longitude: ' + str(long))
37     return lat, long

```

Figure 45: Geocoding

Using the coordinates of the address obtained from geocoding, the node closest to the location was then computed using the function in Appendix C. - . The function uses SQL to compute the shortest distance to a point. The function loops over all the node coordinates to obtain the one closest to the user's location.

### 5.2.5 Creating a Database

In section 5.2.2, the necessity of creating a database was discussed as arising from the need to store the user's location so that it may be used in computations for the proceeding intents in which the user does not specify the location anymore. Also, since the application is to be used by many users concurrently, special care was taken to keep track of which requests belong to which user, so as not to give user B data for user A's location. The session-id was used to identify which user the particular request belongs to. Using this session-id obtained from Dialogflow, each user's session id, the intent, and the location name are stored (Figure 46). These would then be used in the computations for the other intents. The session-id was used to identify which user the particular request belongs to, so as not to give user B data for user A's location.

```

| session_id | intent | location_name |
+-----+-----+
| projects/swmmbot-isvg/agent/sessions/webdemo-4a0bd029-f92d-dc9e-a7d0-69d5e16d80cc | Flooded_point | N22-1 |
+-----+-----+
mysql>in set (0.00 sec)
mysql> select * from swmm_dingsbums;
+-----+-----+
| session_id | intent | location_name |
+-----+-----+
| projects/swmmbot-isvg/agent/sessions/webdemo-4a0bd029-f92d-dc9e-a7d0-69d5e16d80cc | Flooded_point | N22-1 |
| projects/swmmbot-isvg/agent/sessions/webdemo-650ac592-5377-5ea7-5f16-9e5ff5032e32 | Flooded_point | N10-1 |
+-----+-----+

```

Figure 46: Different location names stored in the Database

The database was created using MySQL in the Google console. MySQL is an open-source “relational database management system”. It organizes data into data tables. The relations between the data assist in structuring the data. It allows one to create or modify and extract data using the Structured Query Language (SQL). The steps involved in developing the database were based on (Google, 2022a, Oracle, 2022) and included the following:

The first step involved creating a Google Cloud instance and Cloud SQL instance. Creating the database using the CREATE DATABASE statement. This creates the directory without any tables. The second step involved creating the table which was done using CREATE TABLE and the columns were session\_id, intent and location\_name. Now that we have a table, when the user uses the chatbot, data is inserted into the database using the INSERT INTO statement to insert into the columns the respective values for the session\_id, the intent and location\_name (Figure 47). To retrieve the required data from the database the SELECT statement was used.

#### Inserting values into the table

```

with pool.connect() as con: # its a way to connect to the db faster.pool is so that we can have many connections to the database.
    # establishes a connection with the db. first you specify the headings and then what should go under the headings ie values
    con.execute(text("INSERT INTO swmm_dingsbums(session_id, intent, location_name) values(:session_id, :intent, :location_name)"),
                session_id = sessionID, intent = intent, location_name = node_name )

```

#### Retrieving the last location the user inputs

```

if node_name == "" or node_name is None:
    with pool.connect() as con: # its a way to connect to the db faster.pool is so that we can have many connections
        resultrows = (con.execute(text("SELECT location_name from swmm_dingsbums WHERE session_id =:sessionID"),
                                    # we filter the users based on session ID.
                                    sessionID=sessionID)) # SQL will take the parameter name after the : and look for
        # we want to extract the latest location input from the user.
        for row in resultrows:
            node_name = row[0] # gets all the node_names which the user inputs, which the only column. and since we

```

Figure 47: Inserting values into the database table and retrieving the user's location.

### 5.2.6 Designing the Webpage

The webpage was developed using the HyperText Mark-up Language(HTML) and it was styled using the Cascading Style Sheet (CSS). HTML has a head element which is a container for what is called metadata which is information about other data. The link tags in the head, link to

external files like the style sheet and some fonts that were used in styling the text e.g. google fonts. The viewport was set to adjust the size of the web page to suit all devices.

The head is followed by the body, which is what is displayed on the webpage. At the top of the webpage is the navigation bar to which the UNESCO-IHE logo was added together with some text. The whole web page was set as a grid-container with margins on the left and right to leave some space so that the whole page is not filled with content. The page was then split into three sections. Inside the first section, was a paragraph describing the chatbot and then an unordered list was used to create the bulleted points. Below this, an image was used to describe how the chatbot and the map can be used and below the image, a weather forecast widget was added as a JavaScript element obtained from w3schools (2022). The column on the right consists of two inline frames or iframes. The first one is from the Dialogflow web demo integration which is the SWMMBot chat application. This is where all the communication with the chatbot takes place. The second iframe is a map of all the nodes in SWMM plotted on Google maps. The use of iframes allows us to include content generated elsewhere for use on our website.

The second section is a description of the study area with two columns having images of the study area. The image on the right shows the flooded areas from a 2D SWMM model simulation of the rainfall event used in the study. Below the two images, a paragraph was added to describe the study area. The last section consists of an image and a description of the processes taking place in Dialogflow and the backend until a response is sent back to the user. At the bottom of the webpage is a LightGray footer with contact details to allow users to reach the researcher if they have any questions.

### 5.3 Chatbot Integration Using Webhooks

As discussed in Section 3.3, the way webhooks work is that if something happens i.e. an event is triggered in the ‘source’ application (Dialogflow), it will be serialized and sent to the webhook URL of the ‘receiving’ application (Flask) i.e. the one that acts on the information sent. The ‘receiving’ application will then send a confirmation (callback) response message to let the ‘source’ application know whether the information it sent was successfully received or not. This would be done using an HTTP status code like 200 for successfully received and 400 if not successful. If it successfully received the request, then actions are triggered and a response is generated. If not, then the request may be resent again. Figure 48 shows links between the user input and the processes in the Flask web framework.

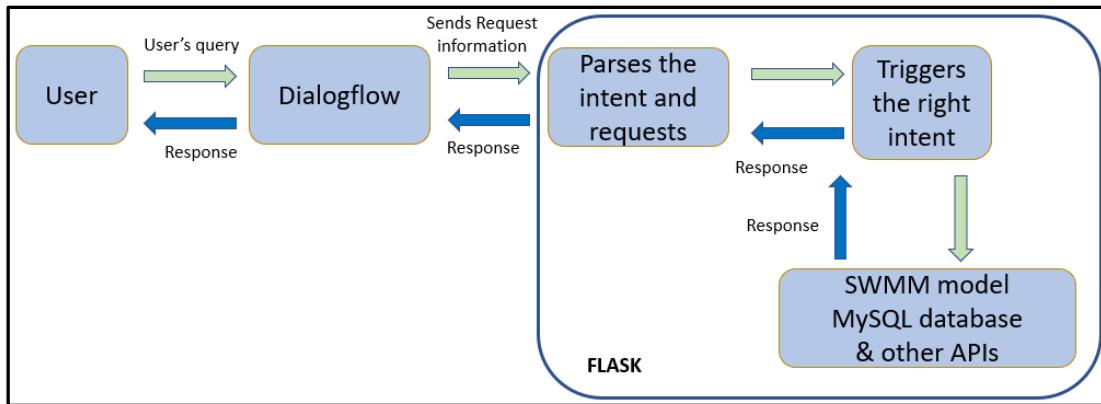


Figure 48: The Conversational flow

Adapted from: (Raturi, 2020)

Since the intents developed in Dialogflow required responses from the SWMM model, the two needed to be linked somehow. The Flask web framework was used to create a Python application which includes the webhook. In Flask, the `route()` annotation is used to specify the URL and the function which should be executed when that URL is accessed. The `run()` method is for running the application on a server, that is starting the webserver.

As outlined in section 3.3, webhooks communicate through a web protocol. So when one enables webhook fulfillment in Dialogflow, a URL has to be provided where a POST request will be sent from Dialogflow. To get this URL, the developed application has to be linked to a server and the Google App Engine was used to host the application. The App Engine is a cloud platform used for web application development and hosting. It supports the running of the Flask web framework and also allows the uploading of third-party libraries as long as they are written in Python. The installation process of the Google App Engine followed the one described in the QuickStart guide of the Google cloud platform. Firstly a project was created and named SWMMBot, after which the App Engine Build API was enabled and the App engine application created. This API is a service that allows the user to build, test and deploy their applications.

The files required included an '`app.yaml`', a '`requirements`' text file and the '`main.py`' where the webhook is developed. The '`app.yaml`' file is used to create configuration files e.g. python 37 which is the Python version. The '`requirements`' text file was also created to specify any libraries (dependencies) that were essential for the application, in this case, flask, pyswmm, shapely etc. Together with these a '`main.py`' file was created; which is the Python file where the application code is built using the Flask web framework. After this, since Google App Engine is regionalized, the West Europe region was selected for locating the app engine. The application was deployed and the resulting application can be viewed in the browser: <https://swmmbot.ew.r.appspot.com/> To integrate the webpage in Flask another route was created and the 'template renderer' was used (Figure 50). This requires the creation of a template folder for the HTML file and a static folder for the pictures and the CSS file.

Figure 49 shows the developed webhook function which starts with the instance of the class flask. Then a route is created for the webhook and the function is the webhook. This route links the application's URL with the webhook function meaning that if we visit the URL for the SWMMBot application: <https://swmmbot.ew.r.appspot.com/> we will get the output of the webhook function. The request in the webhook function is in JSON format and comes from

Dialogflow based on what the user typed and which intent it was matched to and therefore which actions are to be triggered. The silent = “True” means that if the JSON does not conform to our expectations, error messages will be ignored. The other variables are assigned information from the JSON. These will then be used to perform different actions required per intent. The Python script showing some of these actions is shown in Appendix G. -

```
app = Flask(__name__, static_folder= "static", static_url_path='')

#create a route for webhook
@app.route('/webhook', methods = ['POST'])

def webhook():
    req = request.get_json(silent=True, force= True) # request is
    fulfillmentText = '' #initialise the
    query_result = req.get('queryResult') # retrieve the
    sessionID = req.get('session')
    intent = query_result.get("intent").get('displayName')
    params = query_result.get('parameters')
    action = query_result.get('action')
    query_text = query_result.get('queryText')
    node_name=params.get('Location_name') # to get node r

    #Welcome
    if action == 'input.welcome':
        output = AllFlood_nodes()
        fulfillmentText = "Hi! My name is SWMMBot, I can provide yo
```

Figure 49: Creating the Webhook

