UE18CS390B – Capstone Project Phase – 2

SEMESTER - VII

END SEMESTER ASSESSMENT

Project Title	:	Speak Pseudocode2c : A framework to convert customized pseudocode to c code
Project ID	:	PW22NVP03
Project Team	:	Raghav Aggarwal (PES1201800312) Rajdeep Sengupta (PES1201800144) Shaashwat Jain (PES1201802346) Srishti Sachan (PES1201802126)
Project Guide	:	Prof. Nitin V Pujari

Outline

Abstract

- Currently, we are surrounded by a world dependent on AI-based speech assistants which are built to make many tasks in our lives easy.

- As the government aims to strengthen India’s competitive edge in terms of IT, they have introduced coding as part of main course curriculum from Class-VI onwards.

- The main objective is for students to not make trivial mistakes but learn the concepts and gain interest.

- At the moment, there is marginal support provided to help these students, to make them understand the programming concepts better.

Abstract

- We propose a framework that retrieves text data from streamed voice input given by user, converts the raw text input to processed natural language formatted output pseudocode.

- Pseudocode makes an algorithm readable to students because it abstracts the details of programming languages and also lowers the programming language barrier faced by the students in learning how to define algorithms.

- Then, it maps the generated customised pseudocode to finished c code and well-defined algorithms to make programming easy for people who are new to the programming world in general and school students in particular.

Team Roles and Responsibilities

Name	Responsibility	Roles
Shaashwat Jain	• Literature Survey • Speech-To-Text • Mapping Pseudocode to Source code • Integration of Framework	• Speech-To-Text (Pre and Post Language Processing) • NLP • Variable Tracker • Integrating Constructs and Testing • Exception class and handling
Srishti Sachan	• Literature Survey • Text-To-Speech • Testing • NLP (Natural Language Processing)	• IF Else constructs • Testing If and Else constructs • NLP • Text-To-Speech (Research) • XML Approach for converting pseudocode to Source code

Team Roles and Responsibilities

Name	Responsibility	Roles
Raghav Aggarwal	• Literature Survey • Speech-To-Text • Mapping Pseudocode to Source code • GUI (Graphical User Interface) - Backend	• Speech-To-Text (Setup and Tuning Parameters) • For and While loop constructs • Testing for and while loop constructs • Implementing and Integration of GUI with Framework • PLY compiler to convert pseudocode to source code
Rajdeep Sengupta	• Literature Survey • Generating Formatted Text • GUI - Frontend • Testing	• Text-To-Speech (Research) • Input Output constructs • Testing Input and Output Constructs • Implementing GUI • ML approach to convert Pseudocode to Source code

Summary of Requirements-I

Following are the requirements for our framework-

- Google cloud is required for Speech-To-Text translation.
- GCC compiler should be installed on the user system.
- User should enable microphone for taking speech as input.
- User should also have Tkinter(python library) installed on his system as it is required for our Graphical User Interface(GUI).

Summary of Requirements-II

Following listed are the constraints:

- Currently, we are using the Google Cloud free tier which restricts us from using their service for any commercial applications. We have to abide by their terms of service for the same.

- Google Cloud free tier has a limitation for Speech-to-Text API which we are using in the development phase of the application, it is free for 60 minutes/month afterwards it will use the credits.

- The hardware requirements for the system are minimal, although there is a requirement for a microphone with a working internet connection.

Summary of Requirements-III

Following are the Risks involved:

- Microphone being used by another application.
- As internet is necessary for translation purpose, loss of internet connection in the middle might cause trouble for the application.
- User trying to implement a functionality which is not yet supported by our framework.
- Retrieving voice input from the cloud with subpar accuracies.

Summary of Requirements-IV

Following are the assumptions made:

- User has a working internet connection.

- The host machine has gcc compiler installed for execution of the c programs along with python3 and google cloud python library which is necessary for running the framework.

Summary of Design Approach-I

- We have decided to follow the object-oriented design approach where the system is viewed as a collection of objects (i.e., entities). The tasks are defined for one purpose and cannot refer to or change the data of other objects.

- An object is an encapsulation of state (data values) and behavior (operations). The state is distributed among the objects, and each object handles its state data.

