Group 28

**Intelligent System of Voice Recognition**

Guide: Prof. Saurabh Tiwari

**Group Members**

Divyansh Khatri 201801012

Nishit Jagetia 201801027

Manali Reddy 201801118

Jay Shah 201801141

Parthav Kansagra 201801145

Kishan Gadhiya 201801147

Jeet Umat 201801232

Naman Dave 201801439

Aakash Panchal 201801459

Siddharth Moradiya 201801470

Table of Contents

1. SECTION 1 ………………………………………………............................................ 5

1.1. Problem Statement

1.2. Functional and Non-Functional Requirements

1.3. Elicitation Techniques

2. SECTION 2 ……………………………………………….......................................... 6

2.1. Process model

3. SECTION 3 ............................................................................................................ 7

3.1. Concept Map

3.2. Use Case

4. SECTION 4 ............................................................................................................. 8

4.1. Design Document

4.1.1. Use Case Diagram

4.1.2. Activity Diagram

4.1.3. Sequence Diagram

4.1.4. State Diagram

4.1.5. Class Diagram

5. SECTION 5 ............................................................................................................. 12

5.1. Testing

5.1.1. Static Analysis Testing

5.1.2. Unit Testing

5.1.3 Integration Testing

6. SECTION 6 .......................................................................................................... 15

6.1. Part of the requirements not completed/implemented

6.2. Future Goals

Minutes of Meeting ................................................................................................ 15

YouTube Video

SECTION 1

1.1. Problem Statement

Our project is a desktop application that will perform the task given by the end-user through voice command. This project aims to ease and fasten the process for the end-user and therefore make life simpler.

1.2. Functional/Non-Functional Requirements

Functional Requirements:

* Open various tools and apps pre-installed in the system
* Open Chrome, Eclipse, VS Code, Firefox, WebEx, Notepad, Gmail
* Current date and time
* Create and delete files
* Change system settings such as brightness and volume
* Search queries in Wikipedia
* Perform mathematical queries
* Play music or video
* Tell jokes
* Information about weather and news
* Location on Google Maps
* Search file or folder
* Alarm
* Currency converter
* Phone management app
* Find remaining battery
* Merging of 2 or more pdfs
* Show downloads
* Take a picture of yourself using webcam
* Play game like rock paper scissors
* Get to know about the internal systems information
* Unit Converter

Non-Functional Requirements:

* Data privacy and security
* Scalability
* Portability
* Performance: response time
* Light - Weight Process

1.3. Elicitation Techniques

(1) Stakeholder Analysis:

* Accountant
* Businessman
* Student
* Developer
* Analyst
* Professor
* Software Industrialists
* People concerned with the system’s security
* investors
* Employees

(2) Use Case Approach:

In this we identified the actors, use cases and use case diagrams.

Actors are:

* Users
* AI
* Database Manager

(3) Questionnaires

(4) Risk Analysis

(5) Quality Function Deployment

(6) Brainstorming Sessions

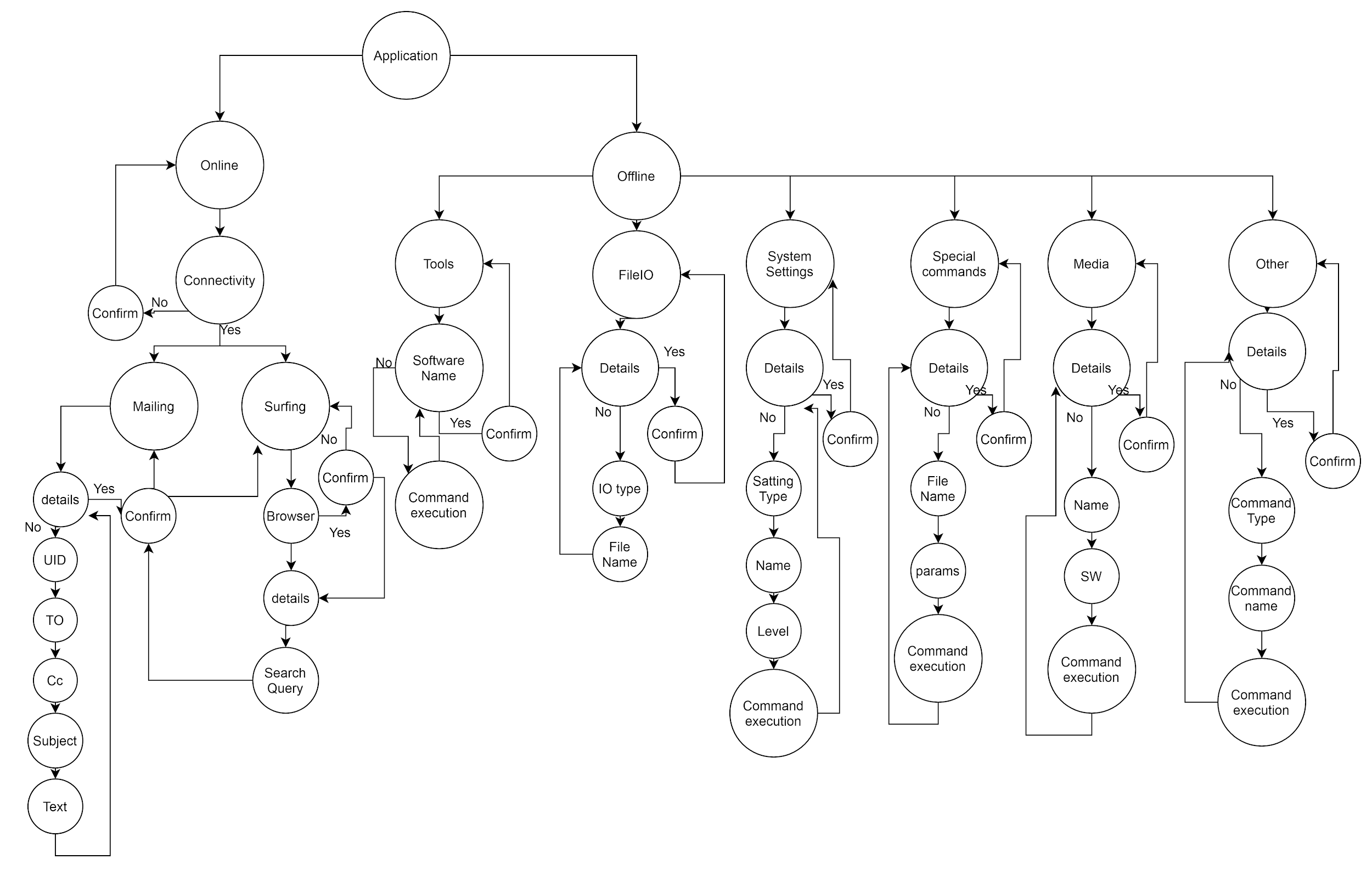
SECTION 2

2.1. Process model: Prototype model

The reason behind choosing this model is quite simple. While doing the project, we had to make changes to our requirements and our prototype according to the suggestions made by the customer (here our TA) and then redefine the requirements and the prototype and then the final prototype was selected for the project. The second reason is that it is easy to make new changes on the model and the requirements in the prototype model, as we can see that after every few months the changes in the voice assistant apps are being implemented and as new technologies arise, we need to make changes in our system according to that. In fact, we may need to add new features to our system or we may need to correct some of the errors till now so it is quite easy to do that in the prototype model when compared to other models. The reason behind choosing the incremental prototyping is that we have divided the coding of the end product into different small parts, and each of us have developed and tested it individually and then we have merged the code and tested it again, this thing has reduced the time complexity of the development process significantly and also the complexity of our code is reduced by dividing it in subparts. So, these are some reasons due to which we chose to work with incremental prototype model.

