Serverless Implementation of Conversational Agent for Delivering Customizable Surveys and Assessments

A Thesis

Presented to

The Faculty of the Graduate School

At the University of Missouri

In Partial Fulfillment

Of the Requirements for the Degree

Master Science Computer Science

by

Yohanes Patrik Handrianto

Prof. Yi Shang, Advisor

# table of content

[Table of content 2](#_Toc13791721)

[Dedication 4](#_Toc13791722)

[Acknowledgement 5](#_Toc13791723)

[Abstract 6](#_Toc13791724)

[1. Introduction 7](#_Toc13791725)

[1.1. Problem Description 10](#_Toc13791726)

[1.2. Proposed Solution 12](#_Toc13791727)

[2. Background and Related works 16](#_Toc13791728)

[3. TigerAware Assistant Design 19](#_Toc13791729)

[3.1. Conversational Agent Grammar 19](#_Toc13791730)

[3.2. TigerAware Assistant Architecture 23](#_Toc13791731)

[*3.2.1. Centralization of Database* 23](#_Toc13791732)

[*3.2.2. TigerAware Assistant Interpreters* 25](#_Toc13791733)

[3.3. New TigerAware Assistant Design 28](#_Toc13791734)

[*3.3.1 TigerAware Assistant Design* 28](#_Toc13791735)

[*3.3.2. TigerAware Assistant Architecture* 30](#_Toc13791738)

[*3.3.3. TigerAware Assistant system methodology* 31](#_Toc13791739)

[*3.3.3. Technology Stack* 32](#_Toc13791740)

[*3.3.4. Front-End and Client Side* 32](#_Toc13791741)

[*3.3.5. Back-End and Server Side* 32](#_Toc13791742)

[*3.3.6. Other utilities* 33](#_Toc13791743)

[4. TIGERAWARE Assistant SYSTEM Implementation 34](#_Toc13791744)

[4.1 TigerAware Web Dashboard 34](#_Toc13791745)

[*4.1.1. Account Management* 35](#_Toc13791746)

[*4.1.2. Builder* 36](#_Toc13791747)

[4.2. Server-side Implementation 39](#_Toc13791748)

[*4.2.1. Firebase Realtime Database and Types Implementation* 40](#_Toc13791749)

[*4.2.2. Authentication function Implementation* 43](#_Toc13791750)

[*4.2.3 Firebase Cloud function Implementation* 44](#_Toc13791751)

[*4.2.4. External API’s function Implementation* 53](#_Toc13791752)

[*4.2.5. DialogFlow Implementation* 54](#_Toc13791753)

[*4.2.6. Firebase cloud function fulfillment* 58](#_Toc13791754)

[5. TigerAware Applications 61](#_Toc13791755)

[5.1. TigerAware Assistant Based Diabetes Self-Management Study 62](#_Toc13791756)

[5.2. TigerAware Assistant Based PHQ9 questionnaire 63](#_Toc13791757)

[5.3. TigerAware Assistant Based MINI 64](#_Toc13791758)

[6. Conclusion and Future Work 65](#_Toc13791759)

[6.1. Conclusion 65](#_Toc13791760)

[6.2. Future Work 65](#_Toc13791761)

[7. Reference 66](#_Toc13791762)

# dedication

*To Professor Yi Shang*

*To Dr. Timothy Trull*

*To Nickolas Wergeles*

# Acknowledgement

I would like to thank and dedicate this paper to Professor Yi Shang. I’m truly grateful for his mentorship, guidance and support throughout my university day up to my early career. He has given me so many opportunity and support from me to grow and complete this research. I would also like to thank Tim and Denis for the input and advice regarding my research, they had given me so many useful information and advice I couldn’t have done this without their help. I would also like to thank Nickolas Wergeles for the advice regarding my research and being my partner in teaching students.

I would also like to thank everyone from the Computer Science lab, especially Jayanth, Luke, Will, Connor, Rui, Charles and may others. For helping evaluate my decision on the research process.

I would also like to thank Dale Musser and Justin Schuyler, who inspired my me. And giving their guidance since I was in my undergraduate study. Without their support and opportunity their given to me I would not be able to accomplish this research.

Finally, I would like to thank my family and friends. Without their support, trust and belief in me, I would not be here today.

# Abstract

The applications of chatbots, i.e. conversational agents, have accelerated rapidly in recent years. However, it is difficult, if not impossible, for non-technical people without programming skills to create and deploy customized chatbots. In this project, a new system, called TigerAware Assistant, is developed to enable non-technical people to easily create and deploy chatbots for delivering customizable surveys and assessments through conversations in natural languages. The system adopts serverless architecture based on Function-as-a-service (FaaS) cloud services, in particular, Google Cloud Functions, and Dialogflow, which supports natural language conversations. Fulfillment code is implemented to let Dialogflow agents call cloud functions on an intent-by-intent basis. During a conversation, fulfillment allows user to use the information extracted by Dialogflow's natural language processing to generate dynamic responses or trigger actions on the back-end. Building chatbots using this approach allows TigerAware assistants to generate dynamic responses based on information retrieved from databases or endpoint services. In addition, Chatbot creators can modify chatbots on-the-fly without interrupting the deployed chatbots. Chatbots created by the system can deliver surveys for data collection, psychology assessments through questionnaires, and cognitive behavior therapies. They can be deployed on iOS and Android mobile devices, smart speakers, such as Google Home and Amazon Echo, and any devices that support Google Assistants. Compared to existing technologies of building chatbots, this system greatly reduces the time of development, removes the requirement of programming skills, and is scalable, expandable, and robust thanks to the serverless computing architecture.

# Introduction

To be able to maximize the efficiency of providing and delivering information, researcher always trying to find the best way to do this to gather valuable data. Chatbot became very important part of business, research and many other industries. This led researchers on using or building application on mobile devices. It’s smartphones are cheap and widely available considering also have a minor learning curve, but this led us into major problem where there is many kinds of mobile devices and developer needed to do adjustment for each device. This could lead development cost become pretty high and also took some amount of time to develop multiple application. Not to mention we don’t have tools for non-programmer to build a chatbot easily and deploy them to multiple platforms.

Based on the latest survey on smart speaker which is conducted on December 2018 we seen a growth of 78% on ownership of smart speaker over the past year. 53 million people age older than 18 in the US own at least one smart speaker, this estimated to be 21 percent of the population. Researcher would have another medium that they can use collect data using this tool, we not only could take advantage of fast-growing market of smart speaker, but we could also deploy conversational agent to any device that could host artificial intelligent virtual assistant.

However, we don’t have tools for non-programmer to build a chatbot easily and deploy them to multiple platforms. In most cases to build a chatbot required programming intensive process or spending a sum of money to rent a service from chatbot provide. There is no easy way for users with no programming background to create or modify their conversational agent on the go. Current problem of developing stand-alone chatbot is first researcher had to design a conversation, then they had to decide which specific mobile platform they wanted to deliver their conversation in, then they had to hire a programmer to build standalone software that would contain the conversation. This methodology of development have multiple disadvantage including but not limited to; to deliver a product it would take months long software development process, it is not very cost efficient to researcher, if researcher wanted to make any changes they had to ask developer to change the conversation programmatically, conversation targeted into very specific field developer had to build new software to apply different conversation in different field.

In this paper I would like to present a new way of developing a conversational agent, with researcher and software development in mind. TigerAware Assistant designed to be a generic and customizable platform which allow researcher to create conversational agent which can create a survey for data collection, provide psychology assessment through questionnaire, and deliver cognitive behavior therapy without the disadvantage mention above.

TigerAware Assistant is important because it deliver this following feature; provide researcher without programming background to be able to build conversational agent, cut down development time from the researcher end since researcher would be able to create conversational agent without software developer intervention, provide researcher with the flexibility to deploy conversational agent through three most popular mobile platform IOS devices, Android devices, and smart speaker devices with one centralized software, researcher would be able to modified conversational agent anytime they want without needing to change the software programmatically, provide software developer ability to add more functionalities to the software without interrupting current deployed software.

In this thesis I will talk extensively about methodology, software structure, and technologies used. TigerAware Assistant was built from the need of a practical and faster way to build a conversational agent. TigerAware Assistant framework use Function as a service methodology or FaaS, FaaS allowing software developer to develop, run, and manage TigerAware Assistant functionalities without the complexity of building and maintaining the infrastructure typically associated with developing and launching an app. Building an TigerAware Assistant framework following this model is a way of achieving a serverless architecture. Serverless computing architecture let cloud provider runs the server, and dynamically manages the allocation of machine resources. Serverless architecture simplify the process of deploying code into production. Scaling, capacity planning and maintenance operations handled by cloud function provider. TigerAware Assistant written to be purely serverless and use no provisioned servers at all.

TigerAware Assistant framework is hosted using Firebase cloud function that would be the fulfillment to Dialogflow platform. Fulfillment is code that's deployed as a webhook that lets your Dialogflow agent call logic on an intent-by-intent basis. During a conversation, fulfillment allows user to use the information extracted by Dialogflow's natural language processing to generate dynamic responses or trigger actions on your back-end. Building Conversational agent using this structure allowing TigerAware assistant to generate dynamic responses based on information that is provided from database and endpoint services, allow records of user’s orders, and enable researcher to implement rules. TigerAware Assistant cloud function have the ability to utilize serverless NoSQL database that lets researcher store and sync data between your users in real time. All information that the TigerAware Assistant retrieved and serve will be recorded in the database.

TigerAware design is highly modular and generic allowing researcher flexibility to create, edit and deploy conversational agent, and also provide flexibility for developer to modify and adding functionalities to their program without needing massive effort of maintaining a server. TigerAware Assistant is a better solution for the current problem of creating a conversational agent software because it negates all the drawback of creating a conversational agent in traditional manner.

## 1.1. Problem Description

Researchers usually tackle the need to collect empirical data by either incorporating an off-the-shelf solution which are commercially available or creating a custom standalone solution each with their own set of drawbacks. Researchers prefer creating a standalone version of their application and dashboard to deliver a survey and integrate other external sensors necessary for a study, it take months to develop a capable conversational agent application and more time to develop a real time dashboard based on the number of dedicated developers working closely with the research team which consumes a significant portion of research budget. Any major change in the application would need a redeployment of application leading to significant delay in collection and delivery of information.

Commercially available systems on the other hand come with their own set of drawbacks. Firstly, many of these tools could cost a lot per month depending on certain user requirements, deployment platforms, and the application itself. And the ones that come free of charge only provide limited features and only for a certain trial period. Both free and paid platform need you to at least have a certain experience of programming, for researchers from other fields this would be a very steep learning curve. This would cost researcher to spend a lot of time to just build one specific application. They also come with a restriction on customizability.

Current Mobile application that tackling this problem needing still need massive adjustment to develop applications for the two most popular mobile phone application which is Android and IOS. With this software development work in two different platform comes challenges where developers have to constantly finding solution that is available for each platform. Library and capabilities in each operating system are different, one solution might work for one platform, but it would not work in the other platform. This especially appears to be a problem when it comes to developing android application, each mobile phone has their own specification, and this created a unique problem in each device. Different phones manufacturer and different software version created their own unique problem. This proven to costing developer a lot of time to produce robust software.

Current solution also only provides card base interaction, each page would be populated with one question with a set of answers, expanding the medium of delivering question and answer would be a better solution in a particular case. TigerAware Assistant excel in deploying chat base interaction whether through text or voice, this has proven to be a better medium in communication even compare to human to human conversation.

User sent a [greater number](https://www.sciencedirect.com/topics/computer-science/greatest-number) of messages to a chatbot than they did to another human, Specifically, human users(patient), either consciously or unconsciously, may have altered their language because they knew they were being evaluated for “humanness.”. Almost 30-fold average increase in profanity in the human–chatbot conversations compared to human–human conversations confirmed our hypothesis and is consistent with prior research.

This proven that most users that being asked with sensitive information would feel more comfortable to give their sensitive information when they are talking to a conversational agent rather than talking with a regular human being.

## 1.2. Proposed Solution

As Smart speaker devices and AI assistant become common and integral in our daily life, I believe this could be a solution to create multi-platform AI that serve the needs of researcher and also business. Smart speaker and assistant have become ubiquitous to our life almost all modern smart speaker has a version of AI assistant that can fulfil task by speaking to them or sending them instruction in the form of text. This allowed for a novel method of requiring information and providing services to accomplish research needs and also business needs. This not only cover a specific area of research and business but can also be applied and used in a very wide range of scenarios.

Although different research needs are varied vastly regarding this topic but in the most common circumstances they will include:

* + Recurring data input by a user on a device, such as providing answers to questions.
  + Collecting data from a device's internal sensors
  + Deployment to attainable devices without needing a very costly expense.
  + Managing multiple participants and their data across time.
  + Visualizing and analyzing collected data on a Web dashboard.
  + As fast of possible analysis result (on the go analysis)

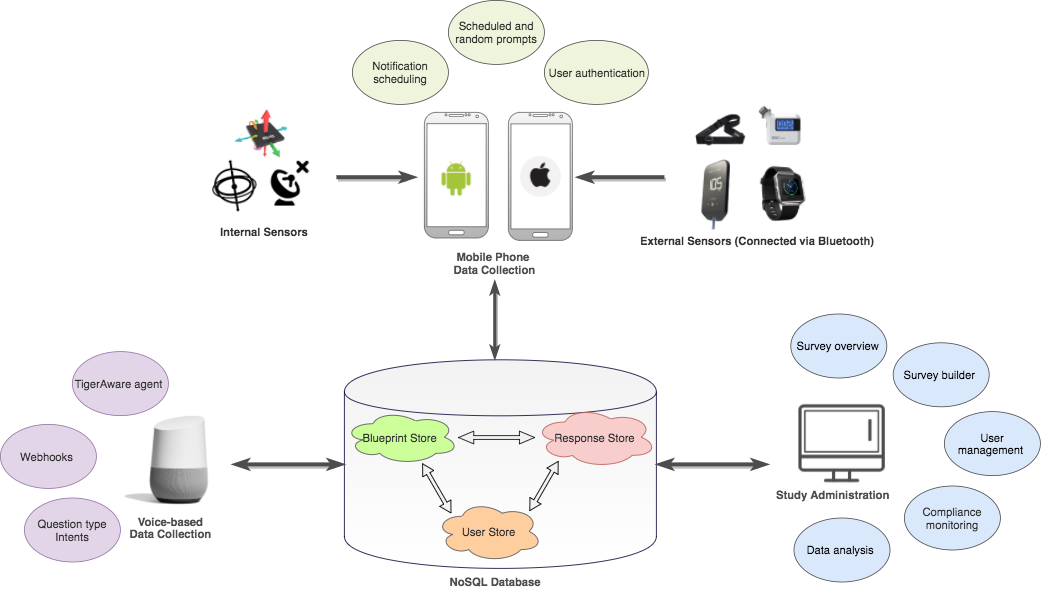


Figure 1. TigerAware System Architecture and Components

This diagram represents the solution to problem from multiple platforms, this represents TigerAware ecosystem as a whole. We had three branches where the center of it all is NoSQL database. This platform we deliver question and also store answers in the NoSQL database. In this paper I will concentrate mostly in Voice base solution where researcher would be able to utilize popular AI assistant to deliver conversation agent. Four main topics will be, how administrator would be able to utilize existing dashboard to create and deliver conversational agent, how cloud function translate stored data from NoSQL database to create chatbot and deliver it to AI assistant, how cloud function receive request from user and provide answers, how cloud function handle on the go analysis by using functionalities from third party API.

The AI Assistant or voice-controlled data collection side of the platform is used in the field to collect all responses and upload them directly to our NoSQL database, while the study administration, creation and management is handled on the web dashboard. During the development of the TigerAware Assistant, constant meetings were held with the client to discuss and understand functionalities that is necessary for them.

The cloud function technology develops using the latest technology of DialogFlow, JavaScript and firebase stack (firebase cloud function and firebase database). TigerAware Assistant provides researchers the ability to send create, deploy, modify any conversational agent through dashboard and send that information into DiaglowFlow to be deployed to Google Assistant where user would be able to use the services provided by researcher. TigerAware assistant services would be able to do real time processes, for example, verified user data in real time to make sure what functions are available for particular user and also send data to third party API to provide real time analysis.

TigerAware Assistant API rely heavily on Node.js as the run time environment which became the control center for all Third-party API services, this made it easy to set up, added features and handle other Application Programming Interface or API’s which improves the ability of the cloud function to perform various other tasks. This made it easier to exchange database platform and also easier to deploy conversational agent to multiple different platform.

This architecture prevents programmer from creating multiple program from different platform, since the API would be able to deploy program through AI assistant this meant it would be available in android, iOS platform and also multiple form of smart speaker without needing any modification towards the code.

The benefit of this deployment also opening TigerAware to another domain, other branch of tiger aware would able to deliver user a text-based solution where each page contains one question with all the choices that the user can select. New Tiger aware assistant allowing researcher to deliver not only text-based interaction but also voice-based interaction. It also provides different experience by delivering the question using a chat flow. Chat bot will provide the experience where user will be having a conversation with the “survey”.

