

## **Virtual Doctor: A System based on Learning Analytics in Healthcare**

### **Abstract-**

Health is the most important aspect of our lives. However, with an increase in the population of India and a decrease in the birth rate. The expansion of medical developments and population has gone out of synergy with the population surpassing the latter making it difficult to obtain doctor consultation. To solve this problem, we propose an AI and NLP (Natural Language Processing) based automated conversational chatbot system which can act as a virtual healthcare expert for pregnant women, family with young kids and mothers that can advise patients about the basic details and information in real-time about the problems that they are facing and reduce the workload and costs of the Healthcare industry and save and impact 100 of lives. The information provided would improve accessibility educate the users on Health resulting in a behavioral change. We concluded that automated medical products may flourish to serve a bigger role in healthcare. In the future, the chatbot can be trained to assist all doctors in providing consultation and performing surgeries by integrating all the stakeholders in the value chain of the Healthcare industry.

Keywords: Learning Analytics, Chatbot, Deep Learning, Healthcare Informatics, Healthcare Communication, Pregnancy

### **Introduction-**

A chatbot is a computer program that conducts a conversation via auditory or textual methods. These programs are designed to provide a clone of how a human will chat and thereby it acts as a conversational partner rather than humans. For various practical purposes like customer service or information acquisition, the chatbot is being used in the dialog system. Most chatbots use natural language processing for interpreting the user input and generating the corresponding response but certain simpler systems search for the keyword within the text and then provide a reply based on the matching keywords or certain pattern. Today, chatbots are part of virtual assistants such as Google Assistant and are accessed via many organizations' apps, websites, and on instant messaging platforms. Non-assistant applications include chatbots used for entertainment purposes, for research, and social bots which promote a particular product, candidate, or issue.

Chatbots are such kinds of computer programs that interact with users using natural languages. For all kinds of chatbots, the flow is the same, though each chatbot is specific in its area of knowledge that is one input from the human is matched against the knowledge base of the chatbot. Chatbot's work basically on Artificial intelligence.

So, using this capability we have decided to add some contribution to Health Informatics.

The high cost of our healthcare system can often be attributed to the lack of patient engagement after they leave clinics or hospitals. Various surveys in this area have proved that that chatbot can provide healthcare in low costs and improved treatment if the doctors and the patient keep in touch after their consultation. To answer the questions of the user chatbot is used. There is a very a smaller number of chatbots in the medical field.

The proposed system provides a text-to-text conversational agent that asks the user about their health issue. The user can chat as if chatting with a human. The bot then asks the user a series of questions about their symptoms to diagnose the disease. It gives suggestions about the different symptoms to clarify the disease. Based on the reply from the user the accurate disease is found, and it suggests the

doctor who needs to be consulted in case of major disease. The system remembers past responses and asks progressively more specific questions to obtain a good diagnosis. The three primary components of our system are (1) user validation and extraction of symptoms from the conversation with the user, (2) accurate mapping of extracted (and potentially ambiguous) symptoms to documented symptoms and their corresponding codes in our database, and (3) developing a personalized diagnosis as well as referring the patient to an appropriate specialist if necessary.

A healthy body relates to a healthy mind. In today's generation, people have become busy chasing and running behind materialistic pleasures in turn forgetting about the most important necessity for their body. There is a major role of information in child and maternal health practices. And apart from these the most important fact is in the reliability and accuracy of the information that is provided as it can influence maternal health behaviors. The going-to-be mothers seek access to this information mainly through doctors. However, this is not the case with the villages that do not have the facilities, hospitals, and doctors to treat their illness or charge a hefty amount of fees that they are unable to afford. So, the proposed solution is a virtual healthcare assistant that can benefit everyone.

The to-be-mothers experience symptoms that may not require medical attention and they may experience symptoms that may require immediate medical attention. The need for proper communication with its proper delivery plays a crucial role to ensure proper health practices and save commuting costs. The maternal industry is flooded with patients that require support outside the hospital setting. It is hence necessary to have a support system that can be accessed by the patients virtually.

Families dealing with pregnancy are very particular concerning the wellbeing of their offspring and themselves making them want to adopt a healthy lifestyle change. An average appointment session with a doctor generally does not last for more than 15 minutes. Such a duration may not be enough to explain everything to a mother. So, with the development of ICT, a lot of women have resorted to searching on websites and apps on the google play store.

In the last few years, the Improvement in Hardware systems and GPUs has led to a rise in the number of Chatbots that are *"Natural language processing systems acting as a virtual conversational. agent mimicking human interactions."*<sup>5</sup> While this technology is still in its developmental phase, health chatbots could potentially increase access to healthcare, improve doctor-patient and clinic-patient communication, or help to manage the increasing demand for health services such as via remote testing, medication adherence monitoring, or teleconsultations.<sup>6–8</sup> Using chatbots in healthcare is a part of Healthcare Informatics. Health Informatics has come out to be one of the hottest fields in the last decade, with a huge role of social media, apps, and websites in helping penetrate through the masses. However, analyzing real-time information that is accurate, reliable, specific, and covers unique cases is one of the biggest challenges that an assistant may face.

There has been a lot of criticism that the chatbot solution faces with an argument that the chatbot can never replace an on-field doctor. However, our solution is just an aid to the doctor-patient relationship by acting as a first-level point of support for the doctors and patients. Nevertheless, the main function of a chatbot of providing information remains useful almost everywhere to everyone provision of this type of assistant will make the patients aware of their health by helping them keep track of their condition with minimal human intervention and reduce stress, anxiety thus simplifying the process of Healthcare by the intervention of technology.

This research paper discussed the development, design, and need of a chatbot. All the challenges, perspectives, stakeholders, and functionalities are introduced. A quality framework has been implemented to analyze the factors influencing the adoption and development practices along-with the necessity of a virtual doctor assistant that can be used to provide first-level support to to-be-mothers via a conversational-based support system.

The paper is further classified as follows section 2 contains the review of literature related to current developments in Health Informatics and chatbot solutions in the field of Healthcare. Section 3 discusses the method of study used. Section 4 presents the results of the output. Section 5 presents the discussions containing the interpretation and description of the significance of the findings. Section 6 provides the conclusions.

### **Literature Review-**

In 1996 Chatbots were introduced and since then these agents have made countless contributions to humanity by making our lives easier. The old chatbots were just a mere question-answer automated tool where your question needed to match with the question pre-feed in the chatbot to be able to provide an answer. But nowadays the chatbot technology has rapidly advanced and FAANG companies like Amazon, Microsoft, and Google are heavily investing in the field of Natural Language Processing for text and chat-based communication which can analyze the context of the conversations to provide customized and personalized answers to different range of questions. (Kandpal et al., 2020) *“Amazon Alexa is a great example of Virtual Assistantship as it provides help in various day-to-day activities like setting alarms, listening to music, news updates, and online shopping just by speaking.”*

### **Dialogflow by Google**

A popular chatbot development tool was introduced by Google as a part of its Google Cloud Platform services. (Kandpal et al., 2020) stated that *“Dialogflow helps businesses in providing latest tech services related to the field of Cloud Computing, Databases, Machine Learning and AI. Dialogflow can analyze both voice and text-based user inputs. Dialogflow can be integrated on whichever platform the developer wants to deploy and it can interact with various devices like mobile phones, car, TV, Google home or phone calls.”*

The concept behind Dialogflow is NLU(Natural Language Understanding) which can take all the other possibilities while the user is interacting with the chatbot without requiring any hardcoding. This is achieved by certain advanced functionalities like brief summarizations, different accents, and sentiment analysis which are prefeed into the Dialogflow model and do not need external addition.

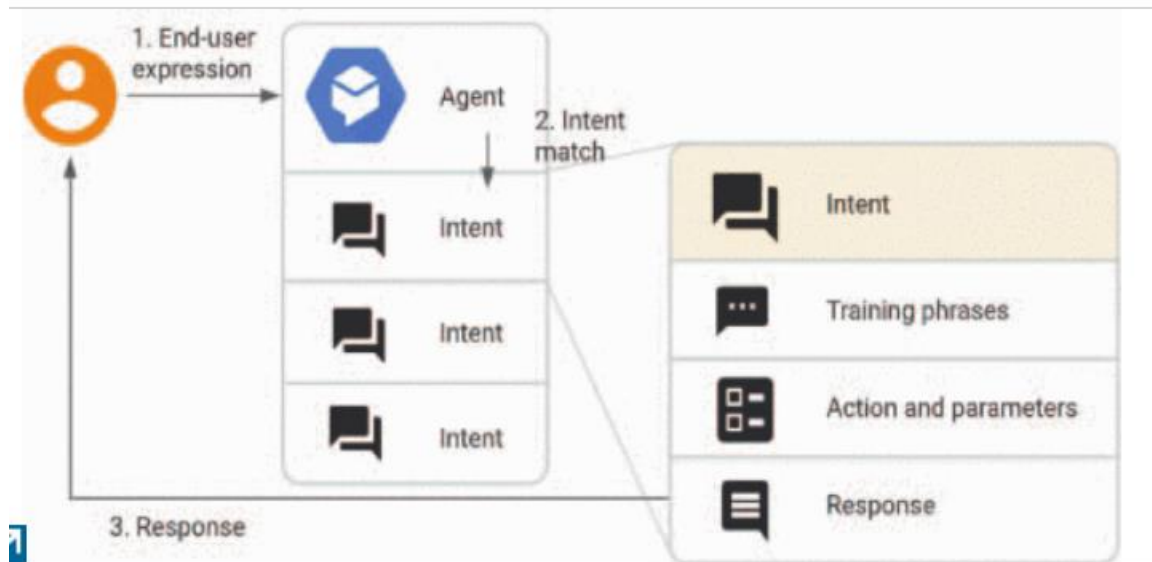


Figure1: Algorithm of DialogFlow

We mention below some common terminologies that are used by Dialogflow in its Design and Implementation.

### Agents

All our conversations are handled by a virtual Dialogflow agent that can handle conversations. The agent uses NLU to interpret the user languages. Dialogflow allows us to build an agent to handle the workflow of all our system conversations and help structure the unstructured data of our text and voice inputs. The agent can be considered as analogous to a virtual assistant to the user trained to handle complex real-time conversations without requiring any explicit training.

### Intents

The intention of the user can be matched and defined with the help of intents. A lot of intents can be defined for each user. The combination of all these intents will help the agent in handling the entire conversation. On receiving a message, the user expression gets matched to the best-suited intent for the incoming user message categorized as intent classification.

Another important functionality of intents is that they can also be used to extract important information from the messages which can be analyzed and utilized by the system for making better and more personalized decisions. The figure given below shows how the data gets extracted from the end-user-defined expression inputs of a weather prediction query.

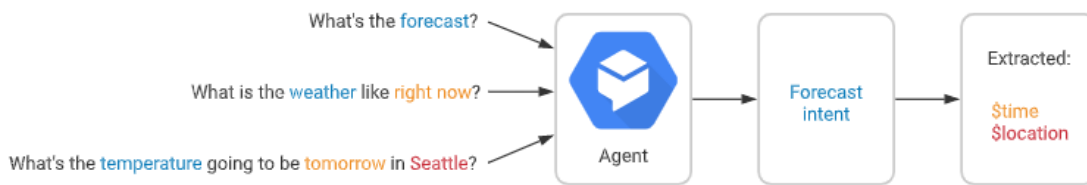


Figure2: Data Extraction

The following are the functionalities and modalities that a basic intent capture.

1. **Training phrases**: Dialogflow asks the developers to put in the phrases to train the model upon so that it can take those and after they are prefeed into the model use those phrases to train the model and if the users end expression is found to be resembling those phrases. The intent gets matched.
2. **Action**: Every intent can be defined with an action on getting matched with the intent Dialogflow gives us the functionality to invoke a set of actions on the system.
3. **Parameters**: On getting matched with the intent some values of data can be extracted from the end-user expression known as parameters. The way the data is to be extracted gets defined by an entity type. The parameters are a set of structured data which are used to build upon the logic of an algorithm.
4. **Responses**: Dialogflow provides us the capability of declaring the answers to the users visually as well as in the form of text and speech. The response can take in more answers by asking a question and can also terminate the conversations.