SECTION 3

3.1 Concept Map



3.2. Use cases

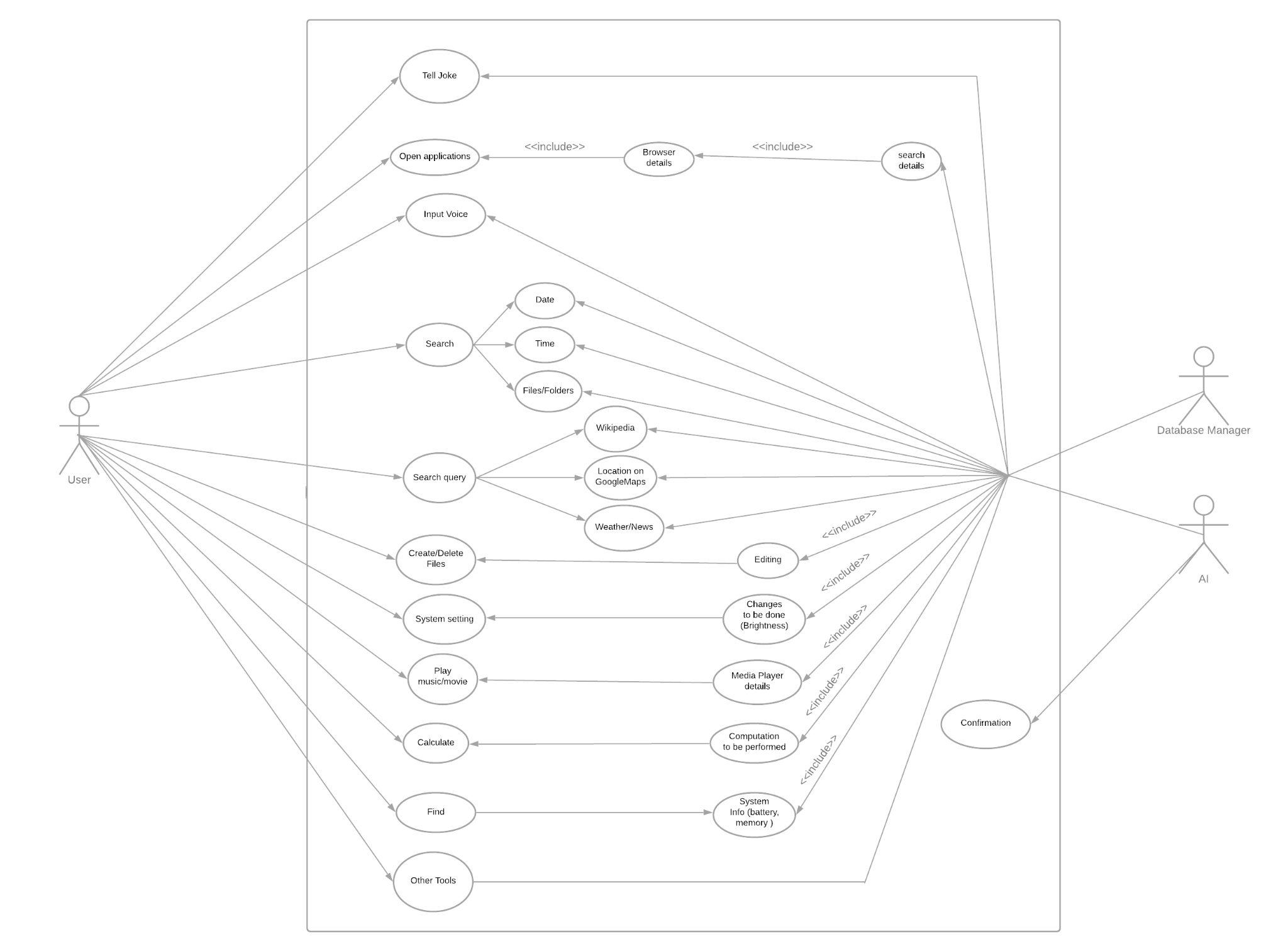
* Tell Joke
* Open applications
* Brower details
* Search details
* Input voice
* Search
* Date
* Time
* Files/Folders
* Search query
* Wikipedia
* Location on Google Maps
* Weather/News
* Create/Delete files
* Editing
* System settings
* Changes to be done (Brightness)
* Play music/movie
* Media player details
* Calculation
* Computations to performed
* Find
* System info (battery, memory)
* Other tools
* Confirmation