The first iteration of TigerAware assistant was to solve one particular problem which is to deliver PHQ9 depression evaluation to diabetes patients. The structure of the program was very rigid and the assistant itself only deliver one set of functionalities (to collect data from user and determine whether the user had a depression or not base of score that they are received).  In this application we are only bale to serve patient with one set of survey.

The new and improved TigerAware assistant improves the research capabilities by providing one stop shop for researchers with the following contribution:

* Create conversational agent.
* Perform real time analysis.
* Real time information delivery.
* Easy to use dashboard that control Conversational agent deployment.

# Background and Related works

Several systems are currently available to try to suit the needs all of researchers providing tools to create questionnaire and survey. Some other platform of smart chatbot also trying to implement this tool. For instance, Woebotis Web-based cognitive-behavioral therapeutic (CBT) apps that fully automated conversational agent to deliver a self-help program for college students who self-identify as having symptoms of anxiety and depression. Thus, it may offer a convenient, engaging way of getting support at any time.

TigerAware is developed to offer a generic and customizable tool, which allows researchers to create surveys to collect a wide range of data, including but not limited to question responses, on device sensor data, such as GPS data, and external sensor data, such as blood alcohol level from a Bluetooth breathalyzer. TigerAware is highly modular and uses advanced Web and mobile technologies to incorporate diverse data sources with a rich set of survey question types, requiring little development work by researchers for their individualized studies.

Healthy Coping Assistant application, this project proposes Healthy Coping with Diabetes, a Google Home assistant application that acts as an innovative intervention strategy to assist elderly patients with self-management of type 2 diabetes. This application framework combines the voice interface of Google Home for hosting the conversational agent and a web interface for data visualization in order to reduce the burden of monitoring diabetic consequences for the user.

Building a Chatbot with Serverless Computing**,** presented a generic architecture for a chatbot framework built on top of Serverless computing platform. This architecture is extensible, inherently scalable, and supports different user interactions mediums. This platform inherently useful to learn how to create architectural structure of building chatbot using serverless computing but does not allowed researcher to easily modified, select and deploy a chatbot

REDCap is another such clinical research tool developed by Vanderbilt University that can be used to author, deploy, and analyze a wide array of clinical studies. It also uses a metadata-driven method of survey creation and deployment but is limited primarily to a medical context. REDCap allows for creation of projects backed by a SQL data store including five predefined tables such as metadata, logging, and regular data.

Each of these previous efforts proposes a system for generating custom surveys and conversational agent to address a specific research question, at times allowing for integrating outside functional sources. These works were essential in identifying the existing approaches, and how they can be utilized for developing the TigerAware Assistant. They were useful in designing an overarching tool for authoring Conversational Agent across various disciplines and yet be flexible to integrate new data sources, external API’s, and deployment devices to build a well-rounded cross-platform conversational agent system, which have been further explored in the following chapters.

Table 1 Comparing traditional and current available chatbot services Systems with TigerAware Assistant

| **Survey Platform’s / Functionalities** | **REDCap** | **Healthy Coping Assistant** | **Woebot** | **Survey Monkey** | **Chatbot with Serverless Computing** | **Existing TigerAware** |
| --- | --- | --- | --- | --- | --- | --- |
| **Text Survey** | Yes | Yes | Yes | Yes | Yes | Yes |
| **Response based Branching in chatbot** | No | No | Yes | No | No | No |
| **Conversational Agent** | No | Yes | Yes | No | Yes | No |
| **Researcher Modifiable chatbot** | No | No | No | No | No | No |
| **Deployable Sources** | Browser | Mobile | Mobile | Mobile | Mobile | Mobile |
| **Survey Templates** | Yes | No | No | No | No | Yes |
| **External Services** | No | No | No | No | Yes | No |

# 

# TigerAware Assistant Design

This chapter discusses the design of the Cloud function API for TigerAware Assistant as well as the factors considered in its design. The chapter also covers the benefits and drawbacks of the development technologies used for the web application.

## 3.1. Conversational Agent Grammar

To talk about data structure of delivering question to the user we have to talk about conversational agent grammar, as many realize this conversational grammar is the iteration of data structure design that has been created for TigerAware ecosystem with minor modification.

|  |  |
| --- | --- |
| Figure 2 An Example of a Step in TigerAware | Figure 3 An Example of a Conditional Step in TigerAware |

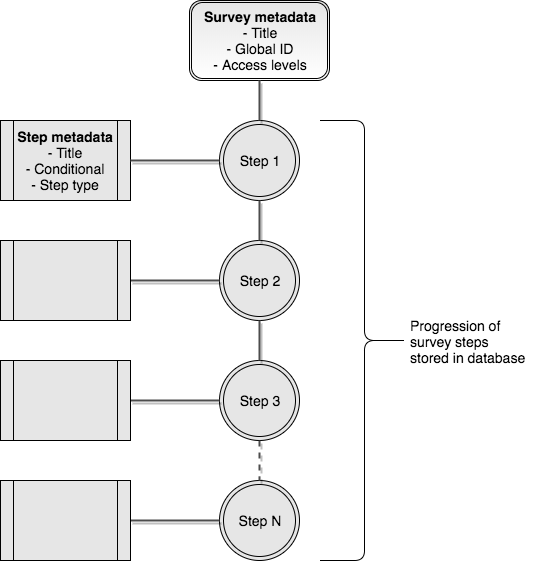


Figure 4 Survey Blueprint Structure in TigerAware Ecosystem

The structure where TigerAware Assistant is stored called blueprint, each conversational agent is composed of a series of questions, instructions, and functions trigger all formatted in JSON form stored in No SQL database. In the two figures above shows the structure that constructed blueprints. In this example of a question we call these steps where you have all the data needed to construct a conversation (question to a user as an example). Figure 2 show a steps where conversational gent simply ask to user a question “what is your name?”, each conversation will have a unique id that to identify which conversation will be delivered next. Type in will be use in conversational agent to determine which answer or information that is expected from the user and the default condition will show the id of next conversation that would be delivered. Figure 3 show conditional conversation in this conversation conversational agent will expect certain answer form the user and when it received answer with the condition it will trigger certain action.

In Figure 3 conversational agent will ask user “are you above 18 years of age?” and the conversation type will be YesNo, if the use answer yes conversational agent will jump to conversation with id q3, if the user answer no then conversational agent will jump to conversation with id q4, and if user decided to not answer the conversation or skip it conversational agent will jump to conversation with id q2. With this structure researcher will be able to create a conversational agent that branch to deliver multiple conversation based on what user said.

Table 2 Survey Grammar that Defines Survey Syntax

|  |  |
| --- | --- |
| ***survey*** | *[steps]* |
| ***steps*** | *conditional-step*  *step*  *step, steps*  *conditional-step, steps* |
| ***conditional-step*** | *{ conditional-step-attributes }* |
| ***conditional-step-attributes*** | *conditional-pairs, conditional-step-attributes* |
| ***conditional-pairs*** | *‘title’: string*  *‘id’: string*  *‘type’: conditional-types*  *‘choices’: choices-value*  *‘conditions’: conditions-value*  *‘conditionalDefault’: string* |
| ***choices-value*** | *array*  *null* |
| ***conditions-value*** | *[ condition ]*  *null* |
| ***conditional-types*** | *“MulitpleChoice”*  *“YesNo”* |
| ***condition*** | *{ “trigger”: value, “toID” : string }*  *{ “trigger”: value, “toID” : string }, condition* |
| ***step*** | *{ step-attributes }* |
| ***step-attributes*** | *step-pairs*  *step-pairs, step-attributes* |
| ***step-pairs*** | *‘title’: string*  *‘id’: string*  *‘type’: type-values*  *‘conditionalDefault’: string* |
| ***type-values*** | *“TimeInt”*  *“NumberPad”*  *“TextField”*  *“TextSlide”*  *“TimeOfDay”*  *“MultipleChoice”*  *“ContinousScale”*  *“Scale”*  *“DateTime”*  *“YesNo”* |

## 3.2. TigerAware Assistant Architecture

TigerAware Ecosystem build with modularity and generic design in mind, this resulted in broad cross platform integration, fast development for adding feature and, easy to use interface.

### *3.2.1. Centralization of Database*

Before talking about TigerAware Assistant it’s important to explain TigerAware centralize database structure that allowing developer to expand and adding functionalities really quickly. TigerAware's central database is a NoSQL database consisting of three key parts: a blueprint store, a response data store, and a user store. The blueprint store holds instructions to define TigerAware modules, as illustrated in Figure 4.In the current version, most modules are conversation or data collection modules. Conversation instructions are comprised of information such as the conversation name and instructions to make each step in the conversation (conversation blueprint).

Each step in a conversation could be a text slide, which delineates a section of a conversation, or a point of input like a yes/no question. Each step is kept as its own unit to allow the interpreter to decide whether it wants to combine the steps into a single interface or separate interfaces.

Currently supported step types, and their configuration. Each step may include configuration metadata, and all step types include step information such as the title and question label which are unique within the conversation. All parts of conversation instructions, including the conversation steps, are left generic to enable different interpretations of conversations. Despite the genericity of the conversation, all the information necessary to define the steps is conveyed in the instruction.

Researchers can add new step types by conforming to the step type interface. These step types could be traditional survey types, such as multiple choice or Likert scale, or more advanced types, such as weather or device location.

The second important component of the database is the data-store which keeps track of conversation response from the mobile device. The storage of response data alongside the corresponding blueprint data is important because the blueprint data enables interpretation of the conversation data.

The last element is the user-store used for storing user information. This is stored separately from the blueprint and the responses. If desired, these database parts can be stored in three distinct databases to ensure privacy of users, improve scalability, or isolate data pertinent to different use cases. In this way, the three stores function independently of one another to enable more modularity and functionality.

A NoSQL database was used because of its high levels of modularity and genericity. This provide developer with flexibility when it comes to refactoring, restructuring and expandability. If TigerAware blueprint created using SQL each survey required a different schema, since each step corresponds to a different column in a SQL style store. This correspond with heavy restructuring of SQL database every time developer needed to provide new field to support new functionalities

Moreover, we use Firebase as our NoSQL data store because it decreased development and deployment time. The process of creating a new Firebase database and connecting it to a new platform takes just a few minutes. Because the database is structured from the frontend interpretive nodes, after a Firebase database has been created and connected to the nodes, backend development is essentially complete. An added benefit is that Firebase can also be used on a large number of diverse platforms, including mobile and the web.

### *3.2.2. TigerAware Assistant Interpreters*

TigerAware's Assistant interpreters are comprised of three key parts: database interface, controller, Cloud function API caller and interpreter. This is the most important part of the program, to be able to translate structure data into actual conversation that is deployable, I create a cloud function that readily available whenever researcher creating a new conversational agent

The database interface reads or writes parts of the Firebase. A TigerAware interpreter Assistant that live as cloud function can trigger any read or write procedure to all of the three segments of the database. Because of the high level of modularity in the Firebase, the database interface segments of the interpreters are consequently highly modular and generic as well. This allowing cloud function extract only necessary information from TigerAware blueprint, confirm user identity, and write conversation result.

The second part of interpreters is the controller, controller takes what has been written and builds TigerAware Assistant conversation or conversation related entities. The controller is modular and comprehensive because the higher levels area based on the same characteristics. These features are also essential in the controller to maximize reusability.

The third part of interpreters is Cloud function API caller, by hosting the application as cloud function TigerAware assistant benefited from the rich API’s that lives in the cloud. TigerAware Assistant have the ability to call external cloud function and fulfill user request, this unlock the ability to do real time analysis and given developer the ability to attach existed functionalities easily. Consequently, this significantly reduce development time.

The final part of interpreters is interpretation. TigerAware Assistant cloud function is primarily used as the interpretation part. TigerAware Assistant cloud function is built for rapid modular development of conversational agent. To be able to deploy conversational agent to google assistant TigerAware Assistant convert blueprint into dialogflow webhook instructions, and dialogflow deploy conversational agent to Google assistant enabled devices.

Once a conversational agent is produced, the controller enables a user to take it at any time also enables researcher to modify the conversation whenever they need to without the needing extra software development. When user takes the survey, Dialogflow record all the conversation and TigerAware Assistant cloud function interpreted those result into the format of the blueprint.

TigerAware Assistant cloud function also control the conversational agent flow, TigerAware assistant is the be umbrella software that hosted multiple conversational agent that is avaiable for user to use. The controller receives the result and passes it to the database interface, which stores it in the Firebase. TigerAware Assistant cloud function thus provides a modular protocol for interpreting conversation instructions and producing conversation data from those instructions.

TigerAware Assistant cloud function was used in place to supplement IOS and android version TigerAware to enable chat flow conversation and voice base interaction. TigerAware Assistant cloud function currently have no other substitutes on the current medium, there is no other Interpreter equivalent that would enable ease of use, support third party functionalities and flexibilities to deploy conversational agent.

## 3.3. New TigerAware Assistant Design

### *3.3.1 TigerAware Assistant Design*

The web dashboard will help researchers view, create and modify their surveys, TigerAware Assistant control access to each survey by taking listed user database, and also handle post data collection to store it back into the database. The major design considerations that were considered right at the beginning of the design of the TigerAware Assistant were:

1. Modularity: as TigerAware Assistant framework is designed to cater a wide range of research studies, it is highly likely that new features will be requested based on the research requirement. By following a modular design philosophy, each new component can be developed and tested in isolation before being integrated to create a desired software system.
2. Extensibility: was vital component design choices made, as the current implementation, takes into consideration future growth. The use of cloud function structure allowing fast deployment and easy connectivity to other external services that is available. It enables plug-in architecture where you can modify data structure efficiently by adding new field without effecting current deployed software.
3. Usability: Users are highly habituated to using software and there is nothing more off putting than using unintuitive software and one’s especially with a learning curve, hence making applications that use intuitive design principles. This open up another medium for TigerAware ecosystem where user can chat with conversational agent and use our services. It enables not only text base conversation but also voice base interaction. This opens up our application to more user.

### 

### *Figure 5 TigerAware Assistant data flow*

### *3.3.2. TigerAware Assistant Architecture*

The architecture of the TigerAware Assistant comprises of a cloud function application hosted on Google cloud function Service (Firebase cloud function). The Assistant, which was developed using the a node.js, dialogflow and integrating with Firebase as the database for the platform.

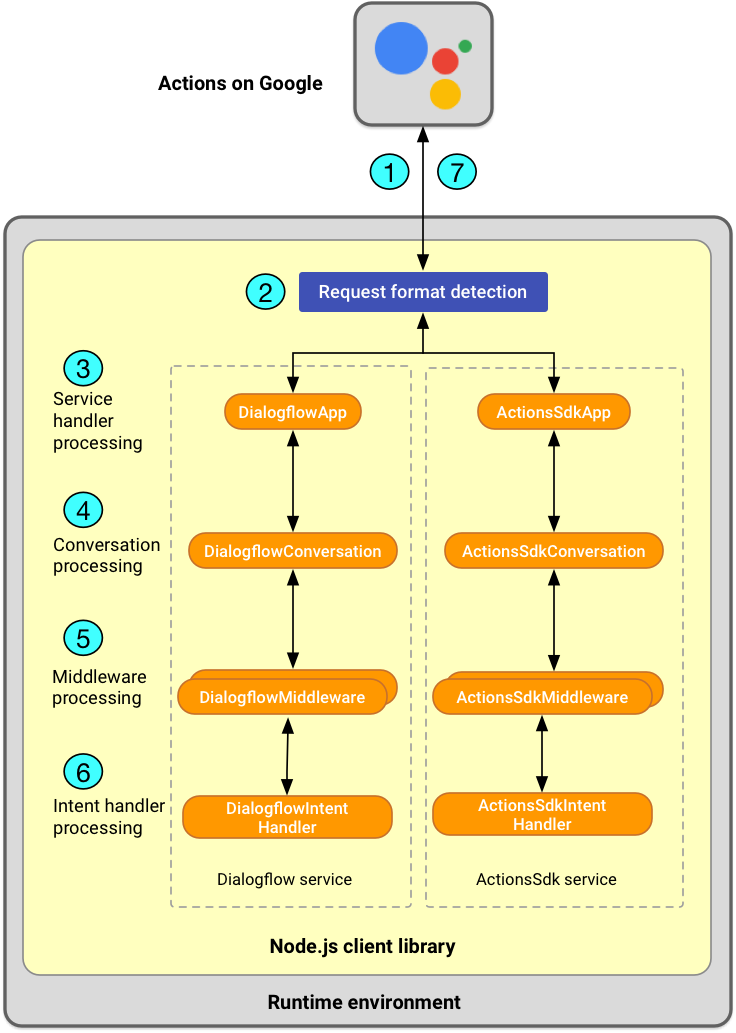
The figure below shows the overall TigerAware API architecture, Firebase Cloud function and node.js based backend provides all the necessary REST APIs which allow for information to be retrieved, saved, updated and transformed on the conversational agent.

Figure 5 TigerAware Assistant Architecture Diagram

### *3.3.3. TigerAware Assistant system methodology*

TigerAware Assistant build using serverless cloud computing methodology which aims to abstract server management and low-level infrastructure decisions away from developers. In this model, allocation of resources is managed by the cloud provider instead of the application architect. TigerAware Asisstant cloud functions allow it to be developed without concerns for implementing, tweaking, or scaling a server. Allowing developers to execute code in response to events without building out or maintaining a complex infrastructure. This Architecture simply allowed developer to upload modular chunks of functionality into the cloud that are executed independently. This allowing developer to build functions which can be scaled automatically and independently.

