**Soothe ‘er**

A PROJECT REPORT

submitted by

**Ankam Mani Kanta (Registration Number: 13MSE1061)**

*In partial fulfilment for the award*

of

**M. Tech. Software Engineering**

**(5 year Integrated Programme)**

**School of Computing Science and Engineering**



**October 2017**

****

**School of Computing Science and Engineering**

**DECLARATION**

I hereby declare that the project entitled **“Soothe ‘er”** submitted by me to the School of Computing Science and Engineering, VIT University, Chennai Campus, Chennai 600127 in partial fulfilment of the requirements for the award of the degree of **Master of Technology -Software Engineering (5 year Integrated Programme)** is a record of bonafide work carried out by me under the supervision of **Prof. Pradeep.K.V, Asst. Professor.** I further declare that the work reported in this project has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma of this institute or of any other institute or university.

Signature

**Ankam Mani Kanta (Registration Number: 13MSE1061)**



**School of Computing Science and Engineering**

**CERTIFICATE**

The project report entitled “**Soothe ‘er**” is prepared and submitted by **Ankam Mani Kanta (Register No: 13MSE1061).** Ithas been found satisfactory in terms of scope, quality and presentation as partial fulfilment of the requirements for the award of the degree of **Master of Technology – Software Engineering (5 year Integrated Programme)** in VIT University, Chennai Campus, Chennai, India.

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_Signature\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

**(Name of the Supervisor/Guide)**

**Examined by**:

**Examiner I Examiner II**

**ACKNOWLEDGEMENT**

I am obliged to give our appreciation to a number of people without whom I could not have completed this project successfully. I would like to place our deep sense of gratitude and thanks to our internal guide **Prof Pradeep K V, School of Computing Science and Engineering (SCSE)**, VIT University whose esteemed support and immense guidance encouraged us to complete the project successfully.

I would like to thank our Program Manager, **Dr Vishwanathan V, School of Computing Science and Engineering (SCSE)**, VIT University, for his valuable support and encouragement to take up and complete this thesis. I also like to extend our deepest gratitude to our panel members and other Professors of our Department for their advices and comments in each review, which motivated me to explore more and impart accuracy in the project.

Special thanks to our Dean **Dr Vaidehi Vijayakumar, School of Computer Science and Engineering (SCSE)**, VIT University, for spending her valuable time and efforts in sharing her knowledge and for helping us. I thank our management of VIT University, Chennai campus for permitting us to use the library resources. I also thank all the faculty members for giving us the courage and the strength that we needed to complete our goal. This acknowledgement would be incomplete without expressing the whole hearted thanks to our family and friends who motivated me during the course of our work.

**CONTENTS**

|  |  |  |
| --- | --- | --- |
| **Chapter** | **Title** | **Page** |
|  |  |  |
|  | Title Page | i |
|  | Declaration | ii |
|  | Certificate | iii |
|  | Acknowledgement | iv |
|  | Table of contents | v |
|  | List of Tables | vi |
|  | List of Figures | vii |
|  | List of Abbreviation | viii |
|  | Abstract | ix |
|  |  |  |
| 1 | Introduction | 1 |
| 2 | Related Work | 2 |
|  | 2.1 Literature Study | 2 |
|  | 2.2 Proposed System | 3 |
|  | 2.3 Problem Statement | 3 |
| 3 | Design and Implementation | 4 |
|  | 3.1 Architecture Diagram | 4 |
|  | 3.2 Detailed Design | 5 |
|  | 3.3 Implementation | 9 |
| 4 | Results and Result Analysis | 18 |
| 5 | Conclusion and Future Work | 23 |
|  | References | 24 |
|  | Appendix – I | 25 |

**LIST OF TABLES**

|  |  |
| --- | --- |
| **Title** | **Page** |
| Table 3.1 : Compatibility and feature support | 4 |

**LIST OF FIGURES**

|  |  |
| --- | --- |
| Title | Page |
| Fig 3.1: Architecture Diagram | 4 |
| Fig 3.2: Use Case Diagram | 6 |
| Fig 3.3: Class Diagram | 7 |
| Fig 3.4: Sequence Diagram | 8 |
| Fig 3.5: Data Flow Diagram | 8 |
| Fig 3.6: Creating a Project for API | 11 |
| Fig 3.7: Naming Project | 11 |
| Fig 3.8: Opening Project | 11 |
| Fig 3.9: API Manager Dashboard | 12 |
| Fig 3.10: Enabling API | 12 |
| Fig 3.11: Finding Google Assistant | 13 |
| Fig 3.12: Enabling API 2 | 13 |
| Fig 3.13: Creating Credentials | 14 |
| Fig 3.14: Providing Details | 14 |
| Fig 3.15: OAuth Client ID | 15 |
| Fig 3.16: Finalizing the details | 15 |
| Fig 3.17: Downloading the Credentials | 16 |

**LIST OF ABBREVIATIONS**

|  |  |
| --- | --- |
| **Abbreviation** | **Expansion** |
| PC | Personal Computer |
| GUI | Graphical User Interface |
| HDMI | High Definition Multimedia Interface |
| SD | Secure Digital |
| USB | Universal Serial Bus |
| GPIO | General Purpose Input/output |
| SDK | Software Development Kit |
| API | Application Program Interface |
| RPC | Remote Procedural Call |
| Wi-Fi | Wireless Fidelity |
| SSH | Secure Shell |
| Pi | Raspberry Pi |
| OAuth | Open Authentication |
| ID | Identity Document |
| JSON | JavaScript Object Notation |
| URL | Uniform Resource Locator |
| PSA | Personal Smart Assistant |

**ABSTRACT**

Now a days we are so connected to our smart phones that we need our mobile to answer even a smallest task like finding the weather or converting 10 US dollars to its equivalent in Indian rupees.