```
#Adding the HTML page
@app.route('/', methods = ['GET'])
def home():
    return render_template("index3.html")
```

Figure 50: Integrating the HTML webpage into the Webhook

This chapter discussed the different methods used in developing the chatbot starting with the definition of a scenario where the SWMMBot can be used. After this, the chatbot development process was explained from NLP processes, the formulation of the different intents, extraction of data from the EPA SWMM, the geocoding process, the creation of a database and the process of integrating everything to produce the chatbot. The next chapter will provide the results of conversations with the chatbot.

# **Chapter 6      Results and Discussion**

---

This chapter presents the final developed chatbot, bringing out the possible application of the technology in different spheres of life. The results are outlined in the form of scenarios encountered in everyday life, showing the relevance of the chatbot application in flood awareness creation. Furthermore, the website developed as a user interface will also be shown. The chapter will also present and discuss different users' experiences as they interacted with the application, obtained from a questionnaire that was administered to assess the effectiveness and efficiency of the technology. Finally, the noted limitations of the technology will also be outlined.

## **6.1 The Developed SWMMBot Conversational App**

Individuals use applications in different ways and for different reasons. To illustrate the possible application of the developed chatbot, different scenarios depicting real-life experiences will be used. The aim here is to show how different users can use the chatbot to address various situations they might face.

### **6.1.1 Scenario 1: Flooding Situation**

Imagine you recently moved into a new house in an area that unfortunately has a high risk of flooding. It is raining cats and dogs and you want to know if your street will be flooded or if your house will get flooded? If so, to what depth the water will rise? For how long will it be flooded. Is it safe to go out for a few groceries? In addition to this, you want to know actions to take to protect yourself and your possessions from flood damage. Since you know about SWMMBot, you go to the application to get some answers to your questions. An illustration of the possible conversation is in Figure 51

In using the chatbot the user starts the conversation with a greeting and the chatbot introduces itself and prompts the user to enter their address. After computations described in the methodology, a response is then sent to the user on whether their location is flooded or not. The user's location was flooded, and after this, they could ask any further questions they may have had about the flooding situation. The chatbot then provided information on the maximum flooding depth, the total time it will be flooded, the maximum flooding rate and the time at which the flooding is expected to be maximum. The user can stop the conversation at any moment if they are not interested in receiving any more information. Based on the information provided by the chatbot, the user can then decide if it is safe to go out or not and also the best time to go out, i.e. at a time that does not coincide with the time of maximum flooding.



SWMMBot

POWERED BY Dialogflow

hi

Hi! My name is SWMMBot, I can provide you with information on the flooding situation in Do Lo, Hanoi, Vietnam. So, which location are you interested in? Enter your address or the pin name closest to you.

QL6, Yên Nghĩa, Hà Đông, Hà Nội, Vietnam

Your location QL6, Yên Nghĩa, Hà Đông, Hà Nội, Vietnam is flooded. What else would you like to know about the flooding situation?

the depth

The maximum depth of flooding at your location is 0.587 metres. Remember not to walk or drive in flood water especially if it's flowing above your ankles. Would you like to know anything else?

yes

What else would you want to know about the flooding situation in Do Lo?

what will be the duration of the flood

Your requested location will be flooded for 9.56 hours. Is there anything else you would like to know?

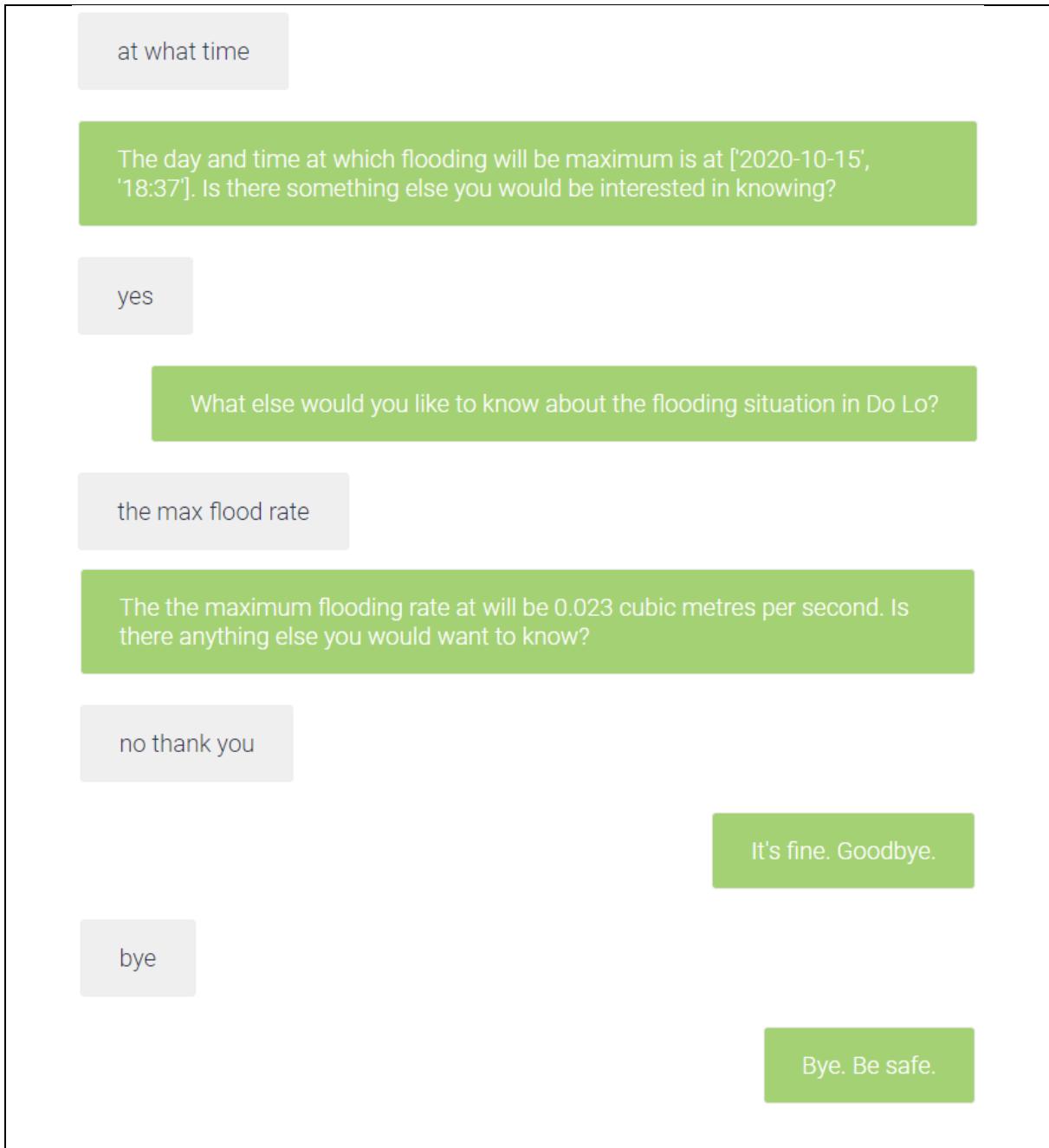


Figure 51: Scenario 1: Flooding situation

The conversation in Figure 51, shows the chatbot successfully understanding the user's questions and then matching them to the accurate actions to provide the right responses.

### 6.1.2 Scenario 2: Flooding situation for Navigation

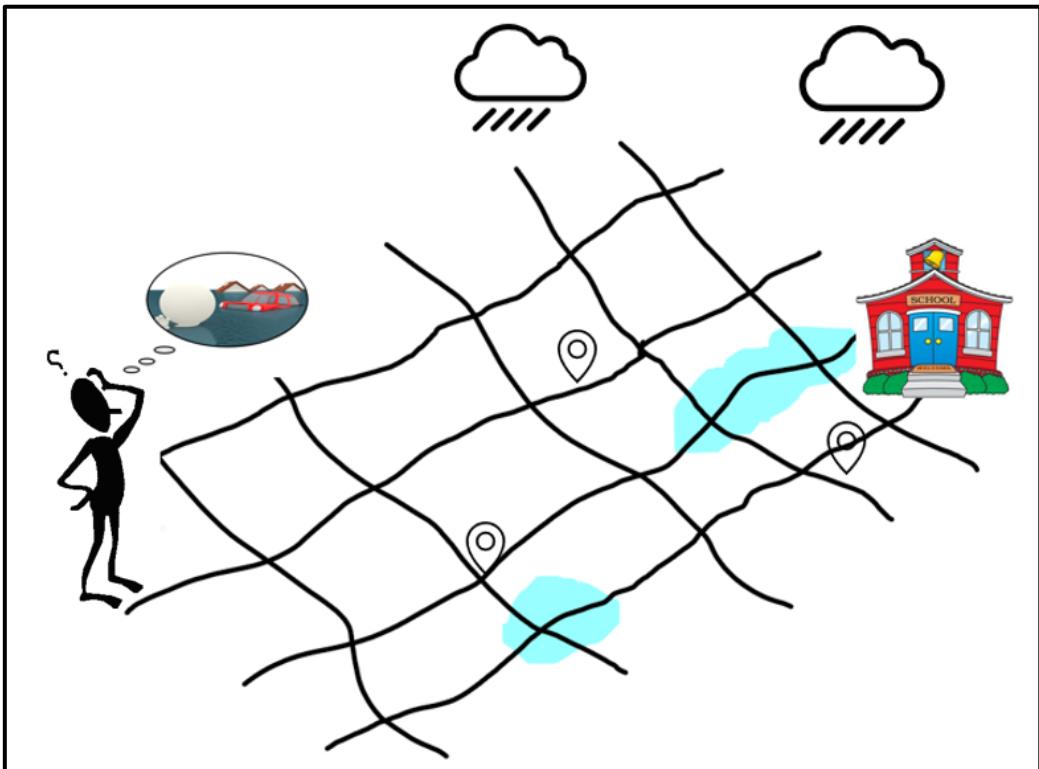


Figure 52: Scenario2: Flooding Situation for Navigation

Imagine it is the end of the day, and after a long tiring day at work and you just want to head home and rest. It is, however, that time of the year when the heavens are open and it has been raining for a few days and today was no exception. Some streets are flooded. On your way home, you have to pick up your son and niece from school and their school is on the other side of the city as shown in Figure 52. Naturally, many questions come to mind: Is it safe to drive? Which parts of the city are flooded? For how long will an area be flooded? How high are the floodwaters? When will the flooding be maximum? So you go to the SWMMBot application to ask your questions to help you decide how to navigate the city. On the SWMMBot webpage, you check the different routes you can use and for every route, you ask the chatbot about the flooding situation on the streets. Figure 53 presents a typical conversation addressing such a scenario.



SWMMBot

POWERED BY Dialogflow

Hi! My name is SWMMBot, I can provide you with information on the flooding situation in Do Lo, Hanoi, Vietnam. So, which location are you interested in? Enter your address or the pin name closest to you.

N33-1

Your location N33-1 is flooded. What else would you like to know about the flooding situation?

to what depth

The maximum depth of flooding at your location is 1.743 metres. Remember not to walk or drive in flood water especially if it's flowing above your ankles. Would you like to know anything else?

and the duration

Your requested location will be flooded for 12.93 hours. Is there anything else you would like to know?

what about N31-22-1

The maximum flooding rate at N31-22-1 will be 0.047 cubic metres per second. Is there anything else you would want to know?

no, the flooding depth

The maximum depth of flooding at your location is 1.72 metres. Remember not to walk or drive in flood water especially if it's flowing above your ankles. Would you like to know anything else?

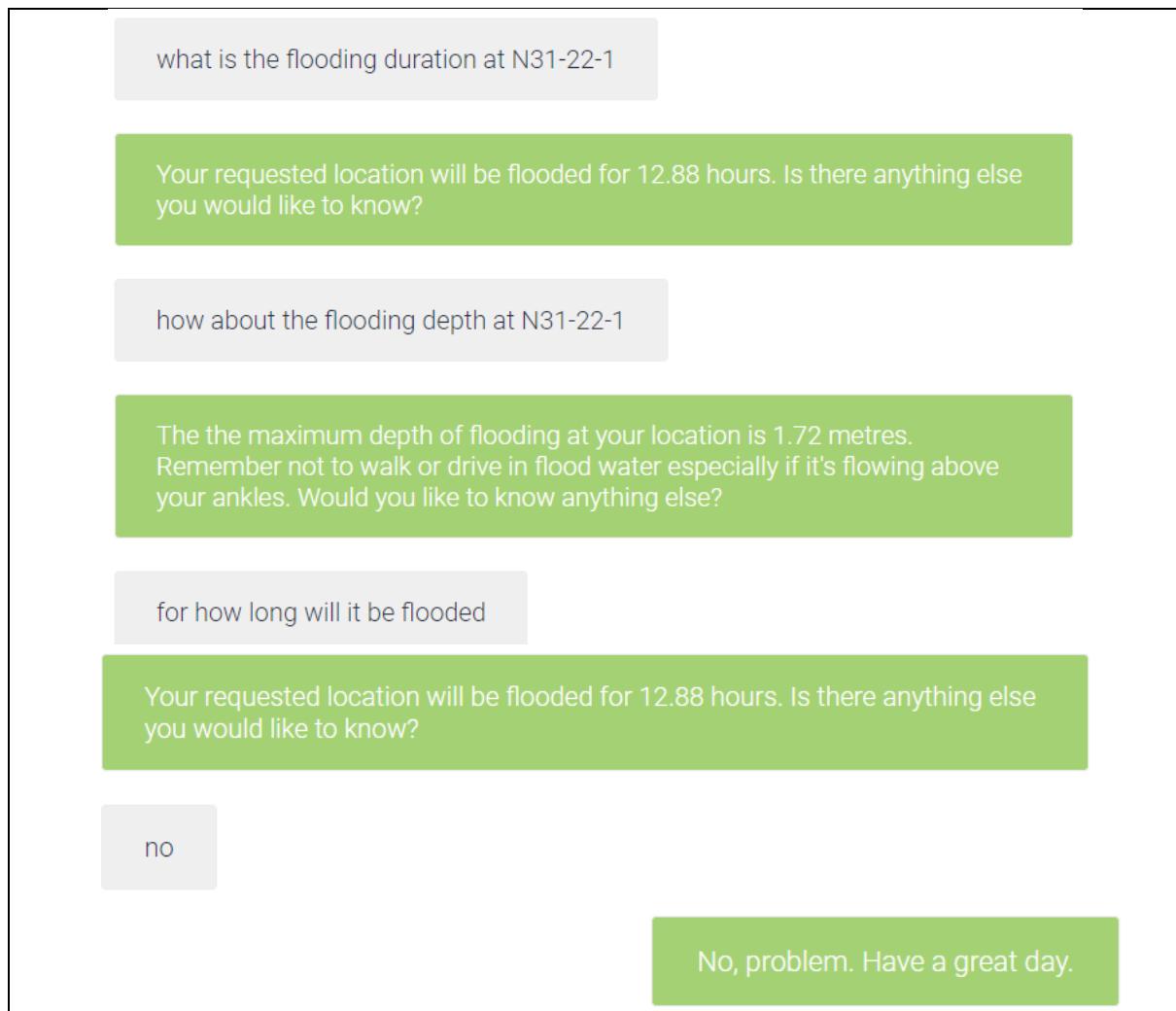


Figure 53: Scenario 2: Conversation to assess the flooding situation in different parts of the city

From Figure 53, the user can use the map to navigate to different locations on the map and find out whether the locations are flooded. Also, using the maximum depth of flooding and the flooding duration, they can find possible routes to follow that are not flooded. Knowing the time of maximum flooding at the different locations would assist in deciding when to leave their location. If it happens that their location is more flooded than their destination (in this case the school), they can then inform the children or their teacher, of the flooding situation and what time they are most likely to be able to pick them up.

Alternatively, if you are new in an area for navigation purposes, since the map is a Google Map, they can get directions from location A to B and then use the SWMMBot application to assess the flooding situation from the starting point, through various points up to their destination. In this way, the user is not just opting for the shortest distance but also for the safest route to follow.

### 6.1.3 Scenario 3: Flooding Situation for a Flood with a Ten Year Return Period

Imagine you want to purchase a house and unfortunately, you cannot find any flood zoning maps for the area you are interested in. You want to assess which areas experience little or no

flooding so you can choose those areas and not have to pay for flood insurance. Or you have already purchased land in a flood-prone area and you're still deciding on the design of the house. You want to know if there will be any need to elevate the house or not and if so to what height. So you ask SWMMBot the flood depth and duration for the current simulation and that for a flood with a 10-year return period. Such a conversation is illustrated in Figure 54.

The screenshot shows a conversation between a user and a bot named SWMMBot. The bot is powered by Dialogflow. The user's messages are in grey boxes, and the bot's responses are in green boxes.

hi

Hi! My name is SWMMBot, I can provide you with information on the flooding situation in Do Lo, Hanoi, Vietnam. So, which location are you interested in? Enter your address or the pin name closest to you.

N23-1

Your location N23-1 is flooded. What else would you like to know about the flooding situation?

what is the flooding depth

The maximum depth of flooding at your location is 0.681 metres. Remember not to walk or drive in flood water especially if it's flowing above your ankles. Would you like to know anything else?

and the duration

Your requested location will be flooded for 7.64 hours. Is there anything else you would like to know?

what would be the depth for a flood with a 10 year return period.

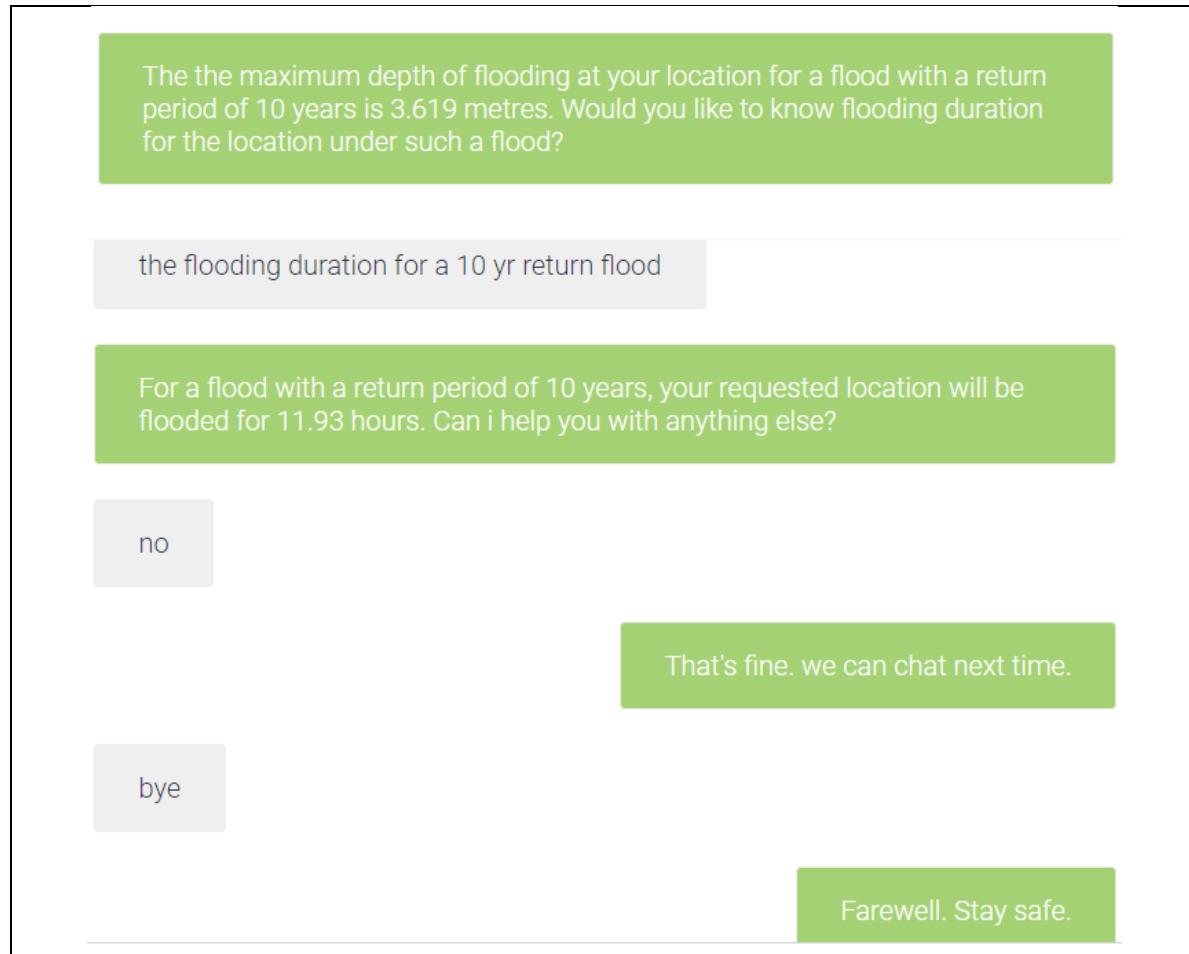


Figure 54: Scenario 3: Conversation to compare the present flooding situation and that for a flood with a 10 year return period

In this case, the user can access flooding data for one location for the present rainfall simulation and for a flood that has the probability of occurring once in ten years. They can then do the same for other different locations and then select the site with the least probability of flooding. The chatbot will thus help the user avoid insurance costs or make an informed decision on the architectural plan of their house.

#### 6.1.4 Scenario 4: Actions to Take During Flooding

This scenario is similar to the first one where it has been raining for hours and unfortunately you live in an area with a high risk of flooding. So in addition to getting information on the flooding situation at your location, you also want to get information on what actions to take as the flooding depth increases. So ask the SWMMBot and the conversation could be as shown in Figure 55. This would be of great benefit for users who may have never experienced flooding and would therefore find themselves stranded, not knowing what to do. It can also serve as a reminder of what to do even to those residents who have experienced some flooding. The solutions provided are obtained from Risicokaart (2022) and they apply to any area in the world.

hi

Hi! My name is SWMMBot, I can provide you with information on the flooding situation in Do Lo, Hanoi, Vietnam. So, which location are you interested in? Enter your address or the pin name closest to you.

Đường Nghĩa Lộ, Yên Nghĩa, Hà Đông, Hà Nội, Vietnam

Your location Đường Nghĩa Lộ, Yên Nghĩa, Hà Đông, Hà Nội, Vietnam is flooded. What else would you like to know about the flooding situation?

depth

The maximum depth of flooding at your location is 0.214 metres. Remember not to walk or drive in flood water especially if it's flowing above your ankles. Would you like to know anything else?

what to do?

Since the maximum water depth will be 0.214 metres, if you receive an 'alert' to evacuate OR your house doesn't have a floor higher than this depth, please go to the nearest Emergency Shelter. If you must drive, the water depth must be below 20cm or below half the height of your car wheels. Otherwise, here's what you can do: 1. when water depth is below 20cm ensure that all important items are stored at a high, dry place, switch off switches, gas and close taps 2. for water depths between 20-50 cm take your emergency package (food, water, warm clothing) with you to a higher floor, 3. between 80cm - 2m, the 1st floor of your house is safe, 4. between 2m - 5m, the second floor of your house is safe. Above 5m, go to the highest point in your house and wait for the emergency rescue team. Check radio and SMS alerts. For more tips visit: <https://www.ready.gov/floods> or <https://www.risicokaart.nl/en/what-risks-are-there/flood>. That is all I have for you today? For information on another location, please enter the address or the pin name from the map. Otherwise, BE SAFE!

Figure 55: Conversation on what to do during flooding

The response provided by the chatbot is quite lengthy but since the chatbot can be integrated for voice conversations the users will then not have to read the lengthy text. Receiving such

information can have a calming effect on the user as they will be in control of the situation and be able to take action to protect themselves and their families and their belongings.

### 6.1.5 Scenario 4: Flood Proofing a House or Neighbourhood

An alert has been sent out by the Disaster Management Authority of a pending flood of high magnitude. As a small neighborhood, you usually build temporary flood barriers around the neighborhood during such times. Now, with the help of SWMMBot, you can obtain information on the maximum depth and the duration of the flooding. Using this information you can then decide on the type of barrier to be used and also, the height to which you should stack the barriers at different sites (Figure 56).



Figure 56: Flood barriers

Source: <https://floodcontrolinternational.com/flood-barriers/>

### 6.1.6 Not Flooded or Location Outside Do Lo

For the flooding data used in the EPA SWMM model for Do Lo, 111 of the 167 locations were flooded. The chatbot was, therefore, able to notify the user if their location was not flooded and also prompt the user to enter another location if they were interested in doing so (Figure 57). Furthermore, the chatbot was developed to identify locations that are outside the study area and inform the user that it could only provide information on the flooding situation in Do Lo (Figure 58).

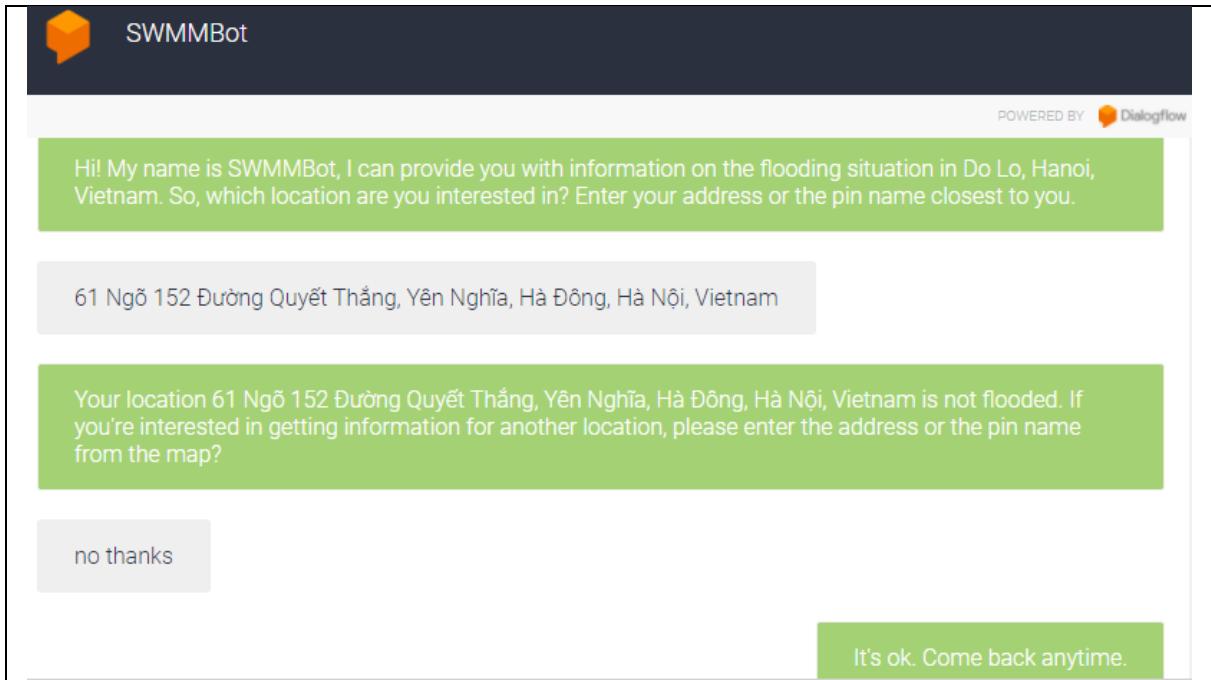


Figure 57: Chatbot conversation if the location is not flooded

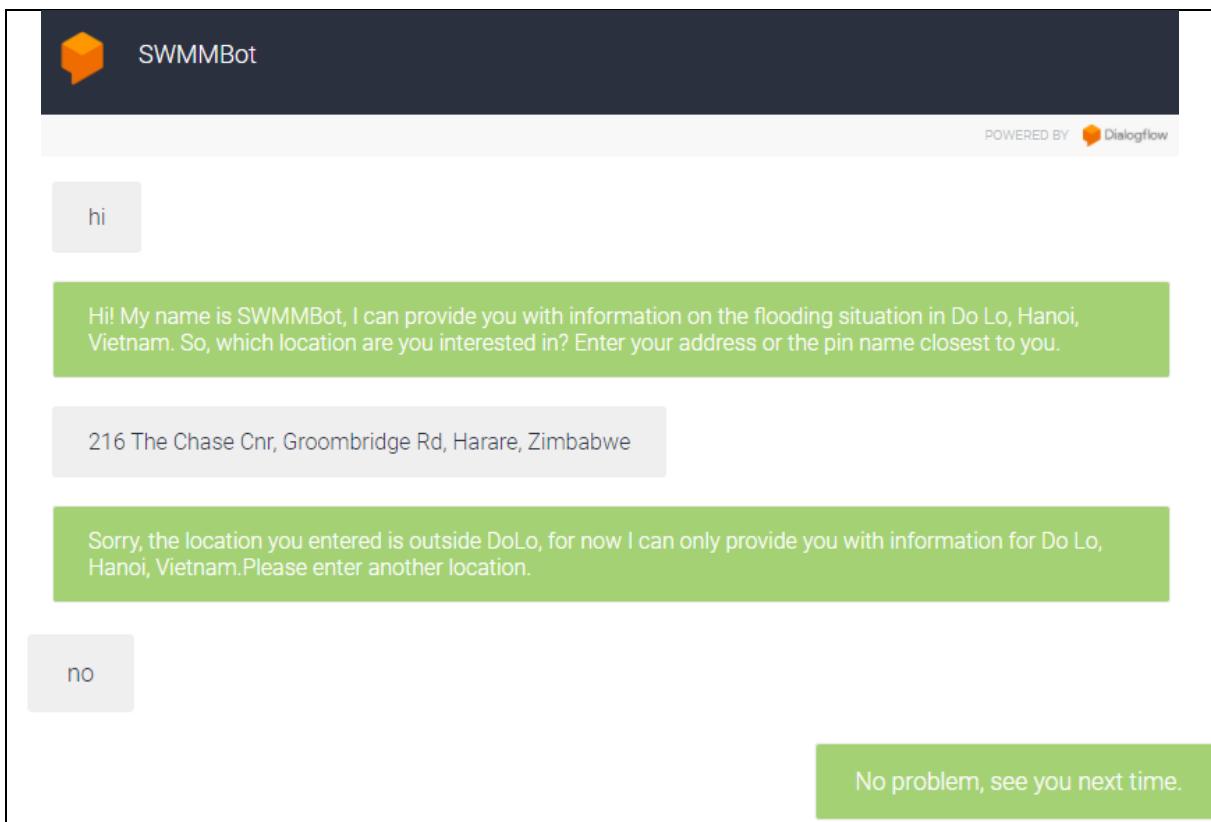


Figure 58: Chatbot conversation for an area outside Do Lo

### 6.1.7 Discussion SWMMBot

As highlighted in previous sections, the SWMMBot chatbot application was developed to answer questions on the flooding situation at various locations based on results obtained from the EPASWMM model. The successful development of this application aimed to contribute to efforts on reducing the impacts of flooding, focusing on improving flood awareness and flood preparedness through disseminating information at a rate that is understandable to the user. The chatbot, therefore, successfully equips the user with knowledge on whether their location will be flooded or not, to what maximum depth, the duration of the flood, the maximum flooding rate and the date and time the flood is expected to be maximum. With this information, the user is then equipped to make informed decisions on how to proceed, if their location is already flooded or a flood is anticipated in the area. This preparedness has been proven to reduce the impacts of floods, for example Owusu, et al. (2015), noted that, in the United Kingdom, for every €1 spent on “Property-level flood Protection”, €5 will be saved from flood damages that would have occurred. Chatbots such as the SWMMBot prototype can help in providing the necessary flooding information which can help citizens in preparing for floods.