The following diagram shows the basic flow for intent matching and responding to the end-user:

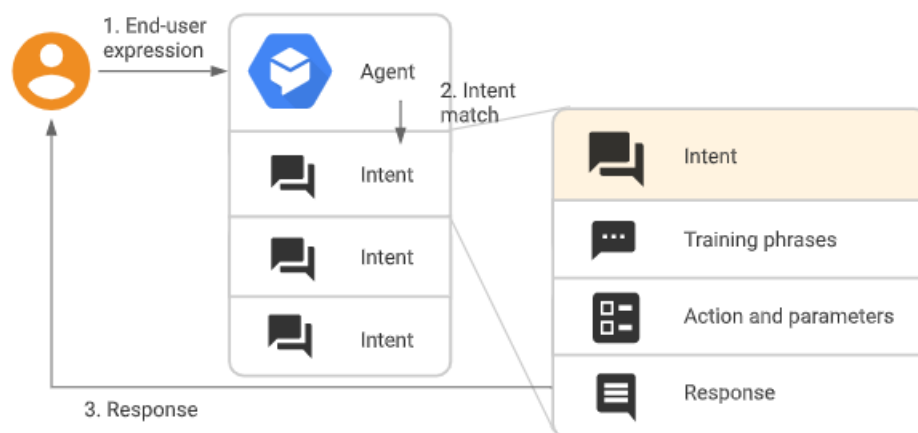


Figure3: Intent Matching

## Entities

There are entity types associated with every parameter that guides the process of data extraction from the end-user expression.

A set of system entities is provided by Dialogflow for matching the common data types with an option of creating custom entities as well for matching with the custom fed data.

## Contexts

Just as we try to get the context of conversation while talking to each other in a similar way Dialogflow takes the contexts for handling the end-user expression for matching it to the optimal intent by analyzing the context.

Contexts are very powerful tools that influence the flow of a conversation. Contexts can be configured with the help of input-output contexts, generally categorized by the name of the strings. An intent context immediately gets activated on getting matched with the intent. Dialogflow directly matches the active input contexts with the matched intents configured input contexts.

The diagram given below gives an example of the usage of context for a banking agent.

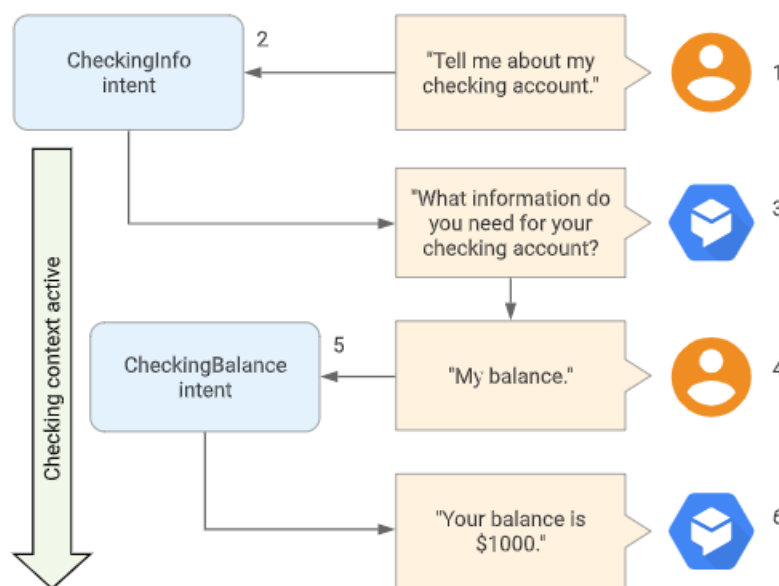


Figure4-Usage of contexts by Banking Agents

The workflow of the above diagram described by(Gelfenbeyn et al., 2018) is as follows:

1. "The end-user asks for information about their checking account.
2. Dialogflow matches this end-user expression to the *CheckingInfo* intent. This intent has a *checking* output context, so that context becomes active.
3. The agent asks the end-user for the type of information they want about their checking account.

4. *The end-user responds with "my balance".*
5. *Dialogflow matches this end-user expression to the CheckingBalance intent. This intent has a checking input context, which needs to be active to match this intent. A similar SavingsBalance intent may also exist for matching the same end-user expression when a savings context is active.*
6. *After your system performs the necessary database queries, the agent responds with the checking account balance."*

#### Follow-up intents

To automatically set contexts for the intents Dialogflow provides us with the functionality of follow-up intents that are the sub-branches of the parent intents. An output context gets added to the creation of a follow-up intent. The matching of the follow-up intent is done only after the matching of the parent intent. The follow-up intents can be used to guide the workflow of the algorithm with the help of nesting and by the creation of multiple levels.

A functionality of pre-defined follow-up intents is provided by Dialogflow for common replies. However, the platform also provides us with the ability to create our follow-up intents to take care of the custom replies.

#### Dialogflow Console

There is a provision of a console that provides a web-user interface for building, testing, and creating agents. The console helps the developer in managing the agents.

#### User interactions with integrations

The integration of Dialogflow with platforms like FB messenger, Slack, and Google Assistant makes it very convenient for the developers as they can solely focus all their energy on developing the agent. A platform-specific way is used in each integration to handle the interaction of the end-user.

#### Fulfillment for integrations

A static response is given with a matched intent. To enable a dynamic response, we use fulfillment that can call the defined service by us. The intent has a setting to enable fulfillment and on getting matched performs the required action by giving the dynamic response. A static response is used in case the matched intent does not have the option of fulfillment enabled. The process that runs behind the background when it gets matched with a fulfillment enabled intent is that the request to the webhook service with information is sent that responds to Dialogflow about the steps needed to progress further. The diagram given below shows the fulfillment procedure.

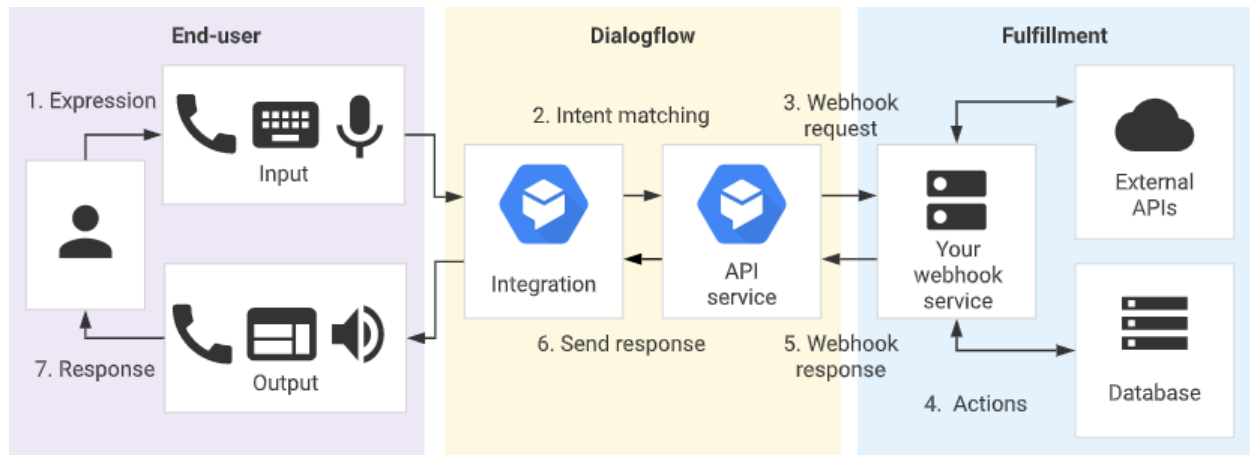


Figure5-Fulfillment Process

1. The parameters are extracted, and end-user expression is matched.
2. A webhook request is sent to the webhook service defining the response, action, parameters.
3. The action needed is performed.
4. A webhook response message containing the response is sent to Dialogflow.
5. The end-user receives the response.

#### Help users interact with technology.

The old computers needed a specific form of input for proper functioning. The proper understanding of structured inputs. A single query can take multiple forms and make the machine confused as to what exactly the user is referring to. The provision of an algorithm at the back that can gather this unstructured data and present it to the computer in a structured form was necessary to ensure that the model operating was robust.

The algorithm is managed in Dialogflow with the help of a virtual agent that handles the user conversations. Once the user messages then the model gets activated and searches for the best and the most suited intent which consists of training phases, trained conversation, responses, actions, and parameters. The appropriate response is replied to by the user once the correct intent is mapped.

#### Introduction to the Messenger Platform

*Working on the messenger platform:* On receiving a message from the user on the Facebook page messenger. The Facebook app is used to automate the conversations. It employs the usage of webhooks to the server URL of the business on which the app is hosted. Send API is employed to send the response to the person. The developers build conversations using the automated app flow to bridge the messenger and the agent's presence.

The Messenger platform is provided free of cost to the customers to help businesses handle customer inquiries.



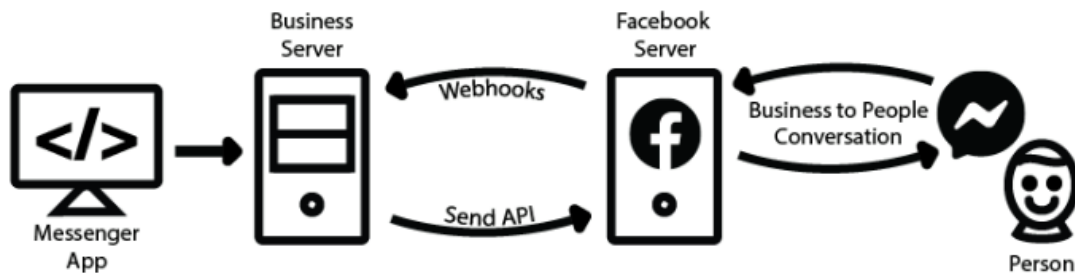


Figure6-Messenger App Workflow

There are 2 types of chatbots that we encounter: Conversational and Task-oriented chatbots. The scope of the activity of these types of chatbots is limited and there are a certain set of pre-defined rules that need to be followed. Whereas conversational chatbots rely on the usage of Deep Learning algorithms and NLP for taking the information context with a response tailored to the specific input. These types of bots also rely on Artificial Intelligence to improve the accuracy of their responses. So, it is recommended to have a chatbot to handle general conversation just like a natural conversation proceeds.

The chatbot uses the input data of the users to interpret, parse and extract the information. The 2 classes which are statistical and rule-based use NLP systems that are text-specific ML-based statistical models and parse through the sentences, phrases, and grammar seamlessly. Certain high-quality practices should be maintained while designing an agent to meet the objectives of the proprietor.

### Objective

The first step is to lay out the objectives that the agent will achieve.

### Platform

The next step is to determine the platform through which the agent will be accessed by the users. The choice of the platform would play a crucial role in determining the content and the support that would be provided.

### Build agents iteratively.

The development process is a never-ending process. However, we can start by building a basic structure that provides the basic functionalities then switch onto a more complex structure with added features and functionalities. This can be done by iterating on all the possible routes of the conversation path of a user.

### Pre-built agents

Certain pre-made agents save the developers from the hassle of creating and designing the agent from scratch. Most of the common queries are covered with the help of intents and entities that contains most of the common queries of the users. The only step that this requires is to add responses to build an agent that is functioning well.

## System entities

During the process of making a request, there are important information called entities. Dialogflow provides pre-built options on these in the form of system entities.

## Small talk

During the development, it may happen in rare cases that the user may be expecting a little off-topic conversation from the agent. This type of feature can be provided with the help of the small talk option. This feature would help the agent to respond to the conversation as naturally as a human by responding to the questions related to the agent as well as provide emotional responses. There is further an option of customizing these small talk responses to suit the brand in a much better way through customization.

## Agent design best practices

The section provided by the Actions on Google team below lists down the design practices that are recommended to be followed by the developers for the development of an accurate agent that is performant, robust, and usable.