Consider a tired person came home and desperately need an answer to a quick task like ‘pounds to inches’ or ‘what’s up with his favourite football team’ or a soothing music to relax, but he/she doesn’t want to interact with the mobile or there is no battery left in the device. Soothe ‘er helps such people to automate things with the help of a voice command or a clap kind of gesture.

Soothe ‘er takes voice commands from the user and provide with the results in voice format. It acts as a personal assistant embedded in an independent device (Raspberry Pi 3) which is always available.

**CHAPTER 1**

**INTRODUCTION**

The Personal Assistant is nothing but an implementation of assistance virtually on the user`s PC (Personal Computer). The software can be used with voice, keyboard input and also using internet for remote access. There are some predefined commands in the system and new commands may be added as and when necessary. System notifies the user about new emails, update changes in social networking sites, news reports, etc. User can set alarms, reminders, appointments etc. Tasks like shutdown, lock system, sleep, file reading and writing can also be executed by the system. There are several modules for each task like time, alarm, search etc. The GUI calls these predefined modules for the given commands. System initiates at start up and voice input to the system can be suspended with a command. All the tasks that require the use of keyboard and/or mouse can be handled by this system.

But in our case Soother ‘er is not capable of handling of any device oriented tasks like shutting down the system, setting up alarms and reminders etc. but it can handle anything which it can find on the internet. Because unlike other proprietary personal assistants which are integrated into the device software Soothe ‘er is merely an application built to sit on top of an operating system.

Hardware Components:

* A [Raspberry Pi 3](http://www.androidauthority.com/raspberry-pi-3-review-678323/)(along with all the normal bits and pieces like a micro SD card, a mouse, keyboard etc. for development purpose).
* A speaker with a 3.5 mm connector or a HDMI Monitor with speakers installed.
* A USB microphone.
* A simple switch and cables that can be connected to the GPIO pins of the Raspberry Pi.

**CHAPTER 2**

**RELATED WORK**

**2.1 Literature Survey:**

Scope:

Anyone who can speak and hear can use this software. Speech commands are given in a specified format or in normal sentence. The software can be given commands by a remote access portal, most commonly a website. It can alert user when specific event occurs making it intelligent.

Mathematical Representation of the system:

Input Set: The personal assistant takes input via speech, text or textual input through remote signal. So the set of inputs will be, I1 = {1 word command, fixed sentence command, random sentences} I2 = {text, speech, remote command} Thus, I = I1 U I2.

Input I = {all sentences in English via speech, all sentences in English via text, remote command}.

Output Set: The output for the specified inputs above will be response determined by the system according to the input given and the database containing all the necessary inputs and their respective outputs. O1 = {speech, display, text} O2 = {GUI, application response} Thus, O = O1 U O2.

Output O = {Response for corresponding input via speech, Response for corresponding input via GUI, application response} There is a one to one relation between input and output as there is only one output for a particular input.

**Operations:**

Following are the operations performed on the input in the system: 1. Recognize (): This operation basically gets the input from the user. For speech input, it converts the speech into text and save it. For text, the input is saved directly while in case of remote signal, the signal is directly converted to the command. 2. Extract (): This operation analyses the input string saved and system gets an idea about which command is expected to be executed for the given input. 3. Search (): This operation searches the local database for the response of the command exacted by the previous operation. 4. Response (): This operation gives the output that we see on the screen or via speech for the given command.

**2.2 Proposed System:**

In this project the main aim is to avoid smartphone to get simple tasks done while providing most of the functionality as that of a mobile phone’s version of a personal assistant. This provides a convenient way to interact with the personal assistant while moving around the environment.

**2.3 Problem Statement:**

In order to recommend assistances that are beneficial for a certain user to achieve his/her goals; the user's interests should be modelled. Furthermore, his/her current behaviour, which represents his/her current status, should also be modelled. When applied to the domain of advertisements, the recommended assistances are in the form of advertising messages. These advertising messages raise the user’s awareness about the various choices that are available to him/her, and can therefore, influence the user’s decisions regarding these available choices. For these advertising messages to be helpful for the user to achieve his/her goal, they need to be relevant to the user’s behaviours and interests.

The work here is aimed at:

1) Proposing a generic personal smart assistant that is able to model the user’s goals, interests and behaviours in an open environment based on the work proposed in, and be able to proactively assist the user in achieving his/her goal.

2) Enabling the personal assistant to match the user model with the contents of the advertising messages that are represented as smart multimedia agents, based on the work proposed in.