FaaS Advantages over traditional application architecture:

1. Server infrastructure management is handled by Cloud function provider minimizing development time.
2. Developer can focus on writing code or app specific logic.
3. Inherently scalable. Rather than scaling your entire application you can scale your functions automatically and independently with usage.
4. Never pay for idle resources.
5. Built in availability and fault tolerance.
6. Business logic is necessarily modular and conform to minimal shippable unit sizes.

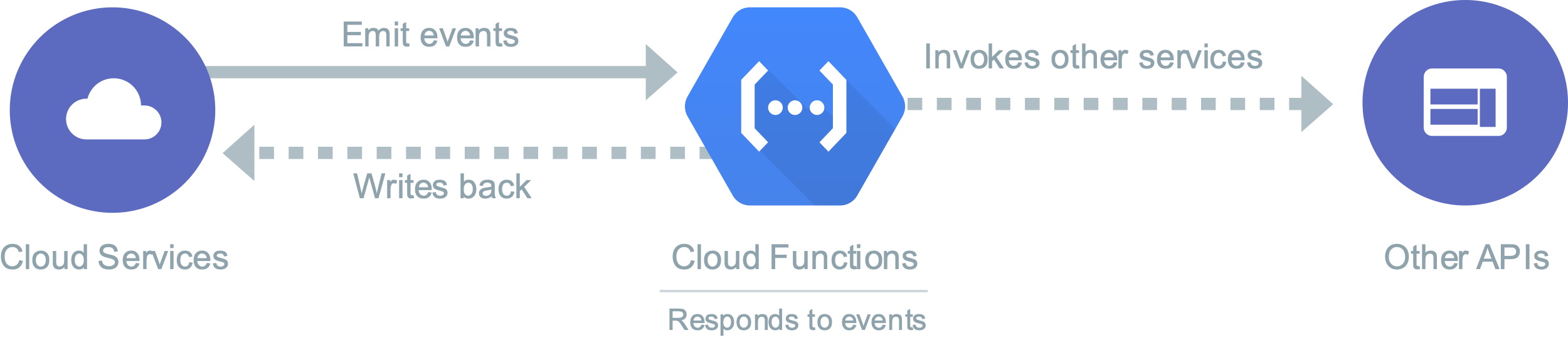


Figure 6 Cloud function diagram

### *3.3.3. Technology Stack*

There were multiple tools used for designing, developing and testing the application. They can be categorized into the front-end and client side, which were used for displaying the interface on the web dashboard and the back-end and server side, which are the storage, and languages used to interact with the storage; and other utilities, like the software programs and cloud services.

### *3.3.4. Front-End and Client Side*

1. HTML5: Markup Language on which the front-end interface is based for both the web dashboard.
2. CSS3: Materialize CSS implements the responsive design on the web dashboard
3. Javascript: AngularJS framework by Google, provides increased interactivity and background processing, with fewer page reloads along with modular features.

### *3.3.5. Back-End and Server Side*

1. node.js: A server-side scripting language capable of multi-processor deployment.
2. Firebase database: Handle data storing
3. Firebase Cloud function: A server-side scripting language enable API to be hosted on cloud to control reading and writing data, interpreting data into conversational agent
4. Google cloud function API: Library of cloud function that provide multiple different services

### *3.3.6. Other utilities*

1. Visual Studio Code: A highly capable and feature-driven text editor
2. Git and GitHub: A file-sharing, collaboration and version control service widely used among the open source community to maintain the code base and back the code up regularly and securely
3. FileZilla: A secure and reliable FTP software to transfer files between the local system and the remote server and vice versa
4. Firebase CLI: technology used to o deploy functions to the Cloud Functions runtime
5. DialogFlow: Natural language processing library that us used as deployment platform

# TIGERAWARE Assistant SYSTEM Implementation

This chapter focuses on the implementation of TigerAware Assistant cloud function and detailed explanation of each and every part implemented in the application. The following topics will discuss features in the applications, key architectural considerations and its implementation and the overall flow of data from front-end dashboard, to Firebase to the backend API and to the google assistant view.

The work done to improve the platform can be divided into improvement made on the server side of the application and the client side of the application. To be able to understand the process of creating conversational agent this section will start with explaining from Administrator side of creating a survey than goes to database and lastly to the server side.

## 4.1 TigerAware Web Dashboard

The TigerAware dashboard plays an important role in the entire platform as it allows the administrators to view, edit and create conversational agent. The dashboard is developed using AngualrJS to follow MVC architecture in the front end of application development. The interface was styled using Materialize CSS3 library to improve usability and the innate responsiveness feature is used for providing multi-platform access to the dashboard like, desktop, mobile phones and tablets. The application relies on APIs that are created using ExpressJS framework to establish a communication channel between the front-end and back-end of the platform.

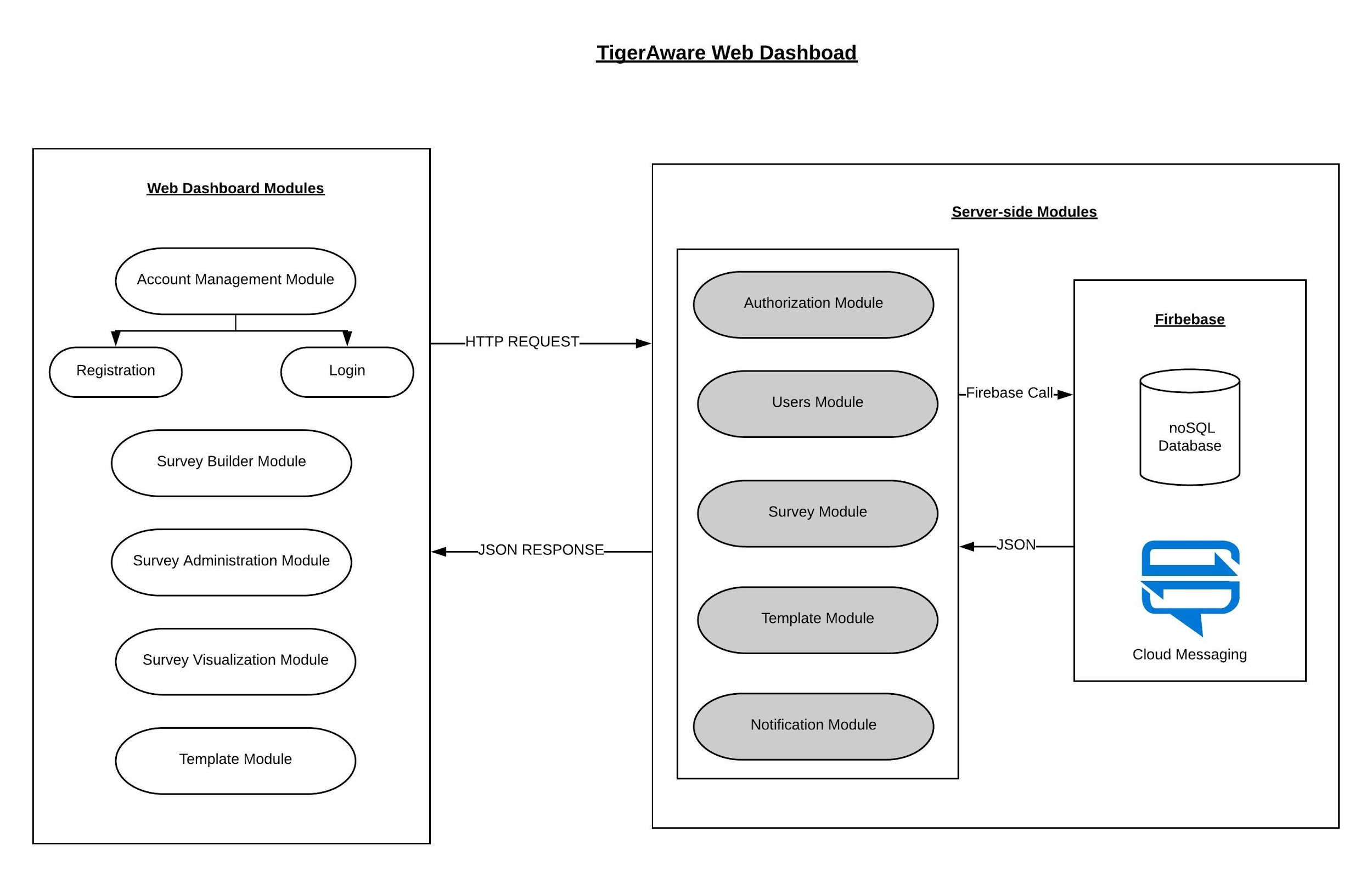


Figure 7 TigerAware Dashboard Implementation

### *4.1.1. Account Management*

Before being able to use the dashboard, users will need to register and login into the application. The first page that any user will come across in the website is the login page. For a first-time user, there is an option to sign up, where they are required to fill out some basic information (name, email address, username and password).

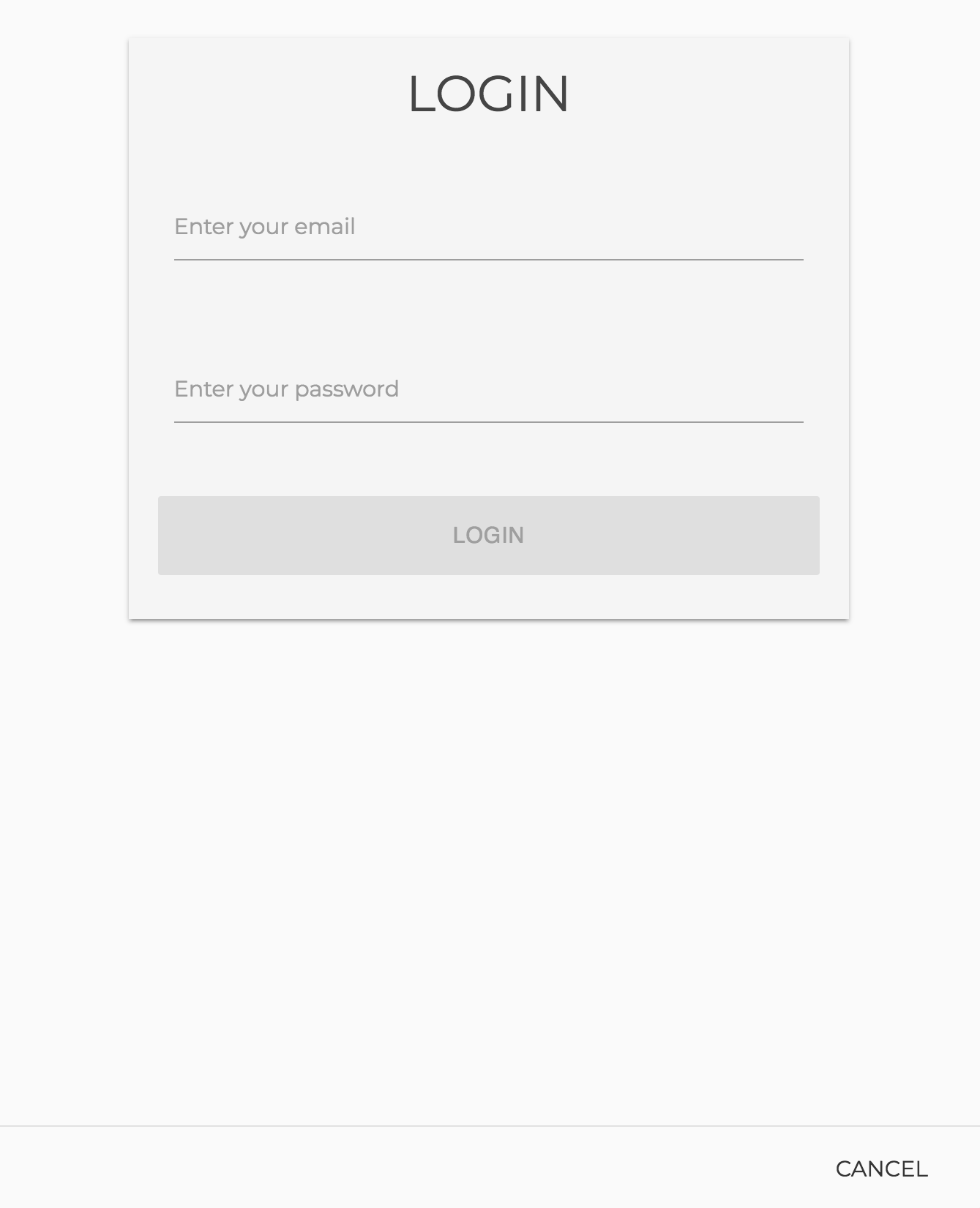
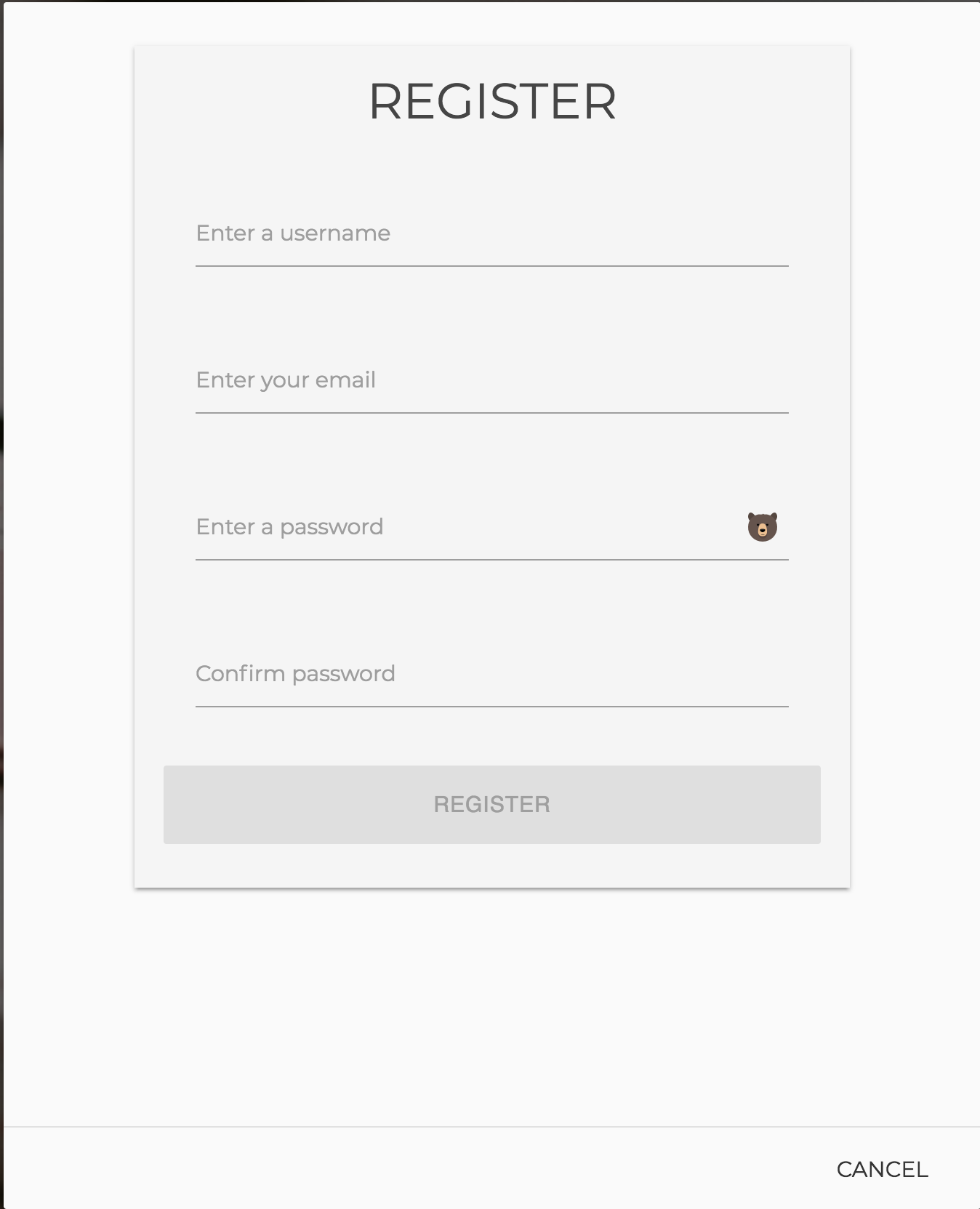


Figure 8 Login and Registration Interface

Security is paramount when creating accounts and creating a secure password prevents any unauthorized access to the application. Firebase as a platform made it easier to create an authentication mechanism for the application and also perform user management using its console. The entire authentication process for registering new users, session management and password management usually implemented on other systems can be reduced few simple function calls.