### ***"Greetings and goodbyes"***

| <b><i>Best Practice</i></b>  | <b><i>Details</i></b>   |
|--|---|
| <i>Welcome intents should let users know about the agent's capabilities with branding in mind.</i> | <i>Your agent's welcome intent should inform the user of 2-3 tasks the agent can help with, as well as brief descriptions (as needed) of how to use these features.</i> |
| <i>Agents should have a suitable exit message when a successful interaction has ended.</i>         | <i>When a user completes a task in your agent, it should summarize the transaction/task and say something like "Until next time", etc.</i>                              |

### ***Machine learning and training***

| <b><i>Best Practice</i></b>   | <b><i>Details</i></b>  |
|---|--|
| <i>Intents should have at least 10-20 (depending on complexity of intent) training phrases.</i> | <i>The complexity of your agent will determine the actual number of <u>training phrases</u> each intent should have, but 10-20 (depending on complexity of intent) is a good minimum. The more parameters you have in your intents, the more phrases you should provide to train the machine learning model.</i> |

| Best Practice  | Details   |
|--|---|
| <i>Training phrases should be varied.</i>                      | <i>Include variations of questions, commands, verbs, and synonyms for common nouns to ensure your phrases cover a broad spectrum of possible requests.</i>  |
| <i>Annotations should be consistent.</i>                       | <ul style="list-style-type: none"> <li>• <i>Review your <u>training phrases</u> and ensure that highlighted annotations are pointing to the correct <u>entities</u>.</i></li> <li>• <i>You should not have text in training phrases that is annotated in some cases but not others.</i></li> <li>• <i>The span of text selected for an annotation should include all of, and no more than, the text that is necessary to match an entity.</i></li> <li>• <i>Be sure that the annotated text in multiple training phrases contains similar portions of the training phrase. For example, consider that you have a training phrase "Set alarm at 6 a.m.", where "6 a.m." is annotated as @sys.date. If you have another training phrase "wake me up at 7 a.m.", do annotate "7 a.m.", but do not annotate "up at 7 a.m."</i></li> </ul> |
| <i>Custom entities should cover a broad range of examples.</i> | <i>Entities are lists of items. Machine learning will take care of grammatical forms, but you have to include all possible items. Also, check the <u>define synonyms</u> option and include some variations.</i>  |
| <i>Disable ML for as few intents as possible.</i>              | <i>Training phrases for intents with ML disabled are not used when training your agent. A user query that is very similar to a training phrase in an intent with ML disabled may be matched to the wrong intent if other intents with ML enabled have a slight resemblance to the user query. If you are having problems with false positives, raise the <u>ML classification threshold</u> instead of disabling ML.</i>  |

| Best Practice  | Details  |
|--|--|
| <i>Do not set a high <u>ML classification threshold</u> for an agent with only a little training data.</i> | <i>If the threshold is high, and there is not a lot of training data, only user queries that have near exact matches to training phrases will result in intent matching. You need to provide a lot of training data if you desire a high threshold.</i>  |
| <i>Agents should have a fallback intent.</i>   | <i>Without fallback intents, unmatched user queries will result in empty responses.</i>  |
| <i>Agents should provide <u>negative examples</u>.</i>   | <i>Negative examples prevent user queries that are slightly similar to training phrases from unintentionally matching intents.</i>   |
| <i>Do not define entities that match virtually anything.</i>   | <i>This degrades the performance and quality of ML. Nearly everything in every training phrase will be evaluated as a possible match. Consider using @sys.any instead. Similarly, composite entities should not contain a single @sys.any as a synonym.</i>  |
| <i>Do not define entities that are composed of filler words or meaningless text.</i>                       | <i>Examples of filler words and meaningless text are: "hmmm", "let's see", "please", "could you please". If you are attempting to use entities like this to introduce variety, you are only degrading the performance of ML. Dialogflow already augments data to handle variety like this. You should add phrases like this to your training phrases, not your entities.</i>   |
| <i>Entities should have a limited scope that captures distinct values of one type of information.</i>      | <i>Keep your entities focused, short, and simple. If your entity values are complicated, it may be because intent training phrases are better suited to your situation. For example, consider end-user expressions like "How can I make an international call with Plan A?" and "Using international data roaming with Plan B". Do not create entities for both the actions ("How can I make an international call" and "Using international data roaming") and the plans ("Plan A", "Plan B"). Instead, you should use training phrases and intent matching to capture the actions and entities to capture the plans.</i> |

| <b>Best Practice</b>  | <b>Details</b>   |
|---|--|
| <i>Annotated text in training phrases should have variety.</i>              | <i>For example, if you are providing time values that should be parsed as @sys.time system entities in training phrases, do not provide the same time in all training phrases. Your training phrases should have a variety of time examples like: "7 a.m.", "8 p.m.", "9 o'clock".</i> |
| <i>Intents with many parameters should also have many training phrases.</i> | <i>As a rule, try to have at least three times as many training phrases as parameters and at least 10-20 (depending on complexity of intent) training phrases.</i>   |
| <i>Each parameter should be used in many training phrases.</i>              | <i>As a rule, each parameter should be used in at least 5 training phrases.</i>  |
| <i>Avoid using multiple @sys.any entities in a training phrase.</i>         | <i>One training phrase shouldn't contain two consecutive @sys.any or a total of three @sys.any entities. Dialogflow may not be able to distinguish them.</i>   |
| <i>Do not use similar training phrases in different intents.</i>            | <i>Different intents shouldn't contain similar training phrases, because this will prevent Dialogflow from learning how to recognize those phrases.</i>  |
| <i>Enable automatic spell correction.</i>                                   | <i>If you are using text input, you should enable <u>automatic spell correction</u>.</i>   |
| <i>Do not nest composite entities</i>                                       | <i>Do not use more than one level of nesting in composite entities. Each level of nesting significantly degrades quality.</i>  |
| <i>Avoid special characters in training phrases.</i>                        | <i>Special characters in training phrases, like {, _, #, and [ will be ignored. Emojis are an exception; they work as expected.</i>  |
| <b>Helpful intent features</b>  |  |

| <b>Best Practice</b>   | <b>Details</b>  |
|--|---|
| <i>Agents should support contextual requests.</i>  | <i>For example, if your agent handles requests for the weather and a user asks "Weather in San Francisco", make sure to add <u>contexts</u> to support further requests like "How about tomorrow?"</i>  |
| <i>Agents should have follow-ups for yes, no, cancel, next, back, etc.</i>                 | <i>Follow-up intents are used to reply to common responses. To add a follow-up intent, hover over an intent and click Add follow-up.</i>  |
| <i>Intents should have at least one text response.</i>                                     | <i>The <u>response section</u> is at the bottom of the intent's page. Adding variations will shuffle the response chosen, making for a less repetitive experience.</i>  |
| <i>Agents should collect all of the necessary information to fulfill a user's request.</i> | <i>Consider making necessary <u>parameters required</u>. Your agent will keep prompting the user until it gets the information it needs. This is called slot filling.</i>   |
| <i>Responses should repeat information as needed, like confirming an order.</i>            | <i>When a user is making a request like placing an order or changing information, your agent should repeat what's happening for confirmation purposes. When creating these confirmation responses, make sure to include all possible combinations of repeated <u>entities</u> and parameters.</i> |

### **Conversation repair**

| <b>Best Practice</b>  | <b>Details</b>   |
|---|--|
| <i>Agents should have helpful recovery prompts for each step of the dialog.</i> | <i>For example, if the initial prompt is "What color do you want?" and the user replies with "jungle parrot", a fallback/follow-up intent should rephrase the question, like "Sorry, what color was that?"</i> |

| <b>Best Practice</b>   | <b>Details</b>   |
|--|--|
| <i>Agents should have customized, brand-specific responses in the default fallback intent.</i>               | <i>When a user says something that isn't matched to an intent, the default <u>fallback intent</u> is matched. This should be customized to reflect your brand, as well as provide information to guide the user to make a valid request.</i> |
| <i>For customized fulfillment, agents should have an intent that allows users to repeat information.</i>     | <i>One intent can handle requests like "say that again", "repeat that", "play that again", etc. This can be a follow-up intent.</i>  |
| <i>Help users to be successful, steer them to tell them exactly what you would like to hear as an answer</i> | <i>For example: if you provide options, don't ask "Would you like A or B?." - because then a user could answer "yes" Instead ask: "I have A and I have B. Which one do you prefer?"</i>  |

## **Persona**

| <b>Best Practice</b>  | <b>Details</b>  |
|---|---|
| <i>Agent responses should have a style and tone that fits your brand and are consistent throughout the agent.</i> | <i>As your users converse with your agent, it should feel like they're speaking to one persona. Make sure the qualities and personality you've chosen are represented in all of your responses.</i> |
| <i>Agents should be sensitive about cultures, genders, religious beliefs, abilities and ages.</i>                 | <i>Stereotyping may offend users, even in jokes, and they might not return to your agent.</i>   |

## **Designing for voice**

| <b>Best Practice</b>  | <b>Details</b>   |
|---|--|
| <i>Avoid content that requires visualization or keyboard and mouse interaction.</i> | <i>Don't use hyperLinks, tables, images, abbreviations. You can refer to a website by name. When presenting a list of options, return the best match and ask if the user would like to hear alternative options.</i> |

### **Best Practice**

### **Details**

*Don't create awkward silences. Always end with a question. Steer the conversation and to get the interactions going.*

*Write compact dialogues which are easy to understand.*

*Use SSML*

*On a screen, text can be long and contain multiple paragraphs. You can skip the parts that are not interesting to you. But hearing a virtual agent talking for too long, won't make your users happy.*

*Use SSML to structure and to change the intonation of your sentences, so your voices will sound more natural.*

## **Protection of consumer privacy**

### **Best Practice**

### **Details**

*Disable data logging in agent settings for GDPR compliance.*

*In the agent settings, you can disable the logging of interactions in Dialogflow. By disabling this feature, no PII data will be stored in Dialogflow. This will also mean that certain features, such as analytics, won't be available.*

*Store chat conversation data in BigQuery to have control over regional storage.*

*Via Cloud Logging or by using the Dialogflow API, you can send incoming chat utterances to BigQuery. By taking this approach, you will have control over the region where you want to store the data. Additionally, you could make use of the Data Loss Prevention API, to mask sensitive information. See the (blueprint for building an AI-powered customer service on GCP)[<https://cloud.google.com/blog/products/ai-machine-learning/simple-blueprint-for-building-ai-powered-customer-service-on-gcp>].*

## **Using the knowledge base connector**



| <b>Best Practice</b>  | <b>Details</b>  |
|---|---|
| <i>When importing public FAQs, use valid HTML5 markup.</i>    | <i>For example, use article elements with schema.org notation such as <a href="https://schema.org/Question">schema.org/Question</a> and <a href="https://schema.org/Answer">schema.org/Answer</a>.</i>  |
| <i>Make sure your FAQ website is indexed by Google Robots</i> | <i>The website will need to allow Google Robots, and needs to be added to the Google search engine via the <a href="#">Google Webmasters tool</a>. Sites like <a href="#">pages.github</a> will not work, because they are not crawlable.</i> |
| <i>Use 1-200 FAQs</i>   | <i>You need more than one Q&amp;A pair and not more than 200 per knowledge base. You can load multiple knowledge bases if you need more.</i>  |

### **Implementing Dialogflow APIs**

| <b>Best Practice</b>  | <b>Details</b>   |
|---|--|
| <i>Do not expose your service account private key in client codebases for mobile or web applications.</i> | <i>This is not considered safe. Anyone who is handy with Chrome Dev tools could steal your key and make (paid) API calls via your account. It's a better approach to always let an API proxy server handle the Google Cloud authentication. This way the service account won't be exposed to the public, and the keys can be safely stored.}</i> |

### **Designing for voice and text in one agent**

| <b>Best Practice</b>  | <b>Details</b>  |
|---|---|
| <i>Don't use SSML in <u>default platform responses</u>.</i> | <i>When your agent can respond with both voice and text, a text response will include the raw SSML code. Use plain text in the default platform response and SSML in platform-specific responses. Alternatively, you can use a webhook to generate SSML only when a voice response is needed.</i> |

## Testing

| Best Practice  | Details  |
|--|--|
| <i>Test your app thoroughly with someone that was not involved in its development.</i> | <i>Having someone unfamiliar with the agent use the app will give you insight to how naturally the conversation flows. Have them look out for accuracy, long pauses, missing conversational paths, pacing, awkward transitions, etc.</i> |
| <i>Test your app on all platforms you plan on supporting.</i>                          | <i>If your agent will be available on one or more <u>platforms</u>, make sure <u>rich messaging</u> and responses show up as expected across all platforms.</i>  |

Table1: Best development Design Practices

In this research the primary goal is to have a chatbot solution for supporting pregnant women and be their virtual companions during pregnancy by answering to their requirements and needs related to the disease of concern, suggesting preventive measures, guiding them to live a healthy lifestyle, etc. The chatbot benefits the mothers, families, and pregnant women with young children the most. It provides different information by starting from a general to a specific set of pathway questions. The chatbot proposed an intelligent agent that represents a virtual healthcare expert and answers the queries of the users remotely from anywhere without getting affected by the geographical location of the user. These types of systems are very critical when it comes to deployment as they require accuracy and robustness.

## Methodology/Design/Implementation-

The objectives that we plan to achieve with a chatbot are listed down.

### Objectives

- Use Dialogflow to build a conversational chatbot that can respond to questions about the pregnancy of to-be-mothers.
- Integrate your Dialogflow model with Facebook by building a Messenger App-based web frontend for a text-chat style interface.

For the development of the model, there was a prerequisite knowledge of using Dialogflow concepts that were needed. The following courses offered by Coursera and Qwiklabs respectively were completed before starting with the development.

