Development of Survey Visualization and Advanced Integrated Data Analysis in TigerAware

A Project
Presented to
The Faculty of the Graduate School
At the University of Missouri

In partial fulfillment
Of the requirements for the degree
Master of science

Implemented and defended by

Rui Huang

Prof. Yi Shang, Advisor

July 2019

TABLE OF CONTENTS

List of Figures

ACKNOWLEDGEMENTS

Firstly, I would like to thank Prof. Yi Shang for his mentorship, guidance, and support throughout my graduate school and through this research project. I would also like to thank Denis for the opportunity to collaborate with the Department of Psychological Sciences at the University of Missouri and their valuable feedback during the process of developing this project. I also want to thank my committee members Professor Trull Tim and Professor Zhao Yunxun for their precious time and supports.

I would also like to thank everyone from the Computer Science Lab, especially Zachary Kipping, Luke Guerden, Connor Rowland and Yohanes P Handrianto, for sharing their precious experience and being an integral part of this project. I would also like to show appreciation to Jodie Lenser, Shirley Holdmeier for guiding me through my application as well as advising me on my courses.

Finally, I would like to thank my family and friends for their support and encouragement during all these years. Without their support, trust and belief in me, I would not be where I am today. I hope I have done you proud.

ABSTRACT

In recent years, smartphone apps have been widely used to collect data in data-driven research. TigerAware is a newly developed system that allows researchers to easily create and deploy surveys on mobile devices, such as iOS and Android smartphones and tablets, which can collect a wide range of real-life data, including survey responses and data from built-in sensors of smartphones and external wireless sensors. In this project, the web dashboard of TigerAware has been enhanced by adding two new components. The first is survey visualization that displays a survey structure as a directed planar graph, where questions are represented as nodes and question ordering or branching as directed edges. Its implementation is based on the D3 visualization library and Tutte's graph embedding algorithm. The generated survey graph is interactive and editable, providing an intuitive GUI for survey creation and presentation. The second new component is an integrated survey data analysis system. In addition to computing typical statistics, advanced data analysis functionalities have been implemented based on state-of-the-art machine learning methods for natural language processing and computer vision, such as various deep learning models for text-based sentiment analysis or object detection in images. They are implemented based on Google Cloud AI and Microsoft Azure Cognitive Service to take advantage of the rich and expanding sets of features these cloud services can offer. With this ability, unstructured data generated from free text and image-type questions can be easily processed to generate useful features for further analysis. These new additions to TigerAware support more effective survey creation, survey administration, and survey data analysis using cutting-edge machine learning methods.

1.INTRODUCTION

From psychology to marketing, researchers in different disciplines are looking for a way to easily deploy survey-based studies, which can help them make insightful decisions. Currently, there are some platforms can help. TigerAware, developed by Distributed and Intelligent Computing Lab at the University of Missouri, is one of them. It is a survey and sensor data collection system which consists of three parts: a survey creation and administration dashboard, a Firebase server for storing data, and Android and iOS mobile applications for data collection[1]. Compared to other commercially available platforms, TigerAware is free to use. Also, some other platforms like REDCap are limited to qualitative data collection, while TigerAware can collect data including but not limited to qualitative data, text, and image. It is designed to offer a generic, customizable survey creation and deployment, data collection from survey and sensors for researchers who want to use smartphone-based surveys in conjunction with various internal or external data sources including wireless-wearable sensors and Internet of Things (IoT) devices[2]. These features make TigerAware rise above others.

However, there still are several parts can be improved. First, TigerAware is not able to provide advanced analysis function. Second, the process of creating surveys is mistake-prone due to the complicated branches. To overcome these issues, this survey visualization and data analysis system is developed. In particular, this system consists of two parts.

The first part is the data analysis system, which contains a standalone backend service and an analysis page in the TigerAware web dashboard. This backend service can provide analysis functions ranging from basic statistical analysis to advanced deep learning analysis by incorporating Microsoft Cognitive Service and Google Cloud AI. The analysis page is built directly into the existing TigerAware dashboard and responsible for displaying questions, communicating with backend, and visualizing results.

The second part is the survey visualization feature, which is also implemented on the dashboard. Visualization feature is added to the TigerAware dashboard by using the state-of-the-art visualization framework. This feature can visualize survey accurately. If researchers make any error when creating a survey, it will show up on the visualization page.

This survey visualization and data analysis system has significant advantages over others.

For the analysis part, Box Skills is a product that provides advanced deep learning analysis function for Box developer community to use. However, Box Skills is for developers to use, not for end-user. If a user uploads an image to Box and he wants to detect emotions, he needs to develop a program to detect emotion and write the result back to Box through Box Skills API. The data analysis system developed in this project is aimed at end-user, not developers. This means there is no need for end-user to develop additional programs. End users just need to create and deploy surveys, all responses collected from participants will be analyzed and visualized automatically.

For the visualization feature, current existing visualization frameworks don't have the ability to draw planar graphs. For example, D3 is the state-of-the-art visualization framework that supports force-directed graph layout, however, it might have crossings between edges even if the graph is a planar graph. The visualization feature developed in this project significantly improved the performance of current state-of-the-art visualization framework by making use of the Tutte embedding algorithm to draw planar graphs.

1.1 Improvement Description

The existing TigerAware platform for researchers to deploy surveys, collect responses and analyze data has two parts that can be improved. First, it lacks the ability to analyze collected survey data. If researchers want to analyze responses from surveys, they have to implement analysis functions or make use of existing analytics software. These solutions are either impractical or high-cost. Second, the process of creating surveys is mistake-prone due to the complicated branches. For example, if the answer to question A is 'yes',the next question is B. However, the next question is C if the answer to question A is 'no'. If researchers make a typo when setting these branch property, it will be very hard for them to find these mistakes due to the lack of visualization.

1.2 Proposed Solution

As deep learning becomes more and more popular in recent years, several technology companies, like Google, Microsoft, IBM and so on, publish their AI service. These services can help to analyze some type of data collected by TigerAware, like text, image, voice, etc. For other types of data, like multiple choice, basic statistics functions, like mean, standard deviation, are implemented manually. Based on the understanding of existing TigerAware system, adding a data analysis page to TigerAware dashboard, and also, building a standalone backend service API, which has the ability to do basic statistics analysis as well as advanced analysis by using AI service published by Google and Microsoft, for analysis page to consume, is proposed as a solution to add data analysis ability to TigerAware.

Adding a visualization page to the TigerAware dashboard is proposed as the solution to solve the mistake-prone issue existing when creating the survey. In this solution, D3.js which is the state-of-the-art visualization framework is used to visualize the survey graph. To improve the performance of D3, the Tutte embedding algorithm is used to find the planar embedding so that the graph rendered by D3 has no crossing.

The whole architecture of the improved TigerAware system is shown as below:

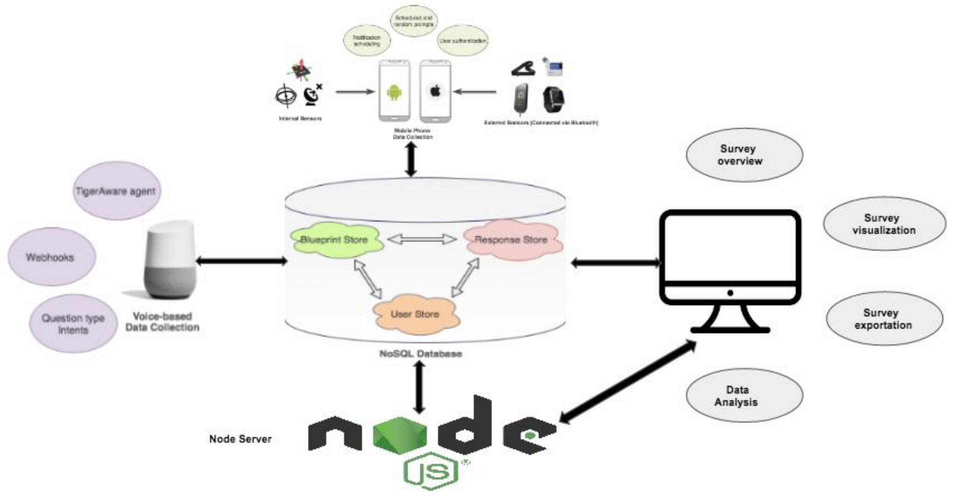


Figure 1 Improved TigerAware System Architecture

The improved TigerAware system has the ability to analyze data as well as visualize survey. This report is organized as follows: In chapter 2, related work about data analysis, data visualization as well as previous TigerAware work are discussed. The design of the survey visualization and data analysis system is discussed in Chapter 3. Chapter 4 presents the implementation of the design. A real-world application of this system is discussed in Chapter 5. Chapter 6 will talk about the conclusion and future work. All references will be listed at Chapter 7.

2.RELATED WORK

2.1 Data Analysis

For the data analysis part in this project, the original idea is from the paper "Deep Learning at Your Fingertips"[1] published by Distributed and Intelligent Computing Lab at the University of Missouri. In this paper, they propose a method to add data analysis ability to the existing TigerAware system by using pre-trained models. The analysis functions work well individually but they haven't moved to a stage to combine the analysis part with the existing TigerAware system. The analysis system implemented in this project uses third party API to analyze data and has been integrated with existing TigerAware system very well.

Several products on the market that can help to analyse data. For example, Box is a platform that can hold and analysis user data through Box Skill. Box Skill is a type of application that performs custom processing for files uploaded to Box. Box Skills leverage third-party AI/ML services to extract information from files upon upload to Box. For example, a Box Skill could automatically label objects in images using a computer vision service[4].The framework allows a Box Skill to receive information about a file as it's uploaded to Box, retrieve the file for processing, and write to the file's metadata in Box[4]. Metadata applied by a Box Skill can be displayed to end-users via the Box application and can drive other capabilities of the Box platform, such as search, workflow, and policies. Even though TigerAware and Box are both platforms that can hold and analysis data, they still have several differences. For instance, Box Skill is an add-on part of Box, while the analysis system implemented in this project is a built-in part of TigerAware. For example, if a Box user wants to analyze his data, he first needs to find an existing Box Skill or implement one if the existing one can't meet his requirement. Then he needs to add the Box Skill to the folder where he wants to apply analysis function. This might be inconvenient for users because they are from different fields and may not have enough programming skills to implement a Box Skill. However, TigerAware users don't need to write any code to analysis data, all the analysis function is provided as a built-in part of TigerAware. As for this aspect, the TigerAware system is better than Box.

Google form is another tool that allows collecting information from users via a personalized survey or quiz[5]. User can create, analysis survey and also visualize the instant result as data come in. However, the shortcomings of the Google form is that it can't provide advanced deep-learning based analysis functions. TigerAware system can provide deep-learning based analysis function and thus better than Google form.

2.2 Visualization

Currently, there are a lot of frontend visualization libraries, like D3.js, Chart.js, Three.js, Echarts.js, and Highcharts.js.

In this project, D3 is used to visualize surveys. D3 is a JavaScript library for manipulating documents based on data. It helps to bring data to life using HTML, SVG, and CSS[6]. It supports lots of layouts, including, but not limited to, chord, cluster, force, hierarchy, and histogram layout. Force layout is used to visualize surveys in this project. However, D3 can't visualize surveys as planar graphs. In this project, The Tutte embedding algorithm is used with D3 to draw planar graphs. Compared to state-of-the-art visualization library D3, the TigerAware system has the ability to draw planar graphs and thus better than D3.

Each of these previous works proposed a solution to address a specific research question. These mentioned works, especially the paper mentioned above, were essential in identifying the existing approaches, and how they can be utilized for the proposed solution. They were used for helping build a powerful data analysis and survey visualization system, which have been further explored in the following chapters.

3.SYSTEM DESIGN

This chapter mainly focuses on the design of the survey visualization and data analysis system as well as the main factors considered in this design. The chapter also covers the technology stack used in this design.

3.1 Data Analysis System Design

The data analysis system is divided into three parts: presentation component , analysis engine, and data storage, shown in Figure 2.

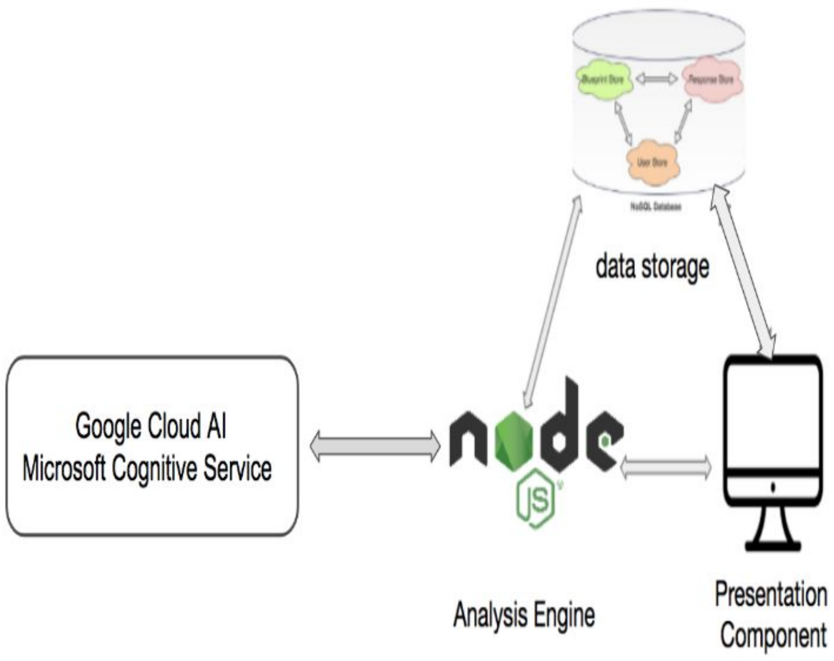


Figure 2 Data Analysis System Architecture

Presentation component consists of Data Access Module, Visualization Module, Parameter Setting Module, Rest API Consume Module. Analysis engine can be further divided into Routing Module, Data Access Module, TigerAware Service Module, Microsoft Azure

Cognitive Service Module, and Google Cloud AI Module. Data storage is a firebase realtime database that hosts all survey data.

3.1.1 Presentation Component

Presentation component is a user-friendly web page in the TigerAware dashboard. On that page, each question is displayed with three dropdowns and a submit button. These three dropdowns are designed for researchers to choose platform, participant, and method. Researchers can select different methods from different platforms and then applied them to responses of selected participants. Once the submit button is clicked, the result will be visualized in different ways depending on the question type and applied methods. Due to complexity, presentation component is further divided into four modules: Data Access Module, Visualization Module, Parameter Setting Module, Rest API Module. The design of each module will be explained as follows and the implementation will be discussed in the next section of this report.

Data Access Module

Data access Module is responsible for retrieving survey data from database. All other modules are built based on the top of this module. This module should provide the following functions:

- Fetch survey blueprint
    - Input: survey ID
    - Output: survey blueprint, including survey name, survey question,etc.

- Fetch survey responses
    - Input: survey iD
    - Output: survey-related data,including survey responses, number of responses,etc.
- Fetch participants:
    - Input: survey ID and question ID
    - Output: a set of participants ID

Visualization Module

Visualization module is used to display questions and results. For each question, it should be displayed with three dropdown selections: platform, method, and participant. Platform provides a set of platforms that can be used, while method and participant specify which algorithm to use and whose response should be analyzed. After setting these three parameters, a submit button should be enabled. User can click this button to view the analysis result. Also, results should be able to disappear and re-display based on users' preferences.

How to display results depends on the question type, selected method, and participant. Following are different visualization forms:

- Text: visualize simple text response
- Image: visualize image response
- Clock: visualize time response
- Word Cloud: visualize simple text response
- Pie Chart: visualize responses distribution

For example, if the question type is Yes-No, and researchers just want to view the response, text-based result should be displayed. However, if researchers want to know the distribution of responses from all participants, a pie chart should be displayed. For Text Field question, results can be visualized as a word cloud. Another example is that, for image type questions, if researchers want to know the emotion, image and detected emotion will be shown.