### *4.1.2. Builder*

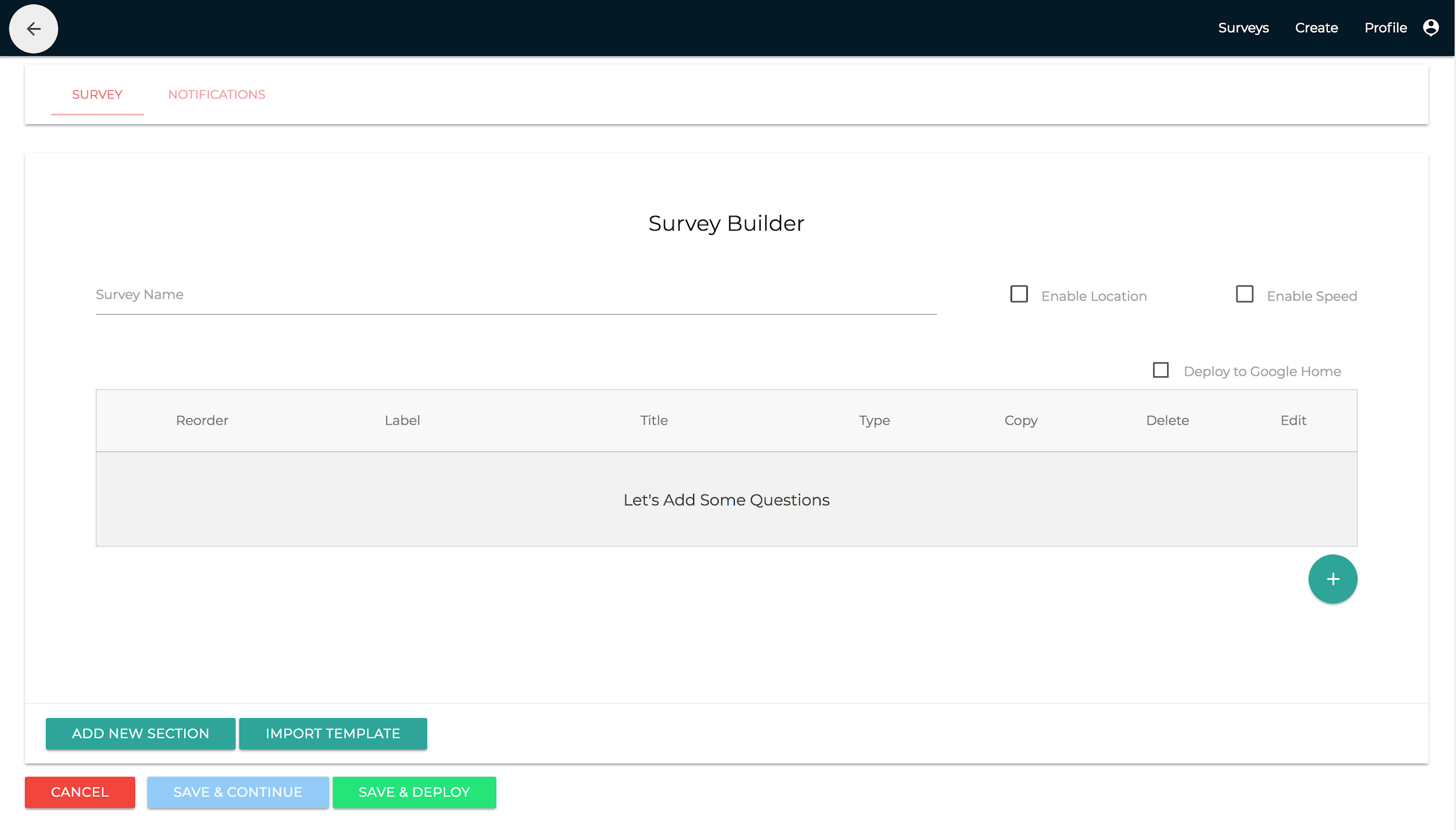


Figure 9 Survey Builder Interface

Current TigerAware Assistant does not have a stand-alone dashboard to conversational agent, currently we are using the same survey builder and standard that has already existed in the TigerAware ecosystem. The new conversational agent builder is in the work to provide administrator with more intuitive experience.

The conversational agent Builder is important part of the dashboard as it allows users to create complex conversation for the participants to undertake. The user can navigate to the conversational Builder page either by clicking on the Create Survey button on the overview page or by clicking the edit button of a pre-existing conversational agent on the overview page.

The Conversational Agent administration module allows administrators to perform administrative operations corresponding to each of their Conversational Agent. An administrator can add users as participant of the survey. It has an interface to show all the users currently in the database. Administrators are the only users who can view, add and edit a participant to their survey.

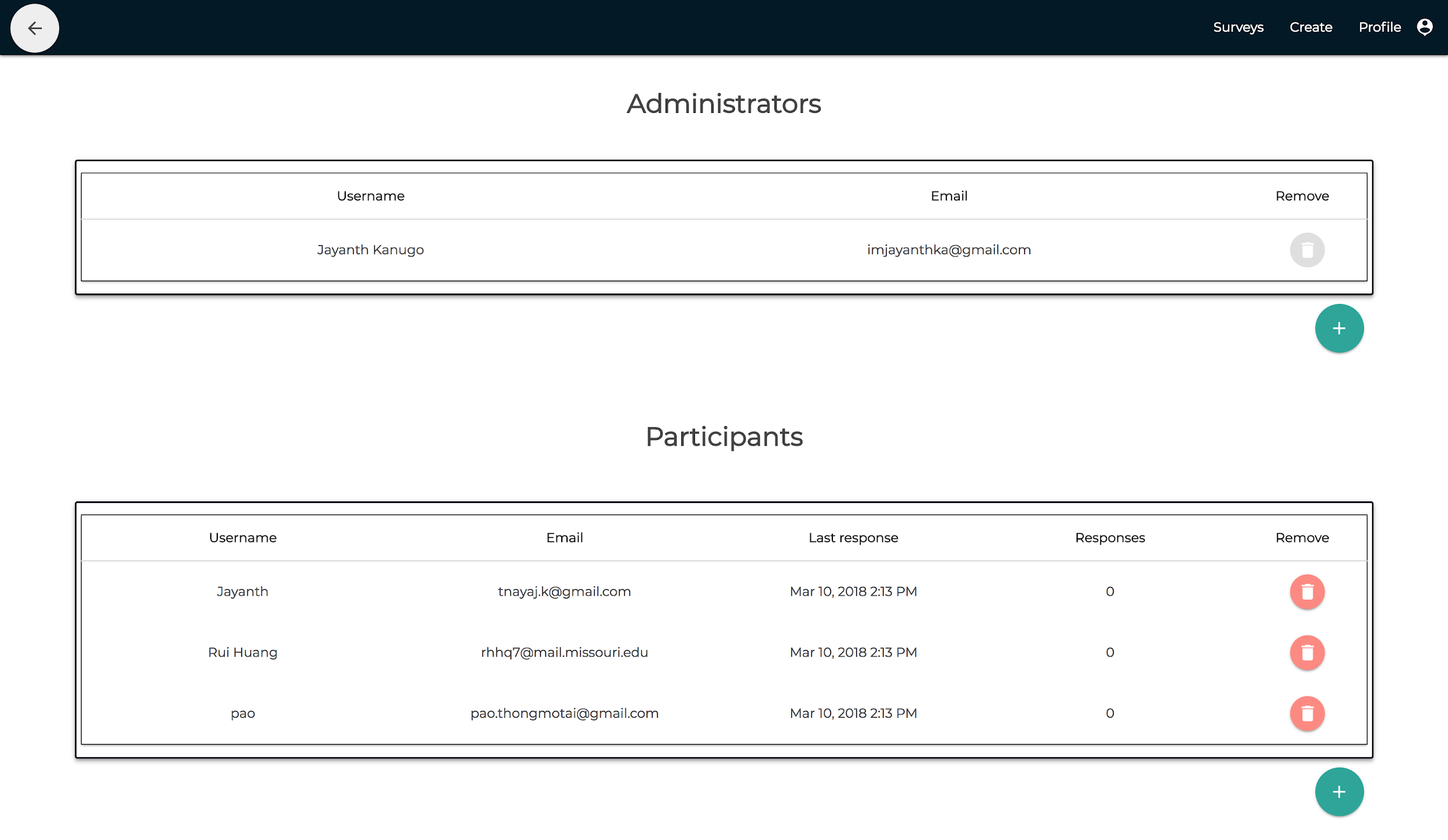


Figure 10 Survey Administration Interface

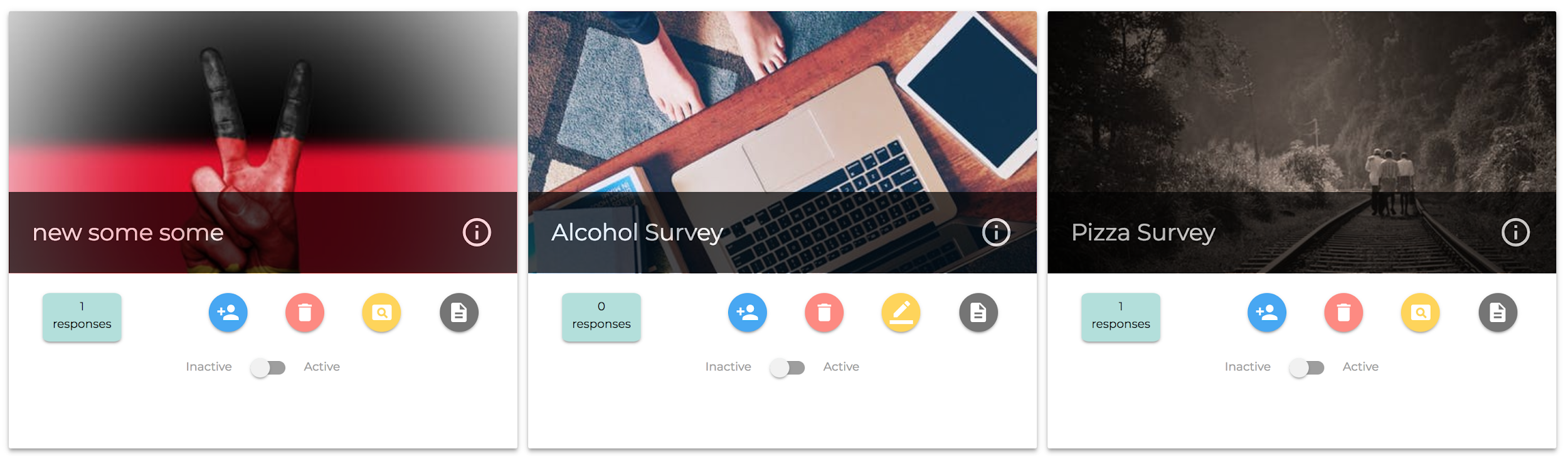
It provides the ability to the administrator to control the access associated to each of their surveys. They can disable any of their currently deployed surveys with the flip of a switch.

Figure 11 Survey Overview Interface with Ability to Activate or Deactivate a Survey

The survey builder is full featured to support development of conversational agent for Google assistant deployment. In the Survey Builder tab, the administrator can add a question by clicking the Green ‘+’ Floating Action Button(FAB). The ‘+’ FAB would launch question-modal show in which would act as the central place for all the information pertaining to a question in the survey. surveys have access to a full array of question types, currently the dashboard allows for 12 different questions types.

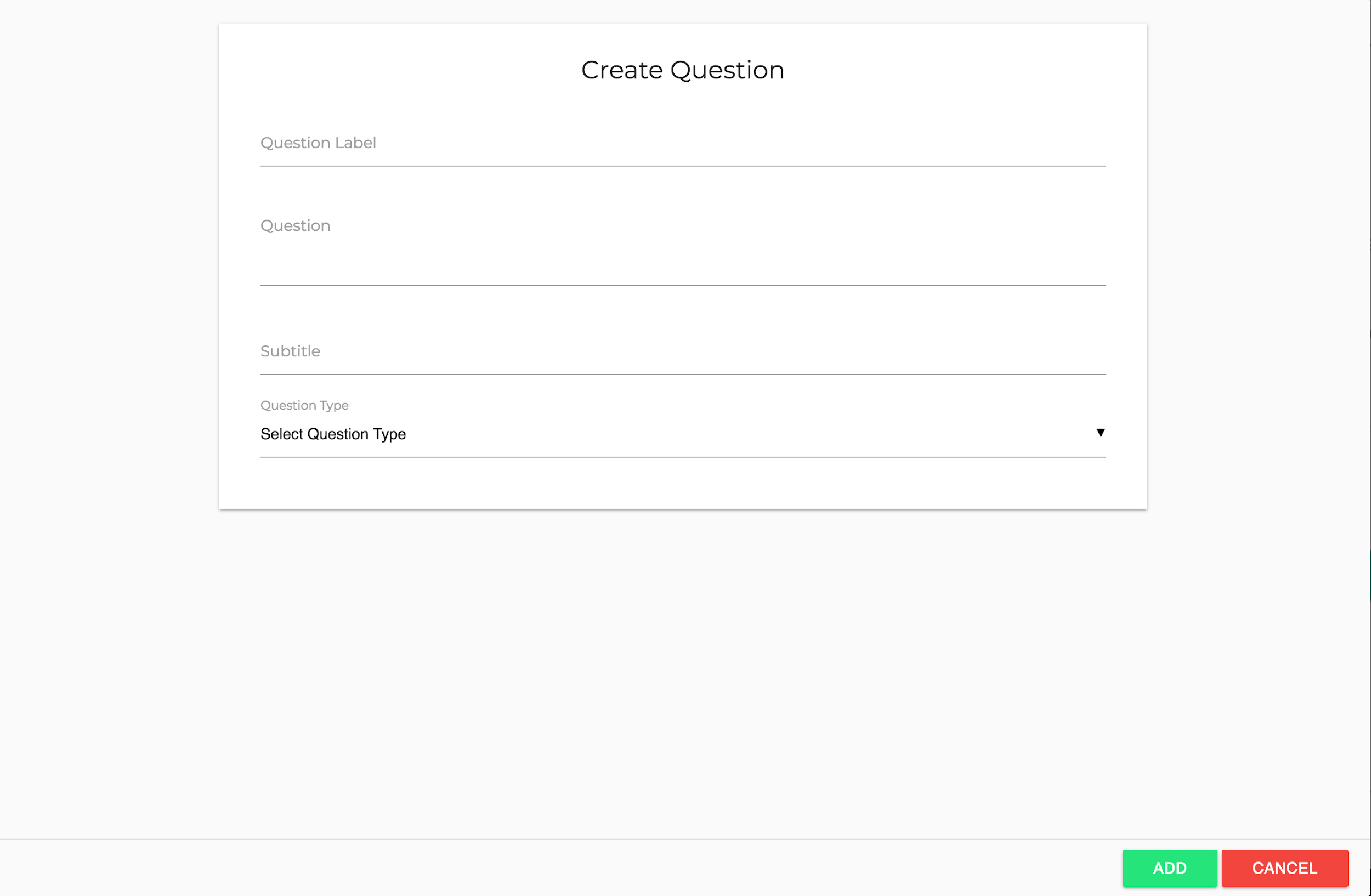


Figure 12 Create Question

Currently the dashboards support branching on four different question types. Any new questions added to the survey is checked to maintain unique question id. Based on survey grammar mentioned, the function uses the *conditionDefault* *step-attribute* is updated either with question id of the next following question or *^eos* a special character used to indicate the end of survey. The convention helps TigerAware Assistant to render the necessary conversation in the required order. Similarly, when the administrator wishes to edit a created survey *deserialize* function flattens graph to recreate all the objects necessary to render the survey builder page.

## 4.2. Server-side Implementation

The server side of the application concentrates on the business logic involved in the application. It holds modules that provide various API’s necessary for the Cloud function to deliver different functionalities. The server relies on the Firebase Cloud function library, a minimal and flexible Node.js web application framework that provides a robust set of features for web and conversational agent applications. It also integrates with Firebase Admin SDK on the cloud function side as it allows us to interact with Firebase from our cloud function environments to provides content management privileges, allowing external API functionalities, data delivery, data retrieval and various other.

### *4.2.1. Firebase Realtime Database and Types Implementation*

The conversation in blueprint holds all the information necessary for the client-side applications to render a survey to the participant. It’s a combination of survey metadata and survey information to handle different use cases corresponding to the survey.

The Firebase Realtime Database is a cloud-hosted database. Data is stored as JSON and synchronized in Realtime to every connected client. When you build cross-platform apps with JavaScript SDKs, all of clients share one Realtime Database instance and automatically receive updates with the newest data.

Structure of the database is can be visualized as a cloud-hosted JSON tree. Unlike a SQL database, there are no tables or records. When you add data to the JSON tree, it becomes a node in the existing JSON structure with an associated key.