SECTION 4

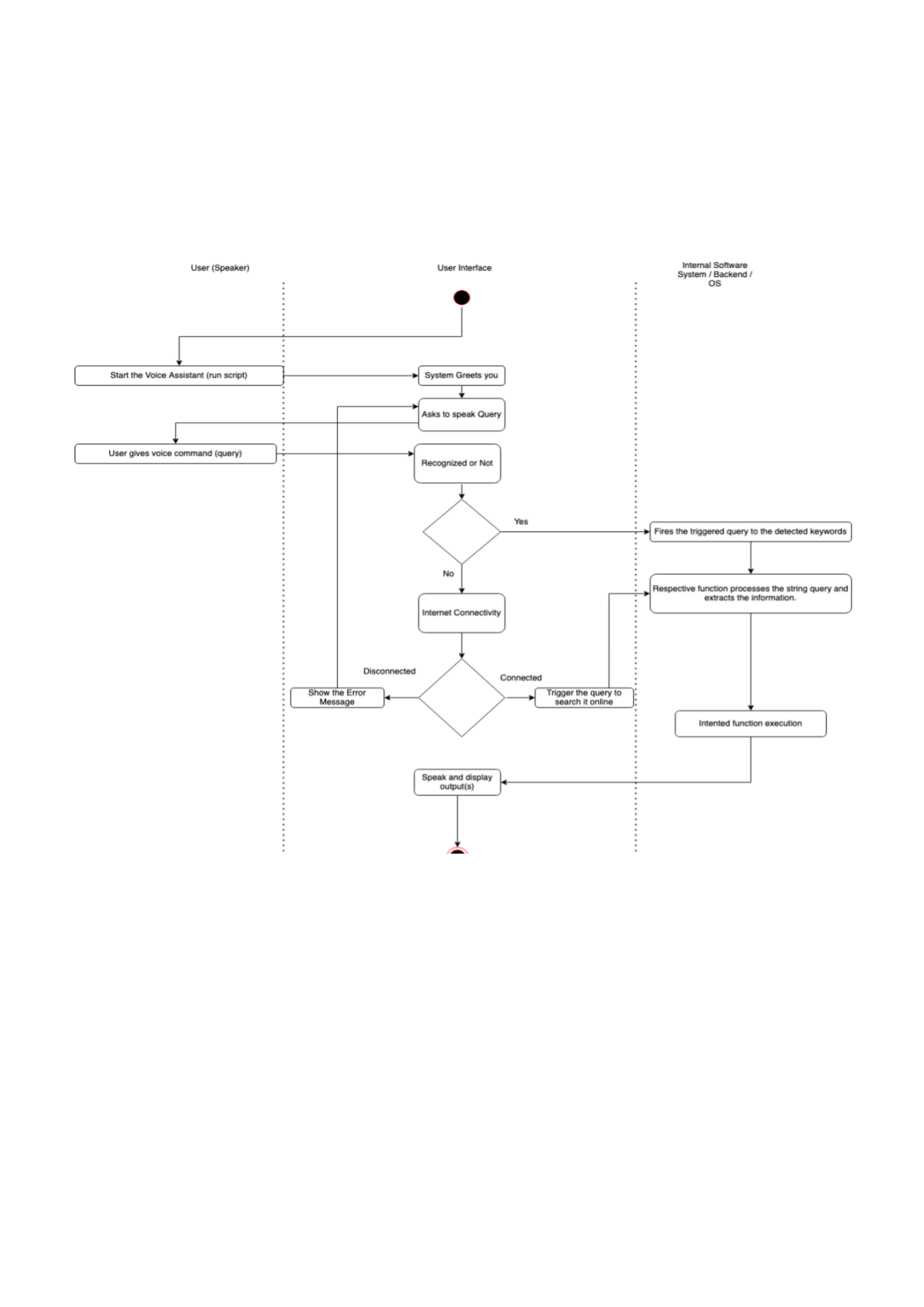
4.1. Design Document

* Use Case Diagram
* Activity Diagram
* Sequence Diagram
* State Diagram
* Class Diagram