Parameter Setting Module

Parameter setting module is responsible for setting the available platform, method, and participants dynamically. For platform, there are three possible choices: TigerAware, Google, Microsoft. Because different platforms provide different functionalities, platforms value for each question will be initialized as a subset of the three choices. For example, for image type questions, Google and Microsoft will be initialized as platforms because only these two platforms provide advanced deep learning functionality to process image. For method, possible values include both basic statistical and advanced deep learning methods. Methods value for each question will be initialized based on question type and platform that has already been set. The reason is that different platforms provide different functionalities and also different types of questions need different methods. For participants, the value is the union of persons who answer the question. If there are more than two participants, we add "All Participants" to the participants list. Also, some parameters should change if other parameter changes. For example, if platform change from Microsoft to Google by researchers, the methods should also change dynamically.

This module should also provide a way for users to choose parameters globally and is designed to simplify researchers' operation especially for surveys that contain a large number of questions. For example, if researchers set global platform to TigerAware, platforms of all questions will be set to TigerAware instantly. This is very useful because manually set the platform value for each question one by one is a time consuming and tedious work.

Rest API Module

Rest API module is responsible for packaging these selected parameters(platform, method, and participant) and creating a request to trigger related analysis at analysis engine. Also, it should be able to parse the response returned from analysis engine to make visualization module display properly. In order to decouple different use cases, three different APIs are designed: individual analysis API, group analysis API, and Export Survey & Response API.

- Individual analysis API
    - Parameter: survey ID, question ID, participant ID, question type, platform, method
    - Return value: text, image URL, choice, time, etc.
- Group analysis API
    - Parameter: survey ID, question ID, question type, platform, method
    - Return value: choice distribution, emotion distribution, etc.
- Export Survey & Response API
    - Parameter: survey ID

- ○ Return value: CSV file containing all persons' response will prompt for user to download. Each row represents a person and each column represents a question.

Individual analysis API is designed to handle the case when the selected participant is an individual and group analysis API is designed for the case when the selected participant is all participants. Export Survey & Response API is designed to export survey questions and responses as a CSV file. Each row represents a participant and each column represents a question. The reason to separate these three APIs is that their logic is quite different and also the parameters needed by them are also different.

Rest API module should also be able to handle error because it might be possible that some exceptions happen during the process of communicating with analysis engine. These possible errors like the null pointer exception, are possible to make the whole application crash. Error handling feature will capture these exceptions to prevent the whole application from crashing. Also, it should be able to give some visual hint to both developers and users.

3.1.2 Analysis Engine

Analysis engine is a standalone node.js express web service that can provide powerful APIs for presentation component to consume. The functions of this application provides including basic statistics as well as advanced deep learning analysis. It is further divided into four modules: Data Access Module, Routing Module, TigerAware Service Module, Microsoft Azure Cognitive Service Module, and Google Cloud AI Module. The following part will discuss the functionality of each module. More implementation details will be talked about in a later section.

Data Access Module

Data access module is responsible for retrieving data from or saving data to database. With the help of data access module, other modules in analysis engine can be easily decoupled with data storage, which will be discussed in the following parts. Data Access Module should provide following functions:

- Initialize Database Connection
    - Input: Database configuration file, database URL,etc.
    - Output: Database object
- Fetch individual response
    - Input: survey ID, question ID, user ID
    - Output: user response
- Fetch all response
    - Input: survey ID
    - Output: all response of that particular survey
- Fetch survey and response
    - Input : survey ID
    - Output: survey questions and response
- Save result to database
    - Input: data, database object, path

- Output: if success, return nothing, otherwise, throw error

These functions are designed to initialize database connection, fetch responses, save results to database, etc. Three functions that can fetch responses from database are designed to meet the requirement of the Rest API Module of presentation component. The function fetch individual is designed for the individual analysis API, function fetch all response is designed for the group analysis API and fetch survey and response is designed for the export survey and response API. These three functions are separated from each other to make the whole application modularized and easy to maintain, understand and modify.

Routing Module

Routing module is designed to parse parameters from incoming request and then route request to a specific function module. These parameters include platform, method, and participants. These values will decide where this request should be routed, which algorithm to use and which data should be used as input to the selected method. Routing rule is listed below:

- If request is group analysis request, it will be forwarded to TigerAware group Analysis module.

If request is individual analysis request, based on platform, it will be forwarded to TigerAware individual analysis module, Microsoft individual analysis module, and Google individual analysis.

- If request is export survey and response request, it will be forwarded to TigerAware individual analysis module.

TigerAware Service Module

TigerAware Service Module is designed to provide non-deep-learning functions. It consists of three submodules: TigerAware group analysis service, TigerAware individual analysis service, and TigerAware survey and response export service. The reason why design in this way is that it will make the whole TigerAware Service module clear , extendable and maintainable.

TigerAware individual analysis service includes the following functions:

- Generate Word Cloud
  - Input: free text
  - Output: a map. Key is word and value is word frequency

TigerAware group analysis service includes the following functions:

- Get Distribution
  - Input: responses, question ID
  - Output: distribution of responses

TigerAware survey and response export service includes the following functions:

- Export survey and response function
  - Input: survey question and response
  - Output: Two dimensions matrix. Row represents participant, Column represents question

Microsoft Service Module

Microsoft service module is designed to provide functionalities to process images or natural language. It contains only Microsoft individual analysis service. Microsoft individual analysis can be further divided into two submodules: Microsoft vision service and Microsoft text service.

Microsoft vision service provides functions to process images. Details about these functions are listed below:

- Emotion Detection
    - Input: Image containing human face
    - Output: One of eight emotions categories including anger, contempt, disgust, fear, happiness, neutral, sadness and surprise

Microsoft text service provides functions to process natural language. Details about these functions are listed below:

- Sentiment Analysis
    - Input: text
    - Output: A score between 0 and 1. Scores close to 1 indicate positive sentiment, and scores close to 0 indicate negative sentiment
- Language Detection
    - Input: text

    - Output: detected language and a numeric score between 0 and 1. Scores close to 1 indicate 100% certainty that the identified language is true. A total of 120 languages are supported

Google Service Module

Google Service module is designed to provide the ability to process images and natural language. It only contains Google individual analysis service. Google individual analysis can be further divided into two submodules: Google vision service and Google text service.

Google vision service provides functions to process images. Details about these functions are listed below:

- Emotion Detection
    - Input: Image containing human face
    - Output: One of four emotions categories including joy, anger, sorrow, and surprise
- Label Detection
    - Input: Image containing labels
    - Output: Name of labels including general objects, locations, activities, animal species, products, and more
- Landmark Detection
    - Input: Image containing landmark
    - Output: Name of landmark. Both popular natural and man-made structures are supported

- Text Extraction
    - Input: Image containing text
    - Output: Text. Both handwriting and printed texts are supported
- Logo Detection
    - Input: Image containing famous logos
    - Output: Name of logos.

Google text service provides functions to process natural language. Details about these functions are listed below:

- Sentiment Analysis
    - Input: Text
    - Output: Score. Score is between -1(negative) and 1(positive).
- Content Classification
    - Input: Text
    - Output: A list of content categories that apply to the text

3.1.3 Data Storage

Data storage holds data for the TigerAware system. These data including but not limited to: survey blueprint, user information, response information and also processed result. Survey blueprint is the metadata about surveys, like questions, branches, conditions. User information contains user account, user name, user's authority and so on. Response information is the response from participants. Further analysis like emotion detection can be done after getting response data. Results from these further analyze are called processed result.

3.2 Survey Visualization Design

This survey visualization system is a single web page on the TigerAware dashboard and can be further divided into two modules: visualize module and file export module.

3.2.1 Visualize Module

Survey consists of questions and these questions connected through branches so that it can be abstracted as graphs. In graph theory, graph consists of nodes and edges. Nodes and edges could be used to represent questions and branches respectively. This module provides three functions: Format Convert, Visualize and Planar Embedding. The following parts will discuss the design of these functions.

Format convert function is designed to be an adapter between data from database and the visualization framework(D3) which will be used in this project. In order to use D3 visualize graph, input data need to be organized in a special format which is much different from the data format in database.

Visualize function is designed to visualize the survey as a graph. In this functionality, D3 will be used. D3.js is a JavaScript library for producing dynamic, interactive data visualizations in web browsers. It supports lots of layouts like force-directed graph layout, hierarchy, etc.

Planar Embedding function is designed to improve the shortcoming of the visualization framework(D3) that will be used in this project. Even though D3 is the state-of-the-art visualization framework, it doesn’t support planar embedding because of the complexity. In graph theory, a planar graph is a graph that can be drawn on the plane in such a way that its edges intersect only at their endpoints[7]. Such a drawing is called a planar embedding of the graph. However, D3 force-directed graph layout locates nodes randomly so that there are lots of cross between edges. In order to reduce the number of crossing, we need to find planar embedding manually. Tutte's embedding algorithm can be used to find planar embedding of graphs. In graph drawing and geometric graph theory, a Tutte embedding or barycentric embedding of a simple 3-vertex-connected planar graph is a crossing-free straight-line embedding with the properties that the outer face is a convex polygon and that each interior vertex is at the average (or barycenter) of its neighbors' positions[7]

3.2.3 File Export Module

File export module is designed to help researchers to present their surveys to others. It provides the functionality to export the visualization page as a PDF File. This is very helpful for researchers especially when there are lots of questions. Without this file export functionality, when researchers present surveys to others, they might need to draw manually, which is a time consuming and error-prone process.

3.3 Technology Stack

In the process of implementation, there were multiple technologies used. They can be categorized into front-end, which is used for displaying the interface on the web browsers; back-end, which provides API for front-end to consume; database, which stores all the data; and other utilities, like the software programs and cloud services.

3.3.1 Front End

1. HTML5: Markup language on which the TigerAware dashboard is based
2. CSS3: Bootstrap and Materialize implements the responsive design on the dashboard.
3. Angular 6: Typescript based open source web application framework led by the Angular team at Google
4. D3: Javascript library for producing dynamic, interactive data visualizations in web browsers.
5. Angular Material: UI component library for AngularJS developers and can help in constructing attractive, consistent, and functional web pages and web applications[x]
6. jQuery: A JavaScript library designed to simplify HTML DOM tree traversal and manipulation, as well as event handling, CSS animation, and Ajax[x]

3.3.2 Back End

1. Node.js: Open source, cross-platform javascript run-time environment that executes Javascript code on the server-side
2. Express: A Node.js web application framework, released as free and open-source software under the MIT License
3. Rsvg: A library that can convert web elements like SVG into other formats like PDF
4. Amazon Web Service: The cloud service provided by Amazon, where analysis engine is running
5. Azure Cognitive Service: Cloud based AI service provided by Microsoft
6. Google Cloud Platform: Cloud based AI service provided by Google

7. PM2: A node.js library which acts as an advanced node.js process manager

3.3.3 Database and other utilities

1. Firebase:Provide real-time database engine for this application
2. Visual Studio Code: Electron based IDE which is used to deploy Node.js application
3. Git and Github: A file-sharing, collaboration, and version control service widely used among the open-source community
4. FileZilla: A secure and reliable FTP software to transfer files between local system and remote server and vice versa
5. Terminus: A secure and reliable SSH software used to login to remote server

4. SYSTEM IMPLEMENTATION

This chapter focuses on the implementation of the survey visualization and data analysis system and detailed explanations of every module implemented in this project.

4.1 Data Analysis System Implementation

This part will focus on the implementation of three components designed in chapter 3, which are presentation component, analysis engine, and data storage.   A module-level design of this system is shown in Figure 3. The following part will discuss the details about each part and the data flow between each part.

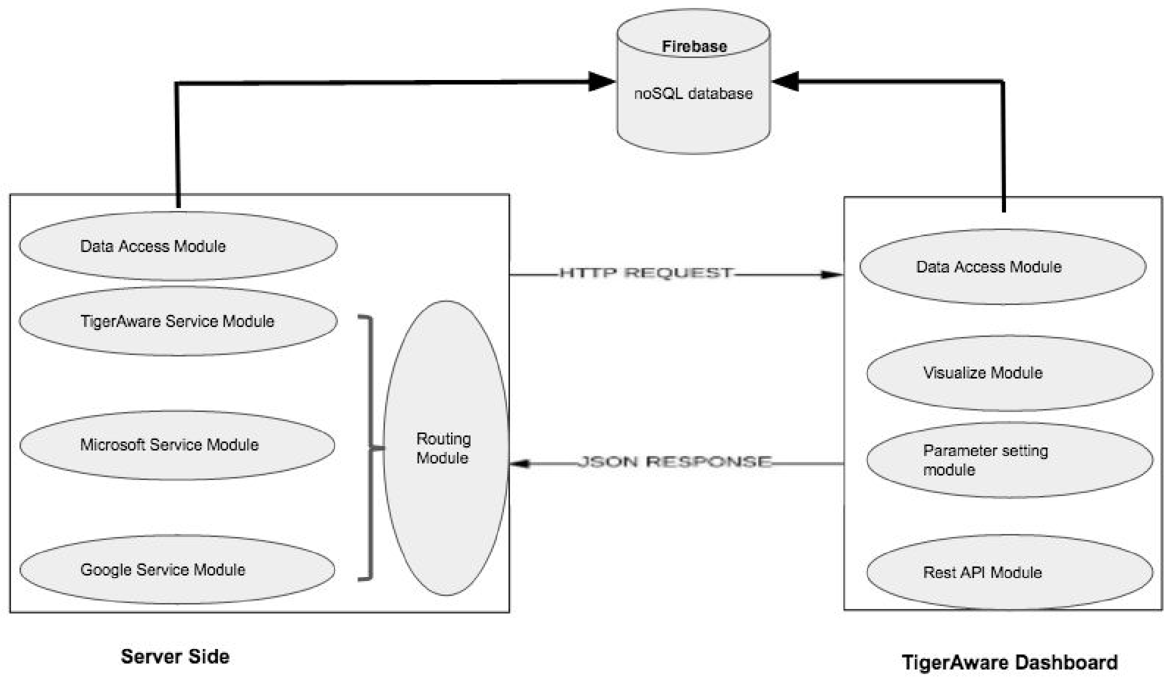


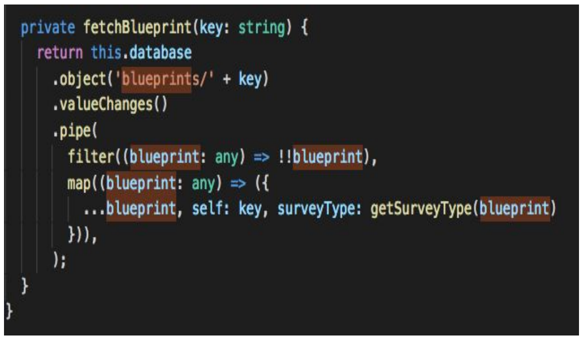
Figure 3 System Architecture Module View

4.1.1 Presentation Component Implementation

Presentation component focuses on the interaction with researchers. Researchers can select different parameters and then apply analysis algorithms to user responses and visualize the result. Presentation component is implemented based on Angular 6 and consists of several modules. The following part will further discuss the implementation of each module.