Since chatbots should be designed to imitate communication between humans, it can be observed from the conversations presented in Figure 51 to Figure 57 that the chatbot starts by greeting and introducing itself, the same way humans do upon meeting. This gives the user an impression that they are talking to a reliable source of information. Also seen from the figures is the fact that the chatbot managed to understand the user’s requests and match them to the appropriate intents and provide accurate responses. This is the ultimate goal for developing the chatbot; getting it to provide the right responses to users.

However, in Figure 53 there is an incident where the user says ‘*what about N31-22-I*’ and the chatbot matches this request to the flooding rate intent and gives the flooding rate at that location. It is only when the user says ‘*no the flooding depth*’ that the communication is successful. This results from the fact that since the question did not have the differentiating entities ‘*depth*’ or ‘*duration*’ the chatbot matched with the most likely intent. However, when the user specifies that they are talking about the depth, then the intent is matched and the proper response is provided. This illustrates that while chatbots are more often accurate, as a technology, they sometimes fail to fully understand the human language. Janssen, et al. (2021) maintain that as long as the intended task is achieved, then the communication between the user and the chatbot was successful. In line with this, it can then be concluded that all conversations illustrated in Figure 51 to Figure 57 were successful and the chatbot functioned well.

Since the SWMMBot, was developed as a prototype to illustrate the possibility of extracting data from SWMM, this goal was met. There is, however, always room for improvement. As more and more people interact with it and provide feedback, it can be improved further, making it address more flooding needs based on feedback from the users. While the chatbot was developed in English there is a possibility to have it translated into any other language. Together with this, since numerous urban drainage models have been developed globally, these could then be coupled to the chatbot instead of SWMM. This would ensure increased access to flooding information for populations in various locations.

### 6.1.8 Limitations of the SWMMBot Application

As humans, we sometimes fail in executing some tasks we set out to do and similarly, technologies such as chatbots also fail in executing some tasks. This is because AI chatbot development greatly relies on Natural Language Processing which is not an exact science but uses different models to do different tasks. Outputs from these models have some level of **uncertainty** which in turn impacts intent matching and therefore sometimes wrong responses may be provided. This, however, is a frustrating experience for the users who may then not use the application in the future.

Also, while chatbots aim to imitate human conversations, they occasionally miss the context of the conversation. Some context can be set between different intents and responses but there is always a possibility that the chatbot will miss this context and match user's requests with the wrong intents and thus provide the wrong response. As discussed in Section ii, intent matching is not fixed and if the NPL processes fail to understand and match the user's question to the right intent then a wrong response is sent or the default response will be sent.

Another challenge could be as shown in Figure 59, where the chatbot failed to connect with the server. In this case, it therefore, cannot execute any of the tasks that are dependent on the HTTP request sent through the webhook. It will then return the default response set in Dialogflow, 'server offline'. This failure usually occurs if there is a Google App engine network error or there is a bug in any of the developed Python code. The webhook then has nothing to send back to Dialogflow so it throws an exception and Dialogflow then sends the response set for that intent to the user. Once the error is resolved or the connection is restored the chatbot will start functioning properly, but this may annoy the user as they would have to repeat their questions or they may stop using the application altogether. In such cases, the goal of improving the users' flood preparedness is not met.

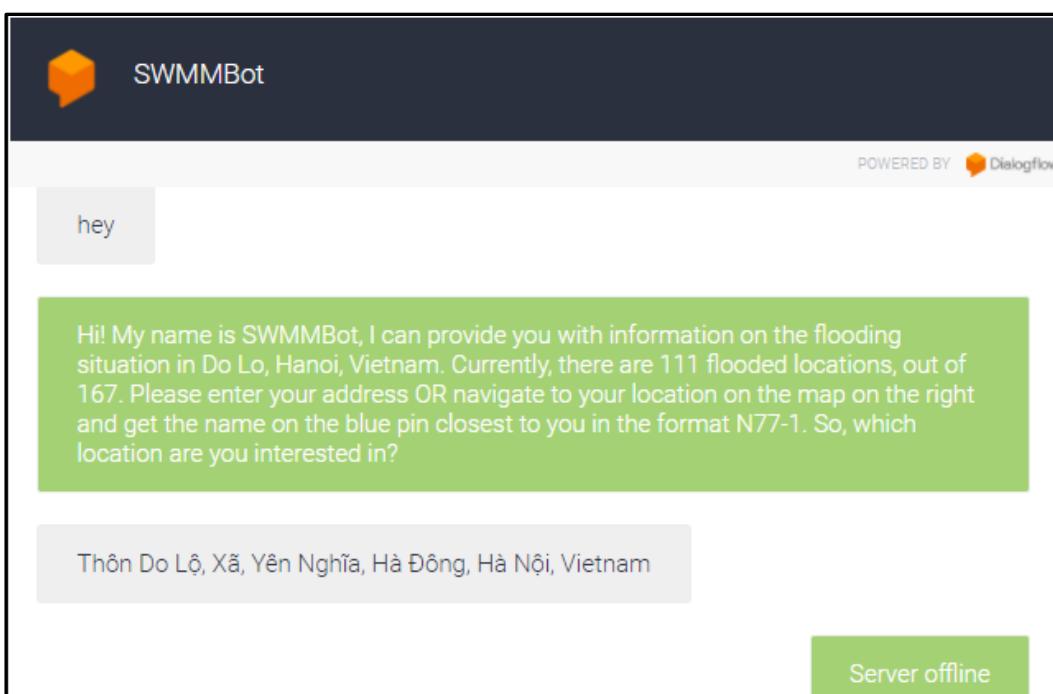
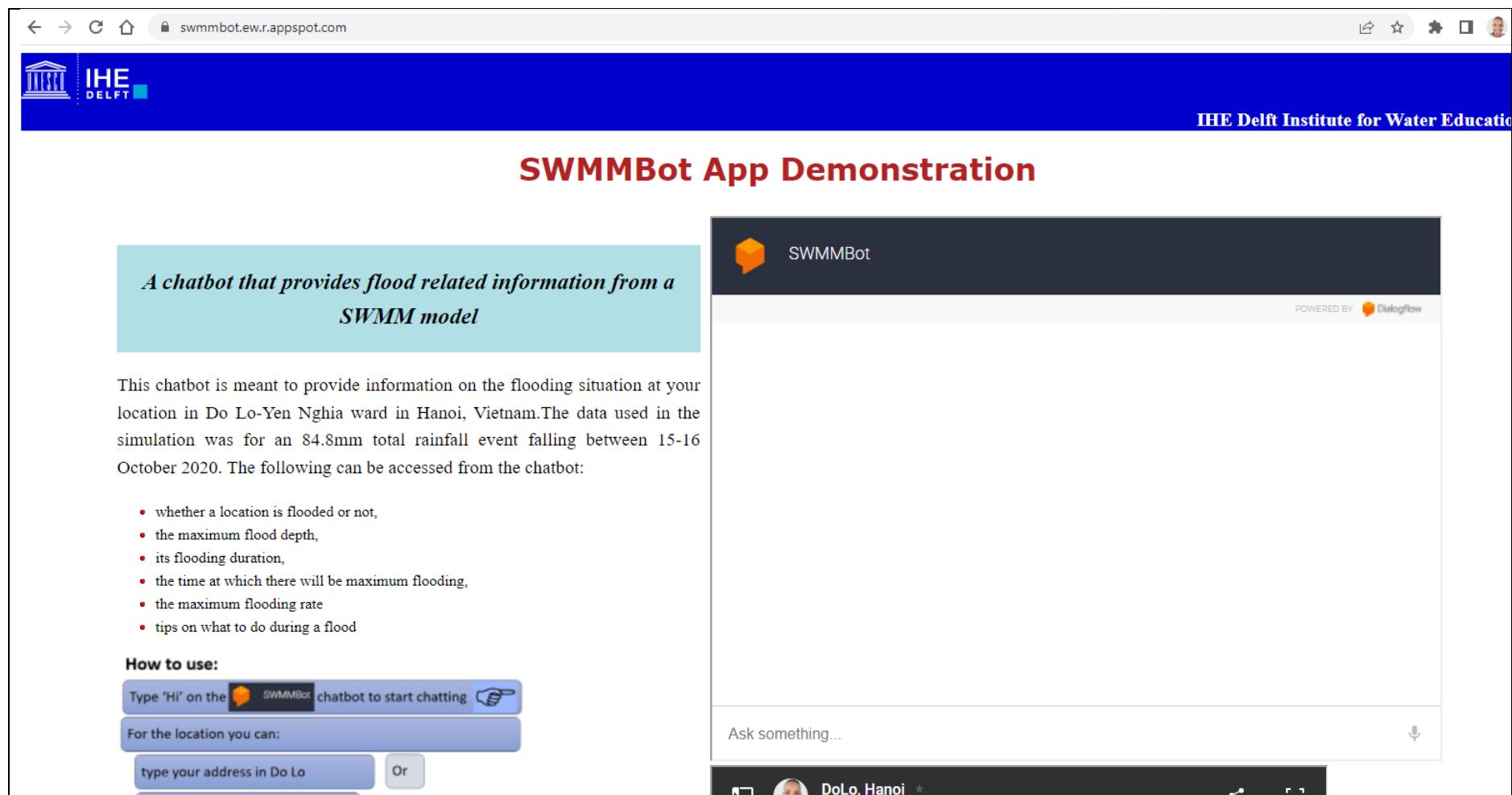


Figure 59: When Chatbot fails to connect to the server.

Another issue is that while the chatbot could provide information on the maximum flood depth, duration and time of maximum flooding, etc, but the velocity of flow could not be provided. This results from the fact that the chatbot used a one-dimensional SWMM model which only computes the velocity of flow within the pipes but not for the ponded water. Therefore while this is an important parameter to consider since the velocity of the water determines its destructive force, it could not be determined in this case. This could be solved by using a 2D model where possible.

## 6.2 The Graphical User Interface: Website

As outlined in Section 5.2.6, the interface developed for users to interact with the chatbot is a website. This was designed with a map to allow the user to navigate to their location of interest and either get the node name to input into the chatbot as they ask their questions or just input the address of their location. The webpage was divided into three sections. The first section displays the chatbot and a brief explanation of how to use it. A flood forecast widget was added to provide the current flood forecast in DoLo. This second section describes the study area. And the last section describes the Dialogflow flow path with a footer for feedback and any contact information. This section will therefore show the outline of the website for clarity on how the whole set-up looks.



The screenshot shows a web browser displaying the [SWMMBot App Demonstration](https://swmmbot.ewr.appspot.com) page. The page has a blue header with the IHE Delft logo and the text "IHE Delft Institute for Water Education". Below the header, the title "SWMMBot App Demonstration" is displayed in red. A teal box contains the text: "A chatbot that provides flood related information from a SWMM model". The main content area describes the chatbot's purpose: providing flooding information for Do Lo-Yen Nghia ward in Hanoi, Vietnam, based on an 84.8mm rainfall event in October 2020. It lists several features: whether a location is flooded or not, maximum flood depth, flooding duration, time of maximum flooding, maximum flooding rate, and tips for flooding. A "How to use:" section shows how to start chatting by typing "Hi" and provides two input fields: one for address and one for location. On the right, there is a screenshot of the SWMMBot interface with a dark background, featuring the bot's name, a "Powered by Dialogflow" badge, and a text input field with the placeholder "Ask something...".

A chatbot that provides flood related information from a SWMM model

This chatbot is meant to provide information on the flooding situation at your location in Do Lo-Yen Nghia ward in Hanoi, Vietnam. The data used in the simulation was for an 84.8mm total rainfall event falling between 15-16 October 2020. The following can be accessed from the chatbot:

- whether a location is flooded or not,
- the maximum flood depth,
- its flooding duration,
- the time at which there will be maximum flooding,
- the maximum flooding rate
- tips on what to do during a flood

**How to use:**

Type 'Hi' on the SWMMBot chatbot to start chatting

For the location you can:

type your address in Do Lo Or

Ask something...

POWERED BY Dialogflow

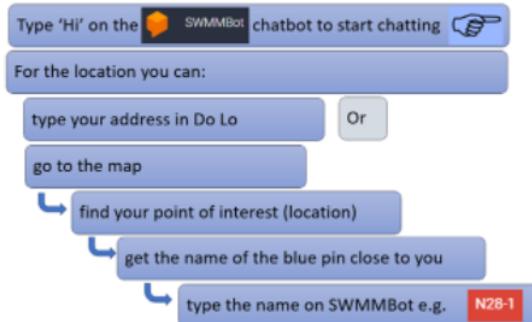
IHE Delft Institute for Water Education

SWMMBot

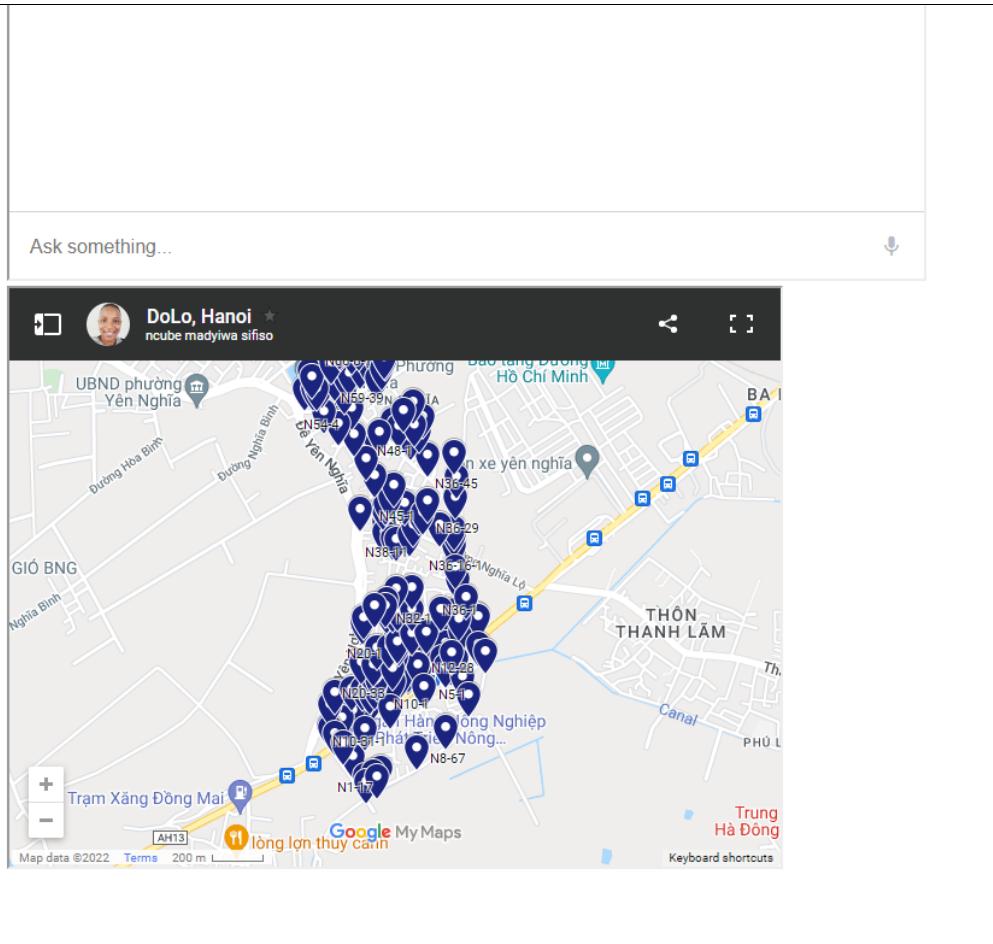
DoLo, Hanoi

- its flooding duration,
- the time at which there will be maximum flooding,
- the maximum flooding rate
- tips on what to do during a flood

#### How to use:

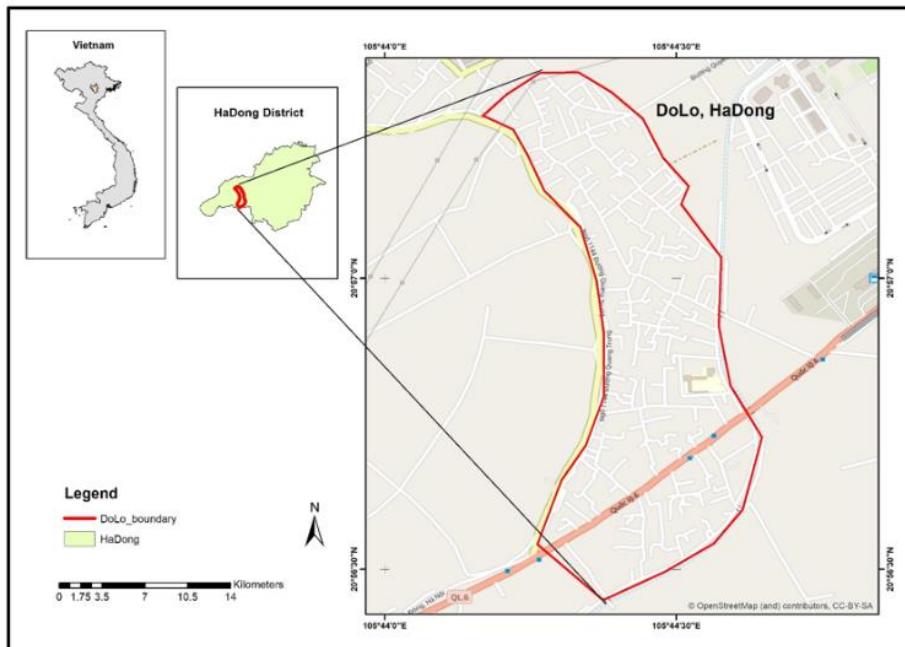


#### Hanoi Weather



## Study Area

Do Lo-Yen Nghia, Ha Dong District, Hanoi, Vietnam



The Drainage System



Do Lo lies to the south of the Yen Nghia ward in the Ha Dong District of Hanoi, the capital city of Vietnam. The average temperature is 23°C, with the lowest being 5°C and the highest 38°C. Do Lo, Yen Nghia is about 61.5 ha in area and is located next to the Yen Nghia dyke. It is this dyke that protects the area from the Day River which lies to the west of the ward. The Co Ban channel, located to the South and the East of Do Lo, is the main output of the drainage system. The drainage system consists mostly of rectangular open channels and circular pipes below roads. The area is experiencing rapid population growth which in turn puts a strain on the drainage infrastructure.

## Dialogflow Conversational Flow Path

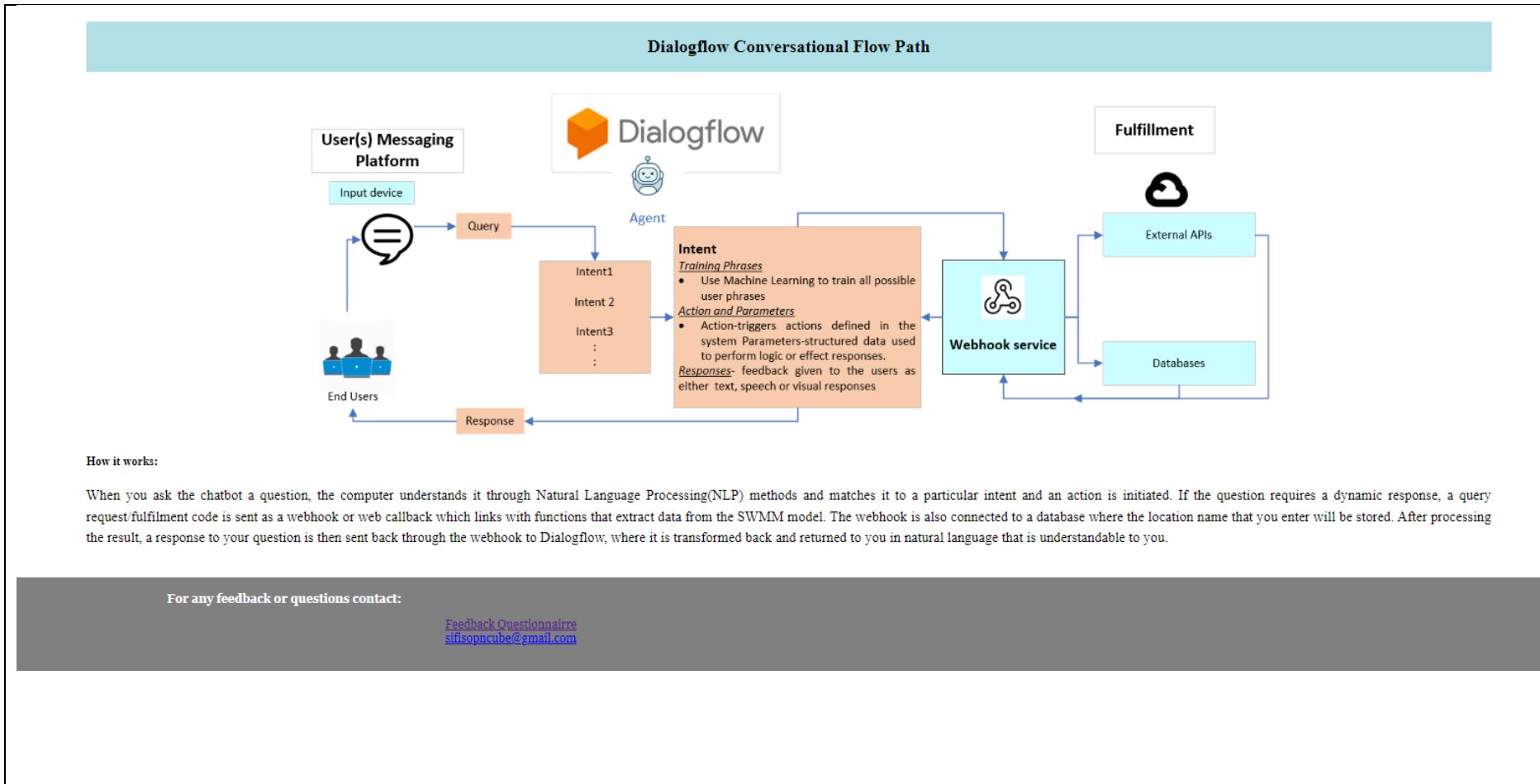


Figure 60: The Developed Website

## 6.3 SWMMBot Pilot Test

To obtain feedback from users on the performance of the chatbot a pilot test was done. A questionnaire was administered online through the Google forms platform to allow the users to interact with the chatbot and document their experiences. The respondents are from different areas of the globe and not specifically from the study area. The questionnaire had 16 questions and is shown in Appendix I. - The test was for testing the computational functionality of the developed SWMMBot chat application. It was purely for academic purposes and no personal information was retained as part of this work. The population sample consisted of a total of 34 respondents and 52.9% of whom were female, 41.2% male and 5.9% chose not to reveal their gender. In terms of age, 2.9% of the respondents were below 20 years, 85.3 % were between the ages of 21 to 40, and 11.8 % were between 41and 60.

### 6.3.1 Ease of use

The first set of questions addressed, the ease of use of the chatbot and the results are presented in Figure 61. From the chart on the left, it can be observed that a majority (97.1%) of the users found the SWMMBot easy to use while a minority (2%) did not. In terms of easily accessing the information they needed, the figure on the right shows that 79.4% of the users could easily access the information they required. Therefore, a majority of the users found the application easy to use, although a minority faced some challenges. In developing the chatbot, the goal was for the application to provide accurate, easy-to-understand responses and these responses from the users confirm that this objective was achieved.

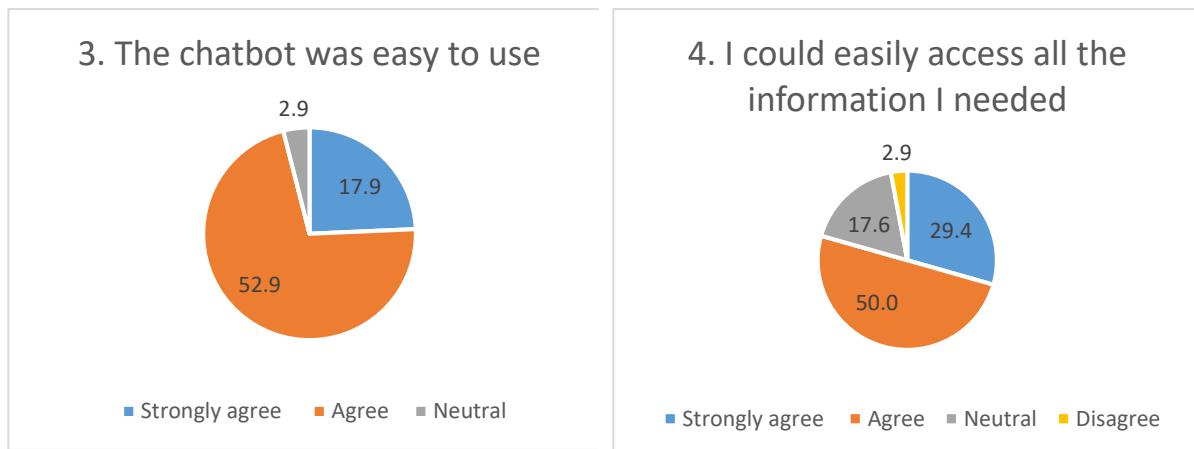
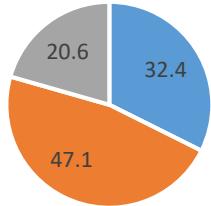


Figure 61: Ease of use

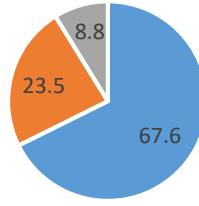
### 6.3.2 Accurate and understandable responses

5. The chatbot gave accurate responses



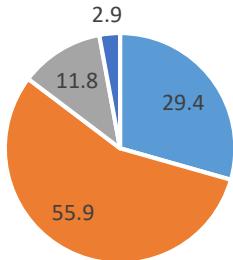
■ Strongly agree ■ Agree ■ Neutral

6. The responses provided were easy to understand



■ Strongly agree ■ Agree ■ Neutral

10. The information obtained from the chatbot was relevant



■ Strongly agree ■ Agree ■ Neutral  
■ Disagree ■ Strongly disagree

*Figure 62: Accurate and easy to understand responses*

The next questions focused on how accurate and understandable the users found the responses provided. To this question, the top left figure shows that 79.4% of the participants received accurate responses while 20.6% had mixed feelings about the accuracy (Figure 62). From the chart on the top right, 91.2% agreed that they could easily understand the responses provided by the chatbot while 8.8% could not. The chart at the bottom shows that some respondents (2.9%) found the results from the chatbot irrelevant while 85.3% found the information relevant. This is a positive outcome since the chatbot was created to provide flooding information to users in smaller ‘chunks’ that they could understand. There were, however, some users who found the responses difficult to understand showing that there are still some improvements that need to be done on the chatbot to address some of the highlighted shortfalls

### 6.3.3 I could easily access my location on the map and other locations

The question of whether the users could easily find their locations on the map led to almost 1/3 of the population stating that they were not able to access their location on the map (Figure 63 top-left chart). On the other hand, about 2/3 of the respondents could easily find their locations

on the map. The top right figure shows that 70.6% of the surveyees could switch and get information for other locations while 29.4 % were not able to make this switch. This would need further investigation into what the problem was. In the bottom left figure, 64.7% of the respondents agreed that the chatbot could identify locations outside the study area. This leaves, more than a 1/3 asserting that the chatbot failed to make this recognition. To address this, the Python functions addressing this aspect in the chatbot we revisited to resolve any errors, and also a pictorial explanation of how users could access their locations was added to make it easier to understand.

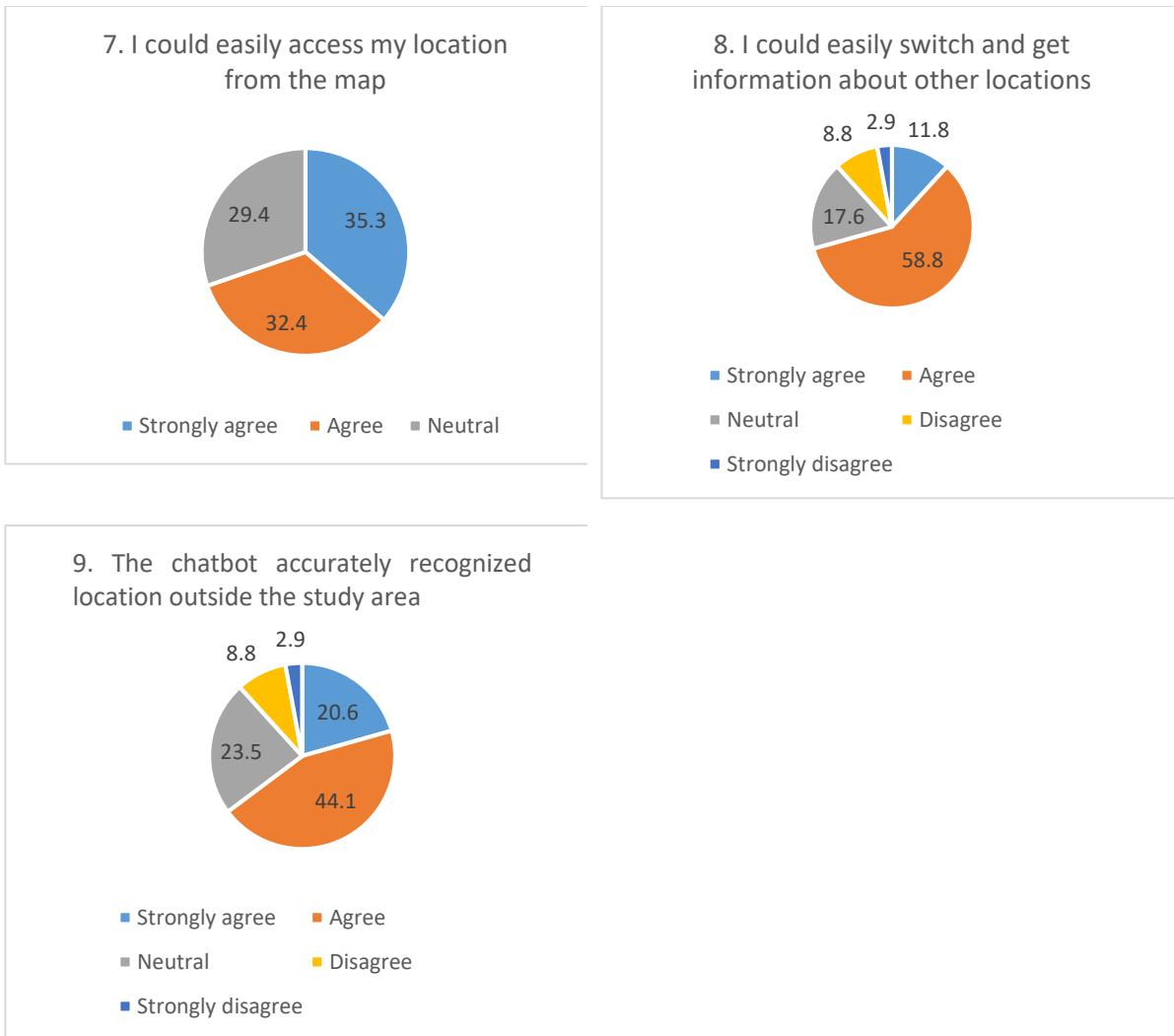
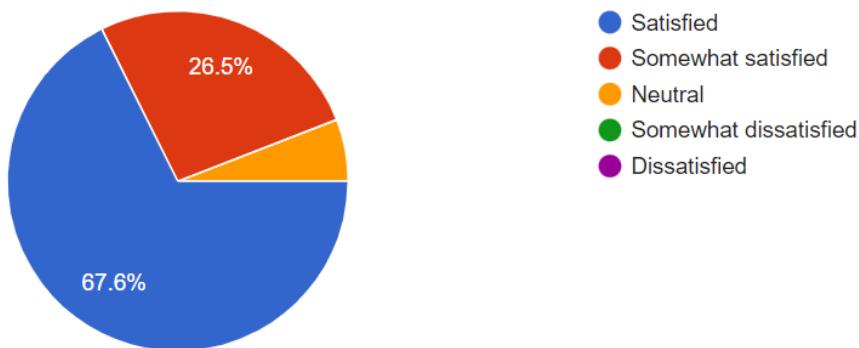


Figure 63: Accessing locations from maps

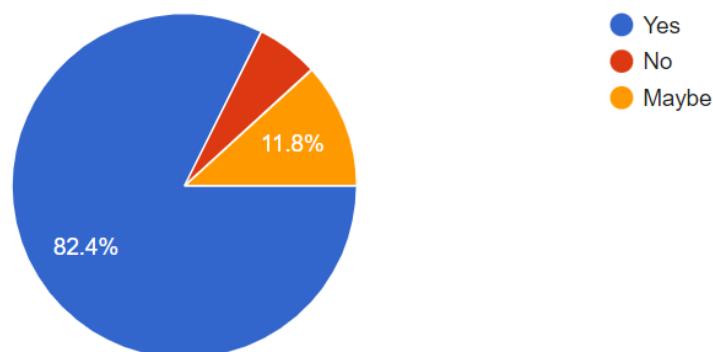
11. How satisfied are you with the SWMMbot application?

34 responses



12. Would you recommend the SWMMBot to a friend?

34 responses



13. How often do you think you can use the chatbot application during the rainy season?

34 responses

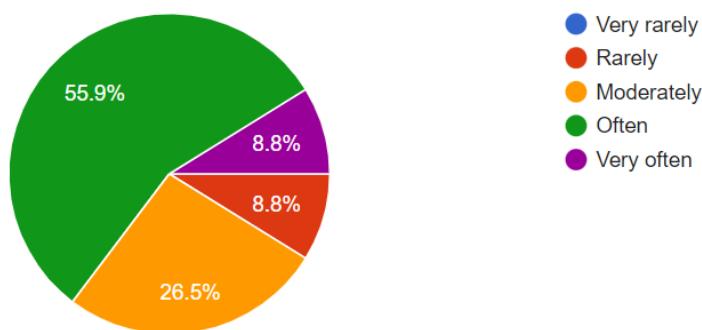


Figure 64: Satisfaction with the chatbot

From Figure 64, it can be observed that a majority (94.1%) of the users were satisfied with SWMMBot and 82.4% agreed that they would recommend the application to a friend. Finally on the frequency with which they could use the SWMMBot application, more than half of the respondents would use the chatbot often. While 8.8% said they would use the application very

rarely. From these responses, a majority of the respondents were satisfied with the chatbot and would refer a friend to use it. This shows the usefulness of the SWMMBot application, although a few aspects still need addressing.

The responses to question 14 on what the respondents liked the most about the chatbot application included: “that it is easy to use, user-friendly, easy to understand, has easy navigation, quick response, safety measures one should take and that it functions as an emergency response and it gives hope to situations and reduces the stress of disaster. Other users also liked the weather forecast widget, the pleasant style of conversation, and that users have the option of typing in or using a mic”. These were the highlights that the respondents liked.

The next question was ‘What features do you think should be added to the chatbot?’ and the participants stated that: “it should cover more areas; maybe provide the tips immediately when the location appears to be flooded; should be able to understand more commands; navigation suggestions; a graphic scheme with an example on how to use it; make it work better on phones, more about early warning, it should understand more answers, zonal flood map; automatically detect my location, information on other hazards.” These are linked to the responses for the final question below.