- Building Conversational Experiences with Dialogflow
- Implementing an AI Chatbot with Dialogflow

As a first step, we created a Google Cloud project and enabled all the APIs, Cloud Storage, Datastore, Compute Engine, and Repositories of Cloud Source.

### Creating a chatbot agent

We created a knowledge base for our model and a flowchart algorithm to get a rough idea of the workflow of the agent. After this, we started building the Dialogflow agent

### Create a topic entity

We then created an entity that could take into consideration all the pregnancy topics that the chatbot could discuss.

### Create a chatbot intent.

We then create a chatbot intent that captures the request and gives the interaction response.

An example, of an intent modeled for interaction, is:

- User: "Hi, I am pregnant."  
This question activates the HR manual intent.
- Chatbot: "OK, I'd be happy to help with that. What assistance can I provide?"

To create a chatbot intent: The procedure given below is followed

### Default intents

When you create an agent, two default intents are created for you:

- **Default Welcome Intent:** This intent is matched when the end-user begins a conversation with your agent. This intent should return a response that lets the end-user know what your agent does or what they can say to begin a conversation.
- **Default Fallback Intent:** This intent is matched when the agent cannot match the end-user expression to any other intent.

To see these intents, *"go to the intent list for your agent:*

1. *Go to the Dialogflow ES Console.*
2. *Select the agent you just created.*
3. *Click **Intents** in the left sidebar menu.*

*The middle of the console shows the list of intents for the agent."*

Search intents



 Default Fallback Intent

 Default Welcome Intent

Figure7- Default Intents

Test the Default Fallback Intent

Try it now



See how it works in [Google Assistant](#). 

Agent

USER SAYS

[COPY CURL](#)

What is your name?



DEFAULT RESPONSE



Say that one more time?

Figure8- Intent Testing Console

The Dialogflow simulator is on the right side of the console. With the simulator, you can test your agent by speaking or typing messages.

Try the agent now:

- Click the **Try it now** field.
- Type `What is your name?`.
- Press enter.

The agent's response appears in the **Default Response** section. Since your input didn't match any intent, the Default Fallback Intent was matched, and you received one of the default replies.

### Create a new intent

The steps in this section create an intent that can answer the question "what is your name?". For each intent, you define many training phrases. A training phrase is an example of what an end-user might type or say to your agent, also known as an end-user expression. You should define many training phrases that provide Dialogflow with a variety of expressions that should match an intent.

### Create an intent:

1. Click the add intent add button next to **Intents** in the left sidebar menu.
2. Enter `get-agent-name` in the **Intent name** field.
3. In the **Training Phrases** section, click **Add training phrases**.
4. Enter the following training phrases, pressing enter after each entry:
  - `What is your name?`
  - `Do you have a name?`
  - `Tell me your name?`

## Training phrases

Search training phrases 



|   |                     |
|---|---------------------|
| ” | Add user expression |
| ” | Tell me your name   |
| ” | Do you have a name? |
| ” | What is your name?  |

Figure9-Training Phrases

**Note:** In most cases, you should enter at least 10-20 (depending on the complexity of intent) training phrases for reliable intent matching.



5. In the **Responses** section, enter the following in the **Text Response** section:

- My name is Dialogflow!

## Responses



DEFAULT GOOGLE ASSISTANT +

| Text Response   |                               |
|---|-------------------------------|
| 1   | My name is Dialogflow!        |
| 2   | Enter a text response variant |


ADD RESPONSES



Figure10-Add Responses Textbox

6. Click the **Save** button and wait until the **Agent Training** dialog indicates that training is complete.

Test your intent.

What's your name?




 See how it works in [Google Assistant](#). 

AgentDomains

USER SAYSCOPY CURL

What's your name?

 DEFAULT RESPONSE▼PLAY

My name is Dialogflow!

INTENT

Name

ACTION

Not available

SHOW JSON

Figure11- Dialogflow Chat ssimulator

In the simulator, type `What's your name?` and press enter.

Your agent responds to the expression correctly, even though the expression was a little different from the training phrases you supplied.

Dialogflow uses training phrases as examples for a machine learning model to match end-user expressions to intents. The model checks the expression against every intent in the agent, gives every intent a score, and the highest-scoring intent is matched. If the highest-scoring intent has a very low score, the fallback intent is matched.

### Parameters and entities

When an intent is matched at runtime, Dialogflow provides the extracted values from the end-user expression as parameters. Each parameter has a type, called the entity type, which dictates exactly how the data is extracted. Unlike raw end-user input, parameters are structured data that can easily be used to perform some logic or generate responses.

When building an agent, you control how data is extracted by annotating parts of your training phrases and configuring the associated parameters.

### Create Parameters

Create a new intent with parameters:

1. Click the plus add button next to **Intents** in the left sidebar menu.
2. Name the intent `EPC-Down Syndrome` at the top of the intent form.
3. Add the following training phrases:
  - `Genetic Disorder`
  - `Trisomy 21`
  - `Downs Syndrome`
4. Click the **Save** button and wait until the **Agent Training** dialog indicates that training is complete.

• EPC-DownSyndrome

” Add user expression

” Genetic Disorder

” Downs syndrome

” Trisomy 21

” Prenatal diagnosis


Figure12-Early Pregnancy Care Down Syndrome intent



5. Dialogflow automatically detects parameters in training phrases that are recognized as system entities. These are entities provided by Dialogflow for many common data types like location, color, and date.

**Note:** If your training phrases are not automatically annotated, you can manually annotate them.

Below the **Training phrases** section, Dialogflow creates a row in the **Action & parameters** table:

Action and parameters 





| REQUIRED  | PARAMETER NAME  | ENTITY  | VALUE       | IS LIST  |
|--|--|--|-------------|---|
| <input type="checkbox"/>   | number   | @sys.number  | \$number    | <input type="checkbox"/>  |
| <input type="checkbox"/>   | Enter name   | Enter entity   | Enter value | <input type="checkbox"/>  |

Figure13-Actions and Parameters

**Required:** The checkbox is not checked, so this parameter is optional.

- **Parameter Name:** This parameter is automatically named as a `number` because the parameter is recognized as a language.
- **Entity:** This is the entity type. It is recognized as a `@sys.number` system entity.
- **Value:** This is the identifier you use when referencing the value of this parameter.
- **Is List:** The checkbox is not checked, so the parameter is not a list.

**Note:** If entities aren't automatically detected, you can highlight text in the training phrase and manually annotate the entity.

#### Use parameter data in a response

The value of a parameter can be used in your responses. For example, you can use the `$number` parameter reference in your responses when building an agent. At runtime, it will be replaced with the language specified in the end-user expression.

Add a response that uses a parameter:

1. Scroll down to the **Responses** section.
2. Add the following text response: `Wow! I didn't know you are $number years old.`
3. Click the **Save** button and wait until the **Agent Training** dialog indicates that training is complete.

Test your parameter

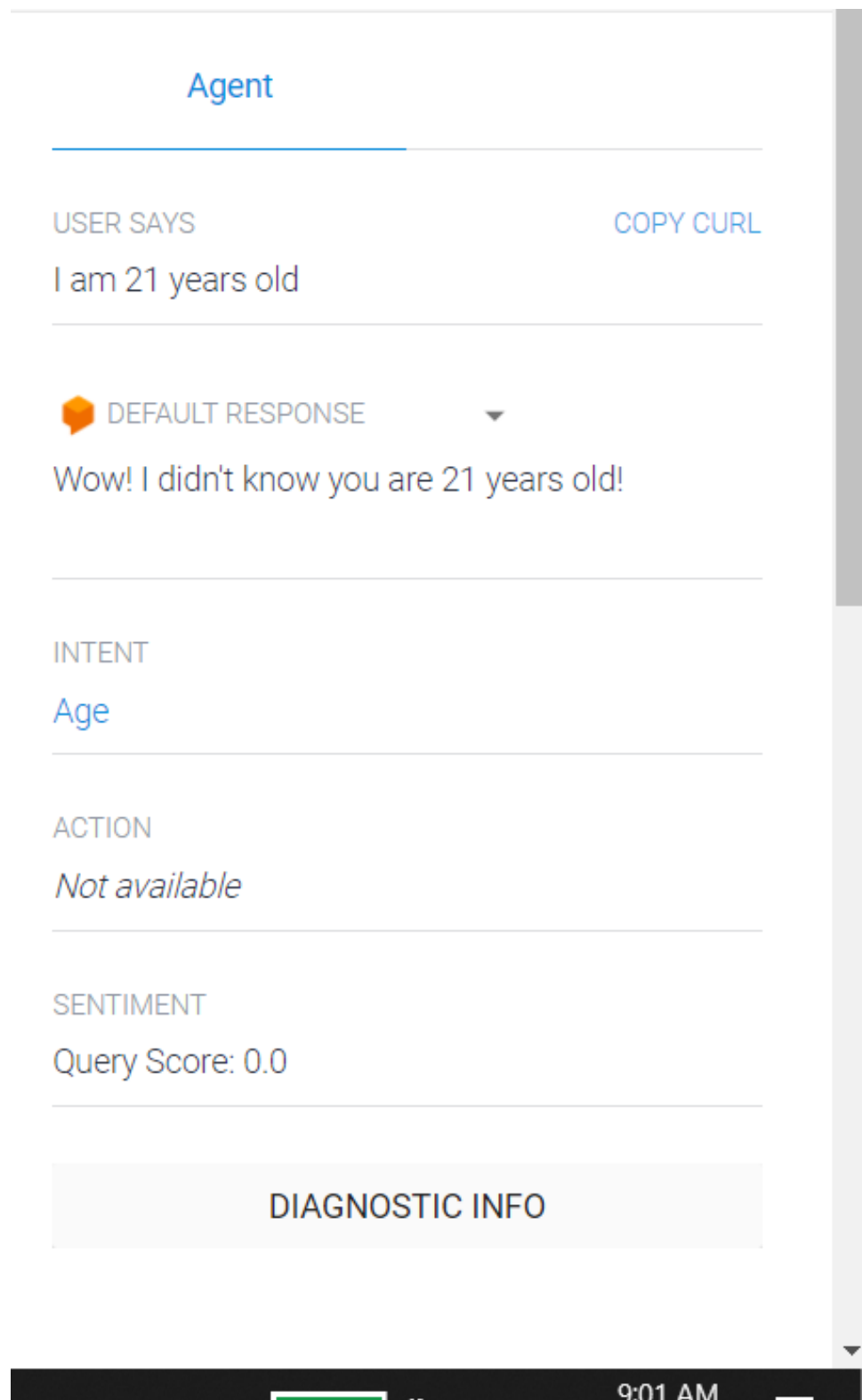


Figure14-Testing Action and Parameters

In the simulator, enter `I am 21 years old.`

You can see that Dialogflow correctly extracts the `age` parameter with the value `21`, and `21` was correctly inserted where the parameter reference was used in the response.

### Create your own entities.

In most cases, you have specific data you need to gather from users that is not provided by system entities. You can create custom entities to handle this.

### Create a custom entity:

1. Click the `add entity` button next to **Entities** in the left sidebar menu.
2. Enter `language-programming` for the name of the entity.
3. Add the following entity entries (rows):

## Pregnancy-abbreviations

SAVE

☒ Define synonyms ? ☐ Regexp entity ? ☐ Allow automated expansion ☐ Fuzzy matching ?

i Separate synonyms by pressing the `enter`, `tab` or `;` key. ×

|                       |                      |
|-----------------------|----------------------|
| BMI                   | BMI, Body Mass Index |
| A/N                   | A/N, Antenatal       |
| BF                    | BF, BreastFeeding    |
| BP                    | BP, Blood Pressure   |
| Enter reference value | Enter synonym        |

Figure15-Entity Creation

Click the **Save** button and wait until the **Agent Training** dialog indicates that training is complete. Dialogflow can handle simple things like plurality and capitalization, but you should add all possible synonyms for your entries. The more you add, the better your agent can determine your entities.

## • Measure

SAVE

” Add user expression



” My BP is 101

” My BMI is 10

### Action and parameters



Enter action name

| REQUIRED ?               | PARAMETER NAME ?        | ENTITY ?                 | VALUE                     | IS LIST ?                |
|--------------------------|-------------------------|--------------------------|---------------------------|--------------------------|
| <input type="checkbox"/> | pregnancy-abbreviations | @Pregnancy-abbreviations | \$pregnancy-abbreviations | <input type="checkbox"/> |
| <input type="checkbox"/> | number                  | @sys.number              | \$number                  | <input type="checkbox"/> |
| <input type="checkbox"/> | Enter name              | Enter entity             | Enter value               | <input type="checkbox"/> |

Figure16-Table of abbreviations, entities, action, and parameters

### Use your new entity

Add training phrases to the Measure intent that make use of the new entity:

1. Click **Intents** in the left sidebar menu.
2. Click the Measure intent.
3. Add the following training phrases:
  - My BP is 101
  - My BMI is 10
4. Notice that the programming languages in these training phrases are automatically annotated and added to parameters in the **Action and Parameters** section.
5. In the **Responses** section, add the following second text response: Your \$pregnancy-abbreviations is \$number.
6. Click the **Save** button and wait until the **Agent Training** dialog indicates that training is complete.