**CHAPTER 3**

**DESIGN AND IMPLEMENTATION**

**3.1 Architecture:**

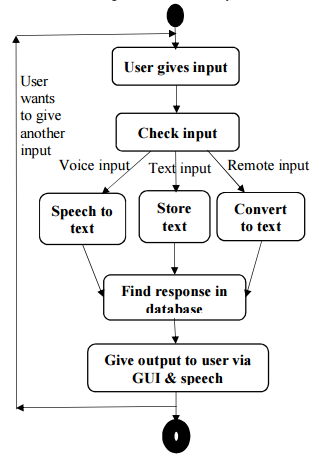


Figure 3.1

**3.2 Detailed Design:**

Modules:

I) Setting up Raspberry Pi and installing dependencies:

Before implementing the actual system Google’s assistant sdk require some pre requisites to fulfil, like setting up audio drivers to listen and speak, python and python virtual environment and it’s grpc api. All these needs to be taken care and installed at right place so that the sdk can make use of this entire environment.

II) Getting around with the Google assistant sdk:

Google assistant come with the minimal features and one can go through the documentation and add their own features accordingly.

Compatibility and feature support:

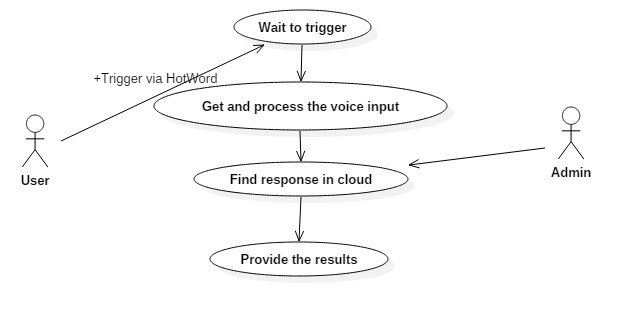
The following table summarizes the platform compatibility requirements and the supported features for the Google Assistant library for Python and the Google Assistant gRPC API:

|  |  |  |
| --- | --- | --- |
|  | Library | gRPC API |
| Supported architectures | linux-armv7l and linux-x86\_64 | [gRPC platforms](http://www.grpc.io/about/#osp) |
| Supported languages | Python | gRPC languages |
| Hands-free activation (*Ok Google*) | Yes | No |
| Audio capture and playback | Built in | Reference code is provided |
| Conversation state management | Built in | Reference code is provided |
| Timers and alarms | Yes | No |

Table 3.1

UML Diagrams:

**Use Case Diagram:**

Figure 3.2

**Description:**

**Wait to trigger:** The system wait for a hot word trigger to get started. User should provide the trigger.

**Get and process the input:** User provides the output in voice format which should be processed for correct assumption and convert to text to find the results in the cloud.

**Find response in cloud:** The processed input from above use case is provided to the cloud to find most relevant results that is helpful to the user.

**Provide the results:** The results from cloud will be in text format which should be conveyed to the user based on his required format like voice or text.