The final question was ‘What aspects of the SWMMBot application would you change?’ and the responses included: “it should generate the answers quicker; should be able to understand more commands; larger map, display the areas more affected, it just needs a wider coverage; use audio input to reach more people; reduction of the lengthy response; when it doesn't understand a question it gets confused and can't recover, include forecasted flooding, to show areal flooding, not at a point, and it could not understand some addresses.” These responses, together with the ones presented in the previous paragraph bring out some limitations of the SWMMBot which could be looked into, and possible modifications that could be incorporated in further research to produce a finer chatbot application.

From all the responses, it can be noted that generally a majority of the users found the application easy to use and would use it again often in the rainy season. There were however some users who faced some challenges while using the application. The conclusion would therefore be that while the chatbot is working well and providing the required answers, there are still some issues that need to be addressed such as why it is failing to notice areas outside the study area.

From the overall results presented in this chapter, it can be concluded that the objectives set out at the beginning of the thesis period to develop a prototype AI chatbot information support system for flood management were successfully achieved. As it is a prototype, showing proof of concept, more can still be done to improve some aspects of the SWMMBot to a point where it can be used by citizens for decision support during flooding.

This chapter presented the results of the study in the form of SWMMBot answering users' questions to address a particular need. From there, a discussion of the presented results was done and after this, the limitations of the study were presented. A display of the user interface was done and finally, pilot test results were presented showing the user's experiences as they used the SWMMBot. Areas that still need to be addressed were identified. The proceeding chapter outlines the conclusions of the research and some recommendations

# **Chapter 7      Conclusions and Recommendations**

---

This chapter concludes on the work done in developing the SWMMBot AI chatbot information system. The conclusion will be outlined first bringing out how the set objectives of the research and research questions were fulfilled. The processes involved are outlined briefly. After the conclusions, some recommended improvements to the chat application are presented together with areas for further research.

## **7.1 Conclusions**

This research aimed at contributing to efforts on mitigating the impacts of floods through increasing the level of preparedness of people living in flood-prone areas. The goal was to provide a solution to the lack of access to flood information through the development of a flood awareness information system. This would help in improving the level of awareness of the citizens which in turn improves their preparedness in case of flooding. With better access to flooding information, citizens can make more informed decisions, in time, which would reduce the impact of flood disasters. The developed SWMMBot AI chat application is a prototype that can therefore assist in providing this information in a fast and easily understandable manner i.e. through simple conversation. The chosen case study was Do Lo in Vietnam, which is a highly flood-prone area that would benefit greatly from such technological initiatives. This study area was therefore ideal for addressing the research objectives to develop the conversational chat application.

As outlined in the research, an AI chatbot information system was successfully developed to address possible questions users might have about the flooding situation at their location. To achieve this, firstly, since SWMM was to be used as a data source for the responses, it was therefore paramount that an analysis be done to assess which SWMM variables are important for flood analysis. After figuring these out, some intents were then developed to address questions users might pose to the chatbot about the flooding situation. For this, a ‘bank’ of numerous different terminologies was therefore created to cover the various ways users might ask different questions. Since it is impossible to have an exhaustive list of these phrases or terminologies, machine learning algorithms were adopted. These algorithms were studied to understand how they are applied in training models that understand natural language and can then match questions asked by users to the terminologies or ‘training phrases that were input into the Dialogflow platform. This platform, therefore, proved priceless as it allowed for the definition of training phrases together while also training these provided phrases using machine learning.

After developing the intents, responses matching these intents had to be provided. To obtain these responses, Python functions were successfully used to extract flooding information from SWMM simulation results. Now, having on one side, the intents and on the other, possible responses from SWMM, the challenge was then how to connect the two in order to have an integrated water resources information system. The Dialogflow platform came in handy as it allowed for the definition of some actions per intent. The actions were then linked to SWMM simulation results through a webhook. This then ensured the successful coupling of SWMM to the chatbot enabling the chatbot to respond to users' questions based on different flood-related parameters obtained from SWMM.

Finally, for users to be able to interact with the chatbot a graphical user interface was designed and integrated into the webhook to provide a complete functional chatbot application. The final developed chatbot SWMMBot is able to understand the user's questions and perform the necessary steps to respond to users' questions. So, the response to the question on whether an AI chatbot information system prototype was developed to respond to users' questions based on SWMM simulation results would be a definite 'Yes'. It is also worth mentioning that while the prototype chatbot used the EPA SWMM, other Urban drainage models like MOUSE, Sobek Urban, etc. can be used. This would allow for the application of this chatbot technology in other areas.

After the successful development of the chatbot application, it had to be tested to establish if it was indeed providing accurate results and to ascertain its limitations and strengths. This was achieved through a pilot test where users interacted with SWMMBot and expressed their experiences with the chatbot. While there were some challenges particularly with the chatbot occasionally missing the user's location, the consensus among users was that the chatbot performed well and provided accurate responses. The users reported general satisfaction with the application and were inclined to use and refer the chatbot to others. It is however still a work in progress and more improvements may still be done on it.

Looking at the research as a whole, it can thus be concluded that the AI chatbot technology can indeed be applied in water resources management and particularly for flood awareness creation through the development of AI chatbots that extract information from modelling systems. This chatbot technology has a great potential for integration with other modelling systems and also with decision support systems for easier and faster relaying of information in a conversational manner where the user also gets to ask questions. This then implies great potential for application in Hydro-informatics

## 7.2 Recommendations

Every research, no matter how successful has some limitations, and this research was no exception. Limitations of the research are outlined in section 6.1.8. The limitations of any research bring out opportunities for further research and improvements. The developed chatbot can therefore benefit from further developments in AI which could make it smarter so that it

can provide improved responses to the users. Recommended improvements to the chatbot application and possible areas of further research are be discussed in this section:

### **Dynamically Change Parameters in the SWMM Model.**

Firstly, while the developed chatbot managed to simulate SWMM, in addition to this, the simulation could be run for varying rainfall input as requested by the user. This would increase the functionality of the chatbot application allowing the user to assess the impacts of different rainfall quantities on the flooding in their location. The ability to dynamically change parameters would also be beneficial for water managers who, after having set up their model, may want to alter a certain parameter and then run the simulation. This advancement would ensure a faster and easier generation of results. Furthermore, this improvement could also be useful in the model calibration process where different parameters could be changed remotely.

### **Incorporate Rainfall Forecast Data**

Since the SWMMBot prototype has demonstrated all the necessary connections and integration necessary for dynamic model simulations using data for one rainfall event, another improvement to the chatbot could be to use rainfall forecast data. This will ensure that users can receive real-time simulations based on rainfall forecast data. Since the goal is to increase flood preparedness, the provision of real-time simulations will be priceless to users as it will allow them to respond timely in reaction to the anticipated flood information that they will be receiving from the chatbot application.

### **Connect to a 2D Model to Show the Actual Spatial Distribution of the Flood.**

As highlighted in section 5.2.3 the SWMM model used in the research was 1D and as such the flooded area in the simulation had to be fixed, this is of course not a true depiction of the real situation. It could therefore be more ideal to try using a 2D model. Since the simulation time for such models is fairly long the model could be run just once and then results can then be extracted from the simulation result files. This will indicate the spatial extent of the flood, therefore, giving users a clearer picture of the impacts of the flood.

Together with this, while the EPA SWMM was used in this research the application can be integrated with any other Urban drainage model. Further research could integrate the chatbot with other urban drainage models and assess if comparable results will be obtained.

### **Flood Navigation System**

One of the scenarios described in the results showed the possible use of SWMMBot for navigational purposes. This idea could be improved on by automating this process and coming up with a navigational system that automatically detects the best route considering the non-

flooded or less flooded locations. Such a technology would significantly lessen the risk faced by Rescue workers who often have to navigate through flooded roads to rescue citizens.

## Provide Time Series Results

Although the chatbot could accurately provide answers on the flooding situation it would have been ideal to provide the responses per given time and not only the maximum expected. The Pyswmm library allows for stepping through the simulation and thus responses could be sent back to the user at a given time interval showing the changes in the flooding situation per given time frame.

## Display of Responses as Graphic Content

While text responses do serve the required purpose, displaying the responses in graphic form (pictures, graphs, video clips, etc.) would make it easier for users to understand. Pictorial representations of areas that have a higher risk of flooding are more likely to compel people to take the necessary precautionary action and avoid the disastrous impacts of floods.

This chapter concluded on the work done in the entire research. It reported on the steps taken in developing the chatbot from the definition of intents to the integration with SWMM and the webpage through the webhook. The developed SWMMBot chat application can understand the user's questions and provide responses to the questions. When deployed for testing users reported that the chatbot was easy to use and that it provided accurate responses. The only exception was the identification of locations outside the study area where some of the respondents reported non-satisfactory responses from the chatbot in some cases. The chapter then discussed recommendations for possible improvements and some ideas on further research areas that can be explored. It was thus concluded that it is possible to develop an AI chatbot information system for flood risk management that extracts flooding information from an EPA SWMM. This technology still has potential for application in many areas of water resources management.

# References

- Abebe AJ, Price RK (2005) Decision support system for urban flood management. *Journal of Hydroinformatics* 7: 3-15 DOI 10.2166/hydro.2005.0002
- Agarwal S, Kumar S (2019) Applicability of SWMM for semi urban catchment flood modeling using extreme rainfall events. *Int J Recent Technol Eng* 8: 245-251
- Ahamed SMF, Agarwal S (2019) Urban flood modeling and management using SWMM for new RR Pet region, Vijayawada, India. *International Conference on Advances in Civil Engineering (ICACE-2019)*, pp. 23.
- Ahmad S, Simonovic SP (2006) An Intelligent Decision Support System for Management of Floods. *Water Resources Management* 20: 391-410 DOI 10.1007/s11269-006-0326-3
- Al Rasyid MUH, Sukardhoto S, Dzulqornain MI, Rifa'i A (2020) Integration of IoT and chatbot for aquaculture with natural language processing. *Telkomnika (Telecommunication Comput Electron Control* 18: 640-648
- AoDREM (2020) 2020 Global Natural Disaster Assessment Report Academy of Disaster Reduction and Emergency Management, Ministry of Emergency Management - Ministry of Education, National Disaster Reduction Center of China, Ministry of Emergency Management, International Federation of Red Cross and Red Crescent Societies
- Boné J, Ferreira JC, Ribeiro R, Cadete G (2020) DisBot: A Portuguese Disaster Support Dynamic Knowledge Chatbot. *Applied Sciences* 10 DOI 10.3390/app10249082
- Cahn J (2017) Chatbot: Architecture, Design and Development. Senior Thesis University of Pennsylvania
- Cembrano G (2004) Optimal control of urban drainage systems. A case study. *Control Engineering Practice* 12: 1-9 DOI 10.1016/s0967-0661(02)00280-0
- Chen SH, Jakeman AJ, Norton JP (2008) Artificial Intelligence techniques: An introduction to their use for modelling environmental systems. *Mathematics and Computers in Simulation* 78: 379-400 DOI 10.1016/j.matcom.2008.01.028
- Cheng Y, Jiang H (2021) Customer–brand relationship in the era of artificial intelligence: understanding the role of chatbot marketing efforts. *Journal of Product & Brand Management*
- Copperwaite M, Leifer C (2015) Learning Flask Framework Packt Publishing Ltd