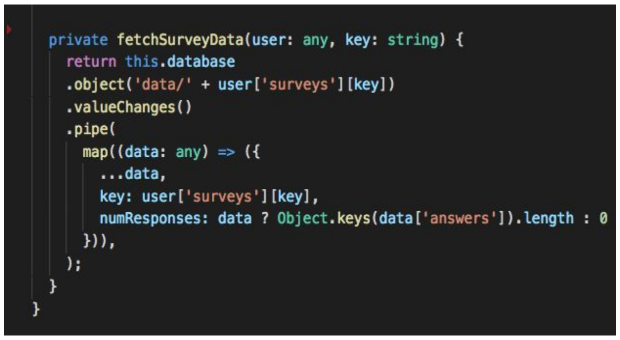
Data Access Module

Data access module is responsible for fetching survey data. In this project, firebase realtime database is used to store data and AngularFireDatabase is used to interact with database. Below are the implementation of functions designed for this module at Chapter 3



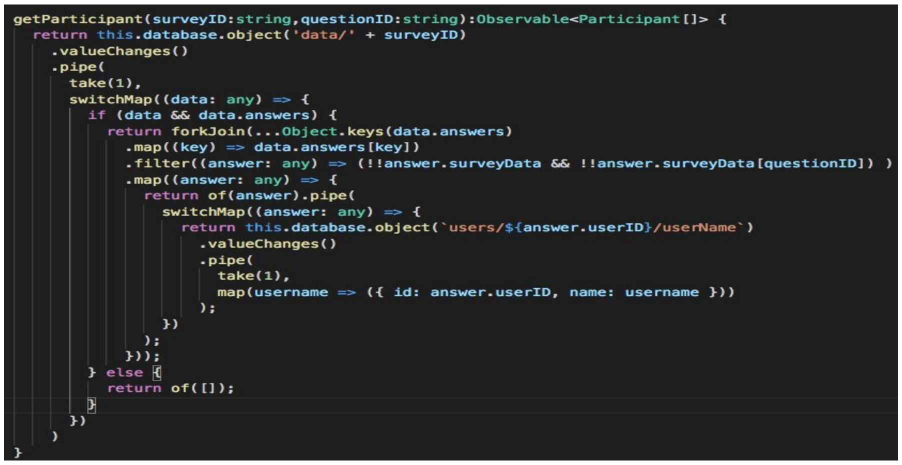
```
private fetchBlueprint(key: string) {
  return this.database
    .object('blueprints/' + key)
    .valueChanges()
    .pipe(
      filter((blueprint: any) => !!blueprint),
      map((blueprint: any) => ({
        ...blueprint, self: key, surveyType: getSurveyType(blueprint)
      })),
    );
  }
}
```

Figure 4 Implementation of Fetch Survey Blueprint



```
  private fetchSurveyData(user: any, key: string) {
    return this.database
    .object('data/' + user['surveys'][key])
    .valueChanges()
    .pipe(
      map((data: any) => ({
        ...data,
        key: user['surveys'][key],
        numResponses: data ? Object.keys(data['answers']).length : 0
      })),
    );
  }
}
```

Figure 5 Implementation of Fetch Survey Response



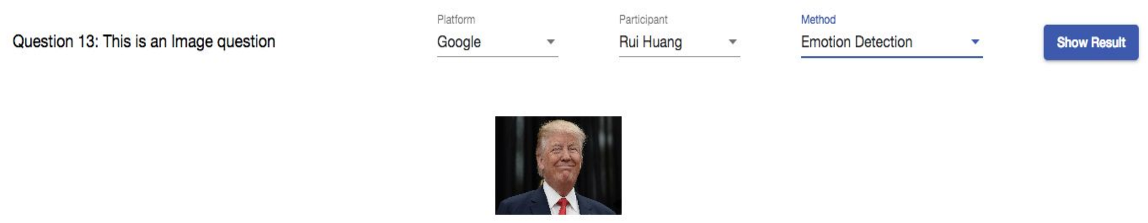
```
getParticipant(surveyID:string,questionID:string):Observable<Participant[]> {
  return this.database.object('data/' + surveyID)
    .valueChanges()
    .pipe(
      take(1),
      switchMap((data: any) => {
        if (data && data.answers) {
          return forkJoin(...Object.keys(data.answers)
            .map((key) => data.answers[key])
            .filter((answer: any) => (!!answer.surveyData && !!answer.surveyData[questionID]) )
            .map((answer: any) => {
              return of(answer).pipe(
                switchMap((answer: any) => {
                  return this.database.object(`users/${answer.userID}/userName`)
                    .valueChanges()
                    .pipe(
                      take(1),
                      map(username => ({ id: answer.userID, name: username }))
                    );
                })
              );
            }));
        } else {
          return of([]);
        }
      })
    )
}
```

Figure 6 Implementation of Fetch Participants

Visualization Module

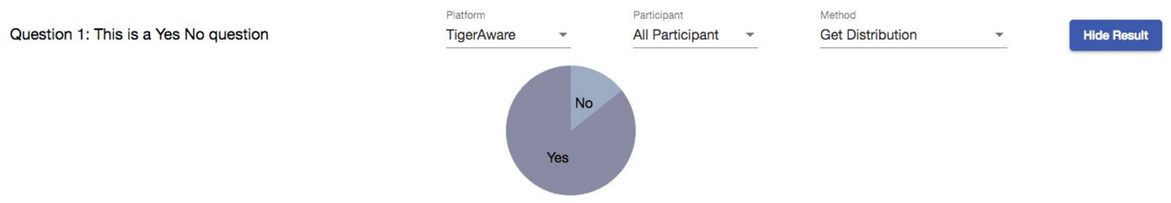
Visualization module is used to display results and questions. Angular Material, which is an attractive, consistent and functional web frontend framework[8], is used to display each question with a submit button and three dropdowns, shown in Figure 7. The button will be enabled automatically by Angular Material framework if all dropdowns' values have been set.

D3.js, which is a JavaScript library for manipulating documents based on data[6], is used to visualize the result. It supports lots of visualization forms including SVG, pie chart, clock, and canvas. Angular tag cloud module, which is a powerful word or tag cloud generating framework, is used to generate word clouds in this project. The following figures show different forms of visualization implemented in this project.


Question 13: This is an Image question

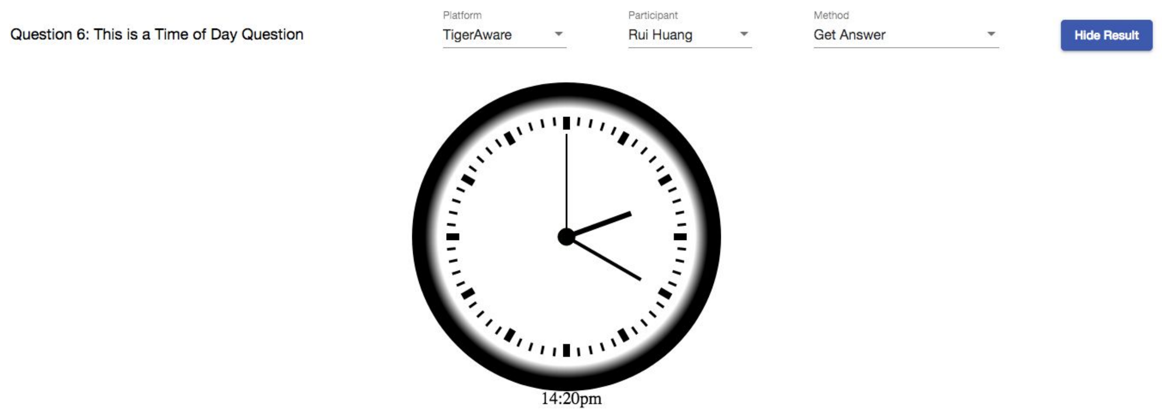
Platform: Google
Participant: Rui Huang
Method: Emotion Detection

Figure 7 Question Visualization


Question 1: This is a Yes No question

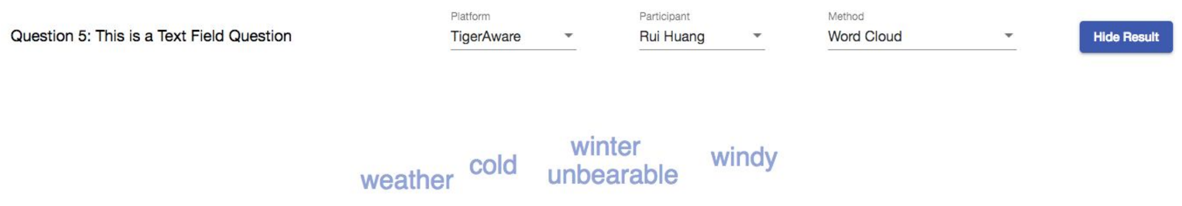
Platform: TigerAware
Participant: All Participant
Method: Get Distribution

Figure 8 Pie Chart Visualization


Question 6: This is a Time of Day Question

Platform: TigerAware
Participant: Rui Huang
Method: Get Answer

Figure 9 Clock Visualization


Question 5: This is a Text Field Question

Platform: TigerAware

Participant: Rui Huang

Method: Word Cloud

Figure 10 Word Cloud Visualization

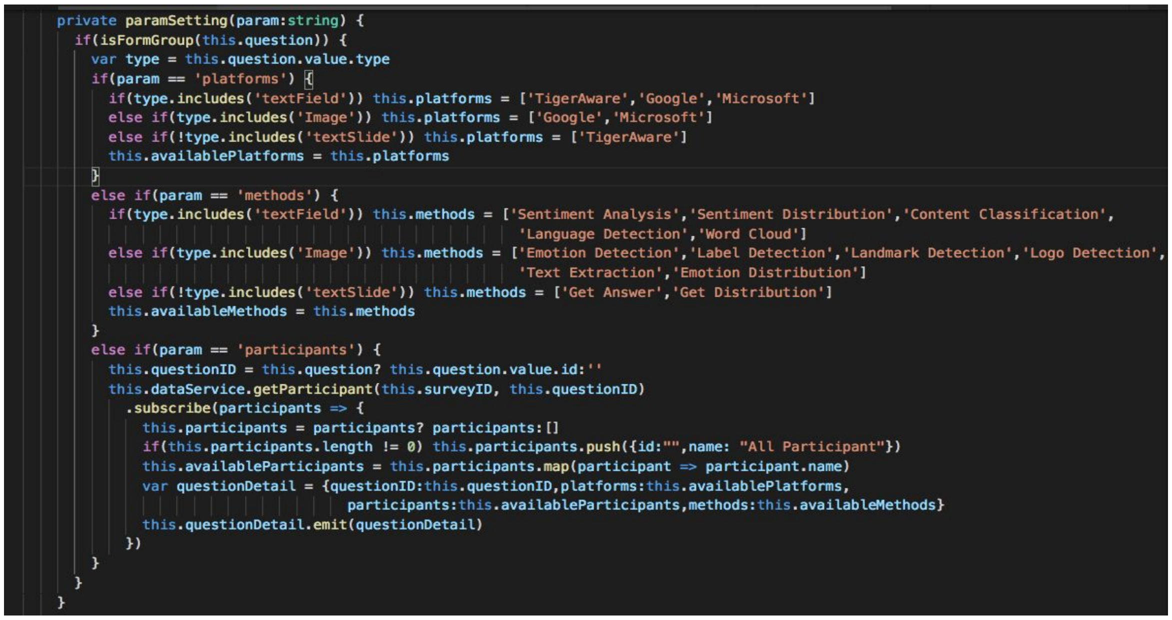
Parameter Setting Module

Parameter setting module is responsible for setting the available platform, method, and participants dynamically. Figure 11 shows the code of initializing these parameters.

For platforms, if question type is image or text field, platforms are initialized to be Google and Microsoft because they provide advanced deep learning functionality like image and natural language processing. For other types(e.g. Yes-No), TigerAware is initialized as the only platform.

For method, possible values include: Get Answer,Get Distribution, Get Emotion, Get Sentiment, Detect Celebrity, etc. Actual methods depend on question type and platform that has already been set. For example, for text questions, if the platform is Microsoft, methods should include Sentiment Analysis, Language Detection, if the platform is Google, methods should include Sentiment Analysis and Content Classification.

For participants, the value is all persons who answer the question. If there is no person, participant will be initialized as an empty list. If there are more than one person answering the question, participants will be initialized as all participants plus “All Participant”. The reason for adding “All Participant” is that researchers want to know the distribution of response sometimes. If researchers choose “All Participants” as the value of participant and choose the method “Get Distribution”, the result will be a pie chart which shows the distribution of responses.

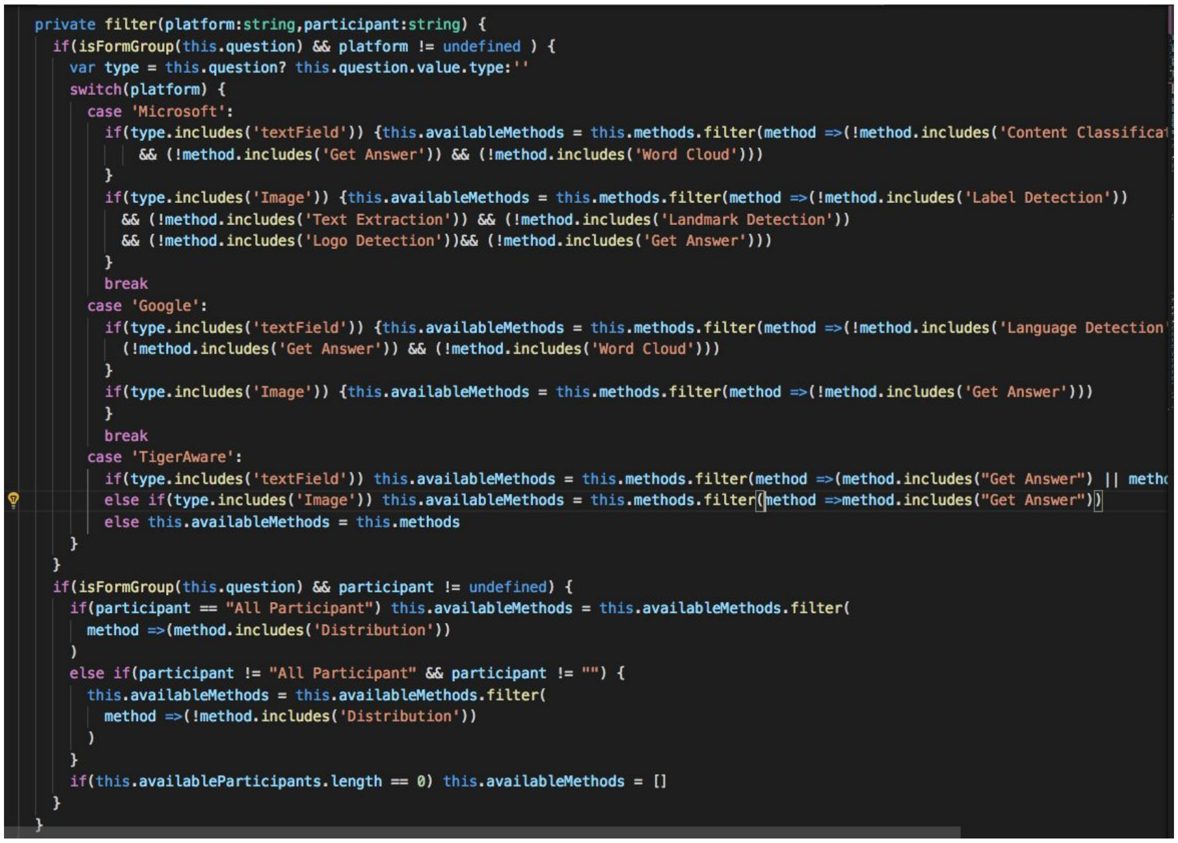


```
private paramSetting(param:string) {
  if(isFormGroup(this.question)) {
    var type = this.question.value.type
    if(param == 'platforms') {
      if(type.includes('textField')) this.platforms = ['TigerAware','Google','Microsoft']
      else if(type.includes('Image')) this.platforms = ['Google','Microsoft']
      else if(!type.includes('textSlide')) this.platforms = ['TigerAware']
      this.availablePlatforms = this.platforms
    }
    else if(param == 'methods') {
      if(type.includes('textField')) this.methods = ['Sentiment Analysis','Sentiment Distribution','Content Classification',
                                                      'Language Detection','Word Cloud']
      else if(type.includes('Image')) this.methods = ['Emotion Detection','Label Detection','Landmark Detection','Logo Detection',
                                                      'Text Extraction','Emotion Distribution']
      else if(!type.includes('textSlide')) this.methods = ['Get Answer','Get Distribution']
      this.availableMethods = this.methods
    }
    else if(param == 'participants') {
      this.questionID = this.question? this.question.value.id:''
      this.dataService.getParticipant(this.surveyID, this.questionID)
        .subscribe(participants => {
          this.participants = participants? participants:[]
          if(this.participants.length != 0) this.participants.push({id:"",name: "All Participant"})
          this.availableParticipants = this.participants.map(participant => participant.name)
          var questionDetail = {questionID:this.questionID,platforms:this.availablePlatforms,
                                participants:this.availableParticipants,methods:this.availableMethods}
          this.questionDetail.emit(questionDetail)
        })
    }
  }
}
```

Figure 11 Parameters Initialization Implementation

Some parameters need to be changed if other parameter changes. For example, for an Image type question, at first the platform is set to Microsoft, the available methods should include Detect Celebrity. However, if the platform change from Microsoft to Google, Detect Celebrity shouldn’t be included in the available methods because Google platform doesn’t provide this functionality. Figure 12 shows the implementation of the parameters dynamically change function.