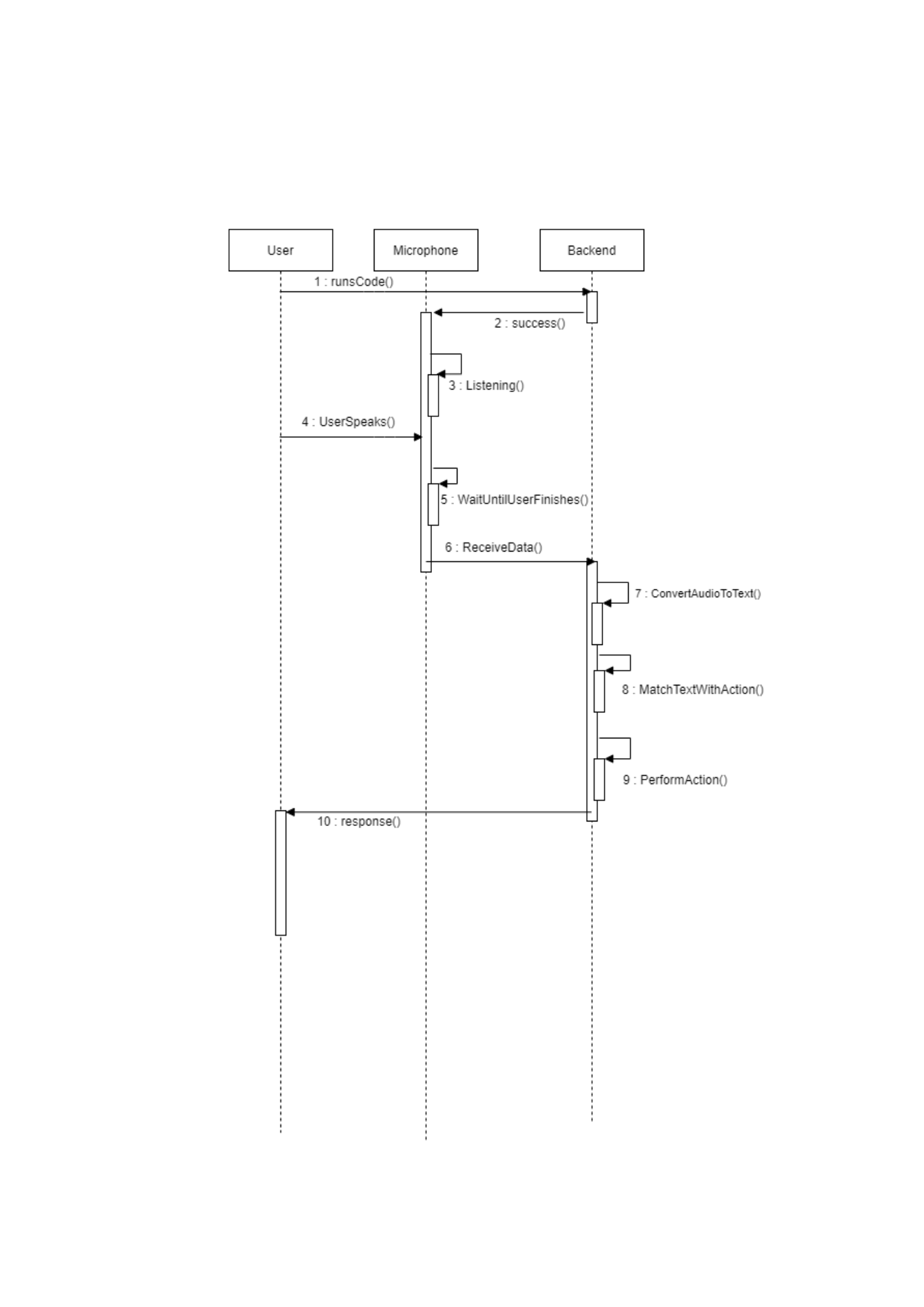
4.1.1. Use Case Diagram



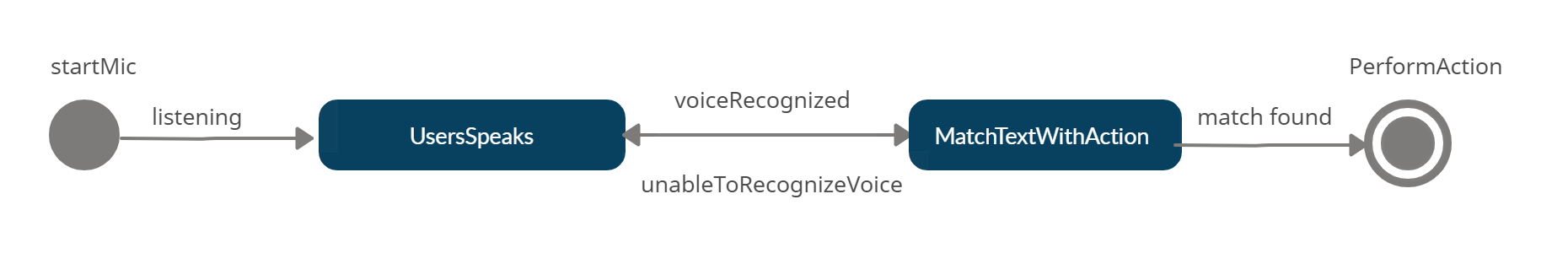
4.1.2. Activity Diagram



4.1.3. Sequence Diagram



4.1.4. State Diagram



4.1.5. Class Diagram

The reason for not implementing the class diagram is that in our project we don't have any class attributes.

SECTION 5

5.1. Testing

5.1.1. STATIC ANALYSIS TESTING

* This kind of test is done at the early stages, before we run the programs. Static analysis Powerful way to improve quality and productivity of software development.
* Static analysis can be performed manually or automatically in our project we have used python tool pylint for static analysis.

5.1.2 UNIT TESTING

* Unit testing is a type of software testing where individual units or components of a software are tested. The purpose is to validate that each unit of the software code performs as expected.
* Unit Testing is of two types automatic and manual but in our project automatic testing is very hard because output is very dynamic and input is in voice, so we have done manual unit testing, so that we can cover maximum functionality in our code.
* The **Unit Testing Techniques** are mainly categorized into 2 parts which are Black box testing that involves testing of user interface along with input and output, White box testing that involves testing the functional behavior of the software application.
* White box testing is very hard in our project so due to lack of time we have performed black box testing.
* We have used the equivalence class method for efficient black box testing. For each functionality we will have three equivalence classes.
* Suppose our query is “open chrome”, so in our hash dictionary we have related functions for “open chrome”



* Even if we say “hey can you please chrome for me”,” please start chrome for me” our chrome caller function will be triggered and chrome will open.
* So, there are equivalence classes for unit testing in our project

**1) We put the exact string which is mapped to the caller function, like “open chrome”,” start chrome”, “begin chrome” in chrome functionality.**

**2) We do not put the exact string which is mapped to the caller function, like for our open chrome functionality we will input strings like** “**hey can you please chrome for me”** **or,” please start chrome for me”.**