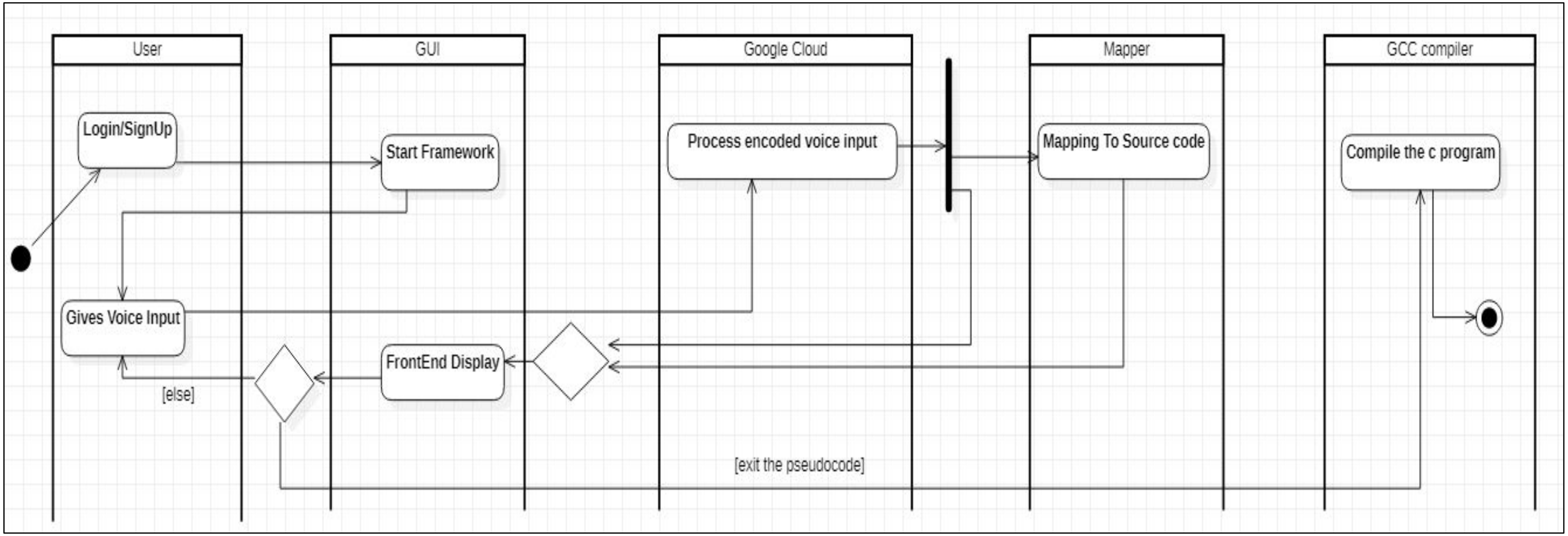
Summary of Design Approach-II

- Object-oriented design is a bottom-up strategy that is more suitable when a system needs to be created from some existing system, where the basic primitives can be used in the newer system.

- Advantages of Object-oriented design:
    - Programs are easy to maintain.
    - Improves code reusability and facilitates the quick development of programs where parallel development of classes is possible.
    - Programs are easier to test, manage and debug.