This module also supports the functionality to choose parameters globally, which can significantly simplify researchers' operation especially for surveys that contain lots of questions. Angular provides a mechanism for parent and child components to communicate. Each Question is a child component and global control component is parent component. Global control component has three dropdown boxes and a submit button. These three dropdown boxes are Platform, Method, Participant respectively. Values of these dropdown boxes are initialized based on the corresponding value of each question. For each question, it will send its' platforms, methods, participants to global control module. After all questions have been initialized, global control module collects all the values for platform, method, and participant. If global dropdown value has been set by researchers manually, an event will be triggered. This event will send the global value to each question component. After receiving the global value, each question will set its corresponding dropdown value to the global value.



```
private filter(platform:string,participant:string) {
  if(isFormGroup(this.question) && platform != undefined ) {
    var type = this.question? this.question.value.type:''
    switch(platform) {
      case 'Microsoft':
        if(type.includes('textField')) {this.availableMethods = this.methods.filter(method =>(!method.includes('Content Classificat
            && (!method.includes('Get Answer')) && (!method.includes('Word Cloud')))
        }
        if(type.includes('Image')) {this.availableMethods = this.methods.filter(method =>(!method.includes('Label Detection'))
          && (!method.includes('Text Extraction')) && (!method.includes('Landmark Detection'))
          && (!method.includes('Logo Detection'))&& (!method.includes('Get Answer')))
        }
        break
      case 'Google':
        if(type.includes('textField')) {this.availableMethods = this.methods.filter(method =>(!method.includes('Language Detection'
          (!method.includes('Get Answer')) && (!method.includes('Word Cloud')))
        }
        if(type.includes('Image')) {this.availableMethods = this.methods.filter(method =>(!method.includes('Get Answer')))
        }
        break
      case 'TigerAware':
        if(type.includes('textField')) this.availableMethods = this.methods.filter(method =>(method.includes("Get Answer") || methc
        else if(type.includes('Image')) this.availableMethods = this.methods.filter(method =>method.includes("Get Answer"))
        else this.availableMethods = this.methods
    }
  }
  if(isFormGroup(this.question) && participant != undefined) {
    if(participant == "All Participant") this.availableMethods = this.availableMethods.filter(
      method =>(method.includes('Distribution'))
    )
    else if(participant != "All Participant" && participant != "") {
      this.availableMethods = this.availableMethods.filter(
        method =>(!method.includes('Distribution'))
      )
    }
    if(this.availableParticipants.length == 0) this.availableMethods = []
  }
}
```

Figure 12 Parameters dynamically change Implementation

Rest API Module

Rest API module packages parameters, create and send requests to analysis engine, and parse the result returned. Three APIs(client-side) are implemented in this module: Individual Analysis API, Group Analysis API, and Export Survey & Response API.

Individual analysis API packages survey ID, question ID, question Type, platform and method and create and send HTTP requests using Angular HttpClient module. Value returned is participant’s response. Like text, choice, image, etc. Implementation is shown in Figure 13.

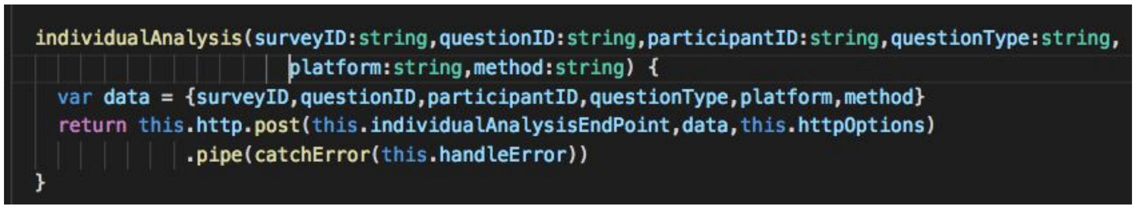

```
individualAnalysis(surveyID:string,questionID:string,participantID:string,questionType:string,
                platform:string,method:string) {
  var data = {surveyID,questionID,participantID,questionType,platform,method}
  return this.http.post(this.individualAnalysisEndPoint,data,this.httpOptions)
          .pipe(catchError(this.handleError))
}
```

Figure 13 individual analysis API client side Implementation

Group analysis API packages survey ID, question ID, platform and method and create and send HTTP requests using Angular HttpClient module. Return value is the distribution of responses. Like choice distribution and emotion distribution. Implementation is shown in Figure 14.

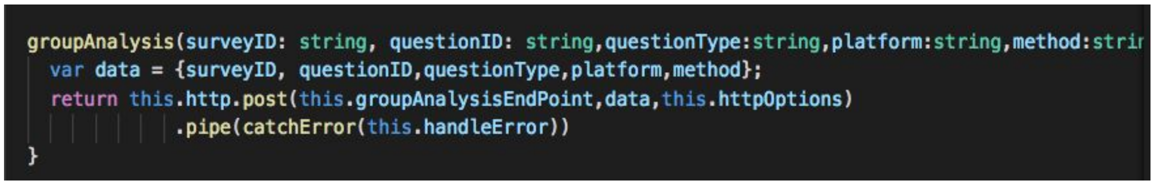

```
groupAnalysis(surveyID: string, questionID: string,questionType:string,platform:string,method:strir
  var data = {surveyID, questionID,questionType,platform,method};
  return this.http.post(this.groupAnalysisEndPoint,data,this.httpOptions)
          .pipe(catchError(this.handleError))
}
```

Figure 14 group analysis API client side Implementation

Export Survey & Response API only requires survey ID to work. Return value is the path of a CSV file. Implementation is shown in Figure 15.

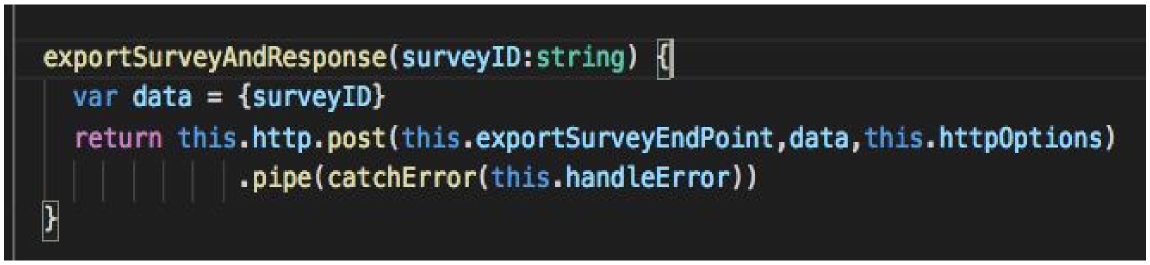
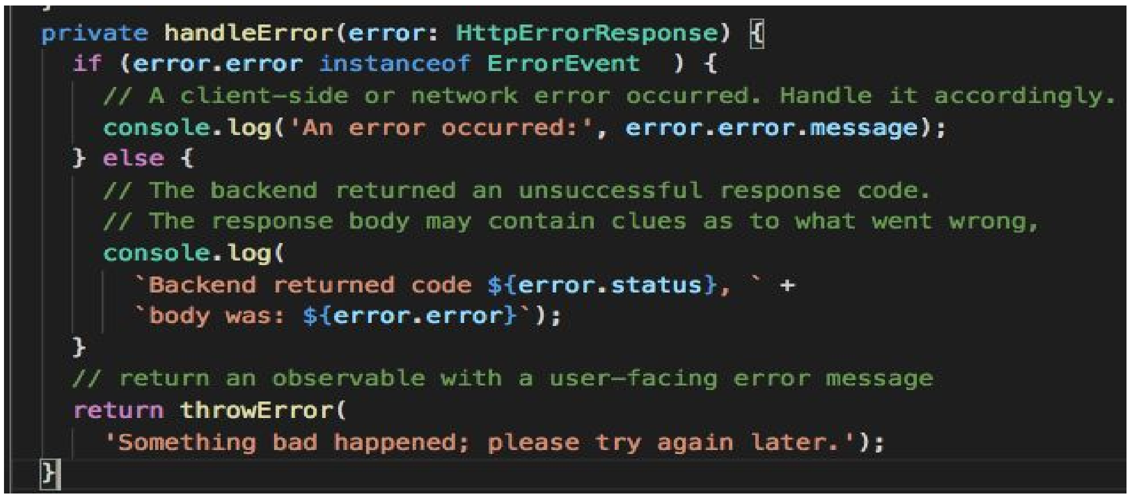

```
exportSurveyAndResponse(surveyID:string) {
  var data = {surveyID}
  return this.http.post(this.exportSurveyEndPoint,data,this.httpOptions)
          .pipe(catchError(this.handleError))
}
```

Figure 15 Export Survey & Response API client side Implementation

This module also supports the error handling function. Based on Angular HTTP Error Response module, errors can be easily classified to client-side error and server-side error. Implementation is shown in Figure 16.



```
private handleError(error: HttpErrorResponse) {
  if (error.error instanceof ErrorEvent  ) {
    // A client-side or network error occurred. Handle it accordingly.
    console.log('An error occurred:', error.error.message);
  } else {
    // The backend returned an unsuccessful response code.
    // The response body may contain clues as to what went wrong,
    console.log(
      `Backend returned code ${error.status}, ` +
      `body was: ${error.error}`);
  }
  // return an observable with a user-facing error message
  return throwError(
    'Something bad happened; please try again later.');
}
```

Figure 16 Error handling Implementation

4.1.2 Analysis Engine Implementation

Building using Nodejs, analysis engine is a REST API for presentation component to consume. It can be further divided into four modules: Data Access Module, Routing Module, TigerAware Service Module, Microsoft Azure Cognitive Service Module, and Google Cloud AI Module.

Data Access Module

Data Access Module is responsible for retrieving data from or saving data back to Firebase using Admin SDK. In order to interact with Firebase realtime database, an asynchronous listener is attached to Firebase realtime database. The listener is triggered once for the initial state of the data and again anytime an event happens[9]. These events include

Value, Child Added, Child Changed, etc. The value event is used to read a static snapshot of the contents at a given database path, as they existed at the time of the read event. It is triggered once with the initial data and again every time the data changes. The child added event is typically used when retrieving a list of items from the database. Unlike value which returns the entire contents of the location, child_added is triggered once for each existing child and then again every time a new child is added to the specified path. The child changed event is triggered any time a child node is modified. This includes any modifications to descendants of the child node. The method Once can help if a callback needs to be called once and then immediately removed. Firebase realtime database also provides several methods to save data, including but not limited to Set, Update, Push, etc. The set method can write or replace data to a defined path. The update method is used to write to multiple children of a database location at the same time without overwriting other child nodes. Push method can generate a unique key for new data and push them to a defined path without overwriting others. Based on these mechanisms Firebase realtime database provides, five functions design in Chapter 3 are implemented as below.

Database Initialization function create the connection between analysis engine and data storage. In this function, service account, which can get from Firebase console, credential, and database url are set up. Implementation is shown in Figure 17.

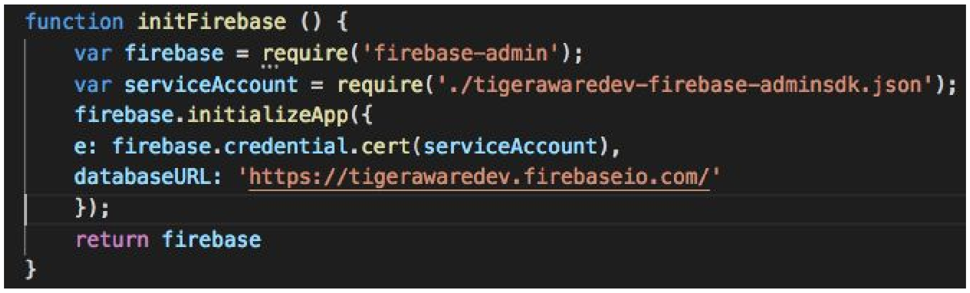

```
function initFirebase () {
    var firebase = require('firebase-admin');
    var serviceAccount = require('./tigerawaredev-firebase-adminsdk.json');
    firebase.initializeApp({
    e: firebase.credential.cert(serviceAccount),
    databaseURL: 'https://tigerawaredev.firebaseio.com/'
    });
    return firebase
}
```

Figure 17 Database Initialization Implementation

. Fetch individual response function can retrieve user response based on survey ID, question ID, and user ID. Keyword async, which provides straight-forward, powerful functions for working with asynchronous events, is used to deal with the asynchronous issue of nodejs. Any function decorated by keyword async will return a promise. Implementation of the function fetch individual response is shown in Figure 18.

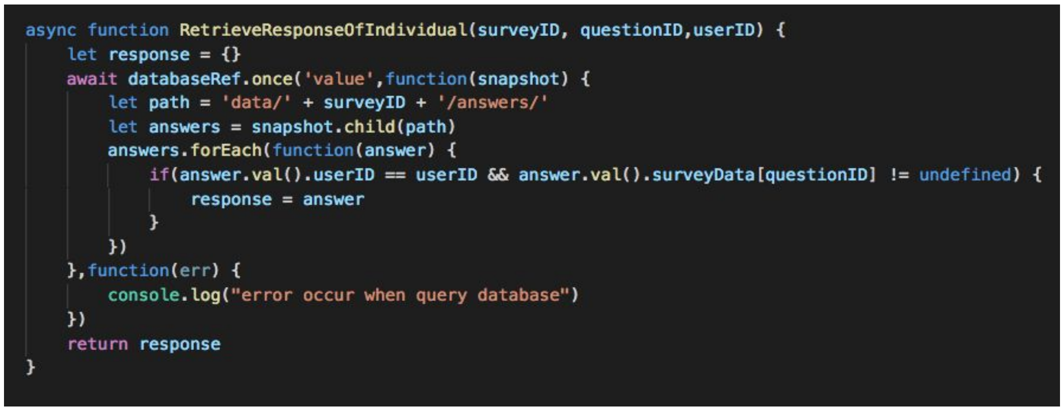
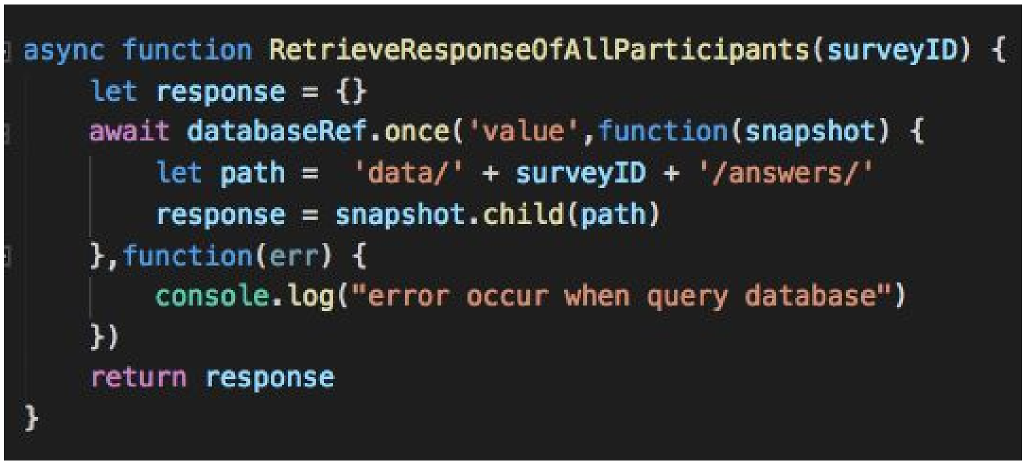

```
async function RetrieveResponseOfIndividual(surveyID, questionID,userID) {
    let response = {}
    await databaseRef.once('value',function(snapshot) {
        let path = 'data/' + surveyID + '/answers/'
        let answers = snapshot.child(path)
        answers.forEach(function(answer) {
            if(answer.val().userID == userID && answer.val().surveyData[questionID] != undefined) {
                response = answer
            }
        })
    },function(err) {
        console.log("error occur when query database")
    })
    return response
}
```