Test your new entity

### Agent


---

USER SAYS

COPY CURL

My BMI is 11.97

---

 DEFAULT RESPONSE

▼

Your BMI is 11.97

---

INTENT

Measure

---

ACTION

*Not available*

| PARAMETER               | VALUE |
|-------------------------|-------|
| number                  | 11.97 |
| pregnancy-abbreviations | BMI   |

---

SENTIMENT

Query Score: 0.1

---

Figure17-Extracting Parameters on Simulator

*In the simulator, enter `My bp is 11.97`.*

*You can see that Dialogflow correctly extracted `bp` for the `pregnancy-abbreviations` parameter, identified it as the `Blood Pressure` entity, and inserted the value in the response.*

### Contexts

*To control the flow of the conversation, you can use context.*

### Add a follow-up intent

*Follow-up intents provide a simple way to control a conversation without having to create and manage contexts manually.*

*When you create a follow-up intent, an output context is added to the parent intent and an input context of the same name is added to the child intent. This means that the follow-up intent is matched only when the parent intent is matched in the previous conversational turn.*

*Add a custom follow-up intent to the `measure` intent:*

- 1. Select the `measure` intent you created in the previous steps.*
- 2. In the **Response** section, update the text response:*
  - `Wow! That's great. Could you help me with your $age?`*
- 3. Click the **Save** button and wait until the **Agent Training** dialog indicates that training is complete.*
- 4. Click **Intents** in the left sidebar menu.*
- 5. Hover over the `measure` intent and click **Add follow-up intent**.*
- 6. Click **custom** in the revealed list.*
- 7. Click the **Save** button and wait until the **Agent Training** dialog indicates that training is complete.*

*Dialogflow automatically names the follow-up intent `measure - custom`.*



Figure18- Custom Intents

#### Intent matching with follow-up intents

Follow-up intents are only matched after the parent intent has been matched. Since the **measure - custom** intent is only matched after the **measure** intent, you can assume that the user has just been asked the question **what is a good \$pregnancy-abbreviations?** Now you can add training phrases for likely user answers to that question:

1. Click **Intents** in the left sidebar menu.
2. Click the **measure - custom** intent.
3. Add the following training phrases:
  - **8 - 12**
4. Click the **Save** button and wait until the **Agent Training** dialog indicates that training is complete.

#### Test your follow-up intent

Enter **My bp is 11.8** in the simulator, then answer the question **Wow! That's great. Could you help me with your \$age? with about 21 years.**

Despite there being no response for the second expression (**about 21 years**), you can see the expression is matched to the correct intent (**measure - custom**), and the duration parameter is correctly parsed (**21 years**).

#### Intents and contexts

Inspect the **measure** intent to see that **measure-followup** is listed as an output context and is prefaced by the number 2. This number is called the lifespan.

Measure - custom

SAVE

priority by pressing this icon

Contexts ?

^

Measure-followup (x) Add input context

Add output context (x)

**Figure19- Custom Intent Context**

After the `measure` intent is matched, the `measure - followup` context is active and attached to the conversation for two turns (lifespan of 2). Therefore, when the user responds to the question, *Wow! That's great. Could you help me with your \$age?* the `measure-followup` context is active.

Inspect the `measure - custom` intent to see that `measure - followup` is listed as an input context, which is the same as the output context for the `measure` intent.

Any intents with an input context that matches an active context are heavily favored when Dialogflow matches intents.

### Contexts and parameters

Contexts store parameter values, and you can access the values of parameters defined in the `measure` intent when its output context is active.

In the `measure - custom` intent, you only asked for the duration the user has known the language, and not the referenced language itself.

To reference the language in the response:

1. Update the `measure - custom` intent text response to *I can't believe your `#measure-followup.pregnancyabbreviations` is \$number!*
2. Click the **Save** button and wait until the **Agent Training** dialog indicates that training is complete.

The `#measure-followup.pregnancyabbreviations` reference is known as a parameter reference for an active context.

### Test the context parameter

Enter `my bp is 11.8` in the simulator, and then respond to the question with `21 years`. Notice that the `pregnancy-abbreviations` parameter value is retrieved from the context.



1. *Intents are created.*

*To answer each types of module. Intent related to that module is created.*

2. *In the **Intent name** field, topics is typed.*

3. **Parameters and Action are added.**

4. *In the **Enter action name** field, lookup is typed.*

5. *The following is entered in the Parameters table*

- *In the **Parameter Name** field, topic.*
- *In the **Entity** field, @Topic.*
- *In the **Value** field, \$topic.*

*The following would create a lookup action passing the topic parameter to the webhook and hence retrieving information.*

## Action and parameters ?

lookup

| REQUIRED ?               | PARAMETER NAME ? | ENTITY ? | VALUE   | IS LIST ?                |
|--------------------------|------------------|----------|---------|--------------------------|
| <input type="checkbox"/> | topic            | @Topic   | \$topic | <input type="checkbox"/> |

+ New parameter

Figure20- Lookup Action

### Train a chatbot Intent.

1. *Now the next step was to train the created intents by adding the training phrases with the following steps:*

- In the **Add user expression** field, I'd like to know about the importance of discipline in pregnancy.*
- Select the word **discipline**.*

## Training phrases ?

Search training phrases



” Add user expression

” I'd like to know about discipline.

| PARAMETER NAME | ENTITY | RESOLVED VALUE |   |
|----------------|--------|----------------|---|
| topic          | @Topic | discipline     | × |

” I want to know about discipline.

” What is discipline?

” Where can I find out about discipline?

Figure21- Resolving values in training phrases.

- c. In the dialog, **@Topic: topic** informs Dialogflow where it must extract the topic parameter exactly from.

Dialogflow now trains the agent based on the example intentions. The training gets completed with a notification message pop-up.

### Test your chatbot.

- In the Dialogflow console, inside the input field, the sample questions are typed in different grammatical forms. This displays a response from the Virtual assistant.

## Agent

---

USER SAYS

[COPY CURL](#)

i am having backache

---



DEFAULT RESPONSE



<https://www.youtube.com/watch?v=sNNi9dj2k44&list=PLPg5Fy2dzqqc05XqXoLp...>

---

INTENT

[EPC-BackPain](#)

---

ACTION

*Not available*

---

SENTIMENT

Query Score: 0.0

---

Figure22- Chatbot Testing



Figure23- Response Video

### Training and testing the chatbot

*After having all the training data of the chatbot. Intent creation was used that could take the queries that the pregnant mothers may ask the virtual assistant. Then, we tested the chatbot using Dialogflow's built-in simulator.*

The built chatbot needs to be integrated with a platform to make it universally accessible to each user. To achieve this Dialogflow Messenger integration is used. A Facebook messenger bot is created to interact with the end-users. Therefore, with the help of Dialogflow Facebook Messenger integration, A Facebook Messenger bot to interact with the end-users is created.

The overview of the integration is that a Facebook app is created using the Messenger Platform and configured with Dialogflow for communication then the Dialogflow integration uses the Facebook API for sending messages which get received from the end-user through the Dialogflow integration with Facebook Messenger webhook. The code which powers this experience is open-source and helps businesses to start delivering a great messaging experience. This will enable to automation of the responses from the server and keep the page engaged.

These are the following requirements needed for deploying the Messenger app

- There needs to be a Facebook page representing the business identity and connecting to the people on Messenger.
- A Facebook Developer Account is needed for creating new apps that are the core of Integration provided by Facebook. Facebook Developers website can be used to get started on this.

- A Facebook app containing the setting for the app including the access tokens is needed for creating a new app by visiting the Facebook developers webpage.

#### For Creating a Facebook Page

- Make a Facebook account and log into the account. Create a page by giving out the required category and name and thus, creating a test page.

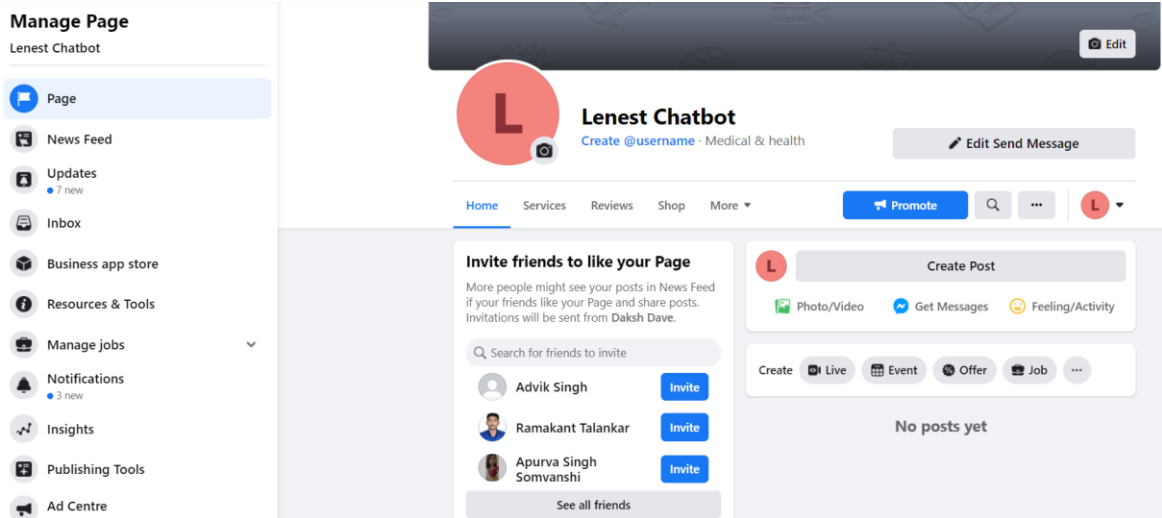


Figure24- Created Facebook Page

- Add the messenger button on Facebook named **Send Message**.
- The button added opens up the messenger chat. However, doesn't reply to the messages as the Dialogflow agent has not been integrated.

#### APP creation in Facebook Developer Account

- Register as a Facebook developer and create an app by visiting the create new app page on the Facebook Developer webpage.

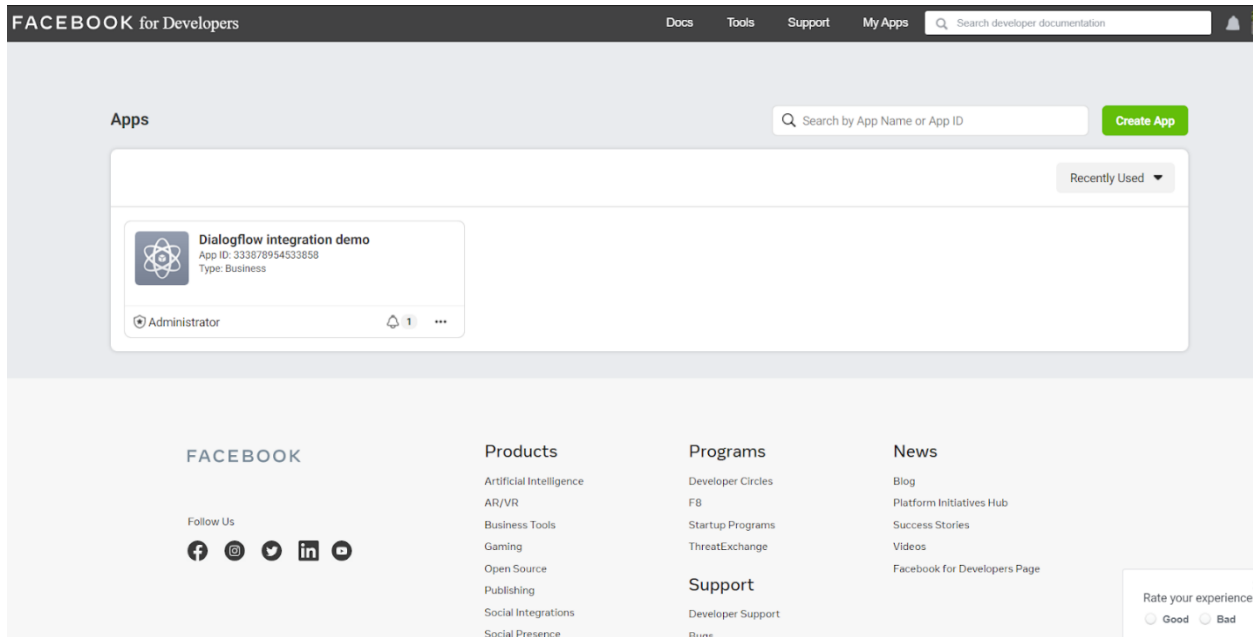


Figure25- App Page

- Click on **Create App** button.