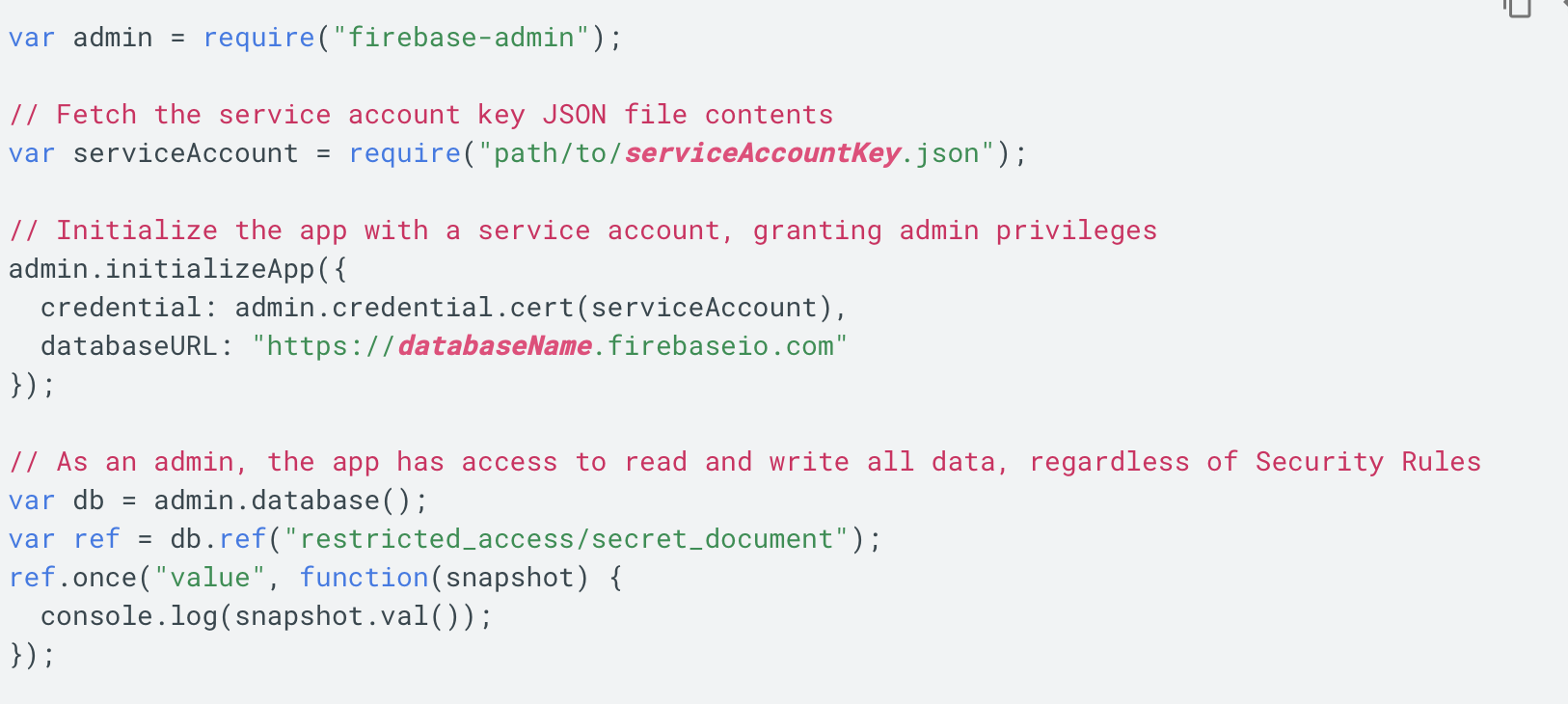
Before you can access the Firebase Realtime Database from a server using the Firebase Admin SDK, you must authenticate your server with Firebase. When you authenticate a server, rather than sign in with a user account's credentials as you would in a client app, you authenticate with a *service account* which identifies your server to Firebase.

Figure 13 Authorization method to let cloud function communicate with database

When an administrator created a conversational agent in dashboard it will be saved into a structured data in blueprint, the following is the sample conversational agent that is created by dashboard and deployed.

Figure 14 Blueprint data structure and user data structure

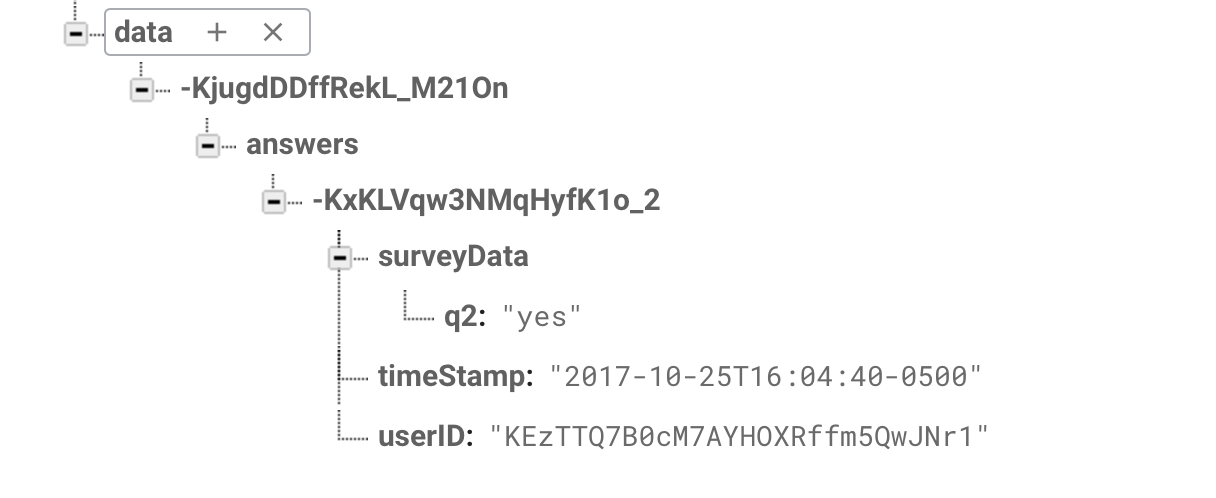
In the survey section it represents the steps of each conversation, each conversational agent represented with a unique id when stored into the blueprint, and in the user key section. From the picture shown above in the data structure we also stored user information which consisted of email, surveys that they are created, survey they are participated (Taking) and username.

Figure 15 Structure of data stored

Lastly but not least we also have data section which store all user responses, each data related into the conversational agent that created and identified using a randomly generated key.

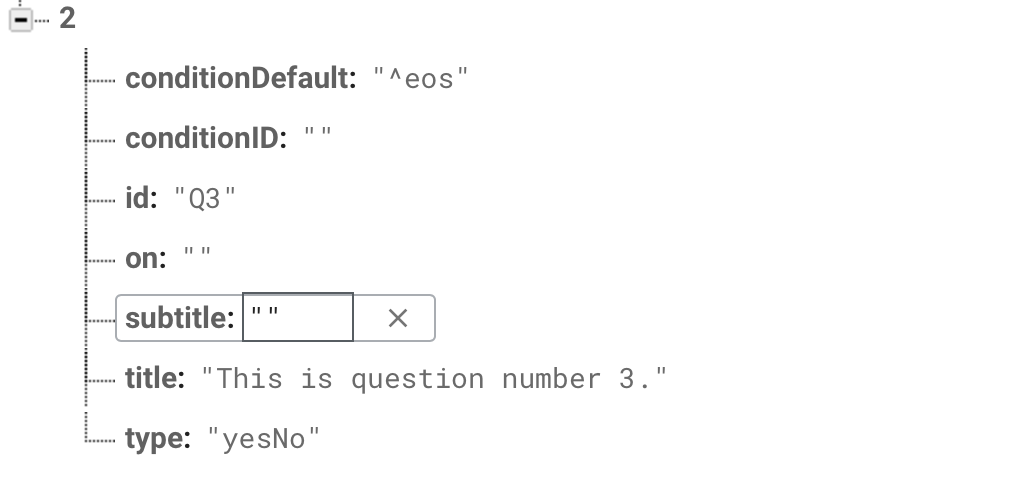
In each conversational step we have a field called type, this is defining the expected response that the conversational agent expected to receive. In the example below the type is yes or no question this particular conversation conversational agent supposed to received yes or no as an answer of the conversation.

Figure 16 Example of conversational step

There are multiple data type that the conversational agent would be able to receive. These nine types of data type would be able to be interpreted by the conversational agent:

* “TimeInt” : Duration of time answer
* “Number” : Taking number answer
* “TextField” : Taking free form answer
* “TextSlide”: Deliver voice or text response
* “TimeOfDay”: Taking time answer
* “MultipleChoice”: Taking multiple choice answer
* “Scale”: Taking scale answer
* “DateTime”: Taking date and time answer
* “YesNo”: Taking yes or no answer

### *4.2.2. Authentication function Implementation*

You can use account linking to connect your users' Google accounts with user accounts in your authentication system. This allows you to build richer experiences for your users; for example, you can save the user's food or music preferences, history of transactions, and other information that you can use to provide a more personalized experience.

If your Action is a companion of existing apps on different platforms (for example, the web or Android), you can use account linking to securely make users' preferences available to all platforms, which ensures a consistent cross-platform experience. Account linking for Actions on Google uses Google Sign-In, Google's secure authentication system, and optionally, OAuth 2.0, the industry-standard protocol for authorization.

### *4.2.3 Firebase Cloud function Implementation*

The firebase cloud function plays an important role. It provides endpoints to perform the core business logic necessary for the platform. It works in tandem with the front end of the application to provide vital functions like addition, deletion and modification of the survey blueprint.

This firebase cloud function host all the functionalities including but not limited to:

* Write and read from database
* External API call
* Interpreter to receive data from dialog flow
* Interpreter to deliver deployable conversational agent to dialogflow

Cloud Functions for Firebase lets you automatically run backend code in response to events triggered by Firebase features and HTTPS requests. Your code is stored in Google's cloud and runs in a managed environment. There's no need to manage and scale your own servers.

Cloud Functions gives developers access to Firebase and Google Cloud events, along with scalable computing power to run code in response to those events. Typical use cases fall into these areas:

* Notify users when something happens.
* Perform Realtime Database sanitization and maintenance.
* Execute intensive tasks in the cloud instead of in your app.
* Integrate with third-party services and APIs.

Developers can use Cloud Functions to keep users engaged and up to date with relevant information about an app. Consider, for example, if there is changes to the conversational agent through the dashboard. In conversational agent, a function triggered by Realtime Database read to provide user with the new available conversational agent.

1. The function triggers on read to the Realtime Database path where conversational agent stored.
2. The function retrieves a conversational agent.
3. Then available conversational agent is read to the user so user can choo the appropriate conversational agent they wanted to use.

With Cloud Functions database event handling, you can retrieve only specified data type that the researcher intended to get. keeping the system up to date and clean. For example, in a yes or no step conversation on Realtime Database, you could write only the answer of yes or no and do not have to waste time to clean up the data or store non intended data. The function's database event handler listens for write events on depending on the specific type and retrieves event data containing the text.

Execute intensive tasks in the cloud instead of in your app

Developers can take advantage of Cloud Functions to offload to the Google cloud resource-intensive work (heavy CPU or networking) that wouldn't be practical to run on a user's device. For instance, you could write a function to listen for a free form answer and retrieve the text directly and di a sentiment analysis without needing do create a model and host a model using traditional route.

Cloud Functions can help your app work better with other services by calling and exposing web APIs. For instance, an app used for collaboration on development could call a google service to retrieve user location and create the street address of located device on the go.

1. A user talk to tiger aware assistant to locate.
2. An HTTPS function triggers via googles API.
3. The function sends back user with the exact location longitude, latitude and street address of the user.

The Cloud Functions for Firebase client SDKs let you call functions directly from a TigerAware Assistant app. To call a function from your app in this way, write and deploy an HTTPS Callable function in Cloud Functions, and then add client logic to call the function from your app. You can trigger a function through an HTTP request by using functions.https. This allows you to invoke a synchronous function through the following supported HTTP methods: GET, POST, PUT, DELETE, and OPTIONS.

With Cloud Functions, you can handle events in the Firebase Realtime Database with no need to update client code. Cloud Functions lets you run database operations with full administrative privileges and ensures that each change to the database is processed individually. You can make Firebase Realtime Database changes via DataSnapshot or via the Admin SDK. You can deploy, delete, and modify functions using Firebase CLI commands or by setting runtime options in your functions source code.

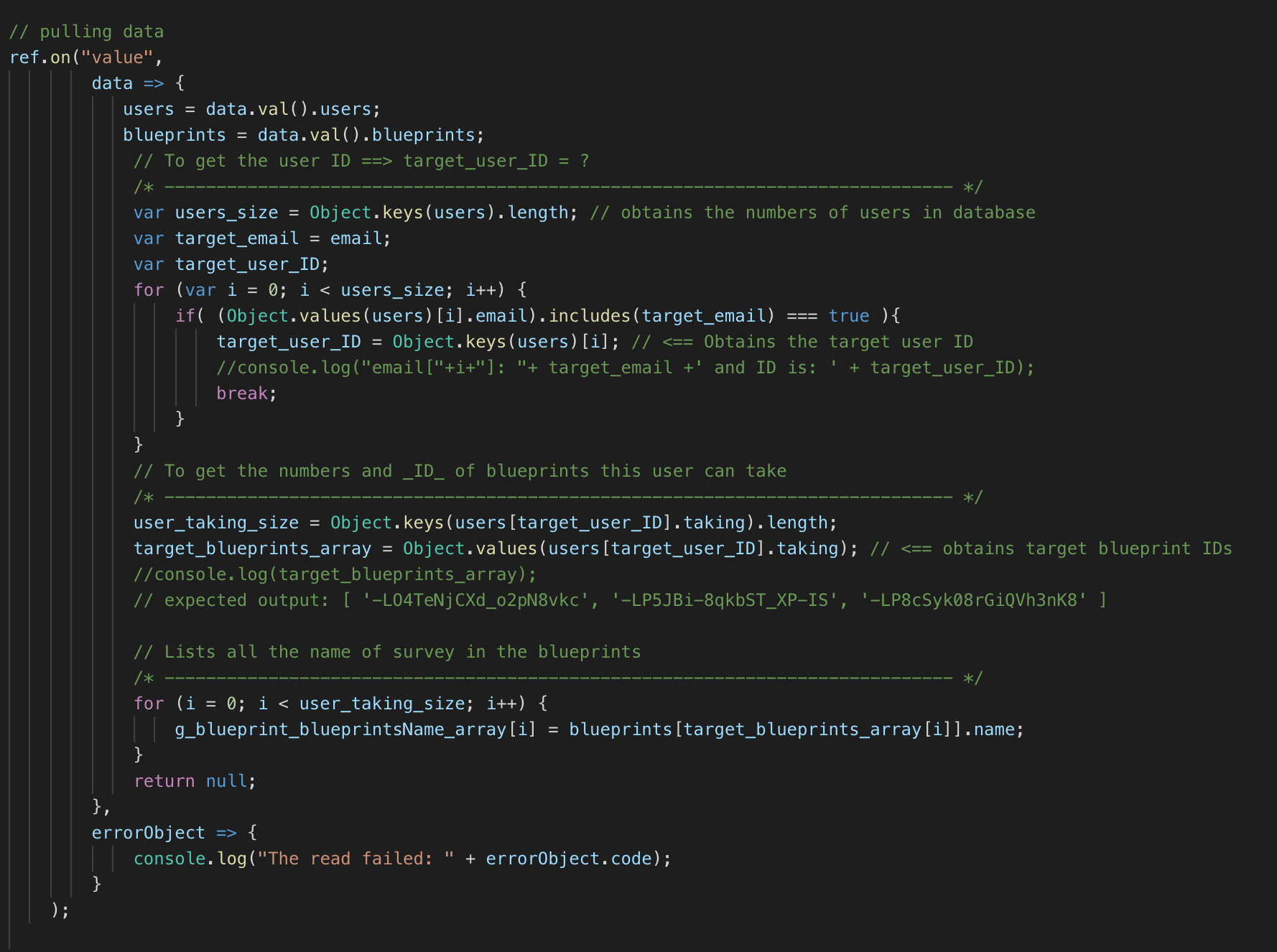
User will be able to use the conversational agent by calling it in the google assistant, the way use call Tiger aware assistant is simply by going into google assistant and saying, “Talk to TigerAware Assist”. The first thing that the cloud function will do is retrieving all user information and sign in the user. Then it will pull all the conversational agent that is available for user to use. The code below shows the process of pulling the data from real time database.