Figure 18 Fetch individual response Implementation

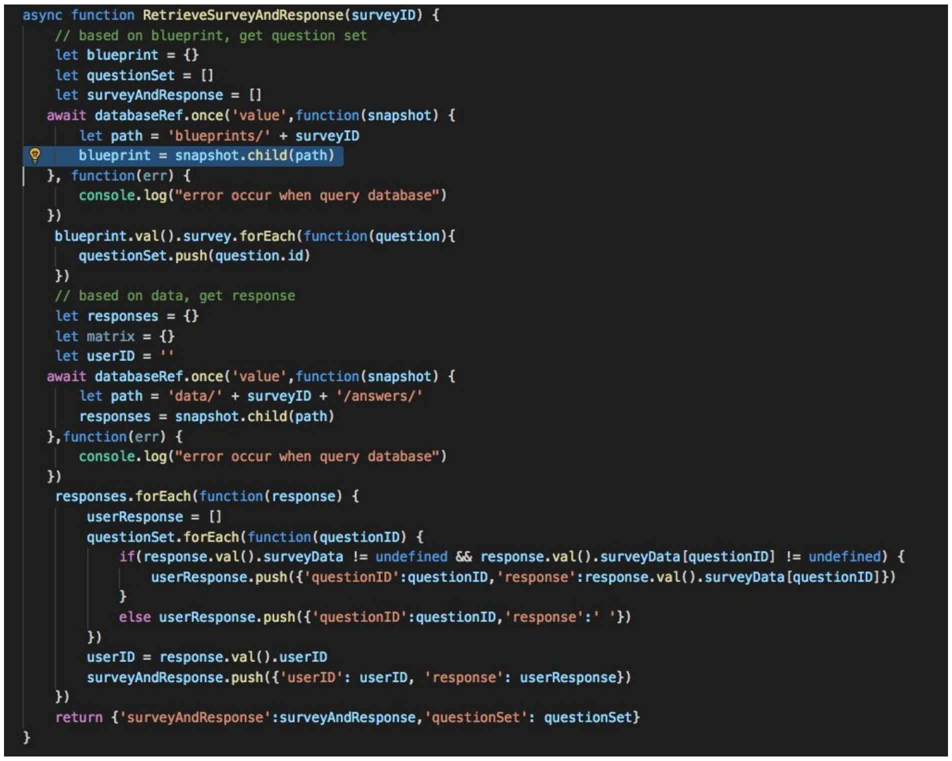
Fetch all response function can retrieve all responses based on survey ID. Once event is used to get the data from Firebase and then remove the listener right away. Implementation is shown in Figure 19.



```
async function RetrieveResponseOfAllParticipants(surveyID) {
    let response = {}
    await databaseRef.once('value',function(snapshot) {
        let path =  'data/' + surveyID + '/answers/'
        response = snapshot.child(path)
    },function(err) {
        console.log("error occur when query database")
    })
    return response
}
```

Figure 19 Fetch all response Implementation

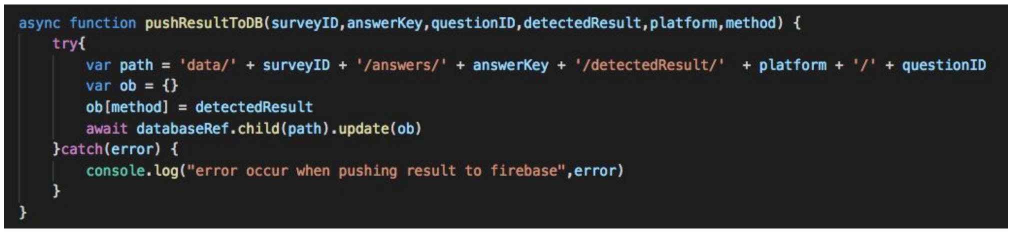
Fetch survey and response function fetches the response for each question. Question and response are organized as a map. The key is question ID and value is response. Implementation is shown in Figure 20.



```
async function RetrieveSurveyAndResponse(surveyID) {
    // based on blueprint, get question set
    let blueprint = {}
    let questionSet = []
    let surveyAndResponse = []
   await databaseRef.once('value',function(snapshot) {
       let path = 'blueprints/' + surveyID
       blueprint = snapshot.child(path)
   }, function(err) {
       console.log("error occur when query database")
   })
    blueprint.val().survey.forEach(function(question){
       questionSet.push(question.id)
    })
    // based on data, get response
    let responses = {}
    let matrix = {}
    let userID = ''
   await databaseRef.once('value',function(snapshot) {
       let path = 'data/' + surveyID + '/answers/'
       responses = snapshot.child(path)
   },function(err) {
       console.log("error occur when query database")
   })
    responses.forEach(function(response) {
        userResponse = []
        questionSet.forEach(function(questionID) {
            if(response.val().surveyData != undefined && response.val().surveyData[questionID] != undefined) {
                userResponse.push({'questionID':questionID,'response':response.val().surveyData[questionID]})
            }
            else userResponse.push({'questionID':questionID,'response':' '})
        })
        userID = response.val().userID
        surveyAndResponse.push({'userID': userID, 'response': userResponse})
    })
    return {'surveyAndResponse':surveyAndResponse,'questionSet': questionSet}
}
```

Figure 20 Fetch survey and response Implementation

Save result to database function uses the update function provided by Firebase realtime database to push data to a path. Implementation is shown in Figure 21.



```
async function pushResultToDB(surveyID,answerKey,questionID,detectedResult,platform,method) {
    try{
        var path = 'data/' + surveyID + '/answers/' + answerKey + '/detectedResult/'  + platform + '/' + questionID
        var ob = {}
        ob[method] = detectedResult
        await databaseRef.child(path).update(ob)
    }catch(error) {
        console.log("error occur when pushing result to firebase",error)
    }
}
```

Figure 21 Save data to database Implementation

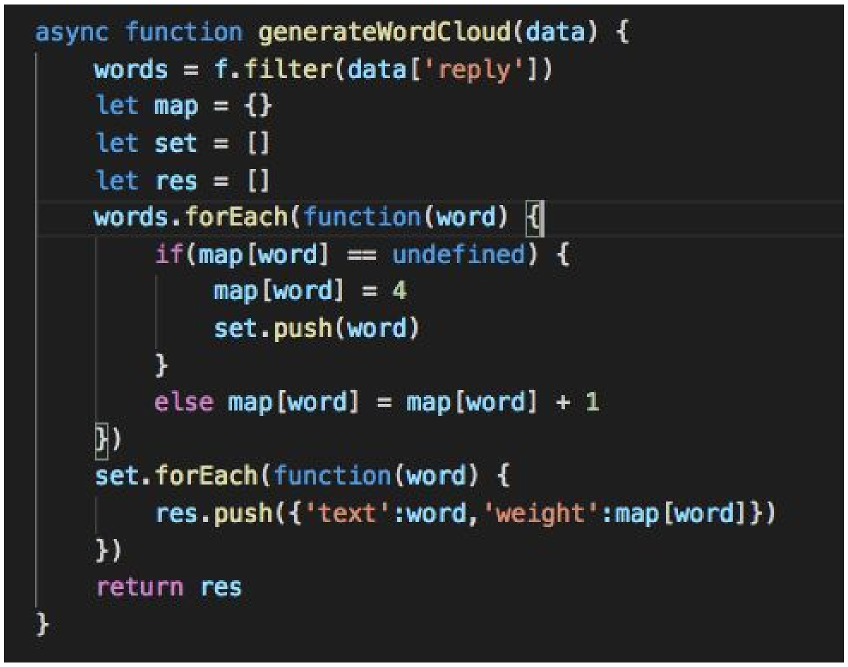
Routing Module

Routing module extracts parameters from requests and routes the request to different functional modules. There are three different requests designed in Chapter 3: individual analysis request and distribution analysis request and export survey & response request. Based on the request parameters, these requests will be forwarded to different functional modules. The main server program is responsible for receiving requests and responses. After receiving the request, requests will be forwarded to different platforms based on platform value. Then each module will call a specific method based on the method value and return the result to the main server program. Main server program then returns the result to presentation component. The reason why designing in this way is that it can make different parts decoupled from each other and only focus on its own task. For example, the main server program’s task is receiving requests and returning a response, so it doesn’t need to consider which functional should the request be forwarded to.

TigerAware Service Module

TigerAware service module, which provides non deep learning functionality, is designed as a complementary module for Microsoft service module and Google service module, which provide deep learning based functionality. TigerAware service module implements three functions: Generate Word Cloud, Get Distribution and Export Survey and Response. Details about these implementations are discussed below.

Generate word cloud function is responsible for splitting input text based on punctuation mark,filtering the stop words and then building a word frequency map. Stop words are the most common words that most search engines avoid, saving space and time in processing large data during crawling or indexing[10]. Usually, stop words don't contain much information. Node-stopwords-filter module is used to split text and filter stop words. Map is used to calculate the word frequency. This function only needs to return word frequency map to presentation component because presentation component will take care of the formation of word clouds. Implementation of this function is shown in Figure 22.

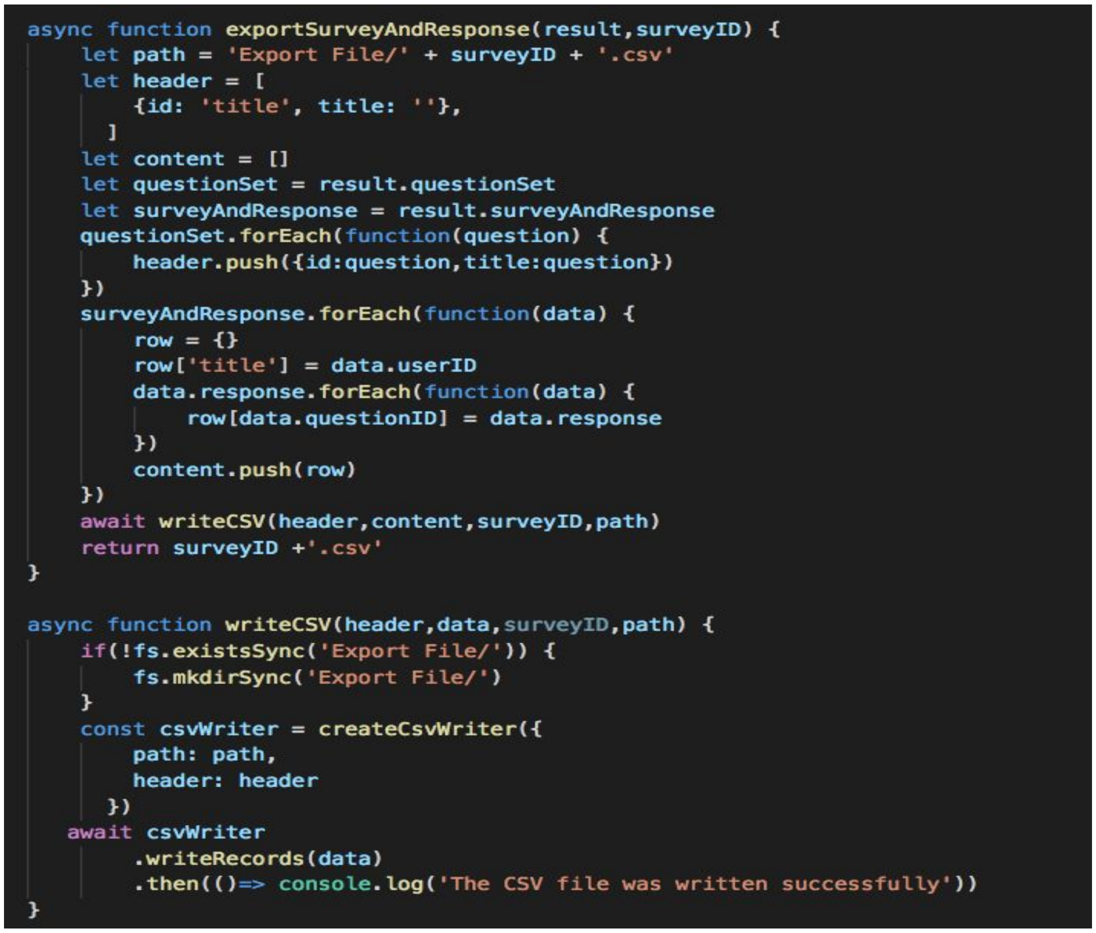


```
async function generateWordCloud(data) {
    words = f.filter(data['reply'])
    let map = {}
    let set = []
    let res = []
    words.forEach(function(word) {
        if(map[word] == undefined) {
            map[word] = 4
            set.push(word)
        }
        else map[word] = map[word] + 1
    })
    set.forEach(function(word) {
        res.push({'text':word,'weight':map[word]})
    })
    return res
}
```

Figure 22 Generate Word Cloud Implementation

Get distribution function is responsible for calculating the distribution of response based on survey ID and question ID. For example, it can calculate how many percent of participants choose A and how many percent choose B if the question is a multiple choice question. In this implementation, map is used to calculate the frequency of each response.

Based on survey ID, export survey and response function can create a CSV file whose row represents participant and column represents column for that survey. CSV file is stored at the folder called ‘Export File’ and the name of the CSV file is the id of the survey. In this implementation, the node module csv-writer is used to generate CSV file. Implementation is shown in Figure 23.



```
async function exportSurveyAndResponse(result,surveyID) {
    let path = 'Export File/' + surveyID + '.csv'
    let header = [
        {id: 'title', title: ''},
      ]
    let content = []
    let questionSet = result.questionSet
    let surveyAndResponse = result.surveyAndResponse
    questionSet.forEach(function(question) {
        header.push({id:question,title:question})
    })
    surveyAndResponse.forEach(function(data) {
        row = {}
        row['title'] = data.userID
        data.response.forEach(function(data) {
            row[data.questionID] = data.response
        })
        content.push(row)
    })
    await writeCSV(header,content,surveyID,path)
    return surveyID +'.csv'
}

async function writeCSV(header,data,surveyID,path) {
    if(!fs.existsSync('Export File/')) {
        fs.mkdirSync('Export File/')
    }
    const csvWriter = createCsvWriter({
        path: path,
        header: header
      })
   await csvWriter
        .writeRecords(data)
        .then(()=> console.log('The CSV file was written successfully'))
}
```

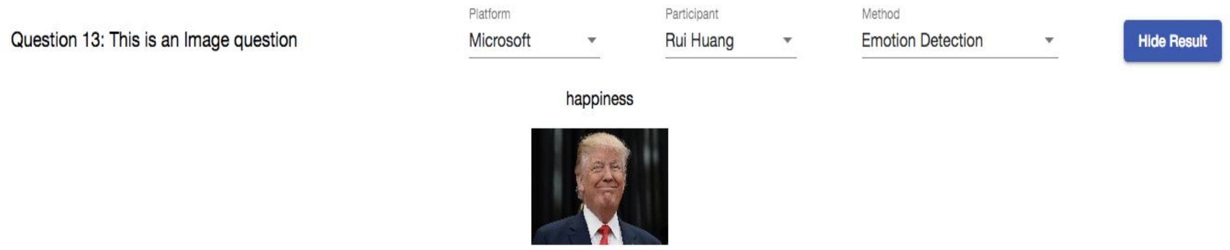
Figure 23 Export Survey and Response Implementation

Microsoft Service Module

Microsoft service is designed to extract information from images and natural language based on deep neural network algorithms. Instead of training network models manually, Microsoft's online services, which called Microsoft Azure Cognitive service is used. These services are exposed to users through REST API and include Microsoft vision service and Microsoft text service. Implementations of these services are discussed below.

Microsoft vision service provides the ability to extract rich information from images to categorize and process visual data and perform machine-assisted moderation of images[11]. Currently, only emotion detection is supported.