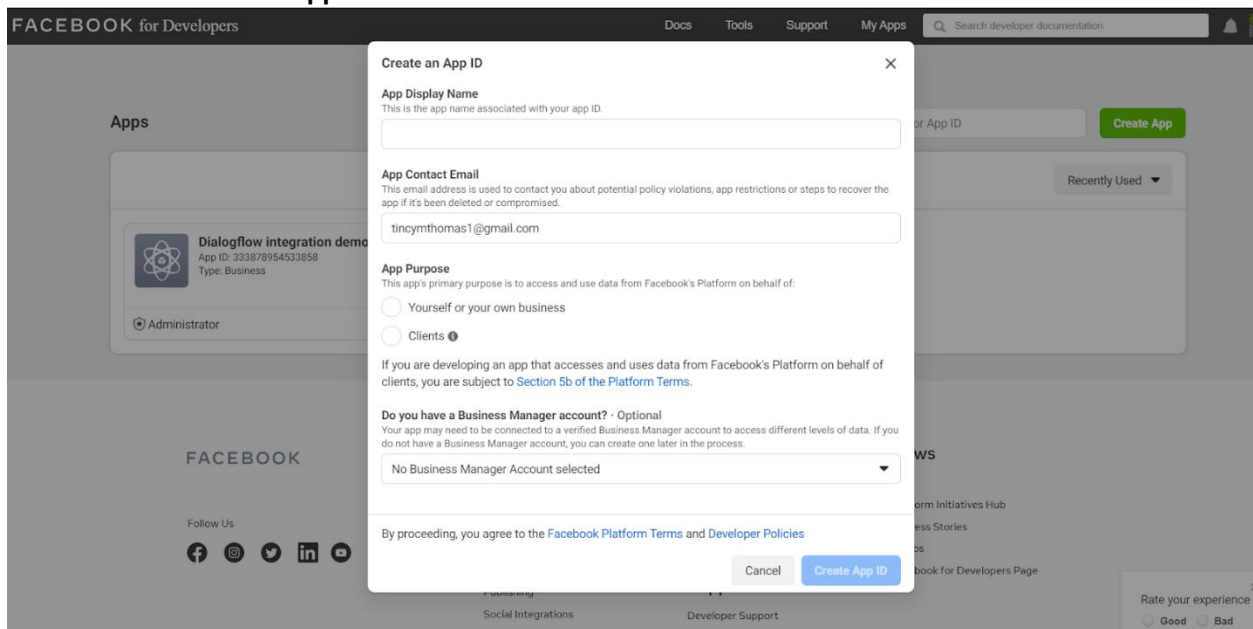


Figure26- Create an App

- Lenest Chatbot app name is selected, and email id is filled. The procedure gets finished by a security check. The app is created.

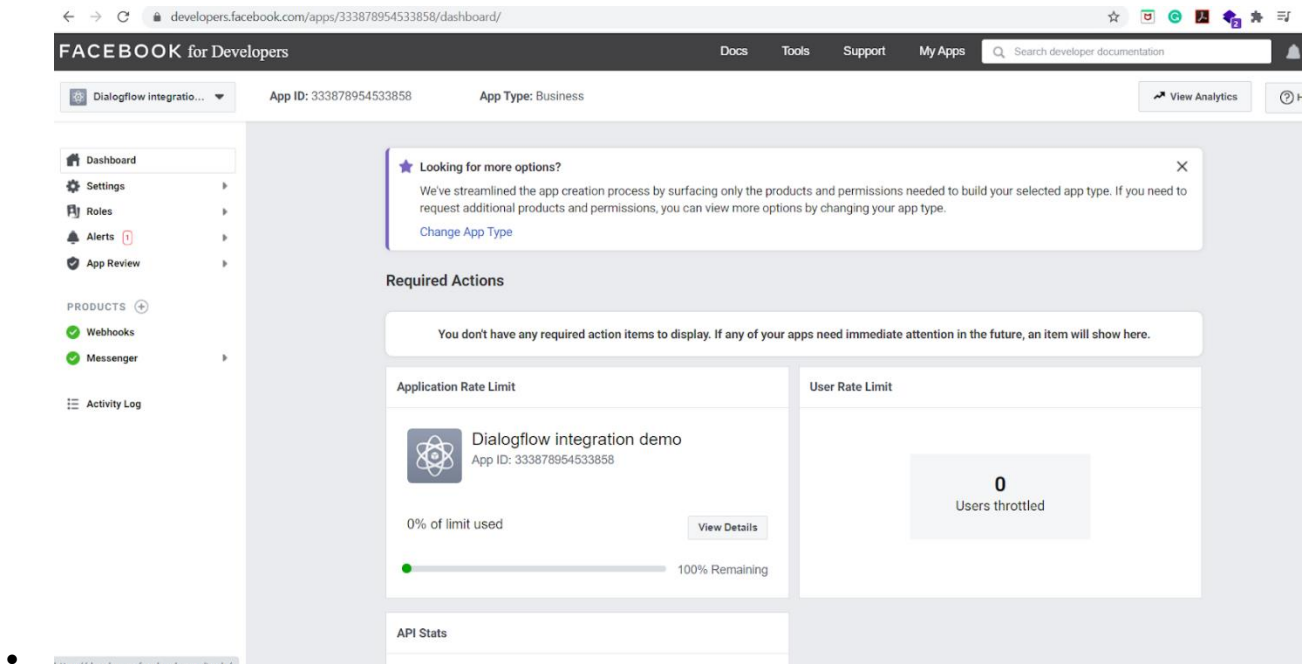


Figure27- The Created App

The Dialogflow agent is used to connect it with the Facebook App.

- To setup, the messenger click on the products

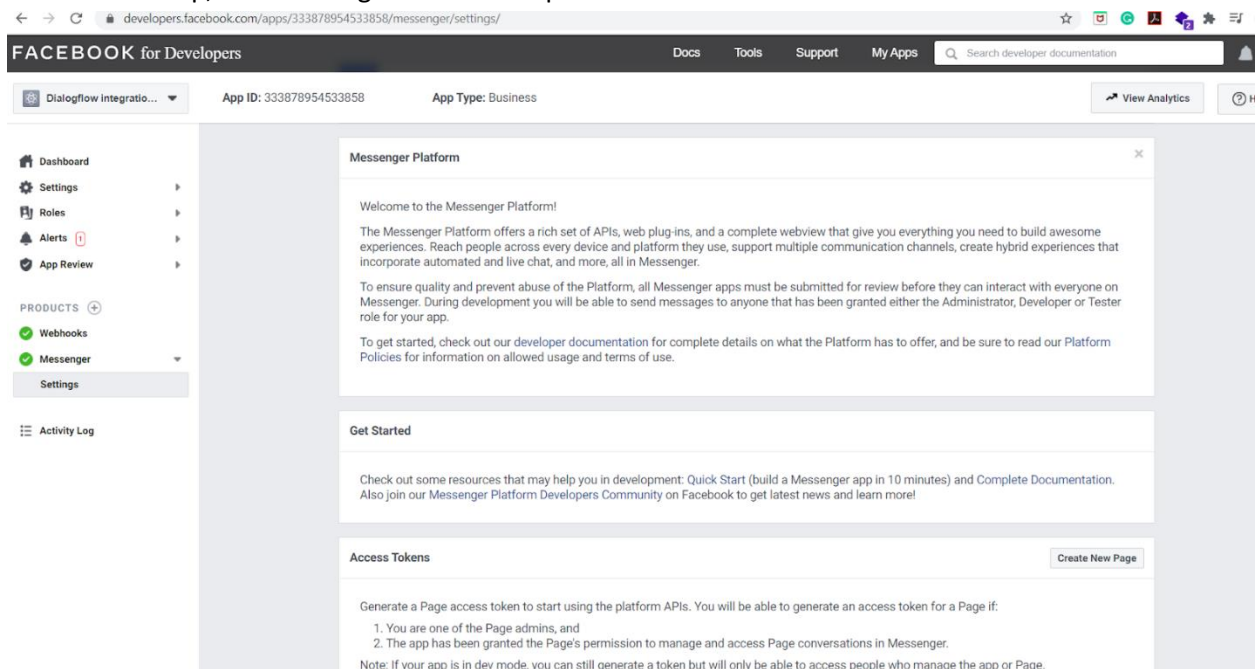


Figure28- Messenger Platform Intgeration Settings

- On the page click the Setup Messenger Button. After this visit the Dialogflow agent by clicking on the integrations button and clicking the Messenger from Facebook.

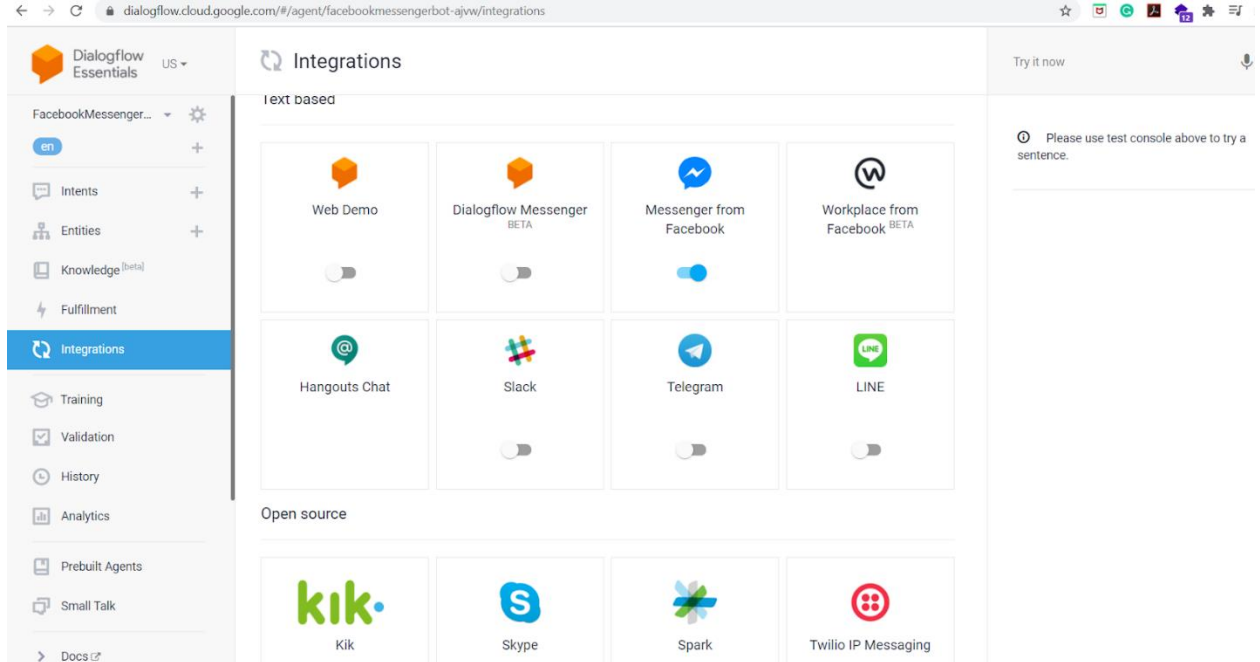


Figure29- Dialogflow Integrations

- The button will redirect to the page shown below that mentions the steps to fill it up.