**Class Diagram:**

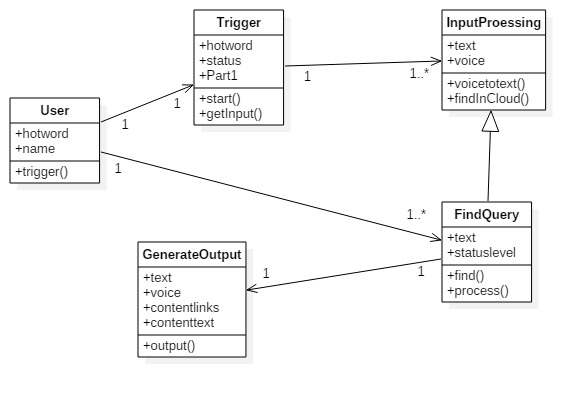
****

Figure 3.3

**Sequence Diagram:**

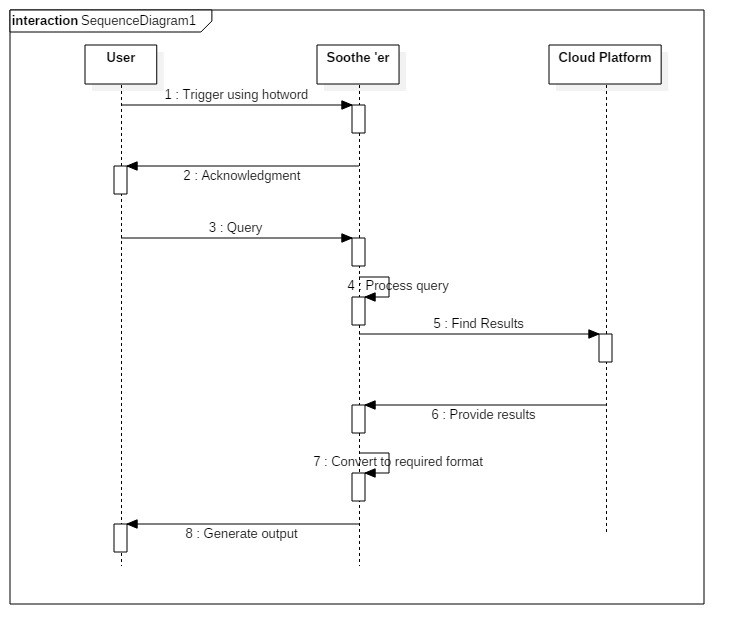
****

Figure 3.4

**Data Flow Diagram:**

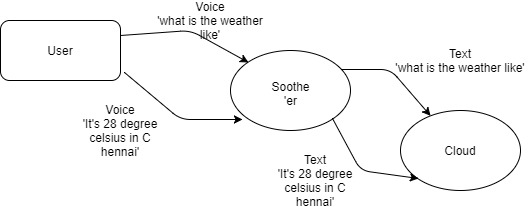
****

Figure 3.5

**3.3 Implementation:**

Step 1: Setting up Raspberry Pi

* Connect the microphone and speaker to the Raspberry Pi.
* Insert the SD card into the Raspberry Pi (with NOOBS or Raspbian with Desktop already loaded).
* Connect a USB keyboard, USB mouse and HDMI monitor to your Raspberry Pi. If you don't have these, you can always connect to the remotely.
* Plug in an Ethernet cable or connect to a Wi-Fi network.

After you configure access, you can connect to the Raspberry Pi via SSH (optional).

Step 2: Configuring audio system

* Verify that recording and playback work.
  + Play a test sound (this will be a person speaking). Press Ctrl + C when done. If you don't hear anything when you run this, check your speaker connection.

**speaker-test -t wav**

* + Record a short audio clip. If you get an error, go to step 2.

**arecord --format=S16\_LE --duration=5 --rate=16000 --file-type=raw out.raw**

* + Check the recording by replaying it.

**aplay --format=S16\_LE --rate=16000 out.raw**

* + Adjust the playback and recording volume.

**alsamixer**

* Find your recording and playback devices.
  + Locate your USB microphone in the list of capture hardware devices. Write down the card number and device number.

**arecord -l**

* + Locate your speaker in the list of playback hardware devices. Write down the card number and device number. Note that the 3.5mm-jack is typically labelled ‘analog’**.**

**aplay -l**

* Create a new file named .asoundrc in the home directory (**/home/pi/**). Make sure it has the right slave definitions for microphone and speaker; use the configuration below but replace **<card number>** and **<device number>** with the numbers you wrote down in the previous step. Do this for both **pcm.mic** and **pcm.speaker**.

pcm.!default {

type asym

capture.pcm "mic"

playback.pcm "speaker"

}

pcm.mic {

type plug

slave {

pcm "hw:<card number>,<device number>"

}

}

pcm.speaker {

type plug

slave {

pcm "hw:<card number>,<device number>"

}

}

* If you have both an HDMI monitor and a 3.5mm jack speaker connected, you can play audio out of either one. Run the following command:

**sudo raspi-config**

Go to Advanced options > Audio and select the desired output device.

* Repeat Step 1 to verify that recording and playback work. If it's still not working, check that the microphone and speaker are properly connected. If this is not the issue, then try a different microphone or speaker.

Step 3: Setting up a Google developer project.

A Google Developer Project gives your device access to the Google Assistant API.

Open Browser in Pi and go to: <https://console.cloud.google.com/cloud-resource-manager>.

* Click CREATE PROJECT for creating new project.

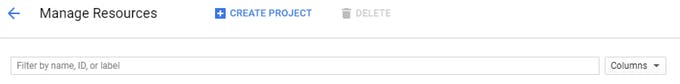


Figure 3.6

* Name your project.

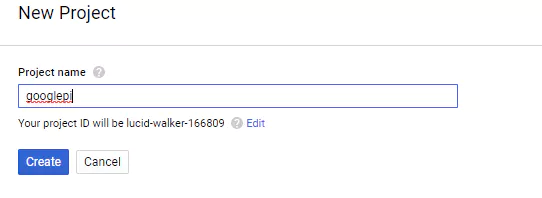


Figure 3.7

* Open your project by clicking your project name.

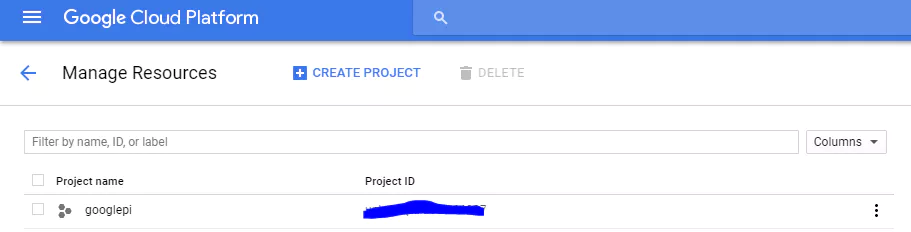


Figure 3.8

* Open API Manager Dashboard,

API Manager => Dashboard.

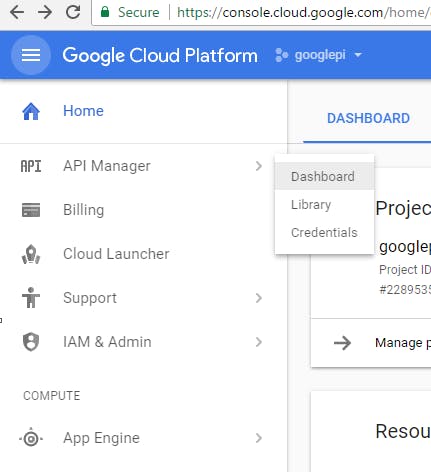


Figure 3.9

Click ENABLE API.