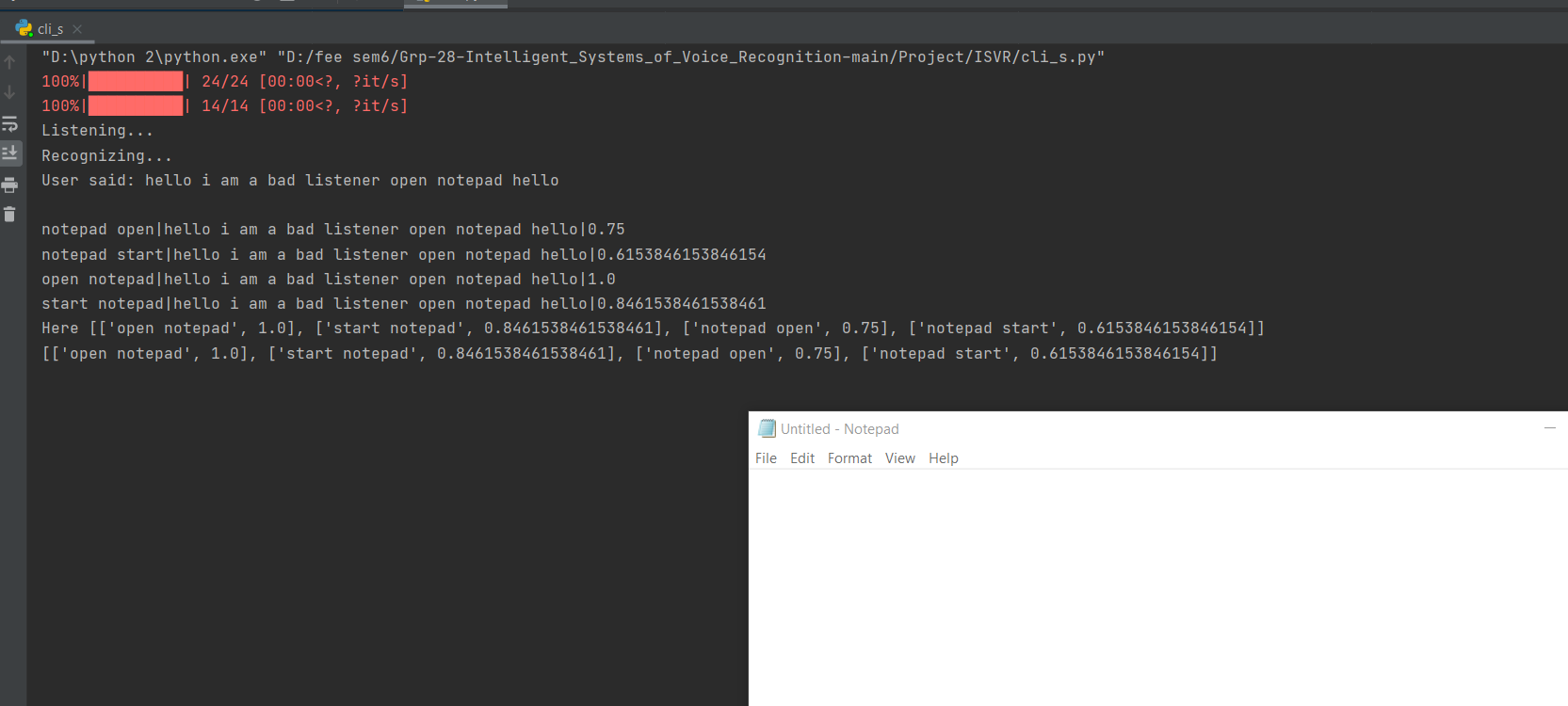
**(3) we put totally unrelated string which has nothing do with our functionality**

* In unit testing we will test individual modules. In our program we have a total of 10 modules. We have tested each module according to equivalence class technique so that we can have a minimum number of test cases for each module.