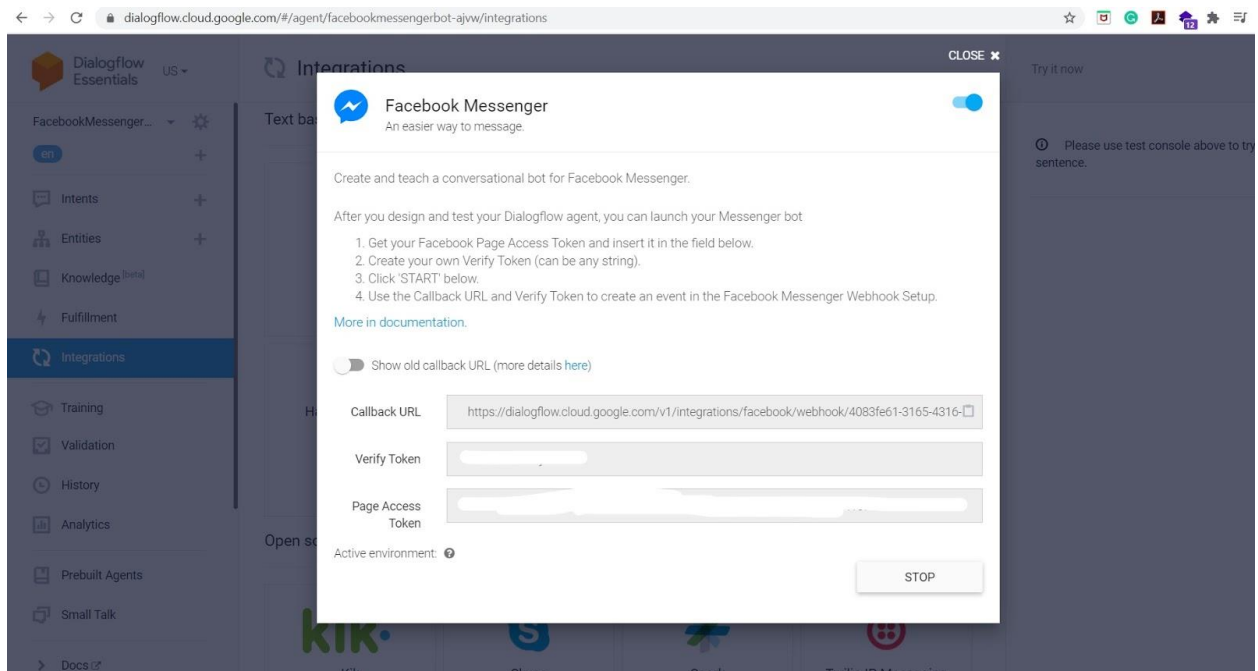


Figure30- Dialogflow to Messenger Integration



From the Dialogflow console, the integration gets configured.

- Dialogflow ES console is visited.
- Integration is selected in the left sidebar.
- Select Facebook Messenger.
- Configuration Dialog opens:
  - Integration toggle is selected at the top of the dialog for enabling the integration.
  - There is also an old callback URL that is shown defaults to off as a temporary option for accommodating the recent modification to the API of Facebook Messenger. The option gets removed after all the agents are updated.
  - A callback URL is provided that is copied for configuring in the Facebook messenger webhook.
  - Any desired private token can be entered and copied to configure the Facebook Messenger Webhook.
  - The copied token is known as the Page access token and is copied when the Facebook page is created.
  - Click to start this integration service for your agent.
- Using the Facebook developer account, the page access token of the Facebook page is obtained. Any name can be given to the Verify token field by providing a secure name. The bot then gets started after clicking on the start button.
- Redirect to the webhooks section in the Facebook Developer's account and copy the Verify token along with the callback URL from the Dialogflow agent and pasting that into the Facebook Developer's account.
- Tick the messaging\_postbacks and messages checkbox and click verify and save.
- In the webhooks section select the Facebook Page to integrate with Dialogflow and click on the subscribe button to connect the agent with the FB Page.

#### In the Facebook Page test the Chatbot

- View the Facebook page as Page Visitor. It is found that the chatbot is not publicly accessible. Only the admins and app creators can chat with the bot. The app needs to be published to make it accessible to everyone.

- Click the send message button and click on the test button. Then test by chatting on the specified intents pre-feed into the Dialogflow agent.

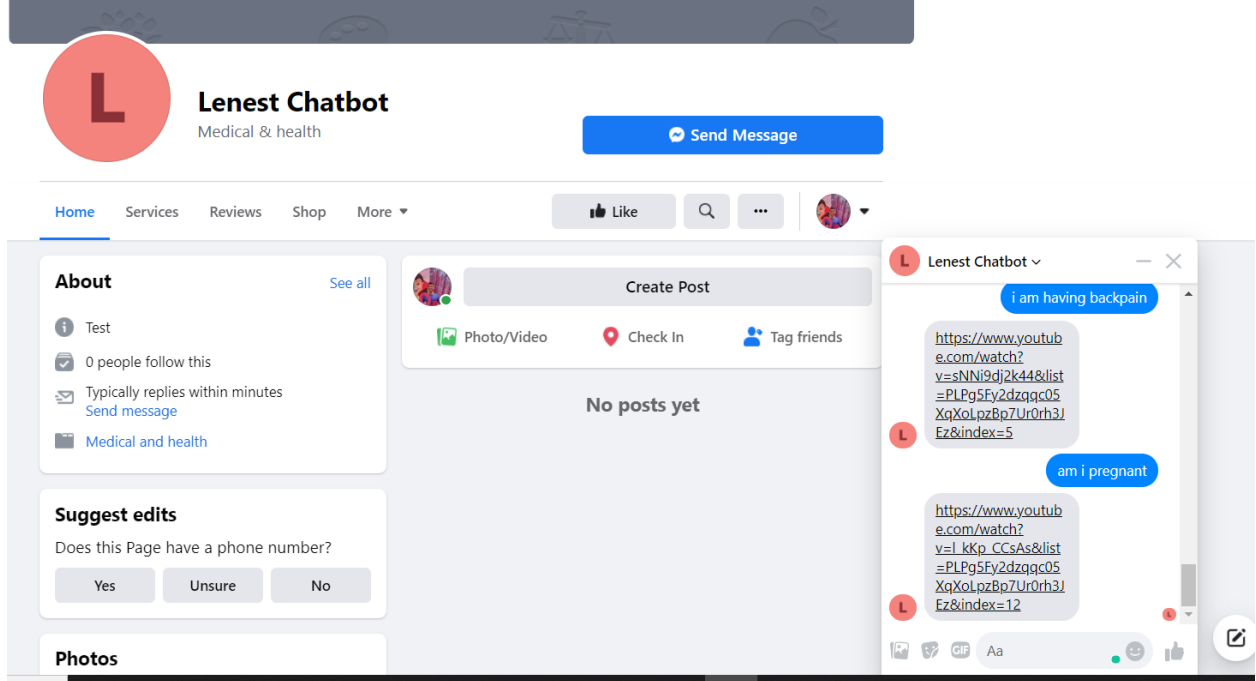


Figure31- Dialogflow Integrated Facebook Page

This completes the process of integrating the Facebook chatbot with the Dialogflow agent. Further, a test page along with an app is created for testing and getting a hands-on experience with Messenger. The app is ready to be deployed and gets submitted for the review process. The app gets ready to interact with the public once the review process is passed.

Now we need to integrate the agent with the website. The website named <https://lenest.in/> is already up and running. There are 2 ways this Dialogflow can be displayed on the agent:

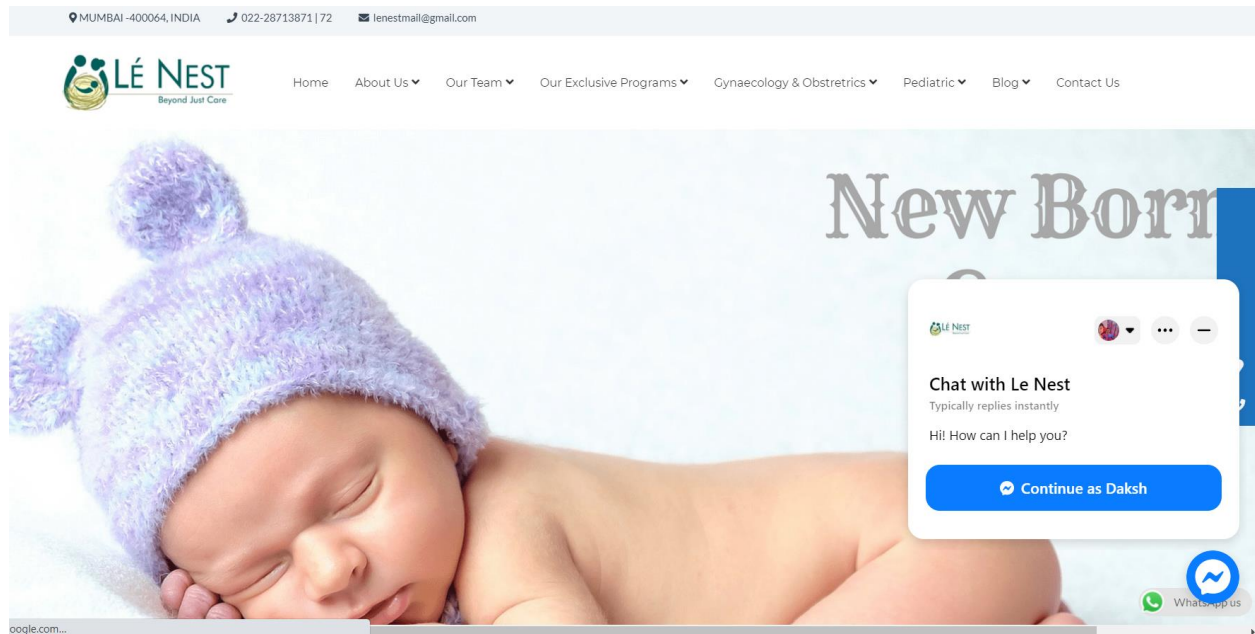


Figure32- LeNest Webpage

1. By directly adding the Facebook messenger bot built on the Lenest Chatbot Facebook page.
2. Going to the integrations in the Dialogflow console and then turning on the Dialogflow Messenger option.

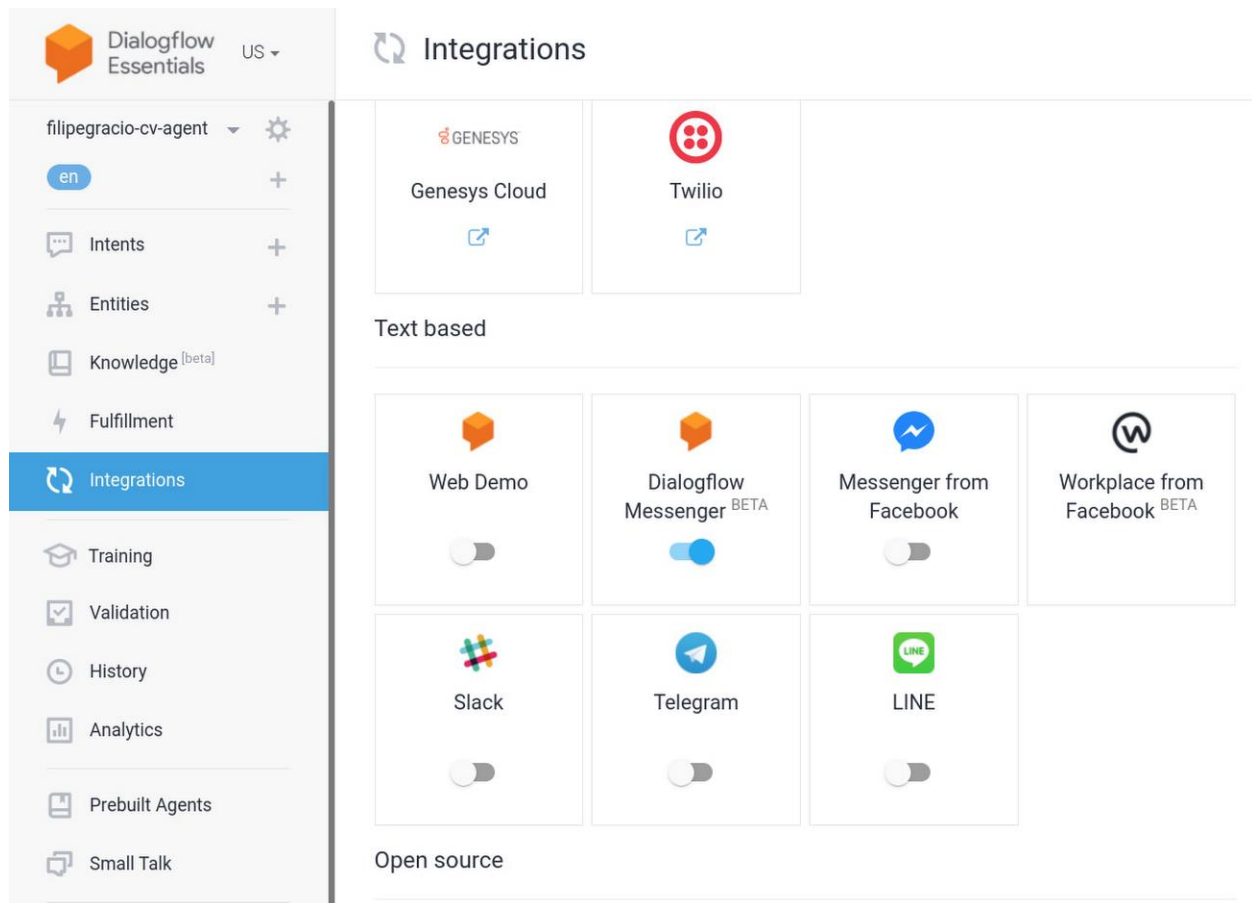


Figure33- Dialogflow Messenger Integration

Clicking on that enables a window containing a piece of code that needs to be embedded into the original website code. The following is the piece of code that is obtained:

Add this agent to your website by copying the code below

```
<script src="https://www.gstatic.com/dialogflow-console/fast/messenger/bootstrap.js?v=
1"></script>
<df-messenger
  intent="WELCOME"
  chat-title=" "
  agent-id=" "
  language-code="en"
></df-messenger>
```

Figure34- Agent code

The code is copied onto the clipboard.