Figure 17 Code to retrieve necessary data of conversational agent from data base

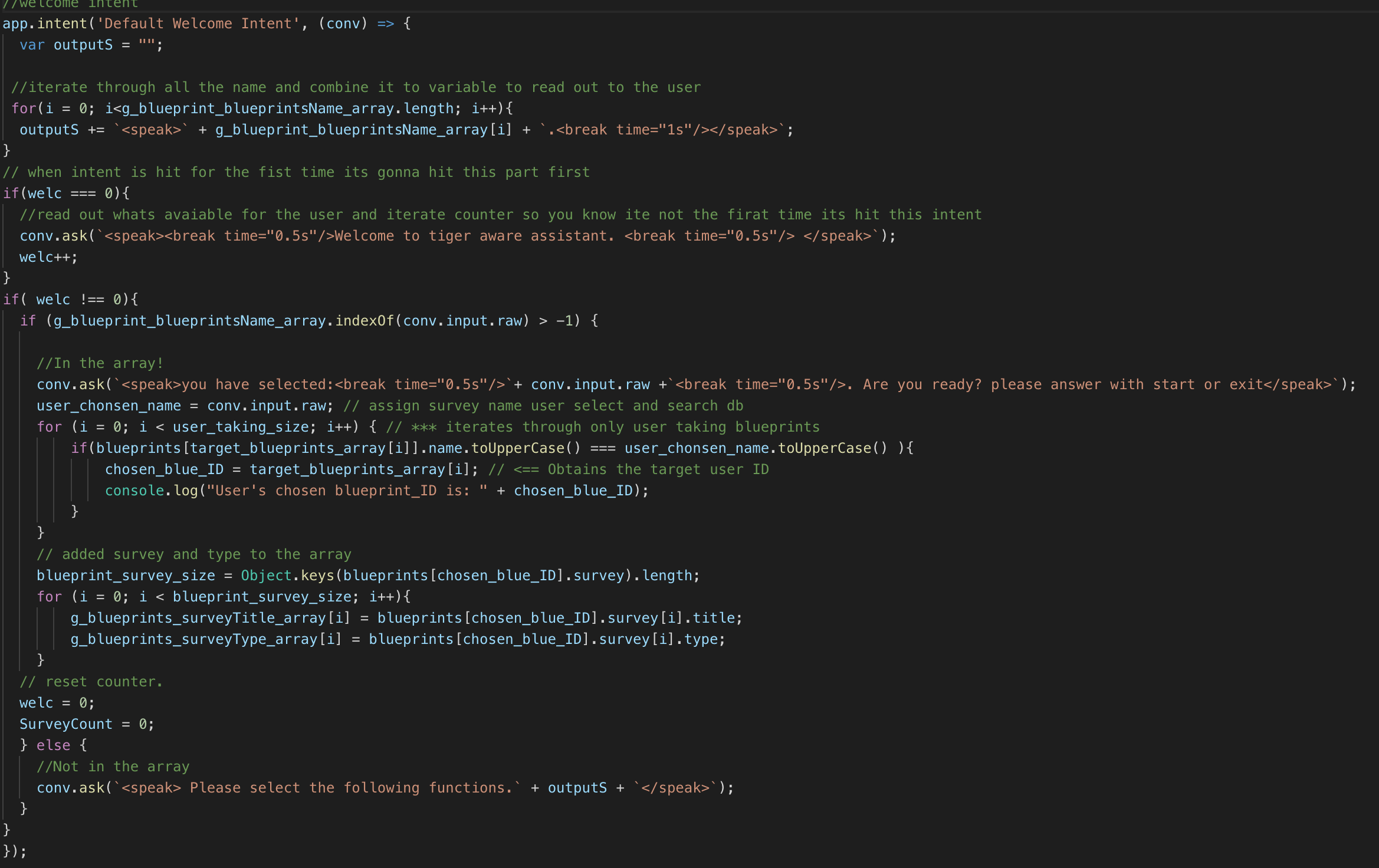
After pulling all the information needed from the blueprint, Cloud function will direct user to the welcome intent. Welcome intent work as a menu for the user, it will receive all the conversational agent that available to the user, aggregate it and read it to the user so the user will be able to select the conversational agent. As the code below show when the user selects a conversational agent it will check whether the conversational agent available or not, then if it available start the selected conversational agent.

Figure 18 Welcome intent code handled the menu and welcome greet for conversational agent

When the conversational agent started, all the user responses and action will be handled through everything intent. Data type handling is very important for to be able to verify the data type received and the data type expected. We need to create a checking function that validate the two of them. The validate type function is shown below.

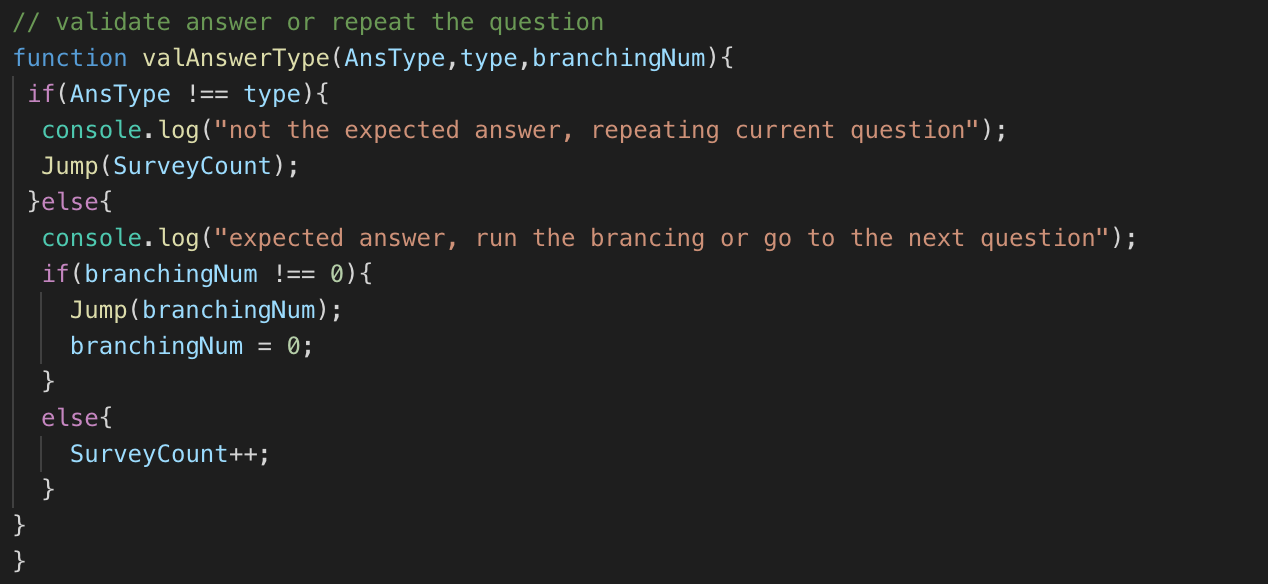


Figure 19 Function to validate answer type with recognize

To be able to provide branching functionalities we have to create a function that will retrieve steps condition and toid, this allow conversational agent to respond with different conversation steps depending on the user answers.

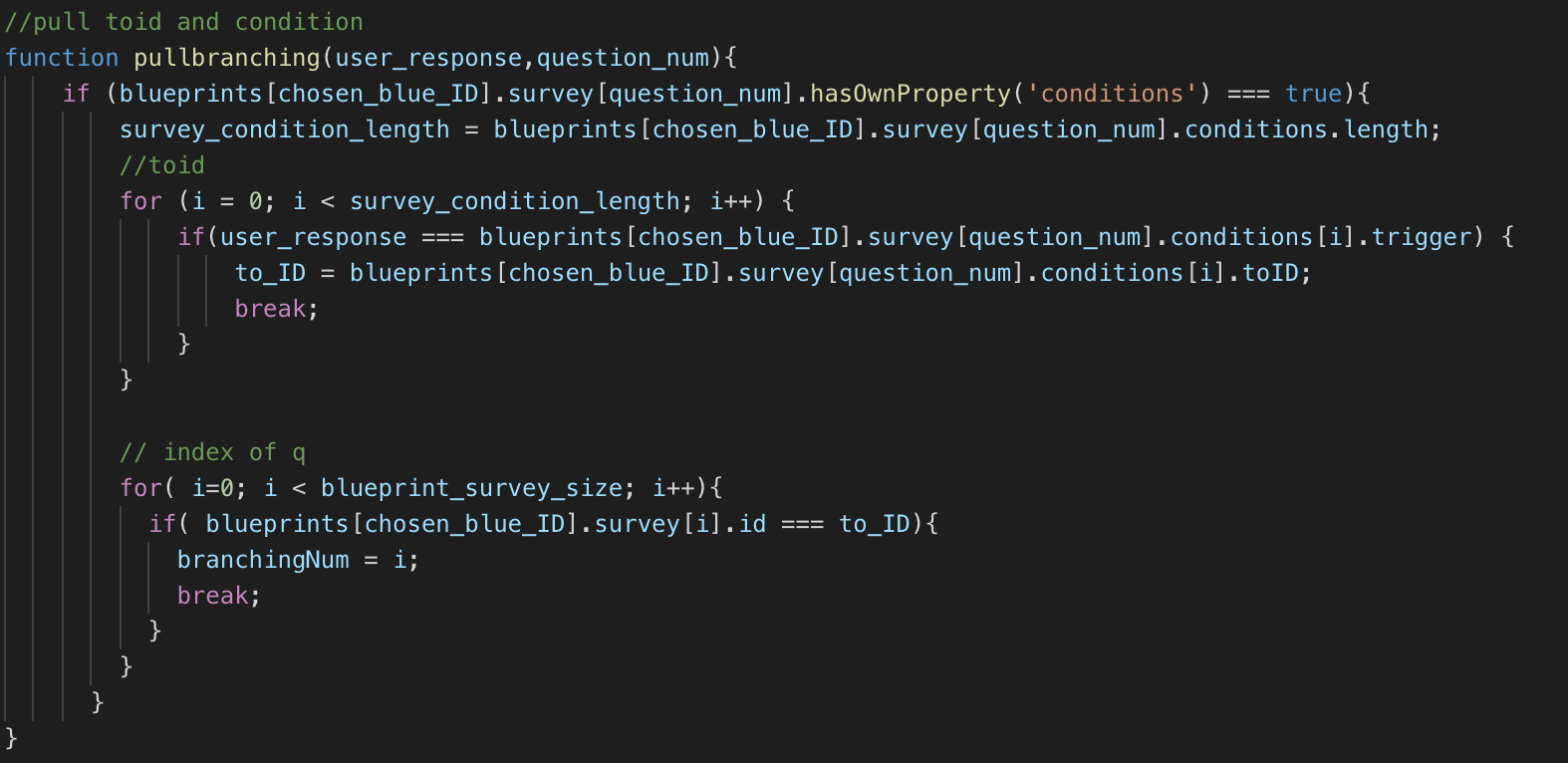
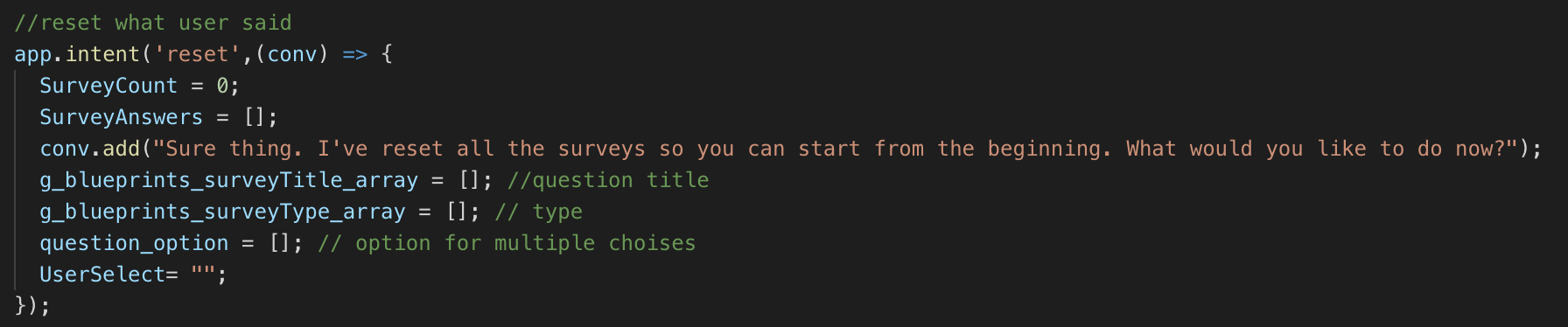


Figure 20 Code handled branching condition

With conversational agent flow in every single steps of conversation we have to enable user to navigate through the conversation we reserve a keyword that would provide user with the abilities to navigate through the conversation. Navigation entities:

* Skip/Next: goes to the next conversation
* Back: goes back to the last conversation
* Repeat: repeat the conversation
* Option: Read out available answers option
* Menu: Go back to the menu and read out the conversation agent available
* Exit: exit program completely
* Reset: reset current conversation and clean up recorded response

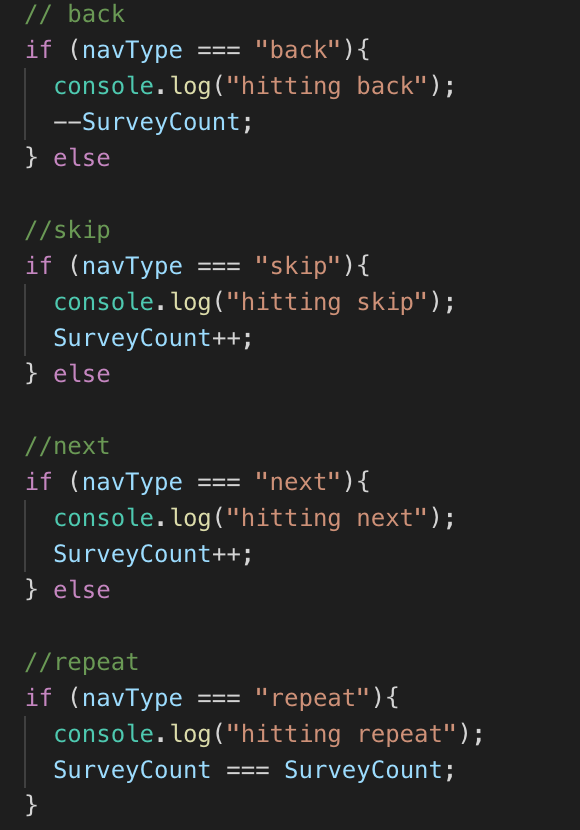
Figure 21 Intent to reset and erase data from conversation

Figure 22 Navigation code to handle conversation

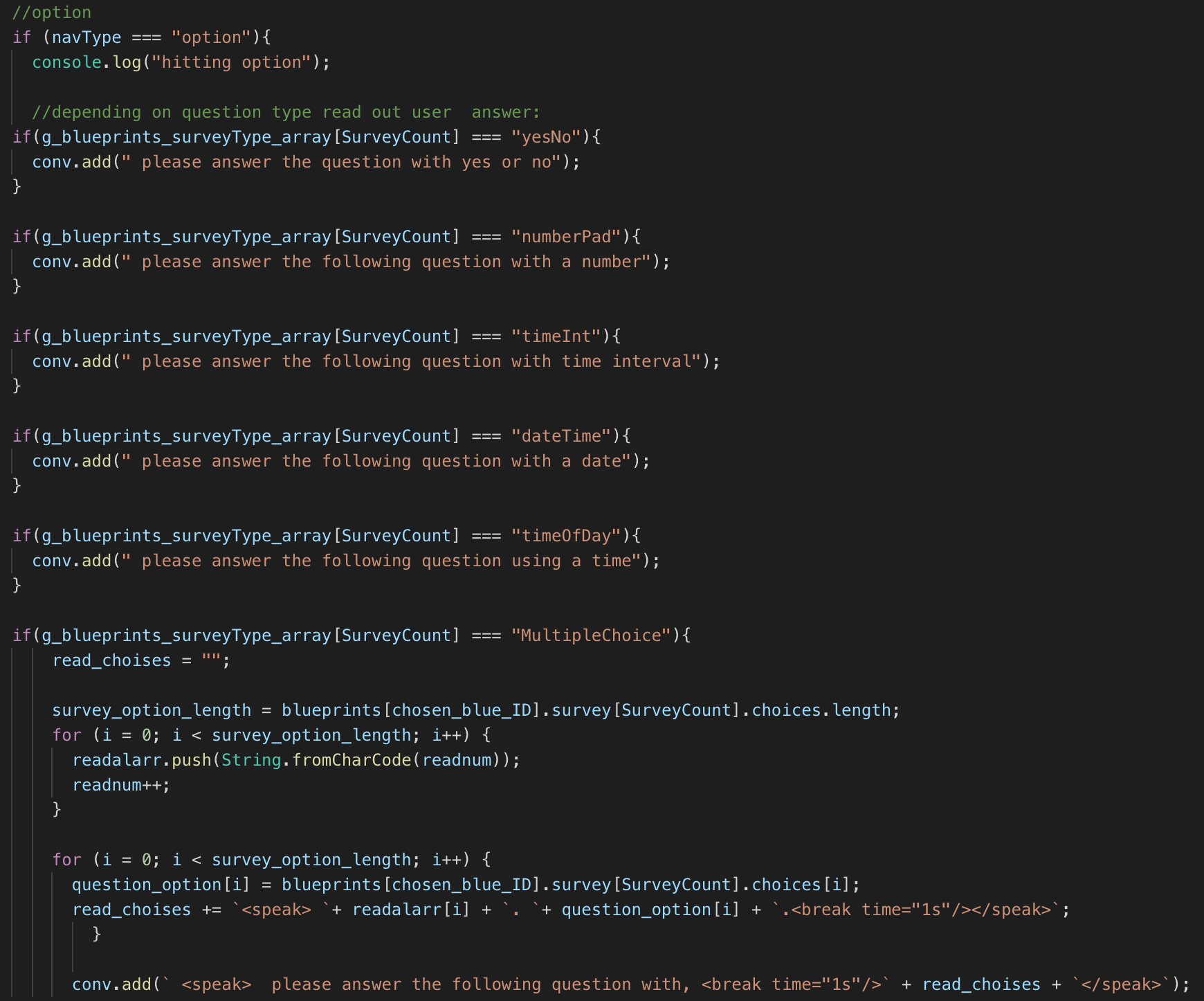


Figure 23 Read out option of the question

For each answer type received it will be processed in the cloud function. Each response has to be validated, collected, checked for conditions and send back to the database to be saved. Then the program will decide whether it needed to re-read the conversation or go to the next conversation. All of this process is contained within each response type program listed below:

* Time Interval Type: Duration of time answer
* Number Type: Taking number answer
* Any Type: Taking free form answer
* Text Slide Type: Required no response usually used to give user instruction or conversation and goes to the next one immediately
* Time Of Day Type: Taking time answer
* Multiple Choice Type: Taking multiple choice answer
* Scale and Continuous Scale Type: Taking scale answer
* DateTime Type: Taking date and time answer
* “YesNo”: Taking yes or no answer

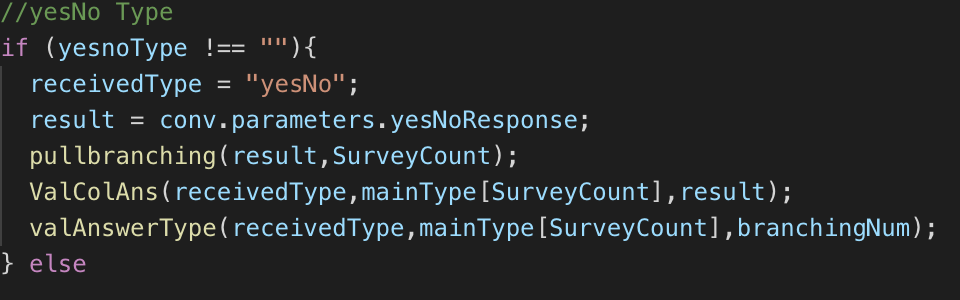
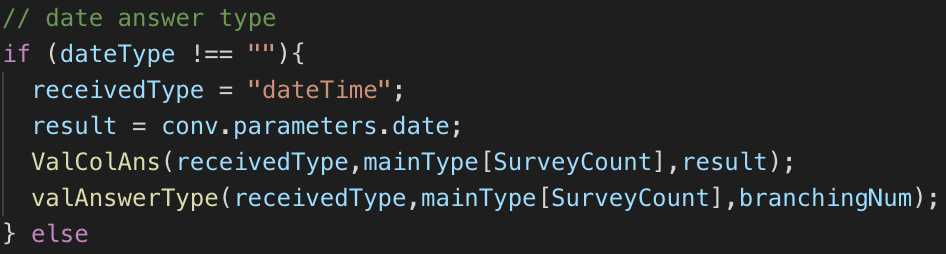
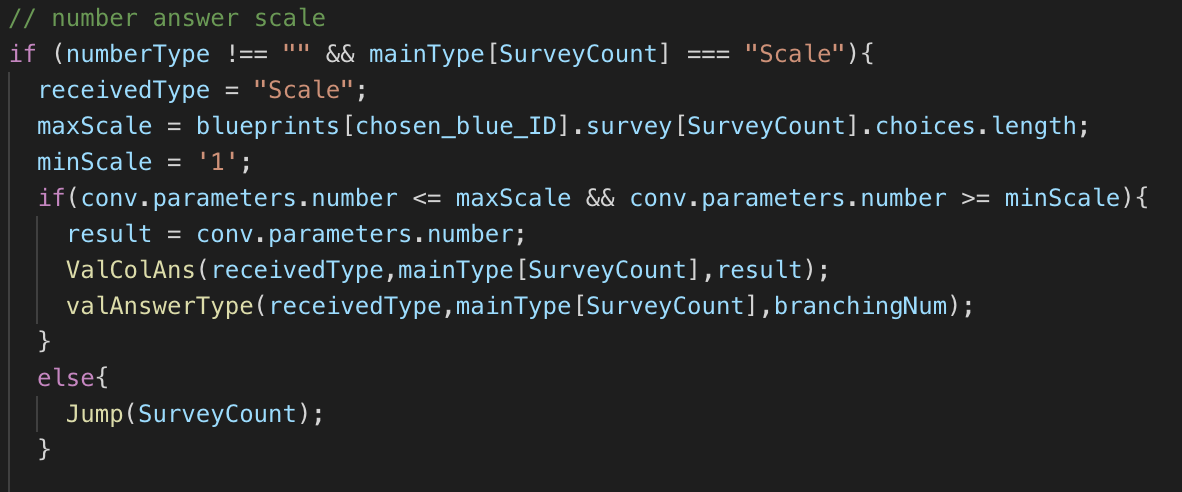
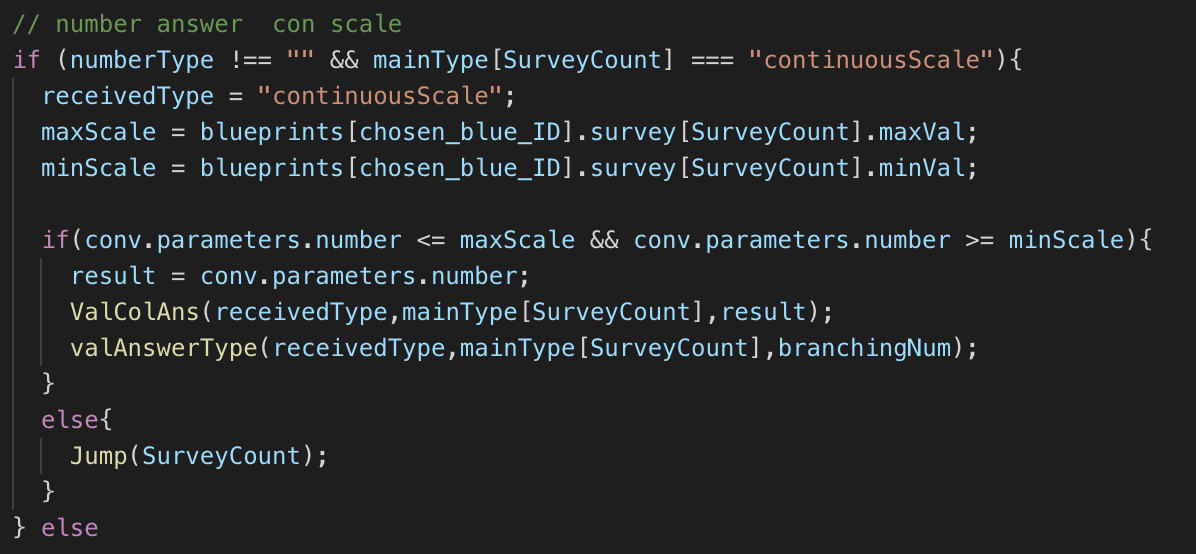
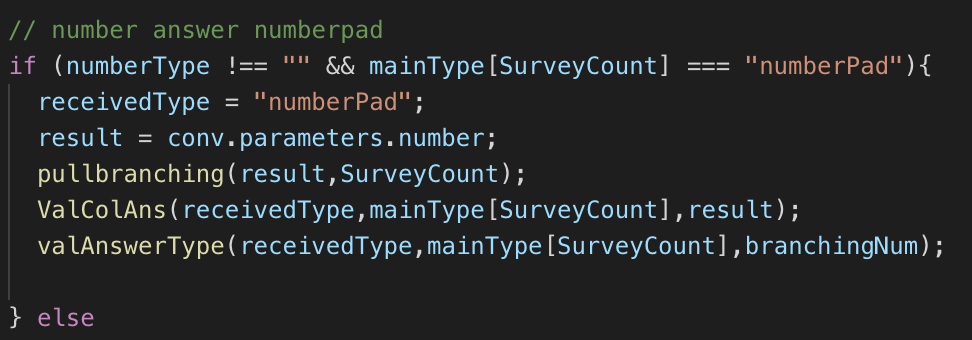


Figure 24 code to handle and interpreted yes no question and date answer type

Figure 25 code to handle and interpreted scale number

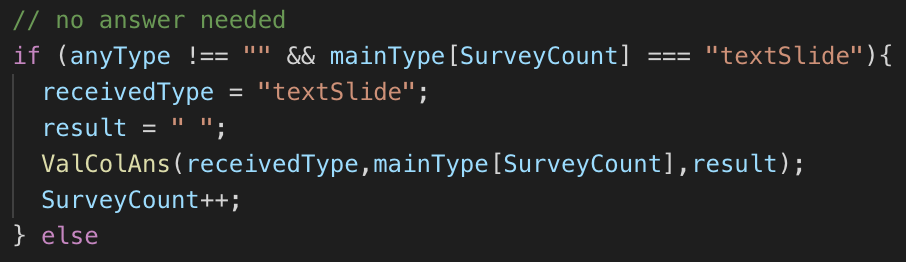


Figure 26 text slide and number

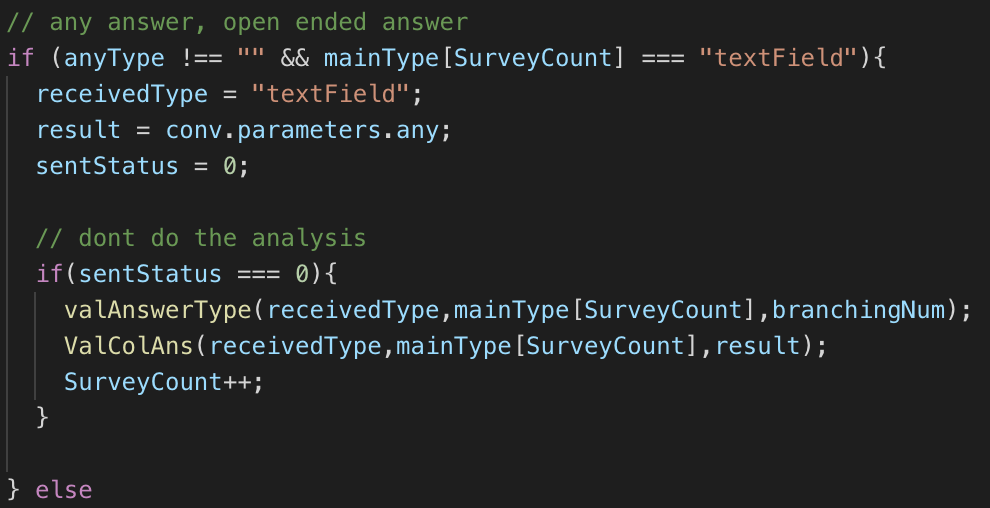
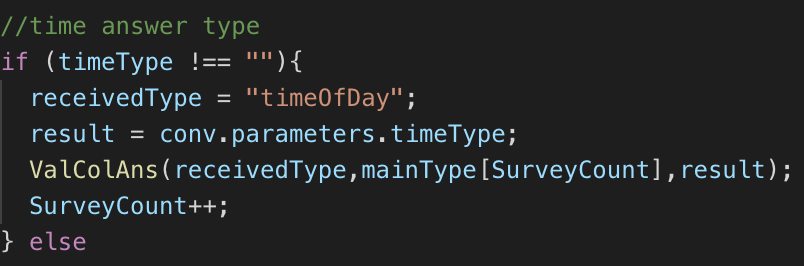
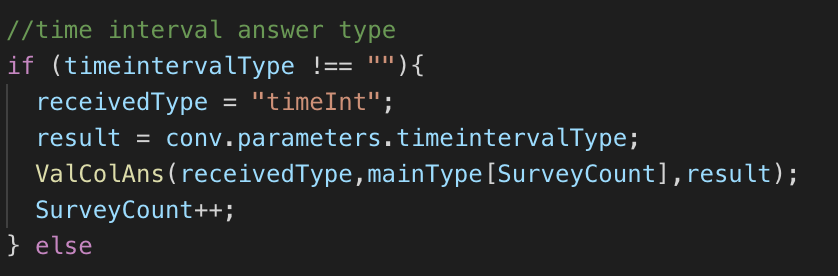


Figure 27 open ended answer and time answer 

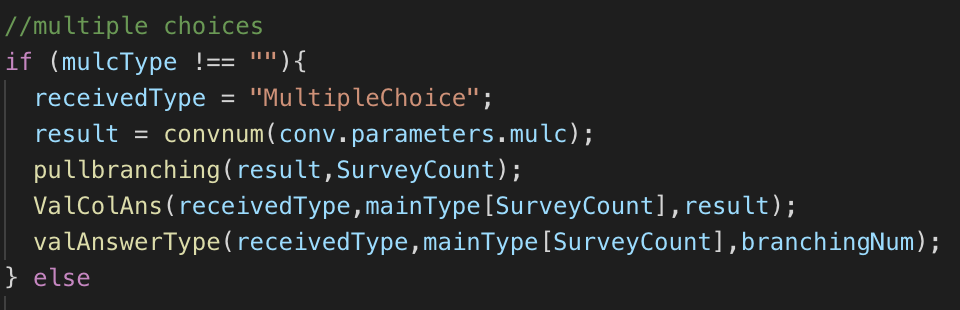


Figure 28 Time interval and multiple choice

### *4.2.4. External API’s function Implementation*

To be able to improve user with external functionalities such as generating user current location we needed to attach and integrate. Some functionalities that is currently available is still in the beta version. These functionalities are usable but currently still under improvement to make sure each one of these functions is robust and would work well under stress test. Current external functionalities that is available:

* Current weather and Temperature Service (Beta)
* Sentiment analysis Service (Beta)
* Location detection and street address generator

### *4.2.5. DialogFlow Implementation*

Dialowflow is used to Build voice and text-based conversational interfaces, such as voice apps and chatbots, powered by AI. This library and development platform enable TigerAware assistant to Connect with users on website, mobile app, the Google Assistant, Amazon Alexa, Facebook Messenger, and other popular platforms and devices. Tiger aware Assistant currently only available in google assistant but have the ability to be deployed into other chatbot interface.

DialogFlow platform build ysing years of domain knowledge and natural language understanding, it analyzes and understand the user's intent to help TigerAware assistant respond in the most useful way. In the future version Tiger aware assistant would be able to be access by users on-the-go or at home, User would be able to engage with it through wearables, phones, cars, speakers and other smart devices. This also broaden TigerAware assistant reach globally with 20+ supported languages including Spanish, French, and Japanese. Although current version only support English.

Dialogflow build conversational interfaces on top of TigerAware Assistant’s products and services by providing a powerful natural language understanding (NLU) engine to process and understand natural language input. Traditional computer interfaces require structured and predictable input to function properly, which makes the use of these interfaces unnatural and sometimes difficult. If users can't easily figure out this structured input, they'll have a hard time figuring out what to do.

Even with simple question, the conversational experiences are hard to implement. Interpreting and processing natural language requires a very robust language parser that's capable of understanding the nuances of language. For this reason, a traditional computer interface would tend to force users to input a well-known, standard request at the detriment of the user experience, because it's just easier.

Dialogflow lets TigerAware Assistant easily achieve a conversational user experience by handling the natural language understanding (NLU). Dialogflow, provide the ability to create agents that can understand the vast and varied nuances of human language and translate that to standard and structured meaning that TigerAware assistant apps and services can understand. An agent helps you process user input into structured data that can be used to return an appropriate response. To build a proper conversational agent builder we need to create intents, which define how to map user input to a corresponding response.

Every intent has a built-in response handler that can return responses after the intent is matched. However, this feature only allows you to construct responses that are static or have minimal logic. TigerAware assistant use cloud function fulfillment to process the intent first, and then return a more intelligent or useful response.

Cloud function fulfillment is custom logic that you implement as a webhook, which services requests, processes them, and returns responses.

1. Agent matches a user utterance to an intent.
2. Your agent extracts parameters out of the user utterance and calls your fulfillment with a JSON payload that contains the parameters along with a host of other useful information about the intent.
3. Your fulfillment processes any necessary information it needs to from the JSON, such as calling another REST API with the extracted parameters.
4. Your fulfillment constructs a response and returns it back to Dialogflow to render to the user. This response can be simple text or a rich response, such as a card with an image.

#### 4.2.5.1. agent

Agents are best described as Natural Language Understanding (NLU) modules. These modules can be included in your app, website, product, or service and translate text or spoken user requests into actionable data. This translation occurs when a user's utterance matches an intent within agent.

The matched intent then delivers a response back to the user. This response can be a simple text or spoken acknowledgment or a webhook response that includes information obtained from another system. E.g. Dialogflow sends a cloud function request with the location and date parameters to a third-party weather service. This weather service returns a webhook response in JSON format. Cloud function custom fulfillment parses the JSON data and delivers a response to the user with the relevant information.

#### 4.2.5.2. Intent

In Dialogflow, the basic flow of conversation involves these steps:

* The user giving input
* Your Dialogflow agent parsing that input
* Processed in cloud function fulfillment
* Your agent returning a response to the user

To define how conversations work, we create intents in agent that map user input to responses. In each intent, define examples of user utterances that can trigger the intent, what to extract from the utterance, and how to respond.

Generally, an intent represents one dialog turn within the conversation. But it work differently in TigerAware assistant, since we created a custom logic we keep intent down to minimum amount and contain all the conversation type in one intent. TigerAware assistant response prompts users for another utterance, which your agent will attempt to match to intent, and the conversation continues.