Emotion detection function takes an image containing human face and returns the emotion. This function is very important in cases when participant's emotion is useful for researchers to make decisions. Microsoft Azure vision service is used to detect emotion. Any facial image will be classified into one of the eight categories: anger, contempt, disgust, fear, happiness, neutral, sadness, surprise. Request, which is a node module, is used to send requests to Microsoft Azure platform. Promise, which is an object representing the eventual completion or failure of an asynchronous operation, is used to handle the asynchronous issue of nodejs. The result of this function is shown in Figure 24.


Question 13: This is an Image question

Platform: Microsoft

Participant: Rui Huang

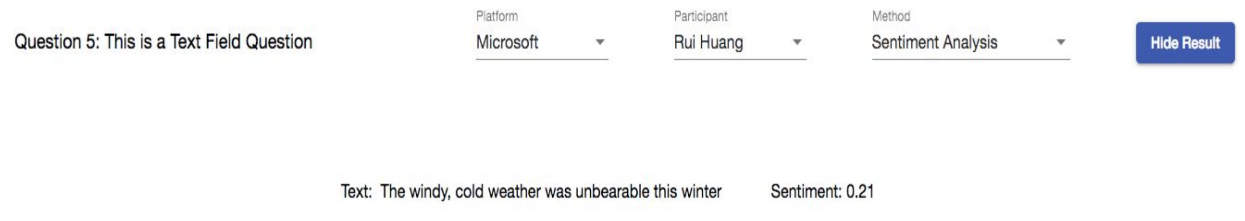
Method: Emotion Detection

happiness

Figure 24 Emotion Detection based on Microsoft Azure

Microsoft text service provides the ability to process natural language. Currently, sentiment analysis and language detection are supported.

Sentiment analysis function takes a text and returns a numerical value which represents the sentiment. The value is between 0, which represents a very negative sentiment, and 1, which represents a very positive sentiment. This function is very helpful if researchers want to take participants' mood situation into consideration. The result of this function is shown in Figure 25.



Question 5: This is a Text Field Question

Platform: Microsoft

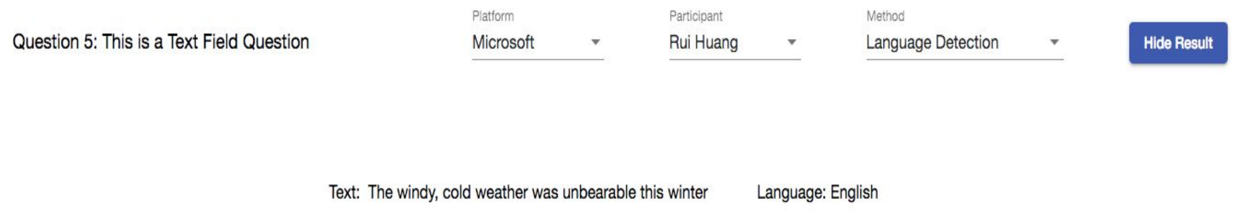
Participant: Rui Huang

Method: Sentiment Analysis

Text: The windy, cold weather was unbearable this winter   Sentiment: 0.21

Figure 25 Sentiment Analysis based on Microsoft Azure

Language detection function takes a text and returns detected language and a numeric score between 0 and 1. Scores close to 1 indicate 100% certainty that the identified language is true. A total number of 120 languages are supported. The result of this function is shown in Figure 26.



Question 5: This is a Text Field Question

Platform: Microsoft

Participant: Rui Huang

Method: Language Detection

Text: The windy, cold weather was unbearable this winter   Language: English

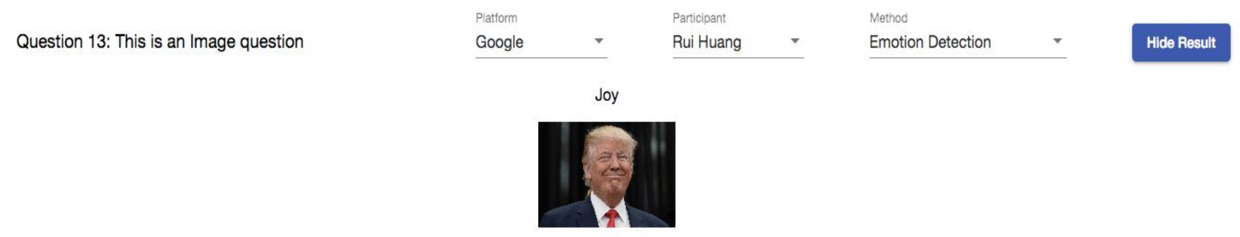
Figure 26 Language Detection based on Microsoft Azure

Google Service Module

Google service module is designed to process images and natural language based on deep neural network algorithms. Google’s online service, which is called Google Cloud AI, is used in this project. These services are exposed to users through client library and include Google vision service and Google text service. Results of these services are discussed below.

Google vision service provides the ability to process images. These abilities include emotion detection, label detection, landmark detection, text extraction, and logo detection.

Emotion detection takes an image containing human face and returns the emotion. Google Cloud Vision service is used. Any facial image will be classified into one of the four categories: joy, anger, sorrow, surprise. Keywords async and await are used to deal with the asynchronous issue. Result of this function is shown in Figure 27.


Question 13: This is an Image question

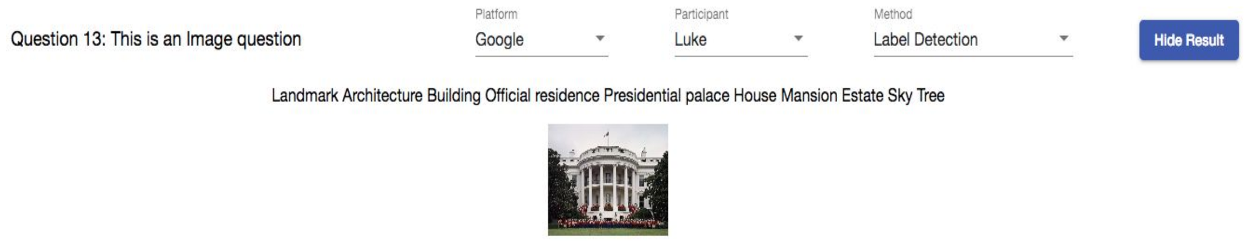
Platform: Google

Participant: Rui Huang

Method: Emotion Detection

Figure 27 Emotion Detection based on Google Cloud AI

Label detection can detect and extract information about entities in an image, across a broad group of categories. Thes labels should identify general objects, locations, activities, animal species, products and more. Result is shown in Figure 28.


Question 13: This is an Image question

Platform: Google

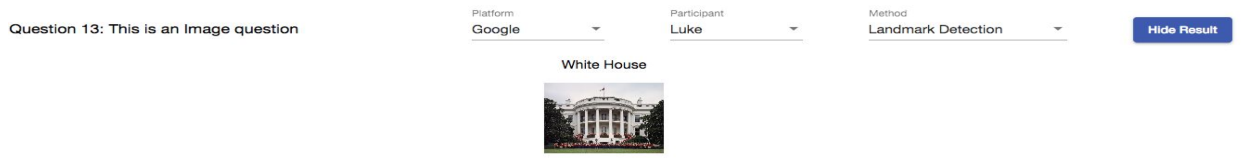
Participant: Luke

Method: Label Detection

Landmark Architecture Building Official residence Presidential palace House Mansion Estate Sky Tree

Figure 28 Label Detection based on Google Cloud AI

Landmark detection could recognize more than 9,000 natural and manmade landmarks from around the world[x]. Result is shown in Figure 29.


Question 13: This is an Image question

Platform: Google

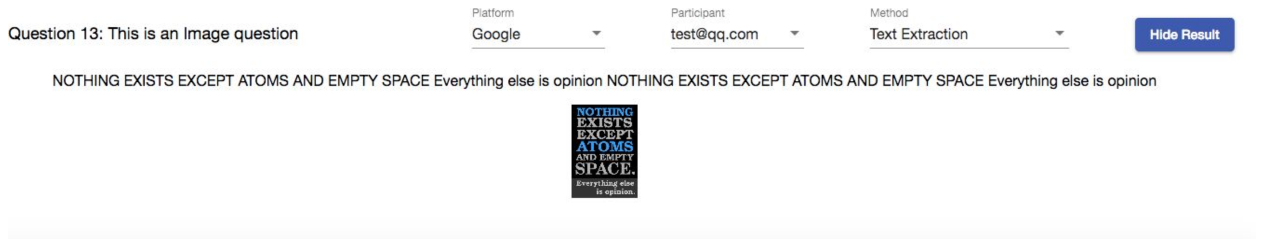
Participant: Luke

Method: Landmark Detection

White House

Figure 29 Landmark Detection based on Google Cloud AI

Text Extraction detects text content in an image. Both printed and handwritten text are supported. Result is shown in Figure 30.


Question 13: This is an Image question

Platform: Google

Participant: test@qq.com

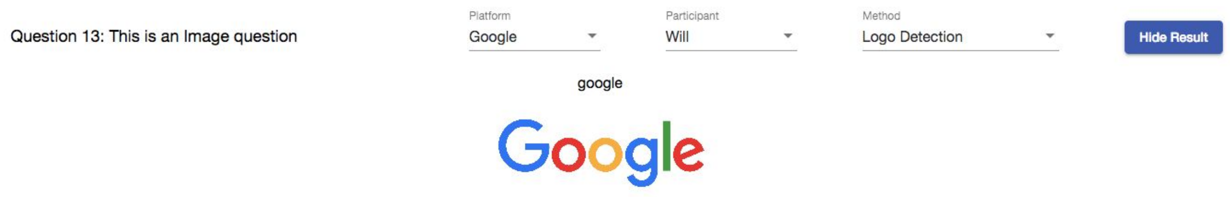
Method: Text Extraction

NOTHING EXISTS EXCEPT ATOMS AND EMPTY SPACE Everything else is opinion NOTHING EXISTS EXCEPT ATOMS AND EMPTY SPACE Everything else is opinion

Hide Result
NOTHING EXISTS EXCEPT ATOMS AND EMPTY SPACE. Everything else is opinion.

Figure 30 Text Extraction based on Google Cloud AI

Logo detection could detect popular product logos within an image. This function is very helpful if researchers want to investigate markets. Google Cloud AI logo detect client library is used in the implementation. Result is shown in Figure 31.


Question 13: This is an Image question

Platform: Google

Participant: Will

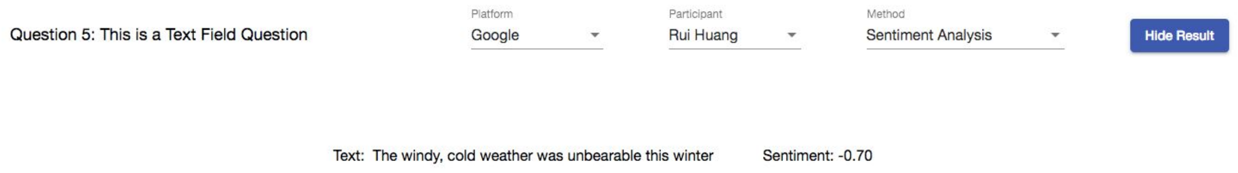
Method: Logo Detection

google

Figure 31 Logo Detection based on Google Cloud AI

Google text service provides abilities to process natural language. These abilities include sentiment analysis and content classification.

Sentiment analysis takes a text as input and returns a numerical value which represents the sentiment. The value is between -1, which represents negative sentiment, and 1 which represents positive sentiment. The result is shown in Figure 32.


Question 5: This is a Text Field Question

Platform: Google

Participant: Rui Huang

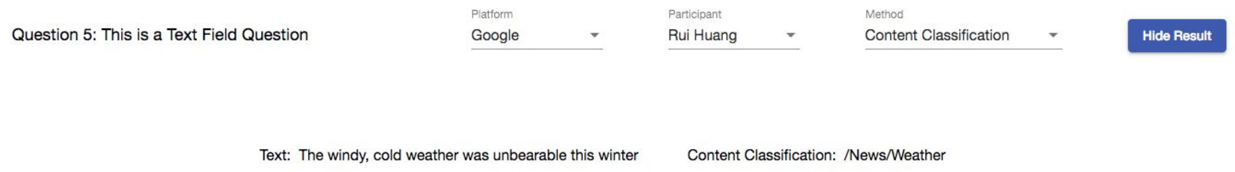
Method: Sentiment Analysis

Text: The windy, cold weather was unbearable this winter

Sentiment: -0.70

Figure 32 Sentiment Analysis based on Google Cloud AI

Content classification analyzes a text and returns a list of content categories that apply to the text found in the document by using the classifyText method provided by Google client library. Result is shown in Figure 33.


Question 5: This is a Text Field Question

Platform: Google

Participant: Rui Huang

Method: Content Classification

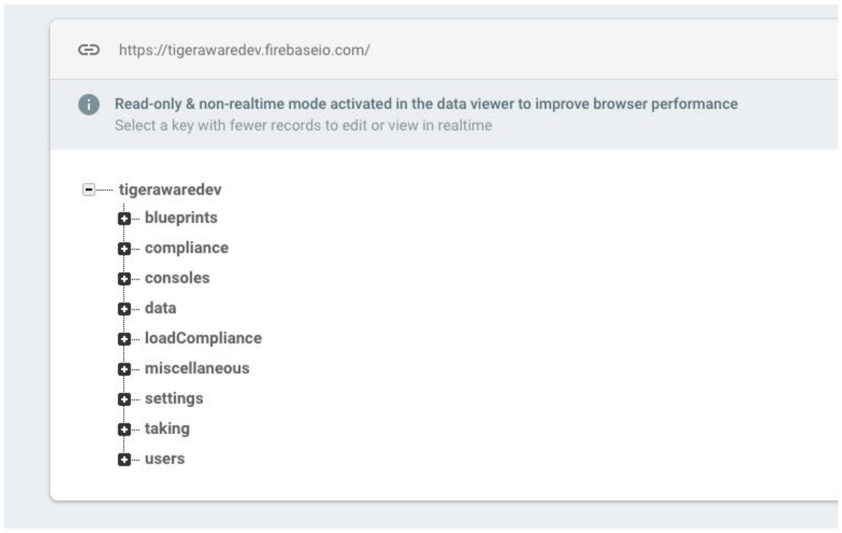
Text: The windy, cold weather was unbearable this winter

Content Classification: /News/Weather

Figure 33 Content Classification based on Google Cloud AI

4.1.3 Data Storage Implementation

Database used in this project is the Firebase Realtime Database, which is a cloud-hosted database. Data is stored as a JSON object and synchronized in realtime to every connected client. The organization of TigerAware system data storage shown in Figure 34. The following part will introduce the structure of some fields that are important and closely related to this project.


https://tigerawaredev.firebaseio.com/

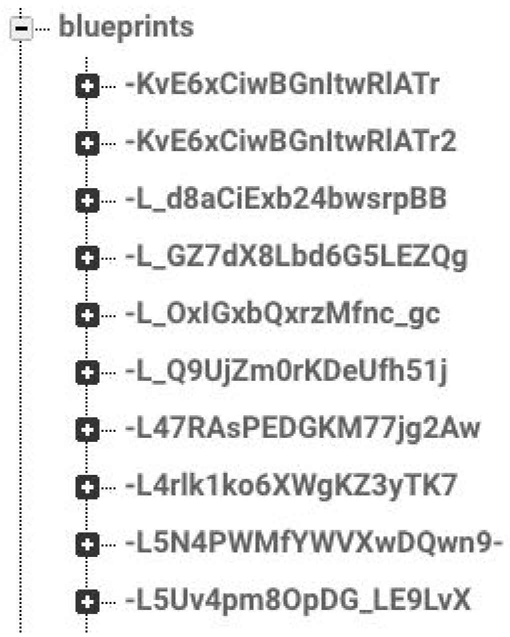
Read-only & non-realtime mode activated in the data viewer to improve browser performance
Select a key with fewer records to edit or view in realtime

```
tigerawaredev
  blueprints
  compliance
  consoles
  data
  loadCompliance
  miscellaneous
  settings
  taking
  users
```