- Dahiya M (2017) A tool of conversation: Chatbot. *International Journal of Computer Sciences and Engineering* 5: 158-161

Deveci Topal A, Dilek Eren C, Kolburan Geçer A (2021) Chatbot application in a 5th grade science course. Education and Information Technologies 26: 6241-6265 DOI 10.1007/s10639-021-10627-8

Divya S, Indumathi V, Ishwarya S, Priyasankari M, Devi SK (2018) A self-diagnosis medical chatbot using artificial intelligence. Journal of Web Development and Web Designing 3: 1-7

Do H (2021) Do Lo, Yen Nghia, Ha Dong, Ha Noi Study Area Description. In: Ncube S (ed), Study Area Description (Do Lo, Yen Nghia, Ha Dong, Ha Noi) edn, Not published.

Eckart K, McPhee Z, Bolisetti T (2018) Multiobjective optimization of low impact development stormwater controls. Journal of Hydrology 562: 564-576 DOI 10.1016/j.jhydrol.2018.04.068

Ellis B, Viavattene C, Revitt M, Peters C, Seiker H (2009) A Modelling Approach to Support the Management of Flood and Pollution Risks for Extreme Events in Urban Stormwater Drainage Systems

Kleppen E (2022). How to Practice Word2Vec for NLP Using Python (2022) <https://builtin.com/machine-learning/nlp-word2vec-python>. Cited 10 April 2022 2022

Ewing G, Demir I (2021) An ethical decision-making framework with serious gaming: a smart water case study on flooding. Journal of Hydroinformatics 23: 466-482 DOI 10.2166/hydro.2021.097

Gao X, Yang Z, Han D, Huang G, Zhu Q (2020) A Framework for Automatic Calibration of SWMM Considering Input Uncertainty. Hydrology and Earth System Sciences Discussions: 1-25

García L, Barreiro-Gomez J, Escobar E, Téllez D, Quijano N, Ocampo-Martinez C (2015) Modeling and real-time control of urban drainage systems: A review. Advances in Water Resources 85: 120-132 DOI 10.1016/j.advwatres.2015.08.007

Geitgey A (2018). "Natural Language Processing is Fun!" ( 2018) <https://medium.com/@ageitgey/natural-language-processing-is-fun-9a0bff37854e>. Cited 17 April 2022

Ghosh SGD (2019) Natural Language Processing Fundamentals : Build Intelligent Applications That Can Interpret the Human Language to Deliver Impactful Results Packt Publishing Ltd, Birmingham

Gironás J, Roesner LA, Rossman LA, Davis J (2010) A new applications manual for the Storm Water Management Model (SWMM). Environmental Modelling & Software 25: 813-814 DOI 10.1016/j.envsoft.2009.11.009

Google (2021) Dialogflow Documentation (2021) <https://cloud.google.com/dialogflow/docs>. Cited 25 October 2021

Google (2022) Connect to Cloud SQL for MySQL from Cloud Shell (2022a) <https://cloud.google.com/sql/docs/mysql/connect-instance-cloud-shell>. Cited 13 March 2022

Google (2022) Dialogflow Documentation: Webhook Service (2022b)  
[https://cloud.google.com/dialogflow/es/docs/fulfillment-webhook#webhook\\_response](https://cloud.google.com/dialogflow/es/docs/fulfillment-webhook#webhook_response). Cited 18 April 2022

Google( 2022) Google Documentation: Python 3 Runtime Environment (2022c)  
<https://cloud.google.com/appengine/docs/standard/python3/runtime#filesystem>. Cited 18 April 2022

Groen ET, Jacobs C (2012) Risk Mapping Vietnam:Sector Disaster Risk Reduction & Emergency Aid Cordaid.

IDMC (2019) Global Report on Internal Displacement Internal Displacement Monitoring Centre.

IPCC (2021) Summary for Policymakers. In: Climate Change 2021: The Physical Science Basis. Contribution of Working Group I to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change. In: [Masson-Delmotte V, P. Zhai, A. Pirani, S. L. Connors, C. Péan, S. Berger, N. Caud, Y. Chen, L. Goldfarb, M. I. Gomis, M. Huang, K. Leitzell, E. Lonnoy, J.B.R. Matthews, T. K. Maycock, T. Waterfield, O. Yelekçi, R. Yu and B. Zhou (eds.)] (ed), pp. 42.

Janssen A, Grützner L, Breitner MH (2021) Why do Chatbots fail? A Critical Success Factors Analysis

Jiang L, Chen Y, Wang H (2015) Urban flood simulation based on the SWMM model. Proc IAHS 368: 186-191 DOI 10.5194/piahs-368-186-2015

Jung D, Tran Tuan V, Quoc Tran D, Park M, Park S (2020) Conceptual Framework of an Intelligent Decision Support System for Smart City Disaster Management. Applied Sciences 10: 666

Kavitha B, Murthy CR (2019) Chatbot for healthcare system using Artificial Intelligence. Int J Adv Res Ideas Innov Technol 5: 1304-1307

Kocornik-Mina A, McDermott TKJ, Michaels G, Rauch F (2020) Flooded Cities. American Economic Journal: Applied Economics 12: 35-66 DOI 10.1257/app.20170066

Li J, Burian S, Oroza C (2019) Exploring the potential for simulating system-level controlled smart stormwater systemWorld Environmental and Water Resources Congress 2019: Water, Wastewater, and Stormwater; Urban Water Resources; and Municipal Water Infrastructure American Society of Civil Engineers Reston, VA, pp. 46-56.

Liddy ED (2001) Natural language processing

Luu C, Tran HX, Pham BT, Al-Ansari N, Tran TQ, Duong NQ, Dao NH, Nguyen LP, Nguyen HD, Thu Ta H, Le HV, Meding Jv (2020) Framework of Spatial Flood Risk Assessment for a Case Study in Quang Binh Province, Vietnam. Sustainability 12: 3058

Ma B, Wu Z, Hu C, Wang H, Xu H, Yan D, Soomro S-e-h (2022) Process-oriented SWMM real-time correction and urban flood dynamic simulation. Journal of Hydrology 605: 127269 DOI <https://doi.org/10.1016/j.jhydrol.2021.127269>

Mattos TS, Oliveira PTS, Bruno LdS, Carvalho GA, Pereira RB, Crivellaro LL, Lucas MC, Roy T (2022) Towards reducing flood risk disasters in a tropical urban basin by the development of flood alert web application. Environmental Modelling & Software 151: 105367 DOI <https://doi.org/10.1016/j.envsoft.2022.105367>

Mikolov T, Chen K, Corrado G, Dean J (2013) Efficient estimation of word representations in vector space. arXiv preprint arXiv:13013781

Mostaco GM, De Souza IRC, Campos LB, Cugnasca CE (2018) AgronomoBot: a smart answering Chatbot applied to agricultural sensor networks14th international conference on precision agriculture, pp. 1-13.

Mufid MR, Basofi A, Rasyid MUHA, Rochimansyah IF, rokhim A (2019) Design an MVC Model using Python for Flask Framework Development2019 International Electronics Symposium (IES), pp. 214-219.

Najibi N, Devineni N (2017) Recent Trends in Frequency and Duration of Global Floods. Earth Syst Dynam: 1-40 DOI 10.5194/esd-2017-59

Nguyen MT, Sebesvari Z, Souvignet M, Bachofer F, Braun A, Garschagen M, Schinkel U, Yang LE, Nguyen LHK, Hochschild V, Assmann A, Hagenlocher M (2021) Understanding and assessing flood risk in Vietnam: Current status, persisting gaps, and future directions. Journal of Flood Risk Management 14 DOI 10.1111/jfr3.12689

Nivedhitha G, Punarselvam E, Aaghash KR, Elayabarathi M, Rahul K, Santhosh R (2021) Ai Consulting Healthcare Chatbot System Using Pattern Matching. International Journal of Scientific Research in Science and Technology: 18-22 DOI 10.32628/ijsrst2182112

Nurdiana A, Wijaya E (nd) Onilne Booking Chatbot Application for Blind People Using Google Dialogflow API and Online Store API Based Android

Okon AE (2021) Exploring the Use of an Artificial Intelligence Chatbot Mobile Application for Providing Early Flood Awareness. Master of Science Thesis, HE Delft Institute for Water Education

Oracle (2022) MySQL Documentation: Reference Manual (2022)  
<https://dev.mysql.com/doc/refman/5.7/en/>. Cited 13 March 2022

Owusu S, Wright G, Arthur S (2015) Public attitudes towards flooding and property-level flood protection measures. Natural Hazards 77: 1963-1978 DOI 10.1007/s11069-015-1686-x

Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay É (2011) Scikit-learn: Machine Learning in Python. J Mach Learn Res 12: 2825–2830

Pilehvar MT, Camacho-Collados J (2020) Embeddings in Natural Language Processing: Theory and Advances in Vector Representations of Meaning Morgan & Claypool

Przegalinska A, Ciechanowski L, Stroz A, Gloor P, Mazurek G (2019) In bot we trust: A new methodology of chatbot performance measures. *Business Horizons* 62: 785-797 DOI 10.1016/j.bushor.2019.08.005

Quah JT, Chua Y (2019) Chatbot assisted marketing in financial service industryInternational Conference on Services Computing Springer, pp. 107-114.

Ranavare SS, Kamath RS (2020) Artificial Intelligence based Chatbot for Placement Activity at College Using DialogFlow. *Our Heritage* Vol 68: 4806-4814

Raturi D (2020) Full video -Dialogflow chatbot using Python | MongoDB | Flask | RapidAPI | Telegram | Pivotal Cloud, YouTube.

Reyes R, Garza D, Garrido L, De la Cueva V, Ramirez J (2019) Methodology for the Implementation of Virtual Assistants for Education Using Google DialogflowAdvances in Soft Computing, Lecture Notes in Computer Science:440-451.

Riaño-Briceño G, Barreiro-Gomez J, Ramirez-Jaime A, Quijano N, Ocampo-Martinez C (2016) MatSWMM – An open-source toolbox for designing real-time control of urban drainage systems. *Environmental Modelling & Software* 83: 143-154 DOI 10.1016/j.envsoft.2016.05.009

Flood (2022) <https://www.risicokaart.nl/en/what-risks-are-there/flood> Cited 29 March 2022

Rossman LA (2015) Storm Water Management Model User's Manual Version 5.1. EPA-600/R-14/413b. In: National Risk Management Research Laboratory. Environmental Protection Agency (ed) U.S. Environmental Protection Agency, United States Cincinnati, Ohio.

Rossman LA, Huber W (2017) Storm water management model reference manual volume II–hydraulics. US Environmental Protection Agency: Washington, DC, USA 2: 190

Sadler JM, Goodall JL, Behl M, Morsy MM (2018) Leveraging open source software and parallel computing for model predictive control simulation of urban drainage systems using EPA-SWMM5 and PythonInternational Conference on Urban Drainage Modelling Springer, pp. 988-992.

Salinas-Rodriguez C, Gersonius B, Zevenbergen C, Serrano D, Ashley R (2018) A semi risk-based approach for managing urban drainage systems under extreme rainfall. *Water* 10: 384

Schmitt TG, Thomas M, Ettrich N (2004) Analysis and modeling of flooding in urban drainage systems. *Journal of Hydrology* 299: 300-311 DOI 10.1016/j.jhydrol.2004.08.012

William Scott (2019) TF-IDF From Scratch in Python on a Real-world Datatset (2019) <https://towardsdatascience.com/tf-idf-for-document-ranking-from-scratch-in-python-on-real-world-dataset-796d339a4089>. Cited 5 April 2022

Setiawan A, Mas YSIGS, Nisa DC, Yuri MIFNM, Hilda SDY (2020) Design and Development of Chatbot Using Dialog Flowin Surya Sembada PDAM Surabaya

CityInternational Joint Conference on Science and Engineering (IJCSE 2020) Atlantis Press, pp. 114-118.

Shukla (2019) Introduction to Dialogflow (2019) <https://medium.com/@sid8491/introduction-to-dialogflow-da6b9cf953b9>. Cited 7 April 2022 2022

Shum H-Y, He X, Li D (2018) From ELIZA to Xiaolce: Challenges and Opportunities with Social Chatbots. CoRR abs/1801.01957

Sörensen J, Persson A, Sternudd C, Aspegren H, Nilsson J, Nordström J, Jönsson K, Mottaghi M, Becker P, Pilesjö P, Larsson R, Berndtsson R, Mobini S (2016) Re-Thinking Urban Flood Management—Time for a Regime Shift. Water 8 DOI 10.3390/w8080332

Tsai M-H, Chen J, Kang S-C (2019) Ask Diana: A Keyword-Based Chatbot System for Water-Related Disaster Management. Water 11 DOI 10.3390/w11020234

w3schools (2022) HTML Tutorial (2022) <https://www.w3schools.com/html/>. Cited 14 April 2022

# Appendices

## Appendix A. - Research Ethics Declaration Form



**Research Ethics Committee**  
**IHE Delft Institute for**  
**Water Education**  
E [ResearchEthicsCommittee@un-ihe.org](mailto:ResearchEthicsCommittee@un-ihe.org)

Date: 2022-03-02  
To: Sifiso Patience Ncube  
MSc Programme: HI  
Approval Number: IHE-RECO 2021-240

Subject: Research Ethics approval

Dear Sifiso Patience Ncube

Based on your application for Ethical Approval, the Research Ethics Committee (RECO), IHE Delft RECO gives ethical clearance for your research topic An AI Chatbot Information Support System for Water Resources Management

This approval is valid until April 19, 2022. Please notify the RECO if your research protocol is modified in any way. If you do not complete your research by the specified date, contact RECO to request an extension for the ethical clearance.

Please keep this letter for your records and include a copy in the final version of your MSc. thesis, together with your personal ethics reflection.

On behalf of the Research Ethics Committee, I wish you success in the completion of your research.

Yours sincerely,

Dr. Angeles Mendoza Sammet  
Coordinator, Research Ethics Committee IHE Delft

## **Research Ethics- Personal Statement**

In conducting the research all ethical guidelines were adhered to maintaining transparency, and honesty throughout the research. The research idea was generated by Dr. G. Corzo who was my mentor guiding me during the research period. Other ideas were obtained from my Supervisor Prof. D. Solomatine and my other mentors Mr. S. Duarte and Ms. H. Do. The research focus is on developing a prototype conversational application (chatbot) that extracts flooding information from SWMM for flood awareness in an area that experiences floods annually. Since the developed chatbot is a prototype, there are still some improvements needed before any deployment for use by the general citizens is possible and therefore the researcher is not responsible for the use of the technology at its current stage of development.

Literature from various authors was used to understand the different concepts relevant to the research and all authors were acknowledged. Most of the information for the study area was obtained from Ms. H. Do based on her Ph.D. research in the same study area. This also includes the complete EPA SWMM model used in this study. The research also used Python codes, some of which were developed by myself while others were modified from Open-Source codes available for use without restriction. Furthermore, to understand concepts on chatbots, the Google Dialogflow platform, the Flask web framework, and other technologies that were new to me YouTube channels such as “Tech with Sach”, “FreeCodeCamp.org”, “TheMarcusProjects”, Danesh Raturi and “Deconstructing Chatbots” with Priyanka Vergadia among others were used.

The developed chatbot was called SWMMBot since it was extracting data from the SWMM model. Other names considered like ‘FloodBot’, and ‘FloodChat’ were already being used by other applications and so the name SWMMBot was used. The application can however be integrated with any other urban drainage model and produce comparable results. The research had its limitations and these were outlined in Chapter 6 of the research.

All methodologies used in the research are well explained in Chapter 5 so that any future advances on this work may have a clear foundation to improve on. Flow charts were also used for easier comprehension of the flow processes. The results obtained are also presented using practical scenarios encountered in real life where the SWMMBot may be applied. This helps in understanding the functionalities of the chatbot. Furthermore, all assumptions made in the research were stated. The research output was not manipulated but presented truthfully. No funding was received for this research and thus no preferences were made on the methods used. The selection of the methods was based entirely on my understanding of the method. The report was presented using standard academic writing in the English language following the ethical guides provided by the Dutch government.

## Appendix B. - Vectorization Output

### TF-IDF