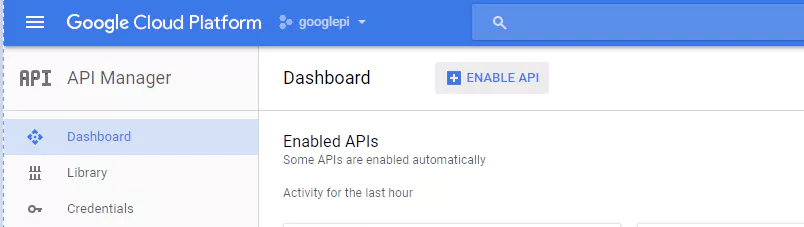


Figure 3.10

* Open Library and Search Google Assistant.

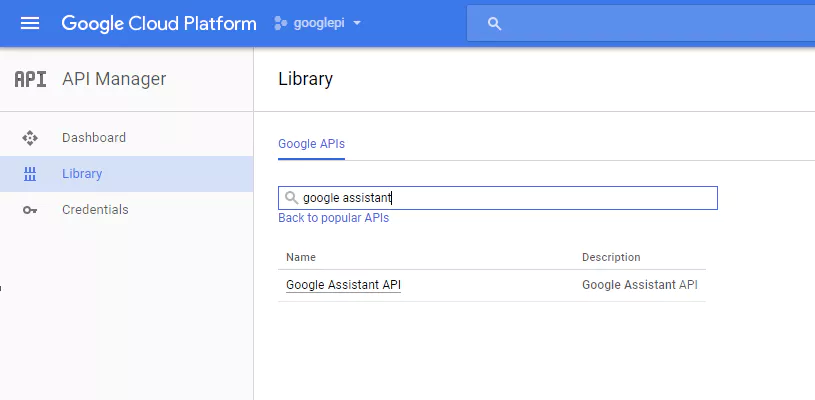


Figure 3.11

* Click the Enable Button for enabling Google Assistant API.

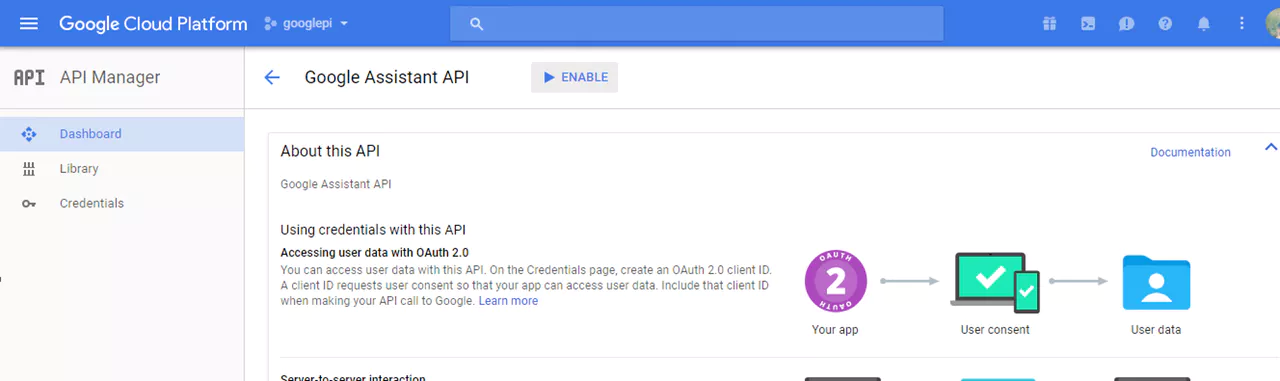


Figure 3.12

* Create credentials for it,

Open Credentials and go to OAuth Consent screen.

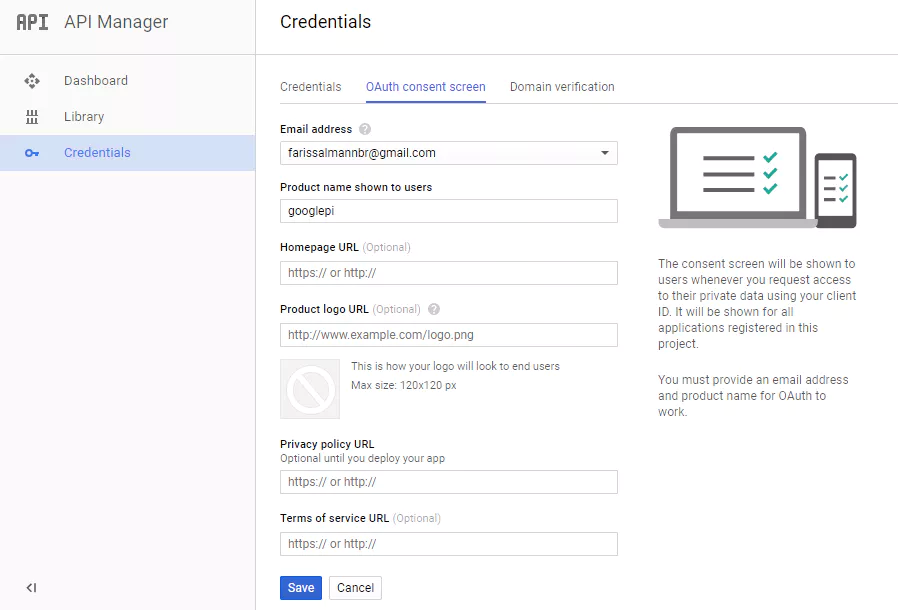


Figure 3.13

* Enter product name and leave everything as default, and save it

Go to credential.