5.1.3 INTEGRATION TESTING

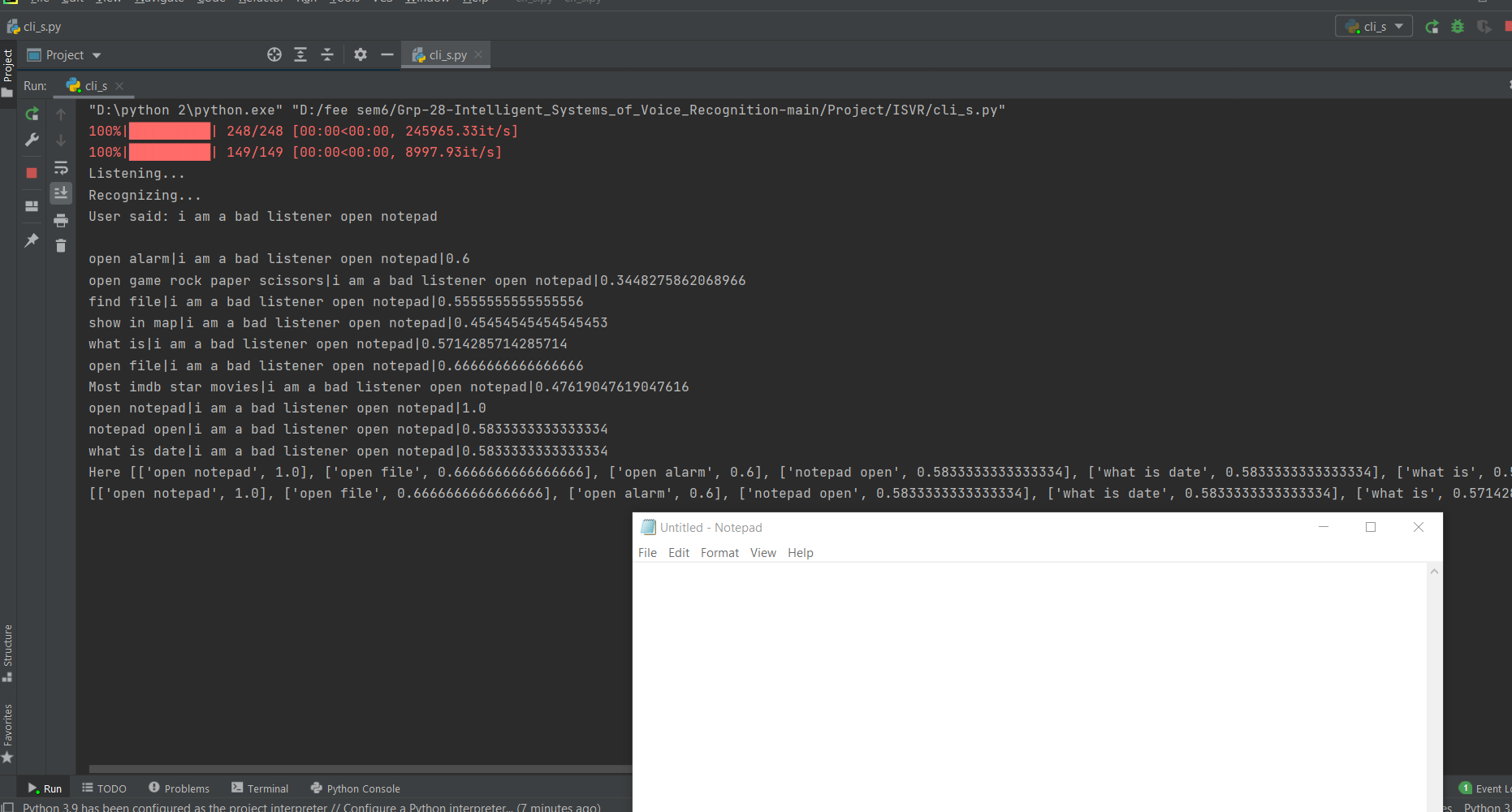
* Integration testing is a level of software testing where individual units / components are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units.
* Now in our code we integrate all the dictionaries and modules so that we can do integration testing.
* The difference between unit and integration testing is that in integration testing sometimes the input query’s caller function can clash with different module’s caller functions. An example is given below

Suppose I say “I am a bad listener, open notepad”. If we run this test in a single module only the functions “open notepad” and “start notepad”, “open notepad” will have some positive weight.



Here we can see functions open notepad has value 1 which is maximum so it will trigger, in above image we can see that start notepad has value 0.84615 , “notepad open” has value 0.75 and so on.

But If I speak the same thing in integration testing some other function will also clash with the given query. Same example is given below



Here we can see some new functions from other modules will also have some positive value, “open file” has value 0.66666, “open alarm” has value 0.6 and so on, but “open notepad” function has value 1 which is maximum so it will be triggered. In integration testing also we have used the equivalence classes given as in unit testing.

* **So, in integration testing, we need to ensure that our code runs for all types of complex queries and some unintended functions from other modules are not being triggered.**
* **For more detailed test cases, static analysis, unit and integration testing check links given below**
* **For functionality unit testing, integration testing and static analysis**

[**https://docs.google.com/document/d/1SpnwgdeV4BMgUAP1JDssuoWcHxROHaxXSHcLvONtghQ/edit**](https://docs.google.com/document/d/1SpnwgdeV4BMgUAP1JDssuoWcHxROHaxXSHcLvONtghQ/edit)

* **For core algorithm unit testing and white box testing**

[**https://github.com/CarbonDDR/Grp-28-Intelligent\_Systems\_of\_Voice\_Recognition/blob/main/Project/ISVR/test\_recombyte1.py**](https://github.com/CarbonDDR/Grp-28-Intelligent_Systems_of_Voice_Recognition/blob/main/Project/ISVR/test_recombyte1.py)

[**https://github.com/CarbonDDR/Grp-28-Intelligent\_Systems\_of\_Voice\_Recognition/blob/main/Project/ISVR/test\_recombyte2.py**](https://github.com/CarbonDDR/Grp-28-Intelligent_Systems_of_Voice_Recognition/blob/main/Project/ISVR/test_recombyte2.py)

SECTION 6

6.1. Part of the requirements not completed/implemented

* Mailing over voice command

We are currently facing some login issues and when there are multiple IDs that are logged in the machine, and also some issues related to attaching documents in the mail.