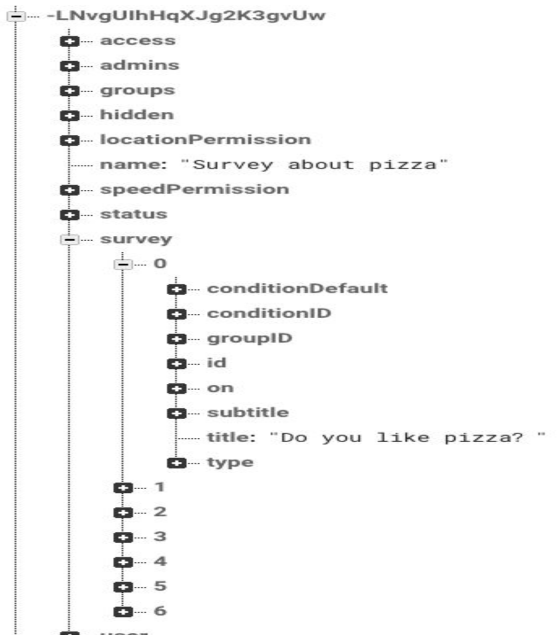
Figure 34 TigerAware System Data Storage

Blueprints

Blueprints is a collection of surveys blueprints, shown in Figure 35. Each blueprint has a unique key generated by Firebase automatically. The inner structure of each blueprint is shown in Figure 36. Each blueprint has a property called name, which is the title of the survey. The field “survey” contains all questions of that survey. Within each survey, all questions are ordered from zero to the end.



```
blueprints
    -KvE6xCiwBGnltwRlATr
    -KvE6xCiwBGnltwRlATr2
    -L_d8aCiExb24bwsrpBB
    -L_GZ7dX8Lbd6G5LEZQg
    -L_OxlGxbQxrzMfnc_gc
    -L_Q9UjZm0rKDeUfh51j
    -L47RAsPEDGKM77jg2Aw
    -L4rlk1ko6XWgKZ3yTK7
    -L5N4PWMfYWVXwDQwn9-
    -L5Uv4pm8OpDG_LE9LvX
```

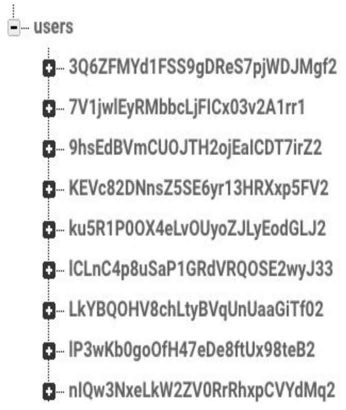


```
-LNvgUlhHqXJg2K3gvUw
    access
    admins
    groups
    hidden
    locationPermission
    name: "Survey about pizza"
    speedPermission
    status
    survey
        0
            conditionDefault
            conditionID
            groupID
            id
            on
            subtitle
            title: "Do you like pizza? "
            type
        1
        2
        3
        4
        5
        6
```

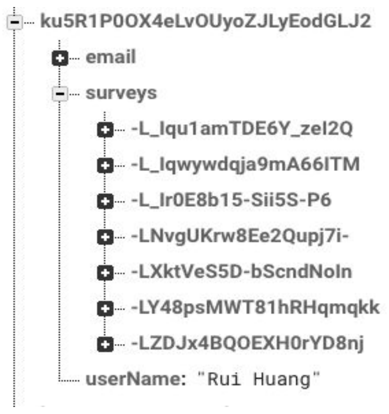
Figure 35 Blueprint List *Figure 36 Blueprint Structure*

Users

Users stores all users and their related information. This related information including but not limited to username, surveys created by that user and so on. The structure of "Users" field and each record within "Users" is shown in Figure 37 and Figure 38.



```
users
    3Q6ZFMYd1FSS9gDReS7pjWDJMgf2
    7V1jwlEyRMbbcLjFICx03v2A1rr1
    9hsEdBVmCUOJTH2ojEalCDT7irZ2
    KEVc82DNnsZ5SE6yr13HRXxp5FV2
    ku5R1P0OX4eLvOUyoZJLyEodGLJ2
    lCLnC4p8uSaP1GRdVRQOSE2wyJ33
    LkYBQOHV8chLtyBVqUnUaaGiTf02
    lP3wKb0goOfH47eDe8ftUx98teB2
    nlQw3NxeLkW2ZV0RrRhxpCVYdMq2
```

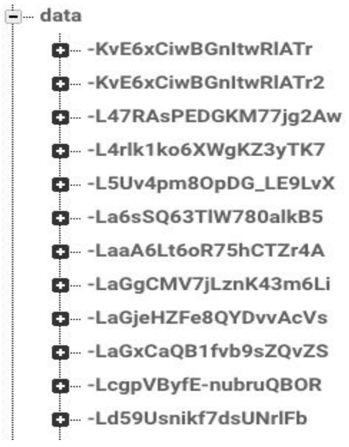


```
ku5R1P0OX4eLvOUyoZJLyEodGLJ2
    email
    surveys
        -L_lqu1amTDE6Y_zel2Q
        -L_lqwywdqja9mA66ITM
        -L_lr0E8b15-Sii5S-P6
        -LNvgUKrw8Ee2Qupj7i-
        -LXktVeS5D-bScndNoln
        -LY48psMWT81hRHqmqkk
        -LZDJx4BQOEXH0rYD8nj
    userName: "Rui Huang"
```

Figure 37 User List *Figure 38 User Structure*

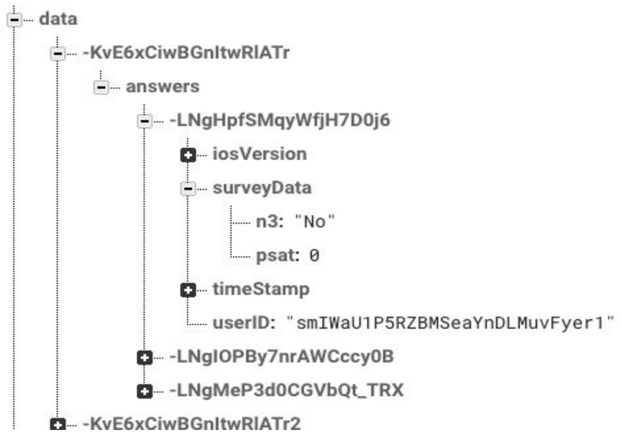
Data

Data field contains all responses collected from participants. This information includes participants' answers for each survey question. Also, some results, which are not collected from participants directly, are also stored in the Data field. For example, for image type question, not only the image itself is stored, the analysis results(e.g. emotion) generated by Google or Microsoft, is also stored. The structure of Data field and the inner structure of each Data is shown in Figure 39 and Figure 40.



```
data
    -KvE6xCiwBGnltwRlATr
    -KvE6xCiwBGnltwRlATr2
    -L47RAsPEDGKM77jg2Aw
    -L4rlk1ko6XWgKZ3yTK7
    -L5Uv4pm8OpDG_LE9LvX
    -La6sSQ63TlW780alkB5
    -LaaA6Lt6oR75hCTZr4A
    -LaGgCMV7jLznK43m6Li
    -LaGjeHZFe8QYDvvAcVs
    -LaGxCaQB1fvb9sZQvZS
    -LcgpVByfE-nubruQBOR
    -Ld59Usnikf7dsUNrlFb
```

Figure 39 Data List



```
data
    -KvE6xCiwBGnltwRlATr
        answers
            -LNgHpfSMqyWfjH7D0j6
                iosVersion
                surveyData
                    n3: "No"
                    psat: 0
                timeStamp
                userID: "smIWaU1P5RZBMSeaYnDLMuvFyer1"
            -LNglOPBy7nrAWCccy0B
            -LNgMeP3d0CGVbQt_TRX
    -KvE6xCiwBGnltwRlATr2
```

Figure 40 Data Structure

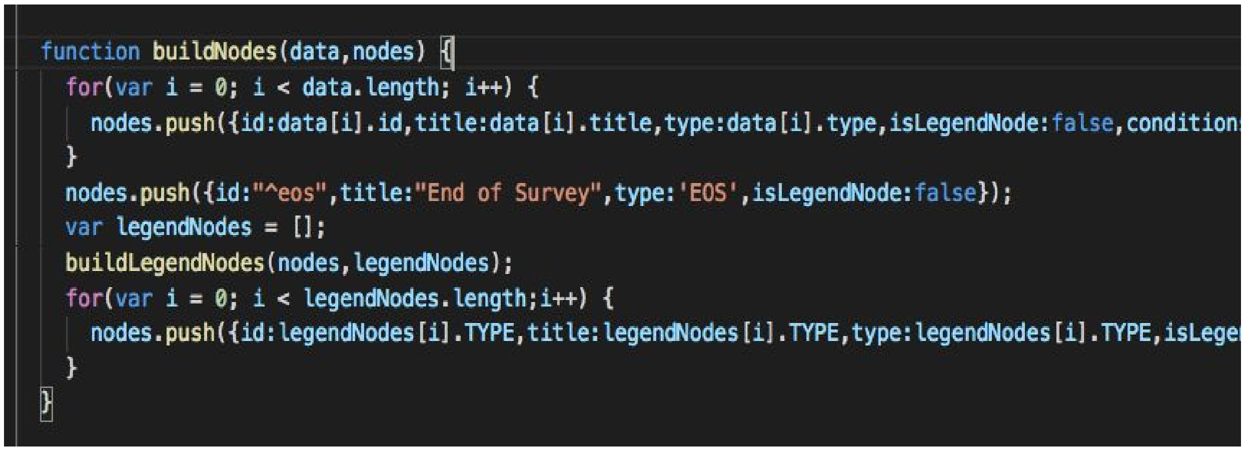
4.2 Survey Visualization Implementation

Survey visualization system is a single web page in the TigerAware dashboard. It can be further divided into two modules: visualize module and file export module. The following part will focus on the implementation of these two modules.

Visualization Module

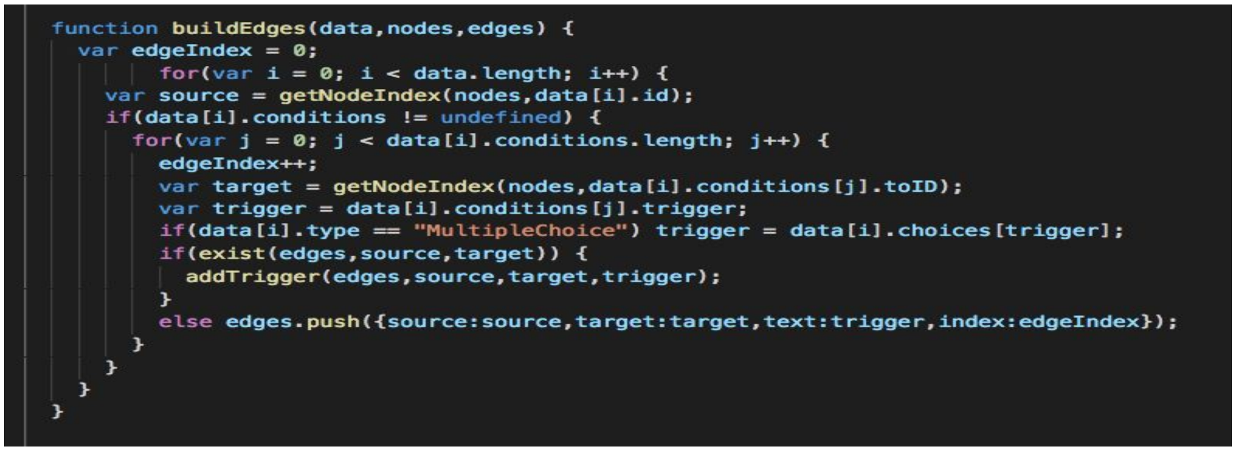
Visualization module consists of three functionality: Format Convert, visualize and planar embedding.

Format Convert function acts as an adaptor between data from Firebase realtime database and the visualization framework D3. D3 requires a specific data format: a set of nodes and a set of edges. However, the data format in database is a whole JSON object. That is the reason why format convert function is designed. The implementation of this function is shown in Figure 41 and Figure 42.



```
function buildNodes(data,nodes) {
  for(var i = 0; i < data.length; i++) {
    nodes.push({id:data[i].id,title:data[i].title,type:data[i].type,isLegendNode:false,condition
  }
  nodes.push({id:"^eos",title:"End of Survey",type:'EOS',isLegendNode:false});
  var legendNodes = [];
  buildLegendNodes(nodes,legendNodes);
  for(var i = 0; i < legendNodes.length;i++) {
    nodes.push({id:legendNodes[i].TYPE,title:legendNodes[i].TYPE,type:legendNodes[i].TYPE,isLege
  }
}
```

Figure 41 Build Node Implementation



```
function buildEdges(data,nodes,edges) {
  var edgeIndex = 0;
        for(var i = 0; i < data.length; i++) {
    var source = getNodeIndex(nodes,data[i].id);
    if(data[i].conditions != undefined) {
      for(var j = 0; j < data[i].conditions.length; j++) {
        edgeIndex++;
        var target = getNodeIndex(nodes,data[i].conditions[j].toID);
        var trigger = data[i].conditions[j].trigger;
        if(data[i].type == "MultipleChoice") trigger = data[i].choices[trigger];
        if(exist(edges,source,target)) {
          addTrigger(edges,source,target,trigger);
        }
        else edges.push({source:source,target:target,text:trigger,index:edgeIndex});
      }
    }
  }
}
```

Figure 42 Build Edge Implementation

Visualize function visualizes surveys as graphs using D3.js force-directed graph layout. Questions and branches are visualized as nodes and edges respectively. Result is shown in Figure 43.

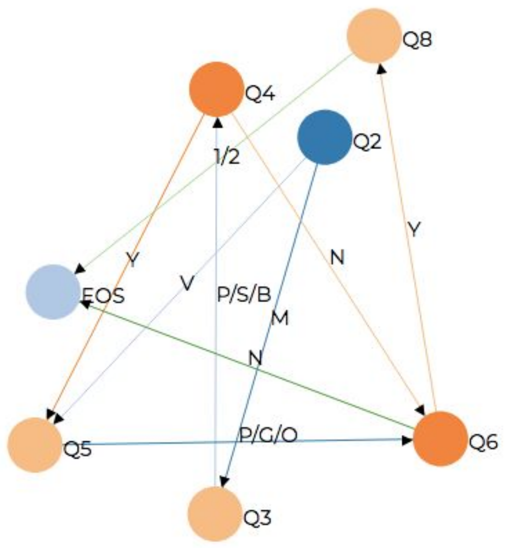


Figure 43 D3 Visualization Without Optimization

Planar embedding function is implemented to optimize D3 visualization. As Figure 43 shows, there are lots of crossing between edges, which makes it look complicated. In graph theory, planar embedding of a graph is a drawing that its edges intersect only at their endpoints so that there is no crossing between edges. In this project, Tutte’s embedding algorithm is used to find the planar embedding. Below is the formula for Tutte’s embedding algorithm. The result is shown in Figure 44.

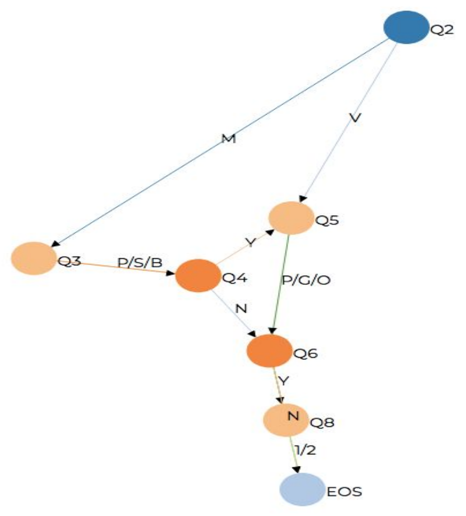


Figure 44 D3 Visualization After Optimization

File Export Module

The module can export graphs visualized by D3 to PDF files. It is designed for researchers to export surveys so that if they want to present surveys to others, they don't need to draw surveys manually.

In front-end, there is a link whose href value is dynamically assigned and points to the path of PDF file. When researchers click this link, it will send a request to back-end, then backend will convert graph to PDF file and save it in file system. Then backend sends the path to PDF file as result to front-end. After receiving returned value, front end will assign the value to the href value of the link.