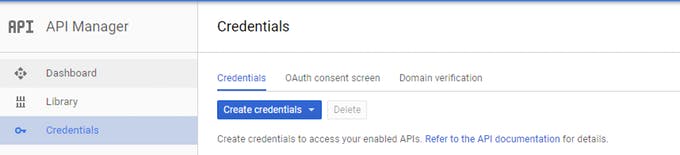


Figure 3.14

* Click OAuth Clinet ID,

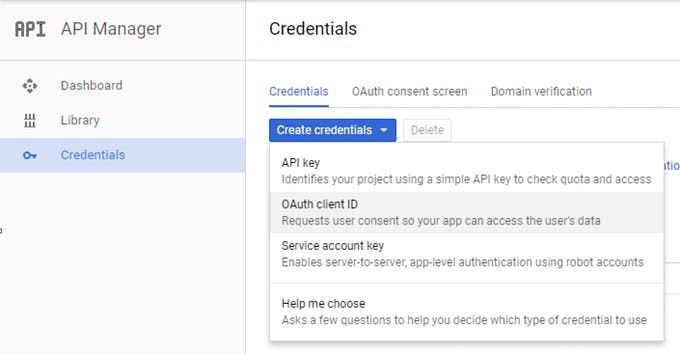


Figure 3.15

* Choose other and give it a name and hit Create button.

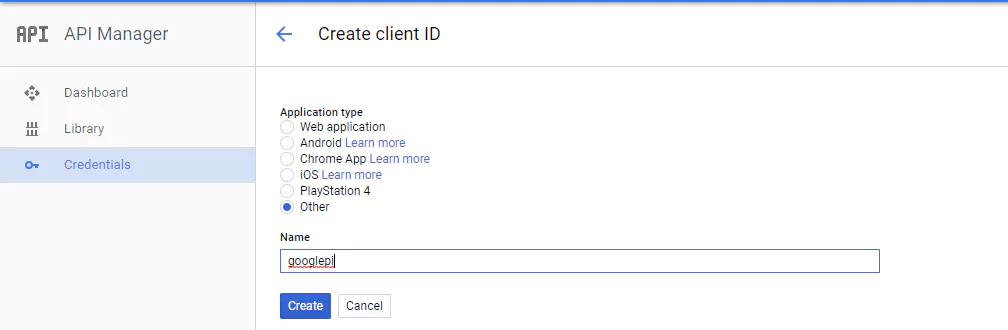


Figure 3.16

Now we get an authentication key.

* Download the Credential by clicking down arrow. You'll get an json file.

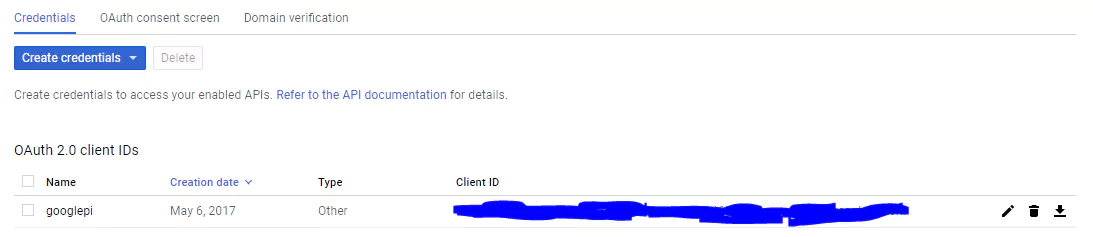


Figure 3.17

Step 4: Setting up SDK

## Configure a new Python virtual environment

Use a python virtual environment to isolate the SDK and its dependencies from the system Python packages.

sudo apt-get update

sudo apt-get install python3-dev python3-venv

python3 -m venv env

env/bin/python -m pip install --upgrade pip setuptools

source env/bin/activate

## (env) $ python -m pip install --upgrade google-assistant-library

* Running the sample

Open a browser and sign into the Google account you want to use to send queries to the Google Assistant. This can be any of your Google accounts, not just your developer account; just make sure that you set activity controls for the account

Authorize the Google Assistant SDK sample to make Google Assistant queries for the given Google Account. Reference the JSON file you copied over to the device in a previous step.

Install the authorization tool:

## (env) $ python -m pip install --upgrade google-auth-oauthlib[tool]

## Run the tool. Remove the headless flag if you are running this from a terminal on the device (not an SSH session):

## (env) $ google-oauthlib-tool --client-secrets /path/to/client\_secret\_client-id.json --scope https://www.googleapis.com/auth/assistant-sdk-prototype --save --headless

## You should see a URL displayed in the terminal:

Please go to this URL: https://...