* Some functionalities are not working properly on all platforms

There are some applications which are platform specific and in implementing some functionalities we need to have a specific format of the path of the application beforehand which is common on the same platform.

* We couldn’t implement the frontend part of the application due to lack of time, though we have the basic structure of it but merging is still pending.

6.2. Future Goals

* Solving the issue related to different platforms and making it as general as possible.
* Addition of more functionalities mainly the mailing over voice command.
* Improving the dictionary of words, because in some cases due to common words sometimes a different function executes instead of the desired one.
* We would like to extend our work in the frontend part too. So, a voice user interface where users can interact with our system with just a single click.

Minutes of Meeting

MEETING 1

Date: 23/1/2021 Timings: 8pm

Location: Google Meet

Agenda: Introduction of group members and discussion about project selection.

Discussion: All the members of the group introduced themselves. Everyone expressed his/her views regarding the projects that they liked. Then, we all came to a common consensus and selected the following three projects: 1. eCommerce site for the clothing retailer 2. Intelligent system of voice recognition 3. Job fiction

MEETING 2

Date: 2/2/2021 Timings: 7pm

Location: Google Meet

Agenda: Discussion about flow of the project

Discussion: The group members gathered to discuss the functional and nonfunctional requirements. Each member gave some ideas regarding the features that need to be implemented in the project. After brainstorming through the ideas presented by the members, a final list of functional and nonfunctional requirements was documented by a team member.

MEETING 3

Date: 7/2/2021 Timings: 7pm

Location: Google Meet

Agenda: Discussion on the software requirements

Discussion: The group members had discussion on the software requirements. We discussed what programming languages to be used for the backend of our software system.

MEETING 4

Date: 13/2/2021 Timings: 9pm

Location: Google Meet

Agenda: Discussion about FR and NFR, requirement elicitation

Discussion: The group members gathered to discuss the functional and nonfunctional requirements. Each member gave some ideas regarding the features that need to be implemented in the project. Further we discussed the requirement of elicitation.

MEETING 5

Date: 14/2/2021 Timings: 10pm

Location: Google Meet

Agenda: Discussion on the Use case diagram, functionalities to be included and distribution of work

Discussion: We discussed what all use cases can be added to our project, along with further discussion on the functionalities of our system. Once the functionalities were decided, allocation of work to individual members was done.

MEETING 6

Date: 10/3/2021 Timings: 10pm

Location: Google Meet

Agenda: Discussion on the implementation of the functionalities

Discussion: The group members discussed how to implement the functionalities.

MEETING 7

Date: 15/3/2021 Timings: 8pm

Location: Google Meet

Agenda: Discussion on the progress made so far in implementing the functionalities

Discussion: The group members gathered to discuss the progress made so far. Further doubts while implementing the code were resolved in this meeting.

MEETING 8

Date: 17/3/2021 Timings: 7pm

Location: Google Meet

Agenda: Discussion of individual code queries

Discussion: The group members gathered to discuss the progress made so far. Further doubts while implementing the code were resolved in this meeting.

MEETING 9

Date: 31/3/2021 Timings: 9pm

Location: Google Meet

Agenda: Mid-evaluation work

Discussion: The group members gathered to discuss the documentation of the mid-evaluation work, which included use case and activity diagram.

MEETING 10

Date: 3/4/2021 Timings: 7pm

Location: Google Meet

Agenda: Discussion related to code

Discussion: The group members gathered to resolve the issues faced while coding.

MEETING 11

Date: 10/4/2021 Timings: 7pm

Location: Google Meet

Agenda: Discussion on resolving errors of individual codes.

Discussion: The group gathered to resolve the errors which were faced by other members throughout coding.

MEETING 12

Date: 3/5/2021 Timings: 8pm

Location: Google Meet

Agenda: Merging final code on Github and running on each machine

Discussion: All the individual codes were merged together and we checked for possible errors.

MEETING 13

Date: 5/5/2021 Timings: 4pm

Location: Google Meet

Agenda: Resolving errors and discussion for all the documentation work and video making

Discussion: We continued our work in resolving the errors, and further distribution of all the documentation work was done. Thinking about what can be demonstrated in the video was discussed.

MEETING 14

Date: 6/5/2021 Timings: 4pm

Location: Google Meet

Agenda: Discussion for all the documentation work and video making

Discussion: Further discussion of all the documentation and presentation was done. Thinking about what can be demonstrated in the video was discussed.

YouTube Video: <https://www.youtube.com/watch?v=elm5ZBhcywU>