In back-end, after receiving requests from front-end, RSVG,which is a free software SVG rendering library written as part of the GNOME project, is used to convert graphs to PDF files. Then back-end will save PDF file in its' file system and send the path to PDF file as result to front-end.

5.Application

This system has been used in a real-world study and the application has shown excellent capabilities in adaptability and deployment for different tasks. The first part of this section discusses the design of the pilot study. Followed by the results getting from participants' responses.

5.1 Survey Design

This pilot study survey is inspired by the paper "Social Factors Influence on Career Choice for Female Computer Science Students", which is designed by Sohail Iqbal Malik from AI Buraimi University College and Mostafa AI-Emran from University Malaysia Pahang. In that paper, a survey was designed to explore the influence of different factors on female students in choosing a career in the IT field[from paper]. The pilot study survey designed in this project consists of parts of questions in that paper and also some questions about students' perception of the College of Engineering of the University of Missouri. This pilot study aims to answer the following three questions:

- RQ1: What led students to choose their major?
- RQ2: How do students feel about Mizzou and College of Engineering diversity?
- RQ3: How do women/female students feel about gender inclusion in the COE?

This pilot study consists of 24 closed-ended questions and 4 random open-ended questions. Totally it has five parts.

The first part covers four demographic questions related to students. These questions include:

- What is your ethnicity (check all that apply)?
    - white(0), Hispanic or Latino(1), black or African-American(2), American Indian or Alaskan Native(3), Native Hawaiian or other Pacific islander(4), Asian(5), other(6), prefer not to respond(7)
- What is your age?
    - 18-19(0), 20-21(1), 22-23(2), 24+(3), prefer not to respond(4)
- What is your major?
    - Computer Science(0), Information Technology(1), Other College of Engineering major(2), Not a College of Engineering major(3), Prefer not to Respond(4)
- What is the gender you identify with most?
    - Male(0), female(1), non-binary(2), other(3), prefer not to respond(4)

The second part covers questions that are mainly related to the influence of different factors in choosing major for female students in computer science. Eleven different factors are included in this section. A five-point scale is used, from not influential at all to extremely influential. Factors considered in this part include: Parents, Teachers/Counselors, Friends, Job Opportunities, Job Prestige, Personal Interests, Personal Abilities, Financial Reasons, Flexible Hours, Make the World a Better Place, Visa/Work sponsorship.

The third part of the survey focuses on questions related to the diversity of the College of Engineering at the University of Missouri. This section includes 7 different questions. Again, a five-point scale is used. Questions include:

- In general, how diverse do you consider Mizzou to be?
    - Not diverse at all(0), Not very diverse(1), Mostly diverse(2), Very diverse(2), Prefer not to respond(4).

- In general, how diverse do you consider the College of Engineering to be?
    - Not diverse at all(0), Not very diverse(1), Mostly diverse(2), Very diverse(3), Prefer not to respond(4).

- In general, do you think Mizzou needs to put more effort into making a diverse and inclusive student body?
    - 1 (Less Effort) to 5 (More Effort)

- Do you think the College of Engineering in particular needs to put more effort into making a diverse and inclusive student body?
    - 1 (Less Effort) to 5 (More Effort)

- Do you think there is an appropriate level of gender diversity in your College of Engineering classes?
    - 1 (No) to 5 (Yes)

- Do you think there is an appropriate level of gender diversity in your non-Engineering classes?

    - 1 (No) to 5 (Yes)

- If you participate in an Engineering-based organization or club, do you think your organization has an appropriate level of gender diversity?
    - No, Not really, Somewhat, Yes, I don’t participate in an organization, Prefer not to respond.

The fourth part of the survey focused on questions related to the gender inclusion of the College of Engineering at the University of Missouri. Questions in this part including do you feel like you are at a disadvantage in your College of Engineering classes because of your gender and do you feel excluded from your peers in the College of Engineering because of your gender.

5.2 Survey Result

This pilot study was performed in the students of class CS2830 and CS3330. Totally 107 responses have been collected through mobile app. This section describes the results of this study and probes Research Questions 1,2 and 3.

Demographic Information

For the question about ethnicity, most respondents (72.9 %) are white people, while only two participants are Black or African-American, shown in Figure 45. For the question about age, 54.2% of participants are 20-21 years old and there is one participant prefers not to respond. For the question about major, 62.6% of respondents’ major is computer science, while 26 participants (24.3 %) are Information Technology major, shown in Figure 46. Most of respondents(81.3 %) identify themselves as males, 17 (15.8 %) participants identify themselves as female and there is one student chooses the option 'Non-Binary'.

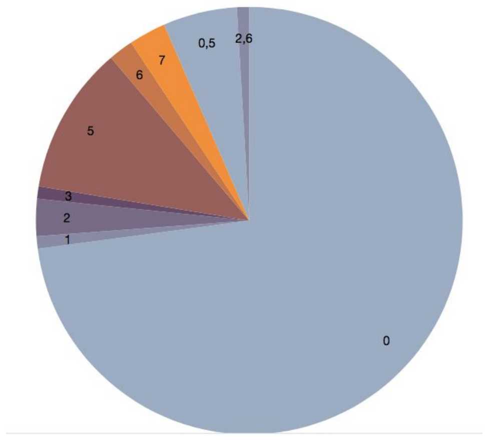


Figure 45 Major Distribution

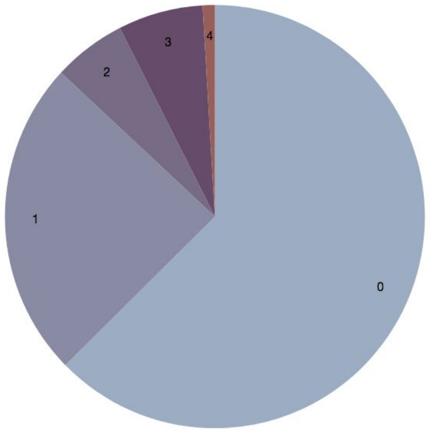


Figure 46 Ethnicity Distribution

Influences on career choices

For the factor parents, most respondents(48.6 %) think it is somewhat influential, shown in Figure 47. For the factor friends, most respondents(43.9 %) think it is somewhat influential. 52.3% of participants believe job opportunity is very influential. For the factor job prestige, 38.3% of participants think it is very influential while 35.5% think it is somewhat influential. For the factor personal Interests, 53 (49.5%) participants think it is the main influential, shown in Figure 48. For the factor personal ability, 50 (46.7%) participants think it is very influential. 36 participants think financial reason is very influential while there are also 31 respondents think it is somewhat influential. For the factor flexible hours, 25 students think it is somewhat influential but there are also 37 students who think it is not influential at all. For the factor making the world a better place, most (60) students think it is somewhat influential. For the factor visa/work sponsorship, most (84) students think it is not influential at all.

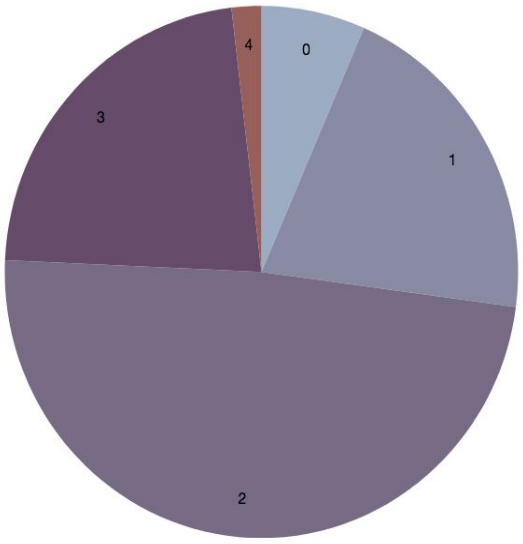


Figure 47 Parent Influence

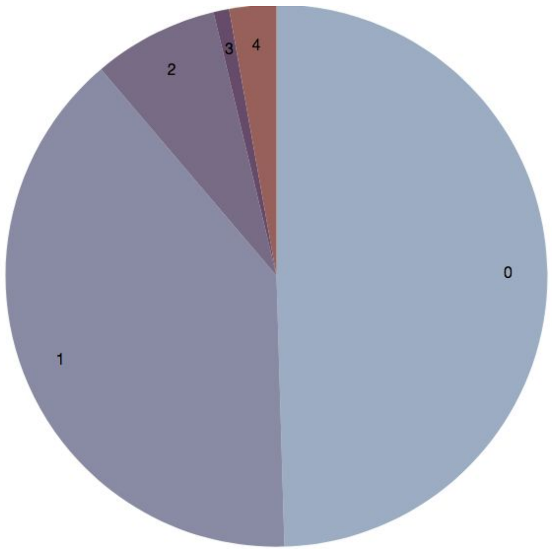


Figure 48 Personal Interests Influence

Diversity of COE

Most (51.4%) participants think Mizzou is mostly diverse and there are 28 participants think MIZZOU is very diverse, the result is shown in Figure 49. About 44 participants think COE is mostly diverse, 28 participants think COE is not very diverse and 26 participants think COE is very diverse, shown in Figure 50. 42 participants think Mizzou needs to make amount of work to make it more diverse. Most (35) participants think COE needs to make more work to make it more diverse. For the question of whether there is gender diversity in Engineering classes, most students (43) think it is slightly diverse, shown in Figure 51. For the question of whether there is gender diversity in non-Engineering classes, most (34) students think it is very diverse, shown in Figure 52. For the question of whether they think the organization has an appropriate level of gender diversity if they join an Engineering-based organization or club, most (49) students' answer is "I don't participant in an organization", while 27 students think there is an appropriate level of gender diversity.

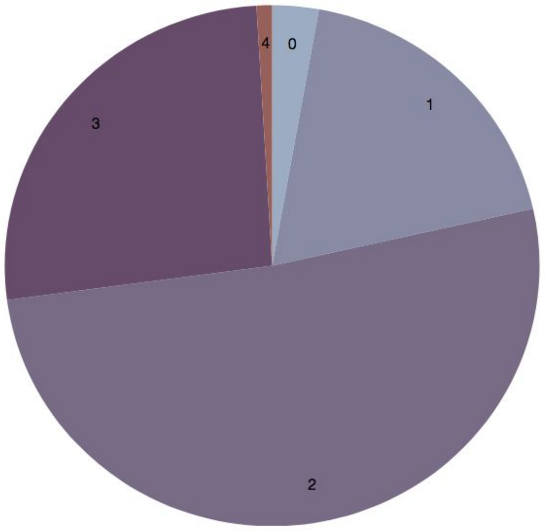


Figure 49 Mizzou Diversity

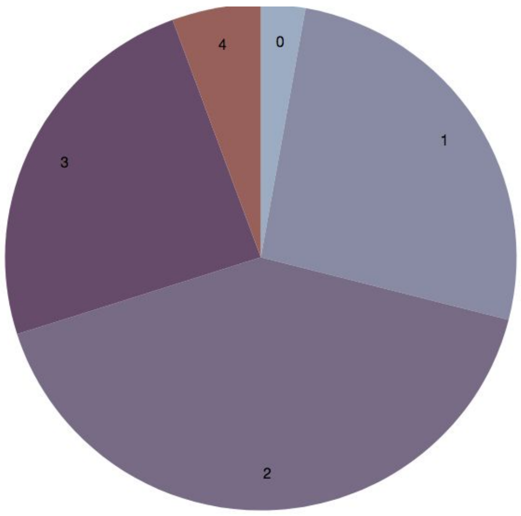


Figure 50 COE Diversity

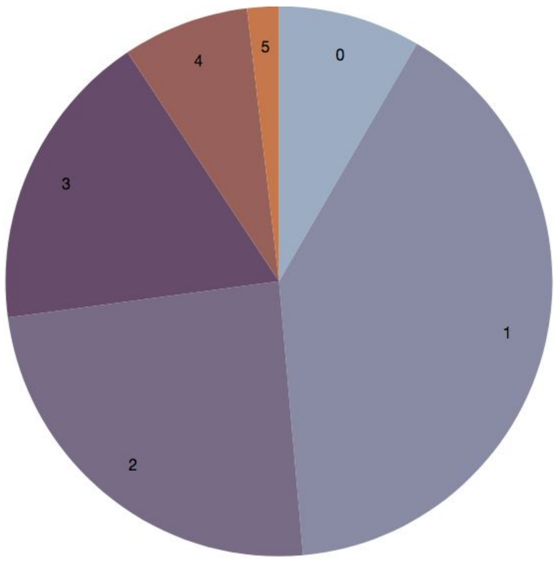


Figure 51 Engineering Class Diversity

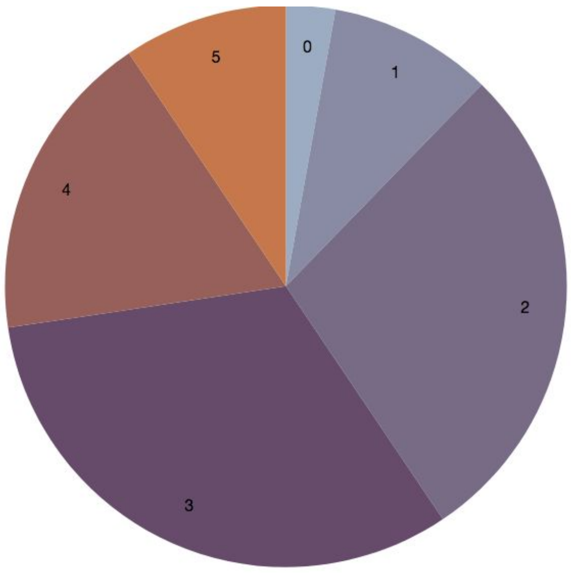


Figure 52 Non-Engineering Class Diversity

Gender inclusion of COE

In this part, only 17 responses have been collected. For the question of whether they are at a disadvantage in the College of Engineering classes because of their gender. Seven of them think it never happens, while there is one student who thinks it often happens. For the question of whether they feel excluded from their peers in the College of Engineering because of their gender. Five of them think it never happens, while one student thinks it always happens.

6.Conclusion and Future Work

6.1 Conclusion

In this master project, a survey visualization and data analysis system has been successfully added to the existing TigerAware system. Before this project, the TigerAware system lacks the ability to visualize surveys and analysis participants' responses. This project provides functionality for TigerAware to visualize surveys and analysis responses using advanced deep learning algorithms.

6.2 Future Work

In the presentation component, the process of reading all participants from Firebase is a time-consuming process, especially if the number of participants is large. It will be much quicker if this task can be moved to analysis engine. In the future, this functionality can be implemented at analysis engine and expose an API for presentation component to use. Also, only pie chart is supported to display distribution, in the future, more form should be supported like histogram.

In the analysis engine, TigerAware platform currently only supports basic statistics method. In the future, deep neural network models could be implemented for TigerAware so that it can provide abilities to process images and natural language.

7.References

[1] Jonathan Rogers, Dylan Simmons, Milesh Shah, Connor Rowland, and Yi Shang, “Deep Learning at Your Fingertips”, 2019 16th IEEE Annual Consumer Communications & Networking Conference (CCNC).

[2] JayanthKanugo, “TigerAware Dashboard: An Improved Survey Generation and Response VisualizaDashboard”, in University of Missouri, Department of CS in Master’s Thesis.

[3] Jonathan Rogers, Dylan Simmons, Milesh Shah, Connor Rowland, Yi Shang, “Deep Learning at Your Fingertips”, Consumer Communications and Networking Conference, CCNC IEEE.

[4] An Introduction to Box Skill: https://developer.box.com/docs/box-skills.

[5] An Introduction to Box Skill Application: https://community.box.com/t5/Using-Box-Custom-Skills/Introducing-Box-Skills-for-Admins/ta-p/64844.

[6] Introduction to Google Forms: https://en.wikipedia.org/wiki/Google_Forms

[7] Introduction to D3: https://d3js.org/

[8] Introduction to planar graph: https://en.wikipedia.org/wiki/Planar_graph

[9] Introduction Angular Material: https://www.tutorialspoint.com/angular_material/

[10] Introduction to Firebase: https://firebase.google.cn/docs/database/unity/retrieve-data?hl=en-au&authuser=3

[11] Introduction to stop words: https://en.wikipedia.org/wiki/Stop_words