## Copy the URL and paste it into a browser (this can be done on your development machine, or any other machine). After you approve, a code will appear in your browser, such as "4/XXXX". Copy and paste this code into the terminal:

Enter the authorization code:

If authorization was successful, you will see OAuth Credentials initialized in the terminal.

* Start the Google Assistant SDK sample.

(env) $ google-assistant-demo

Say Ok Google or Hey Google, followed by your query. The Assistant should respond.

**CHAPTER 4**

**RESULTS AND RESULT ANALYSIS**

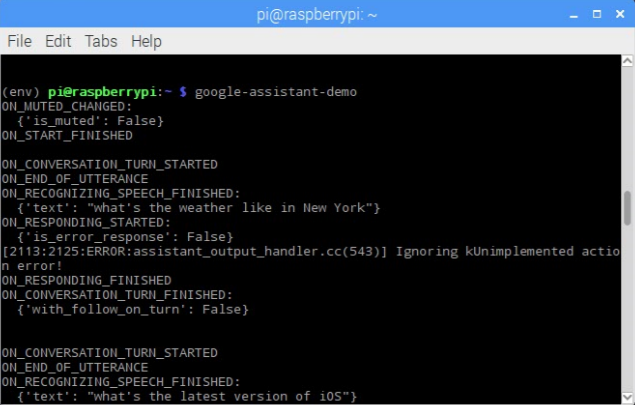
**4.1) Result and result analysis:**

Since the output is in voice format, the output by the Soothe ‘er here will be explained in the text format.

**Query 1:** What’s the weather like in New York.

**Reply:** “In New York city it’s 69 and cloudy” and the other relevant information regarding the forecast

**Screenshot for query 1:**



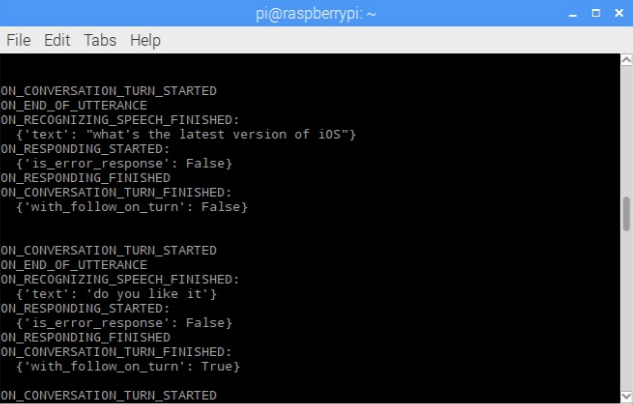
**Query 2:** What is the latest version of iOS

**Reply:** “Wikipedia – Iphone OS 3.0 was officially released on June 17,2009 for iPhone and iPod Touch” and other information regarding security patch release dates.

**Query 3:** Continuing the previous question I’ve asked “Do you like it”

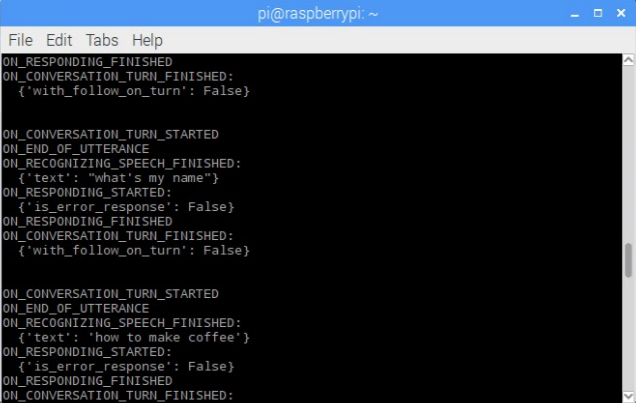
**Reply :** “Great, But I’m an android fan though”.

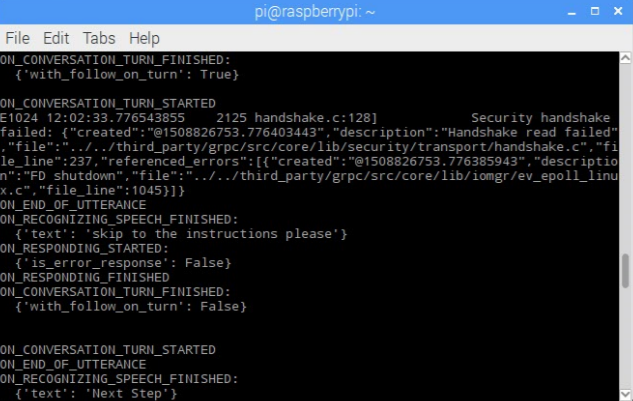
**Screenshot for query 2 and query 3:**

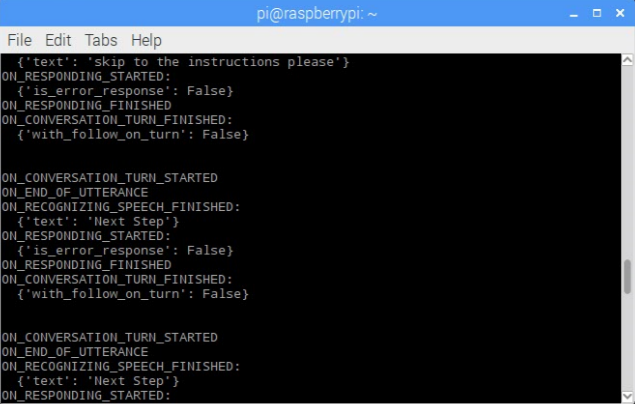


**Query 4: How to make coffee**

**Reply:** The system asked if I need the ingredients as well or just the instructions for which I replied “Skip to instructions please”. Then it suggested me to say “Next step” as we progress one by one. At time I said “After that” and “Then” but the system couldn’t relate them and didn’t continued to the next step until I said “Next step” or “Next step please”