8. The next step involves just putting up the agent on the site. The original code of the website needs to be embedded from the console of Dialogflow.

## Embed from the web

By URL Embed code

```
<script src="https://www.gstatic.com/dialogflow-console/fast/messenger/bootstrap.js?v=1"></script>
<df-messenger
  intent="WELCOME"
  chat-title="xxxxxxxxxxxxxxxxxxxxxx"
  agent-id="xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
  language-code="en">
```

Paste the HTML code from the site you want to embed.

Cancel

Next

Figure35- Embedded Agent Code

Now, the Website is ready to be published and made public with the agent ready to answer all the questions asked by the visitors.

The following training tools were used at different development stages

To refine the training at each stage, Use the Training Tool at various stages of agent development, and refine your training data at each stage:

- Before releasing the agent, it is tested on a small user group.
- After production, the conversations are closely examined to make sure the agent is robust.
- New updates made after production are closely observed.
- A regular analysis is performed by running the tool periodically

We update the agent intents through the Import of quality data.

Some useful data sources used for importing quality data are as follows:

- Conversation logs with humans
- Conversation logs of customer support through online mode like email, forums, FAQs, etc.
- Questions asked by the patients on social media.

The following types of data are avoided.

- Long non-conversational expressions.
- Irrelevant user expressions not related to any of the intents.
- Logs that are considering the end-user responses.

### **Challenges & Limitations of Bots/Study**

Chatbots are yet to be fully explored and are at a very basic developmental phase. The chatbots have deep-strong connected roots to Machine Learning, Deep Learning, and Artificial Intelligence hence there are several challenges that these bots face.

#### **Complex Interface**

There can be instances when a specific query matches multiple queries leading to the chatbot getting confused leading to extra consumption of time. There needs to be proper database management with robust architecture for handling multiple users at an instance.

#### **Time-Consuming Scenarios**

Generally, the user expects a quick reply from the chatbot however at times the right keyword is not found and the chatbot doesn't know what to reply so tries matching all the intents which is a time-consuming process. This delay aggravates when the system has a lot of users leading to a bad user experience.

#### **High Installation Cost**

Designing and developing a chatbot requires skilled expertise and a lot of manpower with constant changes/updates in the front-end and back-end functionalities. As the users increase so does the cost to manage the data and endure the smooth running of the server increases to prevent concurrency issues.

#### **Less Decision-Making Skills**

The chatbots are not extremely reliable in processes involving important decision-making. The architectures need to be more robustly built artificial intelligence instilled in the chatbots to make unbiased gender-neutral decisions and ensure fairness to everyone.

#### **Weak Memory Storage and Processing**

The user history accessibility to chatbots is a challenge as that requires a huge amount of data storage space for thousands of users to avoid sending redundant questions and answers. Previous chat data enhances the user experience and helps give them a personalized assistant feeling. This would also enable the chatbot to make more customized suggestions and recommendations to the users.

## **Benefits**

Chatbots require less manpower to operate when outsourced through an agency and serve as an important tool for E-commerce companies, Customer Support Agents, Online Marketing, and more. Chatbots are an important component in building good customer relationships. Nowadays, chatbots find great engagement rates in website pages and social media handles. The tool comes in handy while scaling businesses and analyzing the customer trends or requirements through chatbot conversations.

The following are the chatbot benefits:

- *Social Media Presence*
- *Quick response time*
- *Easy reach out*
- *Better engagement*
- *Personalized Experience*
- *Gather & analyze customer data*
- *Lead nurturing*

The chatbot supports an Omnichannel network. After a basic foundational network is built it can be deployed anywhere by seamlessly integrating across any of the digital channels and platforms. The end-to-end management system presents all the experiments, analytics, and evaluation without the requirement of custom software.

These bots interact accurately and naturally with the user thus, providing better customer experiences, in turn, improving patient value, adherence, and engagement. These bots are secure and provide constant virtual care to the patient. The data points generated are very useful in providing insights to the doctors through healthcare analytics. The operational data across the systems help the healthcare providers in making effective and confident decisions.

## **Improvements/Innovation/Future Scope of work**

The application promises huge future prospects in updating and scalability. Modifying and capturing appropriate data points like geographic site locations, patient directions from the customers can help modernize the patient-facing applications. The chatbot can be used to rack and integrate all the stakeholders that include the customers, partners, patients, and employees in the process of research studies and clinical trials. The application can easily be updated to accelerate the process of drug development by gathering the medical research data to enroll, engage and recruit participants directly through the application and keep track of their health along with specialized analytics through the data points.

Further, Virtual assistants also known as chatbots have hugely contributed to the digital world market. A research [8] states, *“there are nearly 5.19 billion unique smartphone users, 4.54 billion internet users via smartphones and 3.80 billion users actively using social media. On average every internet user spends around six to seven hours online on daily basis.”* The increasing count of people switching and engaging on online platforms acts as an important tool for connecting the customers to businesses. Almost every small or big firm is using Digital Platforms to stay connected to their customers in a better way and increase the customer-to-company pre and post-sales engagement time. As the markets keep expanding and new players venture out into the market. The need for a virtual assistant would keep on expanding exponentially.

The main challenge currently encountered by mathematicians, companies, scientists, and developers is to make the chatbot more engaging and to keep the conversations helpful without providing an unrelated response on encountering a new or a random query. The vision is to empower the bot into a personal assistant that can sense the need of the user to build human-like intelligence, experiences, and abilities. The next big challenge lies in ensuring that the privacy and data of the user remain intact. The extracted data should be used only to improve the user experience. The chatbots that we have developed are not just limited to a particular sector for a particular set of usage but can be scaled to other fields as well. The Deep Tech and Natural Language Processing technologies are still in their developmental stages to be implemented on a much larger universal scale and it would take a few years to come with a robust assistant that acts as a human-like personal assistant.

## **Conclusions**

The progress made by the present research work and development enterprises looks promising and can only improve our faith in the technology. The future for chatbot technology is heading towards its adoption in almost every small to large scale business. This paper discussed the pipelining and workflow of the code, intent formulation and matching, training, and testing the model to get the necessary output. The paper also presented the existing literature and research proceedings in this domain along with its industrial applications, challenges, and limitations to this technology.

Healthcare is one of the most important domains that would be impacted and greatly benefitted by the growth of technology. The proposed application solution would help the healthcare providers better understand the problems that are being faced by them and identify and rectify the loopholes in the value chain of booking an appointment to getting cured with the help of a personalized Virtual Doctor that would help hospitals in reducing the hassles of handling patients. By deploying this chatbot the hospitals, as well as other stakeholders, can immensely benefit through a positive behavioral change. The adoption of a well-trained chatbot would prove to be the number one priority for all organizations in the near future.



## References:

1. Kandpal, P., Jasnani, K., Raut, R., & Bhorge, S. (2020). Contextual Chatbot for Healthcare Purposes (using Deep Learning). 2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4), 625–634. <https://doi.org/10.1109/WorldS450073.2020.9210351>
2. Gelfenbeyn, I., Goncharuk, A., & Sirotin, P. (2018, April 17). Dialogflow ES DOCUMENTATION | Google Cloud. Retrieved February 24, 2021, from <https://cloud.google.com/dialogflow/es/docs>
3. Siddig, A., & Hines, A. (n.d.). A Psychologist Chatbot Developing Experience. 12.
4. Bahja, M., Hammad, R., & Butt, G. (2020). A User-Centric Framework for Educational Chatbots Design and Development (pp. 32–43). [https://doi.org/10.1007/978-3-030-60117-1\\_3](https://doi.org/10.1007/978-3-030-60117-1_3)
5. Ivanovic M and Semnic M. The Role of Agent Technologies in Personalized Medicine. In: 2018 5th International Conference on Systems and Informatics (ICSAI), 2018, IEEE, pp.299–304.
6. Hoermann S, McCabe KL, Milne DN, et al. Application of synchronous text-based dialogue systems in mental health interventions: Systematic review. *J Med Internet Res* 2017; 19(8): e267.
7. Fadhil A and Gabrielli S. Addressing challenges in promoting healthy lifestyles: the ai-chatbot approach. In: Proceedings of the 11th EAI International Conference on Pervasive Computing Technologies for Healthcare, ACM, 2017, pp.261–265.
8. Comendador BEV, Francisco BMB, Medenilla JS, et al. Pharmabot: A pediatric generic medicine consultant chatbot. *J Automat Control Eng* 2015; 3(2): 137–140. 9. Abashev A, Grigoryev R, Grigorian K, et al. Programming tools for messenger-based chatbot system organization: Implication for outpatient and translational medicines. *BioNanoScience* 2017; 7(2): 403–407.
9. Nadarzynski, T., Miles, O., Cowie, A., & Ridge, D. (2019). Acceptability of artificial intelligence (AI)-led chatbot services in healthcare: A mixed-methods study. *Digital Health*, 5. <https://doi.org/10.1177/2055207619871808>
10. Harwich E and Laycock K (2018). Thinking on its Own—AI in the NHS. (Reform). <https://reform.uk/research/thinking-its-own-ai-nhs> (accessed 2 November 2018).
11. Shangrapawar, A., Ravekar, A., Kale, S., Kumari, N., & Shende, A. (2020). Artificial Intelligence based Healthcare Chatbot System. 07(02), 3.
12. Rarhi, K., Bhattacharya, A., Mishra, A., & Mandal, K. (2017). Automated Medical Chatbot. SSRN Electronic Journal. <https://doi.org/10.2139/ssrn.3090881>
13. Srivastava, P., & Singh, N. (2020). Automatized Medical Chatbot (Medibot). 2020 International Conference on Power Electronics IoT Applications in Renewable Energy and Its Control (PARC), 351–354. <https://doi.org/10.1109/PARC49193.2020.236624>
14. Wang, L., Wang, D., Tian, F., Peng, Z., Fan, X., Zhang, Z., Ma, S., Yu, M., Ma, X., & Wang, H. (2021). CASS: Towards Building a Social-Support Chatbot for Online Health Community. ArXiv:2101.01583 [Cs]. <http://arxiv.org/abs/2101.01583>
15. Mathew, R. B., Varghese, S., Joy, S. E., & Alex, S. S. (2019). Chatbot for Disease Prediction and Treatment Recommendation using Machine Learning. 2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI), 851–856. <https://doi.org/10.1109/ICOEI.2019.8862707>

16. *Intelligent Chatbot for Medical Assistance in Rural Areas*. (2020). *International Journal of Innovative Technology and Exploring Engineering*, 9(6), 24–31. <https://doi.org/10.35940/ijitee.F3083.049620>
17. Vaira, L., Bochicchio, M. A., Conte, M., Casaluci, F. M., & Melpignano, A. (2018). *MamaBot: A System based on ML and NLP for supporting Women and Families during Pregnancy*. *Proceedings of the 22nd International Database Engineering & Applications Symposium on - IDEAS 2018*, 273–277. <https://doi.org/10.1145/3216122.3216173>
18. Kumar, S. A., Krishna, C. V., Reddy, P. N., Reddy, B. R. K., & Jacob, I. J. (2020). *Self-Diagnosing Health Care Chatbot using Machine Learning*. *International Journal of Advanced Science and Technology*, 29(05), 9323–9330.
19. Mugoye, K., Okoyo, H., & Mcoyowo, S. (2019). *Smart-bot Technology: Conversational Agents Role in Maternal Healthcare Support*. *2019 IST-Africa Week Conference (IST-Africa)*, 1–7. <https://doi.org/10.23919/ISTAfrICA.2019.8764817>
20. Omoregbe, N. A. I., Ndaman, I. O., Misra, S., Abayomi-Alli, O. O., & Damaševičius, R. (2020, September 29). *Text Messaging-Based Medical Diagnosis Using Natural Language Processing and Fuzzy Logic [Research Article]*. *Journal of Healthcare Engineering; Hindawi*. <https://doi.org/10.1155/2020/8839524>