Intents consist of four main components that allow you to map what your user says to what your agent responds with. These components include the following:

* **Intent name:** The name of the intent. The intent name is passed to your fulfillment and identifies the matched intent.
* **Training phrases**: Examples of what users can say to match a particular intent. Dialogflow automatically expands these phrases to match similar user utterances.
* **Action and parameters**: Define how relevant information (parameters) are extracted from user utterances. Examples of this kind of information include dates, times, names, places, and more. You can use parameters as input into other logic, such as looking up information, carrying out a task, or returning a response.
* **Response**: An utterance that's spoken or displayed back to the user.

#### 4.2.5.3. entity

Entities are Dialogflow's mechanism for identifying and extracting useful data from natural language inputs. While intents allow your agent to understand the motivation behind a particular user input, entities are used to pick out specific pieces of information that your users mention. Anything from date, number, names or amounts with units. Any important data you want to get from a user's request will have a corresponding entity.

Dialogflow is equipped with numerous system entities, which allow agents to extract information about a wide range of concepts without any additional configuration. For example, system entities are available for extracting dates, times, and locations from natural language inputs.

### *4.2.6. Firebase cloud function fulfillment*

Fulfillment is code that's deployed as a webhook that lets Dialogflow agent call business logic on an intent-by-intent basis. During a conversation, fulfillment allows us to use the information extracted by Dialogflow's natural language processing to generate dynamic responses or trigger actions on the cloud function.

The following are some example cases where you can use fulfillment to extend an agent:

* To generate dynamic responses based on information looked up from a database.
* To trigger write database on Realtime.
* To implement the rules and conditions for conversation.

TigerAware assistant can be agents using the following platforms:

* Actions on Google: Deploy through the Actions on Google console to make your agent available through Google Assistant and related devices.
* One-click Integrations: Make your agent available through one or more of Dialogflow's platform integrations. (Beta)
* Importers and Exporters: Import your Alexa Skill into Dialogflow or export to Alexa and Cortana. (Beta)

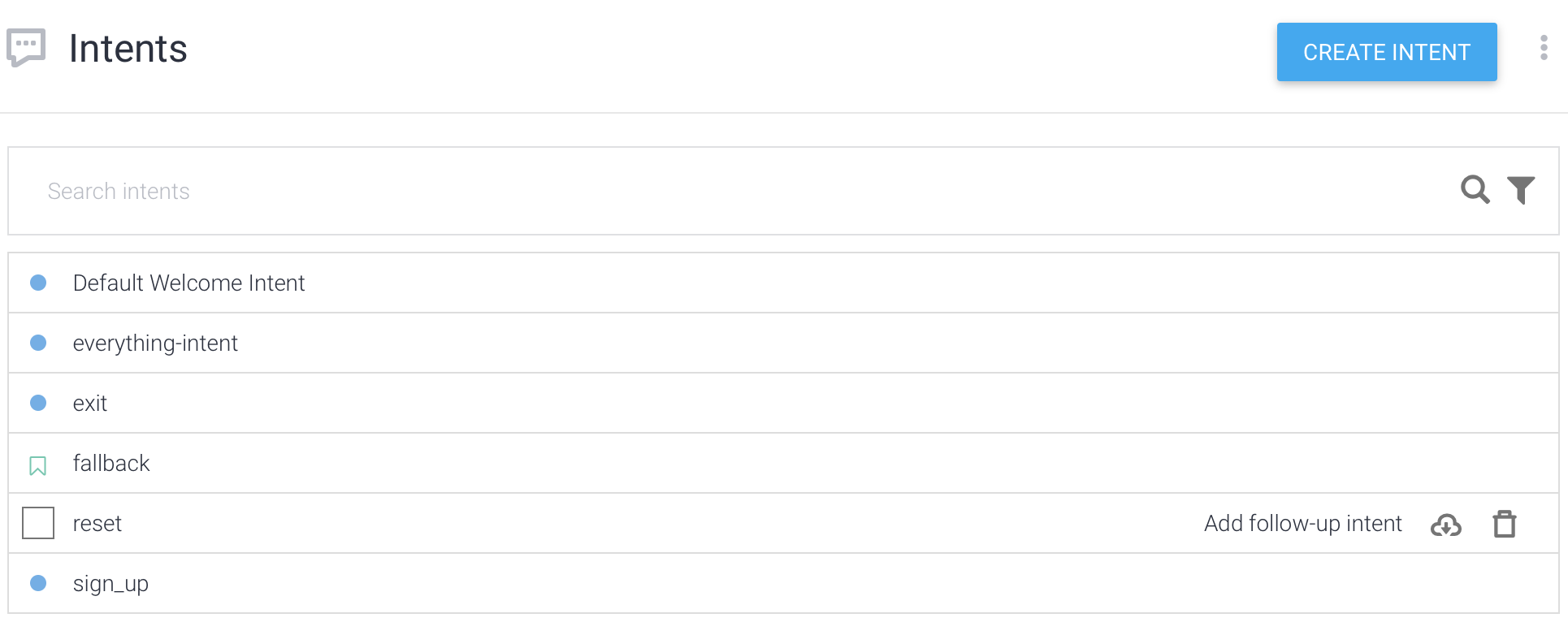
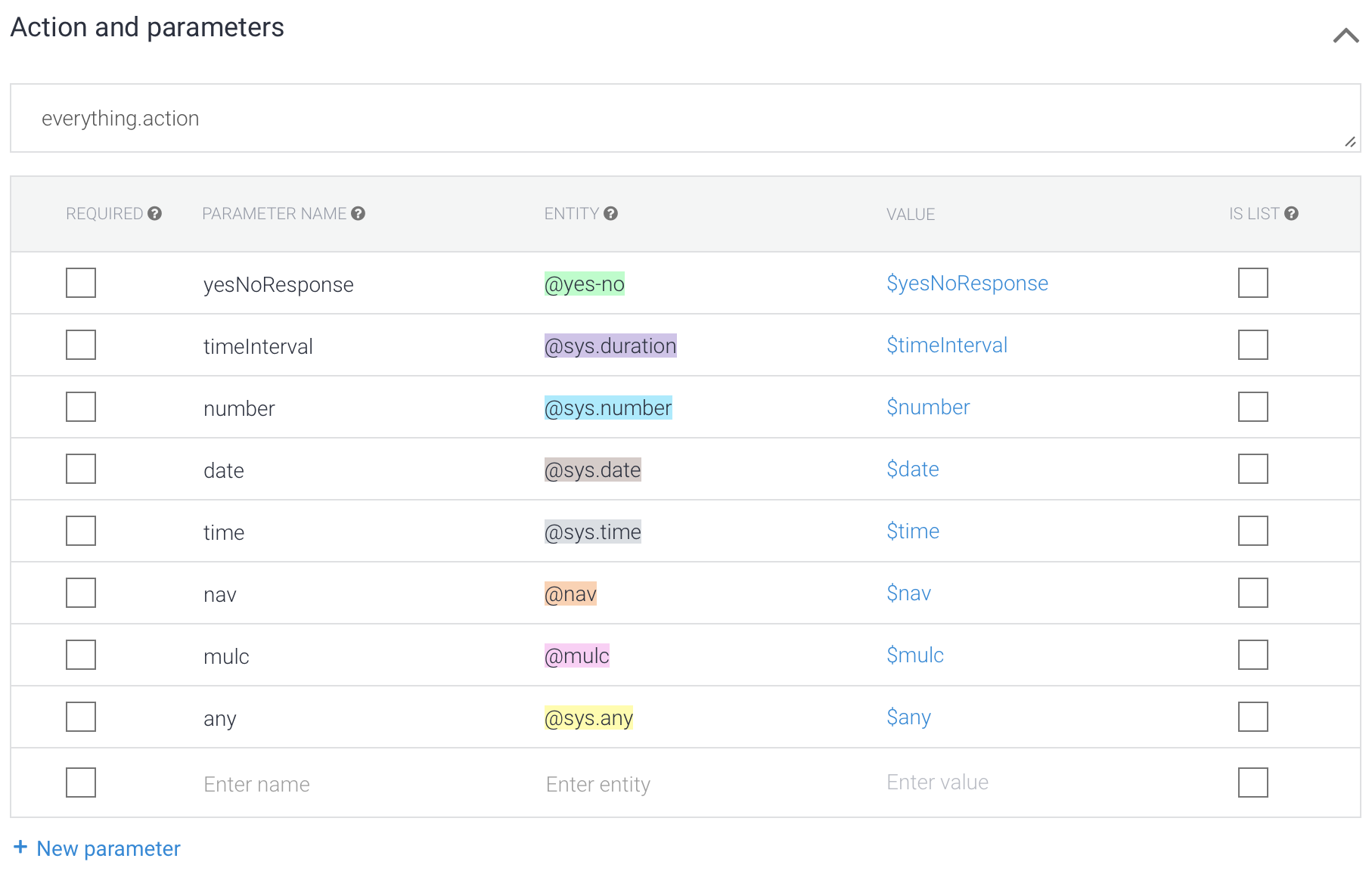
On the DialogFlow side, we created six different intend to provide all the functionalities all the intent is shown in the figure below. DialogFlow also filtered responses to trigger action.

Figure 29 Dialogflow intent

 Figure 30 collection of entities recognize

# TigerAware Applications

Before, we discuss the real-world applications of TigerAware Assistant. The table below describes how the current platform can provide more medium and features compared to the Existing TigerAware and other industry platforms.

Table 3 Comparing traditional and current available chatbot services Systems with TigerAware Assistant

| **Survey Platform’s / Functionalities** | **REDCap** | **Survey Monkey** | **Woebot** | **Chatbot with Serverless Computing** | **Healthy Coping Assistant** | **Existing TigerAware** | **TigerAware ecosystem and TigerAware Assistant** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Text Survey** | Yes | Yes | Yes | Yes | Yes | Yes | Yes |
| **Response based Branching in chatbot** | No | No | Yes | No | No | No | Yes |
| **Conversational Agent** | No | No | Yes | Yes | Yes | No | Yes |
| **Researcher Modifiable chatbot** | No | No | No | No | No | No | Yes |
| **Deployable Sources** | Browser | Mobile | Mobile | Mobile | Mobile | Mobile | Mobile and Smart Speaker |
| **Survey Templates** | Yes | No | No | No | No | Yes | Yes |
| **External Services** | No | No | No | Yes | No | No | Yes |

The platform has been applied in a number of conversational agent and the application has shown excellent capabilities in adaptability and deployment. The first part of the section discusses about the feedback from an initial focus group and a pilot study. followed by several other ongoing test use-cases of TigerAware Assistant.

## 5.1. TigerAware Assistant Based Diabetes Self-Management Study

In collaboration with the Department of Health Management and Informatics at the University of Missouri, the TigerAware system was adopted for data collection and diabetes self-management. TigerAware is used to convert a set of questions into a Google Assistant based application which complements a preexisting mobile application. A chat function is developed to collect diabetes-related information, such as the users' blood glucose readings from their Blood Glucose Monitor, daily meals and activities, and overall well-being of the patient.

The application advices the users based on their response for activity level and diet based on their responses. The mobile application and dashboard can be used by patients and physicians to monitor the information relating to patient’s wellbeing

## 5.2. TigerAware Assistant Based PHQ9 questionnaire

The PHQ-9 is a multipurpose instrument for screening, diagnosing, monitoring and measuring the severity of depression:

* The PHQ-9 incorporates DSM-IV depression diagnostic criteria with other leading major depressive symptoms into a brief self-report tool.
* The tool rates the frequency of the symptoms which factors into the scoring severity index.
* Question 9 on the PHQ-9 screens for the presence and duration of suicide ideation.
* A follow up, non-scored question on the PHQ-9 screens and assigns weight to the degree to which depressive problems have affected the patient’s level of function.

The PHQ-9 is brief and useful in clinical practice. The PHQ-9 is completed by the patient in minutes and is rapidly scored by the clinician. The PHQ-9 can also be administered repeatedly, which can reflect improvement or worsening of depression in response to treatment.

## 5.3. TigerAware Assistant Based MINI

The Mini-International Neuropsychiatric Interview (M.I.N.I.) is a short structured diagnostic interview, developed jointly by psychiatrists and clinicians in the United States and Europe, for DSM-IV and ICD-10 psychiatric disorders. With an administration time of approximately 15 minutes, it was designed to meet the need for a short but accurate structured psychiatric interview for multicenter clinical trials and epidemiology studies and to be used as a first step in outcome tracking in nonresearched clinical settings.

# Conclusion and Future Work

## 6.1. Conclusion

Developing, TigerAware Assistant using serverless architecture has involved using some of the latest and popular web development technologies, understanding how conversational agent are structured, and making design considerations on how conversational agent data needs to be handle and processed is a big task to accomplish.

Considering to be able to build this tool we needed to use multiple different layers of library. All of this has been an enriching process in terms of knowledge gained about developing tools to build conversational agent for service deployment and integrating various API for conversational agent deployment.

This application will open up a new possibility for conversational agent medium to provide ease of use in administrator side, pleasant experience in the user side and reduce development time from developer perspective.

## 6.2. Future Work

There are a number of important enhancements that the TigerAware assistant can use and will be considered for implementation in near future.

* Integrating more machine learning module that will greatly increase the usability.
* Developing standalone dashboard that would make creating conversational agent from administrator side easier.
* Moving Database to Firestore cloud database

# Reference

* + - 1. A. Cheng, V. Raghavaraju, J. Kanugo, Y. P. Handrianto and Y. Shang, "Development and evaluation of a healthy coping voice interface application using the Google home for elderly patients with type 2 diabetes," *2018 15th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, Las Vegas, NV, 2018
      2. Abdul-Kader, SA and Woods, JC (2015) 'Survey on Chatbot Design Techniques in Speech Conversation Systems.' International Journal of Advanced Computer Science and Applications
      3. Lucas F. Albuquerque Jr., Felipe Silva Ferraz3, Rodrigo F. A. P. Oliveira, and Sergio M. L. Galdino, Function-as-a-Service X Platform-as-a-Service: Towards a Comparative Study on FaaS and PaaS (2017)
      4. S. Roos, R. Lochan, “CHATBOTS IN EDUCATION A passing trend or a valuable pedagogical tool?”(2018)
      5. Yan, Mengting & Castro, Paul & Cheng, Perry & Ishakian, Vatche. (2016). Building a Chatbot with Serverless Computing.
      6. Kathleen Kara Fitzpatrick, Alison Darcy, Molly Vierhile. (2017). Delivering Cognitive Behavior Therapy to Young Adults With Symptoms of Depression and Anxiety Using a Fully Automated Conversational Agent (Woebot): A Randomized Controlled Trial
      7. W. Morrison, L. Guerdan, J. Kanugo, T. Trull and Y. Shang, "TigerAware: An Innovative Mobile Survey and Sensor Data Collection and Analytics System," *2018 IEEE Third International Conference on Data Science in Cyberspace (DSC)*, Guangzhou, 2018, pp.
      8. Shawar, BA and Atwell, E (2002) *A comparison between Alice and Elizabeth chatbot systems.* University of Leeds, School of Computing research report 2002.
      9. B. Setiaji and F. W. Wibowo, "Chatbot Using a Knowledge in Database: Human-to-Machine Conversation Modeling," *2016 7th International Conference on Intelligent Systems, Modelling and Simulation (ISMS)*, Bangkok, 2016
      10. Dahiya, Menal. (2017). A Tool of Conversation: Chatbot. INTERNATIONAL JOURNAL OF COMPUTER SCIENCES AND ENGINEERING.
      11. Doshi, Sarthak & B. Pawar, Suprabha & G. Shelar, Akshay & S. Kulkarni, Shraddha. (2017). A comparison between Alice and Elizabeth chatbot system.
      12. Philipp Leitner, Erik Wittern, Josef Spillner, Waldemar Hummer, A mixed-method empirical study of Function-as-a-Service software development in industrial practice, Journal of Systems and Software, Volume 149, 2019, Pages 340-359.
      13. Npr report smart speakers increase YOY [online]
      14. Firebase cloud function [online]
      15. Firebase Realtime database [online]
      16. DialogFlow [online]
      17. AngularJS, "AngularJS," [Online]. https://docs.angularjs.org/api.
      18. E. S. Raymond, "Software Design Principles," [Online].
      19. N. M. Wergeles, "AMD, analysis of mood dysregulation: a machine learning approach," 2016.
      20. M. Guay, "The 20 Best Online Survey Builder Tools," Zapier, 24 June 2016. [Online]. Available: https://zapier.com/learn/forms-surveys/best-survey-apps/.
      21. SurveyMonkey, "SurveyMonkey Plans and Pricing," SurveyMonkey, 2017. [Online]. Available: https://www.surveymonkey.com/pricing/details/.
      22. http://lukeangel.co/cross-platform/docker-servless-faas-functions-as-a-service/
      23. https://martinfowler.com/articles/serverless.html
      24. https://www.devbridge.com/articles/what-comes-after-microservices-serverless/
      25. https://www.iron.io/what-is-serverless-computing/
      26. https://stackify.com/function-as-a-service-serverless-architecture/