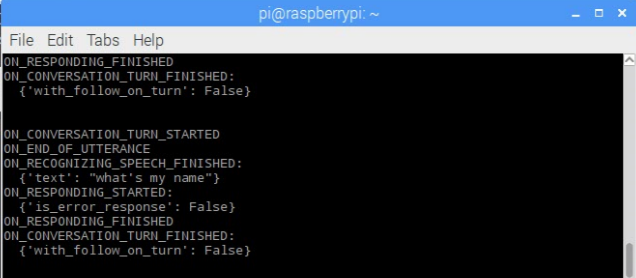






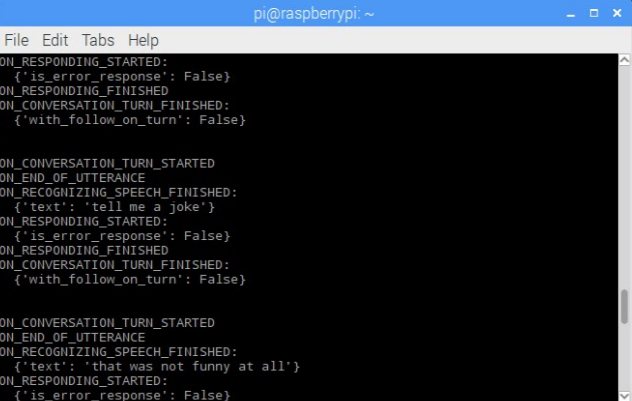
**Query 5:** “What’s my name”

**Reply:** It came up with the first name of my Google account. “Mani”



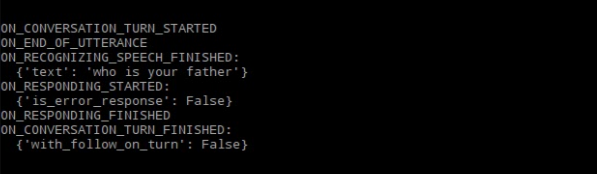
**Query 6**: “Tell me a joke”

**Reply:** “One joke,coming up! If you want to open a store, I recommend selling stoves. You’ll immediately offer a range of hot products”



**Query 7:** At conversational level I’ve asked some personal questions like“Who is your father”

**Reply:** “The Google team is my family. That’s roughly 60000 people”



**CHAPTER 5**

**CONCLUSION AND FUTURE WORK**

**5.1) Conclusion**

The purpose of this project is getting into an approach that can used to develop a personal assistant. Though a number of assistants are available in the market for smartphones, personal assistants for PC’s are not that popular in our country. With technology covering every aspect of our lives, the need of intelligent assistance has increased. However, for this assistance to be beneficial for users, it should be targeted to them based on their needs and preferences. This type of digital assistant agent is a promising approach for personalization and user centric assistance. This project proposed a digital assistant that is able to model the user, and the environment he/she is interacting with, in order to provide the relevant assistance that the user needs to achieve his/her goal efficiently. The work in this project combines a deliberative, goal-driven, approach with a reactive, event-driven approach to model the user’s behaviours. Such a combination guarantees a more accurate modelling for the user behaviours in an open environment that is constantly changing with unexpected events. The modelled user behaviour is then used to predict the user’s next actions using probabilistic approaches.

**5.2) Future Work:**

This section illustrates the future work that can be done to expand the proposed Personal Smart Assistant (PSA).

* + An additional feature of remote access eradicates the limitation of area within which the system can be accessed. Through a site the system can be given commands and this helps to develop an expert system of home automation which is an extended scope of this system.
  + Propose a collaboration component that can be used for the device to collaborate with other devices in the environment in order to better understand the environment’s context and therefore provide a better assistance to the user.

**REFERENCES**

**Book references:**

1. Phil Cohen, Adam Cheyer. “On the Future of personal assistants” – 2006.

2. Rod Williams, Harry Leverette. “Pda Playhouse: The Interactive Book of Personal Digital Assistants”

**Web references:**

[1] https://developers.google.com/assistant/sdk/

[2] https://www.hackster.io/shiva-siddharth/youtube-music-streaming-for-google-assistant- on-raspberry-pi-cba841

[3] https://www.open-electronics.org/google-assistant-sdk-now-available-for-prototyping-voice-agent-technology-on-raspberry-pi-3/

[4] https://www.cnx-software.com/2017/05/03/google-assistant-sdk-turns-your-raspberry-pi-3-into-google-home/

[5] http://www.instructables.com/id/Google-Assistant-on-a-Raspberry-Pi/