```
[[0.          0.          0.57769148 0.          0.          0.
  0.38688672 0.          0.          0.46607785 0.          0.
  0.          0.          0.          0.38688672 0.          0.
  0.38688672 0.          ]]
[[0.          0.          0.          0.          0.57769148 0.
  0.38688672 0.          0.          0.46607785 0.          0.
  0.          0.          0.          0.38688672 0.          0.
  0.38688672 0.          ]]
[[0.          0.37796447 0.          0.          0.          0.37796447
  0.          0.37796447 0.37796447 0.          0.37796447 0.
  0.37796447 0.          0.          0.          0.          0.
  0.          0.37796447]
[[0.          0.          0.          0.          0.          0.
  0.31403664 0.          0.          0.          0.          0.46891321
  0.          0.          0.46891321 0.31403664 0.37831623 0.46891321
  0.          0.          ]
[[0.49389914 0.          0.          0.49389914 0.          0.
  0.          0.          0.          0.          0.          0.
  0.          0.49389914 0.          0.          0.39847472 0.
  0.33077001 0.          ]]]
```

### Word2Vec

```
Word2Vec(vocab=22, vector_size=5, alpha=0.025)
22
[[-0.038489  0.01697034  0.36630523  0.6466623 -0.66774166]
 [-0.48855633 0.44339007  0.6159798 -0.34430027 -0.25834888]
 [ 0.6150826 -0.12779781 -0.37807617  0.54620695 -0.4050401 ]
 [-0.13908541 0.2203119   0.07596576 -0.6345492 -0.72366744]
 [ 0.56778425 0.39369524  0.5247549  0.05921042  0.49371186]
 [-0.31360865 -0.0870167  0.53157   -0.6930861 -0.36213627]
 [-0.52173674 -0.06459854  0.6624952 -0.508372 -0.16209812]
 [-0.17209908 0.7173912  -0.52674794  0.00401096 -0.4221993 ]
 [-0.6574894  0.34299585 -0.59966785 -0.3007598 -0.00190357]
 [-0.01832281 -0.4739489  0.5947988  0.30820602  0.5711919 ]
 [-0.6135584  0.33812967 -0.31115025  0.06201341  0.6391825 ]
 [-0.32604063 0.33008292 -0.495907  -0.25927955  0.68672645]
 [-0.17525713 0.03570026 -0.4599709 -0.8534483 -0.16752052]
 [ 0.3487878 -0.12540856  0.7814713 -0.387367  0.3191695 ]
 [ 0.38065428 0.5823024 -0.10142842 -0.642458  0.30493617]
 [ 0.0483131 0.62943316 -0.06885333 -0.2232388 -0.73952913]
 [-0.07811416 0.2577513  0.49293217  0.6434481 -0.5200528 ]
 [ 0.16980009 0.5570405 -0.4385344 -0.28434208  0.6226643 ]
 [ 0.15738432 0.0188236  0.33495095  0.02072453  0.9285764 ]
 [ 0.36111233 -0.63632333 -0.5024687  0.0643257  0.45615566]
 [-0.6043939 0.25703368  0.36390346  0.40261227  0.5235641 ]
 [-0.55843437 0.10043473  0.54789054 -0.25732094 -0.5582704 ]]
```

## Appendix C. - Python Functions

### To find the closest node

```
204     def Find_nearest_node(node_name):
205         min_distance = 10000000000
206         min_distanode = None          # the node with the min distance
207         lat_user_input, lon_user_input = geocode(node_name)
208         with pool.connect() as con: # its a way to connect to the db faster.pool is so that we can have many connections to the database.
209             for i in nodes_dict:
210                 node = nodes_dict[i]
211
212                 distances = (con.execute(text("SELECT ST_Distance_Sphere(point(:lon, :lat),point(:node_lon,:node_lat ))"), __ # lat n lon ar
213                                         lon=lon_user_input, lat=lat_user_input, node_lon=node.get("lon"), node_lat=node.get("lat")))
214
215                 for row in distances:
216                     distance = row[0] # give the distance between the user input and the selected node
217                     if distance < min_distance:
218                         min_distance = distance
219                         min_distanode = node      #it will go through all the nodes checking the min distances and retains the node with the min
220
221     return min_distanode.get("name")
```

### To Extract Information from SWMM

```
def Ex_NodeFloodHrs(node_name):          #fn extracts number of hours a node is flooded.
    nodes = []
    flooded_hrs = []
    for num in range(758, 869):
        filteredList = []
        result = linecache.getline("swmm.rpt", num)
        list = result.split(" ")
        for element in list:
            if len(element) > 0:
                filteredList.append(element) # append every string with greater than 0
        nodes.append(filteredList[0])
        flooded_hrs.append(float(filteredList[1]))
    index = nodes.index(node_name)           #the location of the node_name
    return flooded_hrs[index]
```

```
def Ex_Time_maxflood():
    nodes= []
    T_maxflood = []
    for num in range(758, 869):
        filteredList = []
        result = linecache.getline("swmm.rpt", num)
        list = result.split(" ")
        for element in list:
            if len(element) > 0:
                filteredList.append(element) # append every string with greater than 0 characters this eliminates the empty spaces
        nodes.append(filteredList[0])
        T_maxflood.append([filteredList[3],filteredList[4]])

    return(nodes, T_maxflood) #max_floodrate,T_maxflood,max_ponded_depth)

nodes, T_maxflood = Ex_Time_maxflood()
x = PrettyTable()
x.add_column("Node", nodes)
x.add_column("Day and Time of Max Flooding", T_maxflood)
print(x)
```



## Appendix D. - Default Fall Back Intent Responses

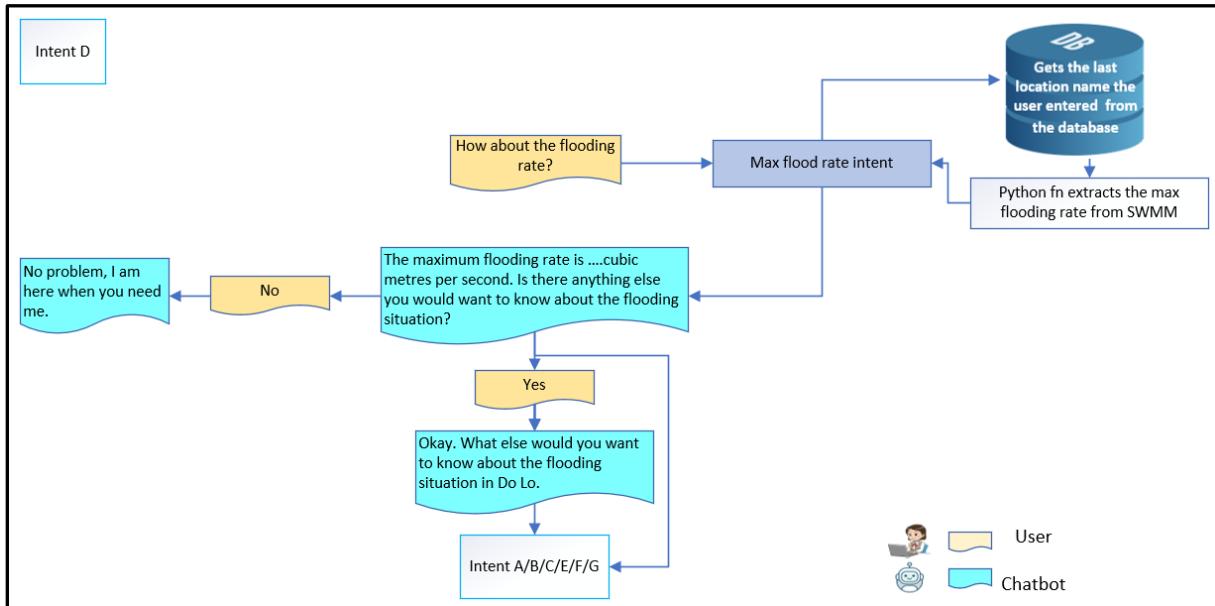
### Default Fallback Intent

SAVE

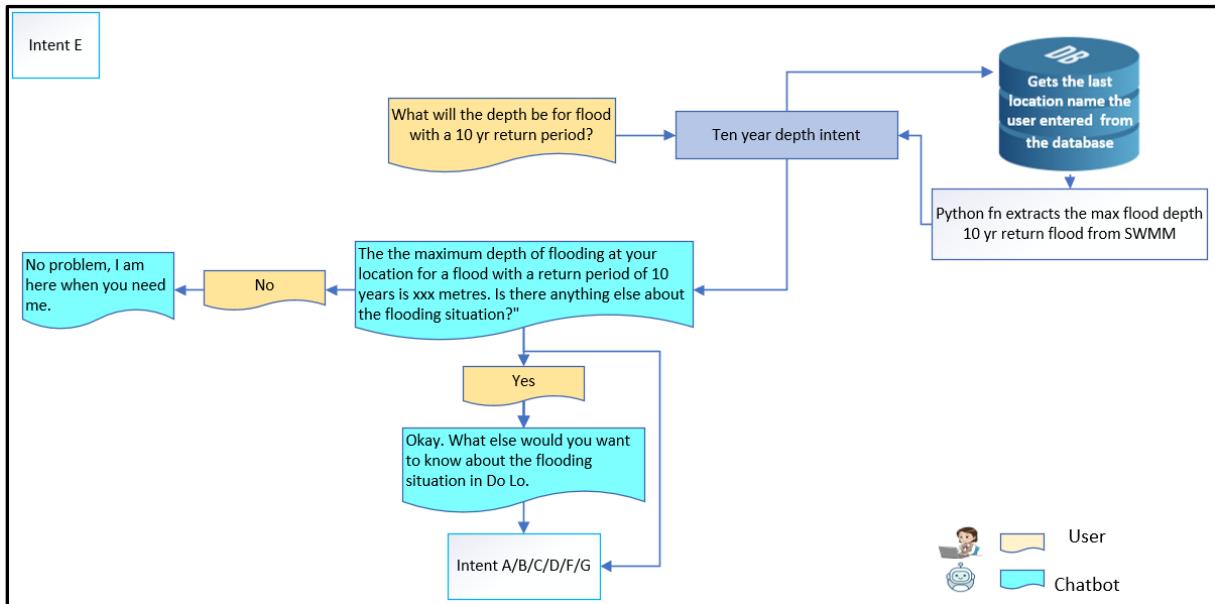
1	Sorry, I didn't get that. Can you say it again?
2	This is embarrassing, I missed what you said. May you kindly rephrase?
3	Goodness me, I missed your question, could you say that again?
4	Sorry, can you say that again?
5	Could you please say that again? Or let's start from the beginning, please type "Hi"
6	Sorry, I didn't get that. Can we please start from the beginning? Please type "Hi"
7	Sorry, what was that? Could you please rephrase?
8	This is embarrassing, could you say that one more time?
9	I did not get that? Could you please rephrase.
10	Could you please say that one more time?
11	I didn't get that. Can you please repeat?
12	I missed that, please say that again?
13	Oh, my! I seem to have missed your question completely, can we please start from the beginning? Please type "Hi"
14	I seem to have not understood your question well, can we please start from the beginning? Please type "Hi"
15	Goodness me! I seem to have misunderstood your question, could you kindly rephrase it? Otherwise, let's start from the beginning? Please type "Hi"

## Appendix E. - The Other Developed Intents

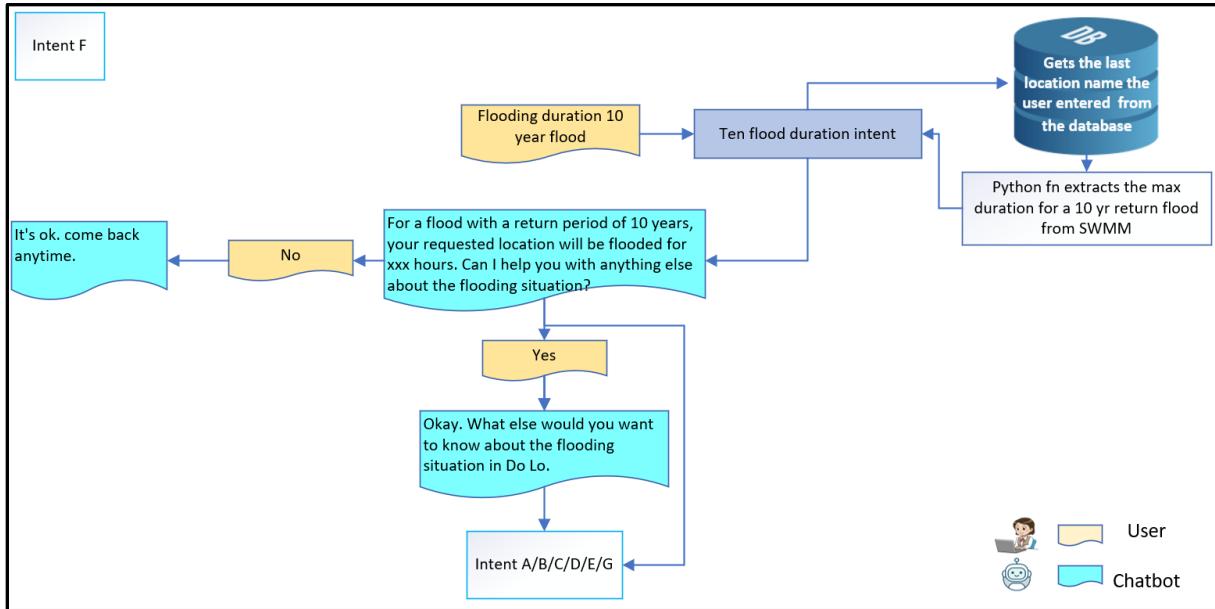
Maximum flooding rate



Depth for a flood with a 10 year return period



## Duration for a flood with a 10-year return



## Appendix F. - EPA SWMM

### Simulation Report

Topic: Node Flooding Click a column header to sort the column.

Node	Hours Flooded	Maximum Rate CMS	Day of Maximum Flooding	Hour of Maximum Flooding	Total Flood Volume 10^6 ltr	Maximum Ponded Depth Meters
N10-1	4.01	0.025	1	18:40	0.037	0.245
N10-17-1	1.57	0.026	1	18:40	0.018	0.105
N10-31-1	0.04	0.004	1	18:45	0.000	0.001
N11-1	10.15	0.030	1	18:39	0.102	1.071
N11-14-1	11.20	0.024	1	17:28	0.089	1.078
N11-19-1	12.41	0.022	1	17:27	0.080	0.993
N11-26-1	9.56	0.023	1	18:37	0.061	0.587

Node flooding (2020)

Summary Results

Topic: Node Flooding Click a column header to sort the column.

Node	Hours Flooded	Maximum Rate CMS	Day of Maximum Flooding	Hour of Maximum Flooding	Total Flood Volume 10^6 ltr	Maximum Ponded Depth Meters
N10-1	7.97	0.071	1	18:40	0.216	1.267
N10-17-1	7.28	0.102	1	18:40	0.273	1.120
N10-31-1	6.07	0.101	1	18:39	0.210	0.855
N11-1	14.76	0.072	1	18:39	0.329	3.543
N11-14-1	15.27	0.061	1	18:39	0.285	3.503
N11-19-1	15.69	0.059	1	18:39	0.255	3.158
N11-26-1	14.23	0.072	1	18:39	0.256	2.423

Node flooding 10yr return flood

**Function for extracting the maximum depth of the flood**

```

def Ex_NodePonded_depth(node_name):
    nodes= []
    max_ponded_depth = []
    for num in range(758, 869):
        filteredList = []
        result = linecache.getline("swmm.rpt", num)
        list = result.split(" ")
        for element in list:
            if len(element) > 0:
                filteredList.append(element) # append eve
        nodes.append(filteredList[0])
        max_ponded_depth.append(float(filteredList[6]))
    index = nodes.index(node_name) #locat
    return max_ponded_depth[index]

```

### Function for extracting hours that the nodes are flooded

```

def Ex_NodeFloodHrs(node_name):      #fn extracts number
    nodes = []
    flooded_hrs = []
    for num in range(758, 869):
        filteredList = []
        result = linecache.getline("swmm.rpt", num)
        list = result.split(" ")
        for element in list:
            if len(element) > 0:
                filteredList.append(element) # append eve
        nodes.append(filteredList[0])
        flooded_hrs.append(float(filteredList[1]))
    index = nodes.index(node_name) #the locat
    return flooded_hrs[index]

```

## Appendix G. - Python Script to Trigger Actions on the Webhook

```
node_name = node_name.capitalize()
# Flooding state 1 location
if action == 'node.location': # checking if the action key matches node.location
    if FloodedLoc(node_name) == True:
        fulfillmentText = "Your location " + params.get('Location_name') + " is flooded. What else would you like to know about the flooding si"
    else :
        fulfillmentText = "Your location " + params.get('Location_name') + " is not flooded. If you're interested in getting information for an"
    
#Max flood depth(level)
elif action =='maxponded.depth':
    if FloodedLoc(node_name) == True:
        depth = Ex_NodePonded_depth(node_name)
        fulfillmentText = "The maximum depth of flooding at your location " + " is " + str(depth) + " metres. Remember not to walk or drive in f"
    else :
        fulfillmentText = "The requested location " + params.get('Location_name') + " is not flooded."
    
## 10 yr return Max flood depth(level)
elif action == 'teny.depth':
    if FloodedLoc10(node_name) == True:
        depth10 = Ex_NodePonded_depth10(node_name)
        fulfillmentText = "The the maximum depth of flooding at your location for a flood with a return period of 10 years is " + str(depth10)
    else:
        fulfillmentText = "The requested location will not be flooded."
    
# flood duration
elif action == 'flood.duration':
    if FloodedLoc(node_name):
        hours = Ex_NodeFloodHrs(node_name)
        fulfillmentText = "Your requested location will be flooded for " + str(hours) + " hours. Is there anything else you would like to know a"
    else :
        fulfillmentText = "Your location is not flooded."
```

## Appendix H. - Sample HTML and CSS

### HTML

```
<html lang= "en">
<head>
    <title>Chatbot Demo</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link href="style3.css" rel="stylesheet" />
    <link href="https://fonts.googleapis.com/css?family=Dosis&display=swap" rel="stylesheet">

    
    <link href="https://fonts.googleapis.com/css2?family=Roboto&display=swap" rel="stylesheet">
    <link href="https://fonts.googleapis.com/css2?family=Kaushan+Script&display=swap" rel="stylesheet">
    <link href="https://fonts.googleapis.com/css2?family=Atma:wght@500&display=swap" rel="stylesheet">
    <script src="https://www.weatherwx.com/js/html5kellycolorpicker.min.js"></script>

</head>

<body>
<nav class="navbar">
    
    | <div class="navbar-text" > IHE Delft Institute for Water Education </div>
</nav >
<div class = "grid-container-main">

    <h1> SWMMBot App Demonstration</h1>
    <div id="section1">

        <div class = "grid-container">

            <div class="column-left">
                <h2><i> A chatbot that provides flood related information from a SWMM model </i></h2>
                <p >This chatbot is meant to provide information on the flooding situation at your location<br/>
                </p>
            </div>
        </div>
    </div>
</div>
```

### CSS

```
h2{
    text-align: center;
    font-family: Times New Roman,verdana,Georgia,Cambria;
    color: MediumBlue;
    font-weight: bold;
    font-size:150%;
    background-color: None;
    padding-top: 20px;
    /*padding-bottom: 20px; */
}

h3{
    text-align: center;
    font-family: Times New Roman,verdana,Georgia,Cambria;
    color: black;
    font-weight: bold;
    font-size:120%;
}

.navbar {
    background-color:MediumBlue;
    width: 100%
}

.navbar-text {
    text-align: right;
    font-weight:bold;
    color:white;
    font-size: 20px;
    vertical-align: top;
}

.grid-container-main
{
    margin-left: 100px;
    margin-right: 90px;
```

## Appendix I. - Questionnaire for the Pilot Test

The screenshot shows a mobile application interface. At the top, a blurred background image of a person's hands interacting with a device is overlaid with a teal bar containing the white text "What Can I Help You With?". Below this, a dark grey header bar contains the text "Section 1 of 3". The main content area has a light grey background and features a title "SWMMBot Pilot Test" in bold black font. To the right of the title are three small icons: a close button (X), a maximize/minimize button (double arrows), and a more options button (three dots). A vertical scroll bar is visible on the right side of the content area. Below the title, a paragraph of text describes the SWMMBot as an AI chatbot developed as a decision-support tool for flood awareness creation, based on an EPA SWMM model for Do Lo, Hanoi, Vietnam. The text is followed by a horizontal line and a blue vertical bar on the left. Under the heading "Scenario", there is a hand-drawn style illustration of a street grid. The grid shows several buildings, including a school building with a red roof and a small house. Two blue thought bubbles are present: one above a car showing a flooded street, and another above a person's head with a large red question mark. Two rain clouds with rain lines are positioned above the grid. Three location pins are placed on the grid at different locations.

## Scenario

Imagine it is the end of day and after a long tiring day at school or at work you just want to head home and rest. It is, however, that time of the year when the heavens are open and it has been raining cats and dogs for most of the day and streets are flooded. Also, on your way home, you have to pick your sibling/child/niece/nephew from school. The school just happens to be on the other side of the city. Naturally many question come to mind: Is it safe to walk/drive? Which parts of the city are flooded? For how long will an area be flooded? How high are the flood waters? If you wait will the water subside or will it increase? When will the flooding be maximum?...

The SWMMBot chatbot application was developed to answer such questions based on results from an EPA SWMM model. To help us assess the effectiveness of this chatbot technology we kindly request that you would access the chatbot app on this link: xxxxxx and then tell us your thoughts about it on this questionnaire. Your responses will be treated as strictly confidential and anonymous.

### 1. Gender \*

- Male
- Female
- Prefer not to say
- Other...

### 2. Please indicate your age \*

- 20 and below
- 21-40
- 41-60
- above 60

## Ease of Use and Accuracy

X ⋮

Description (optional)

3. The chatbot was easy to use. \*

1      2      3      4      5

Strongly disagree

Strongly agree

4. I could easily access all the information I needed. \*

1      2      3      4      5

Strongly disagree

Strongly agree

5. The chatbot gave accurate responses. \*

1      2      3      4      5

Strongly disagree

Strongly agree

6. The responses provided were easy to understand. \*

1      2      3      4      5

Strongly disagree

Strongly agree

7. I could easily access my location from the map. \*

1      2      3      4      5

Strongly disagree

Strongly agree

8. I could easily switch and get information about other locations. \*

- Strongly disagree
- Disagree
- Neutral
- Agree
- Strongly agree

9. The chatbot accurately recognized locations that were outside the study area. \*



10. The information obtained from the chatbot was relevant. \*



## User Satisfaction

Description (optional)

11. How satisfied are you with the SWMMbot application? \*

- Satisfied
- Somewhat satisfied
- Neutral
- Somewhat dissatisfied
- Dissatisfied

12. Would you recommend the SWMMBot to a friend? \*

- Yes
- No
- Maybe

13. How often do you think you can use the chatbot application during the rainy season? \*

- Very rarely
- Rarely
- Moderately
- Often
- Very often

14. What did you like the most about the chatbot application? \*

Long answer text

15. What features do you think should be added to the chatbot? \*

Long answer text

16. What aspects of the SWMMBot application would you change? \*

Long answer text

Thank you very much for your participation.

Description (optional)