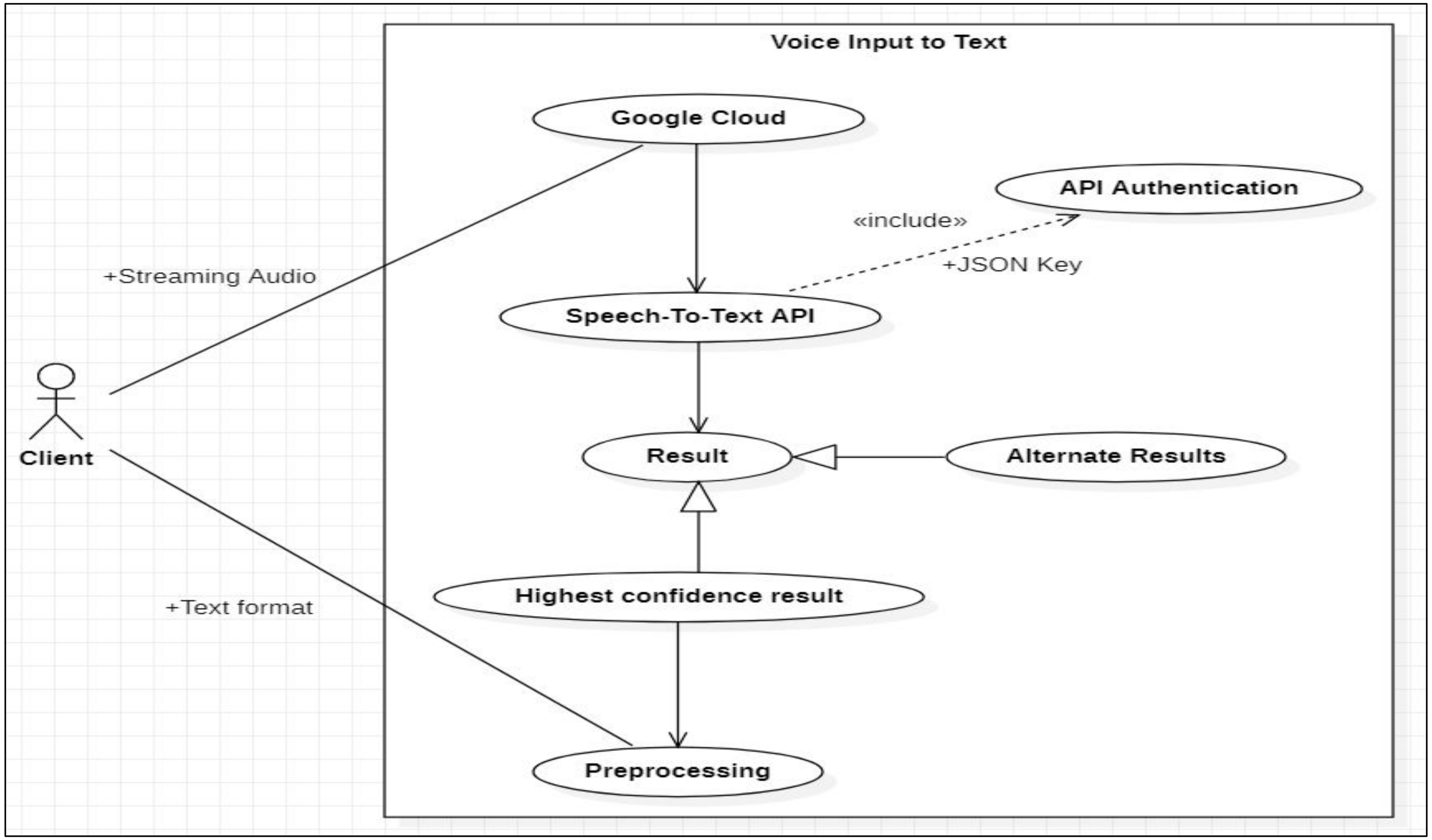
Design Description-I

Activity Diagram



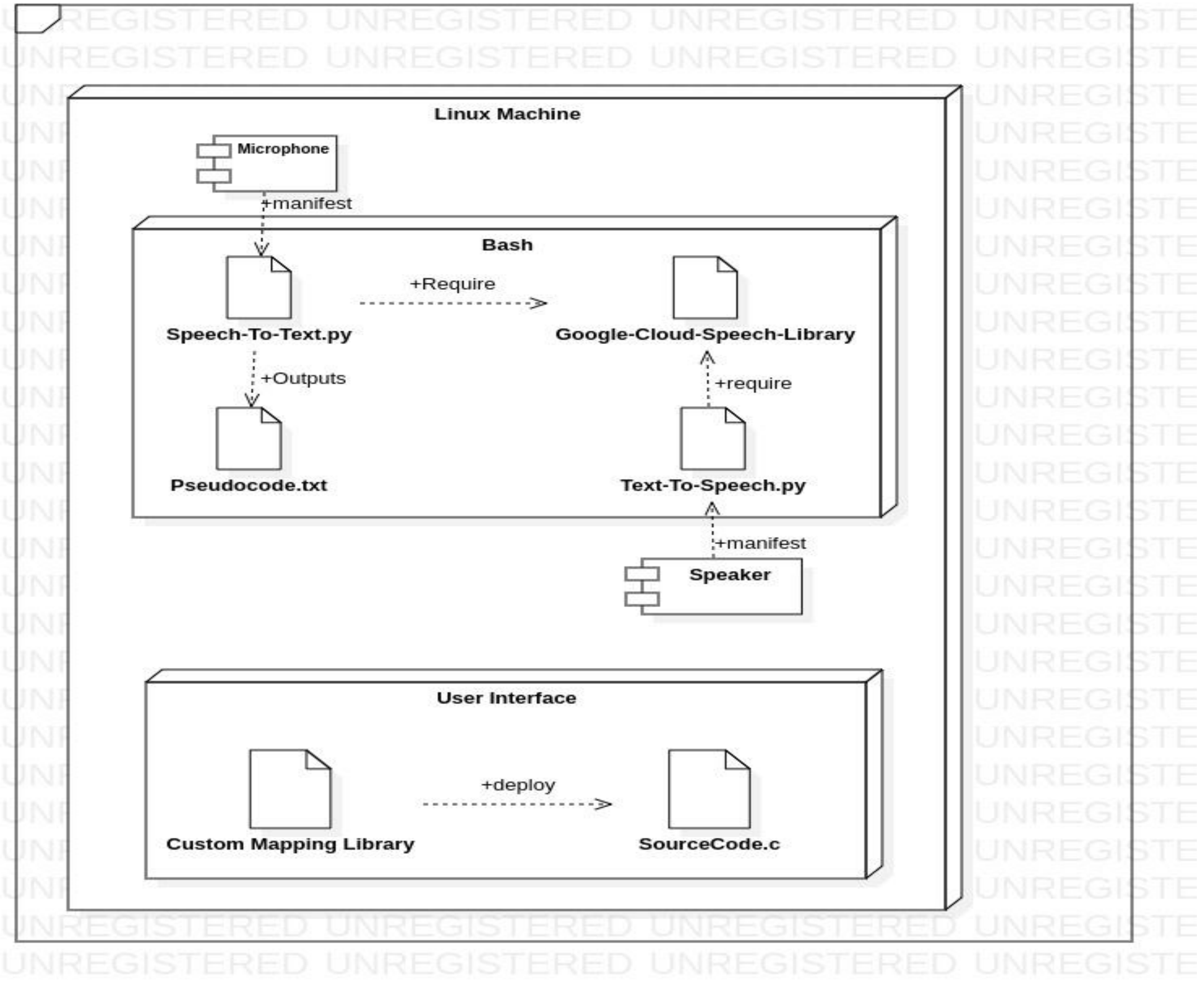
Design Description-II

Use Case Diagram



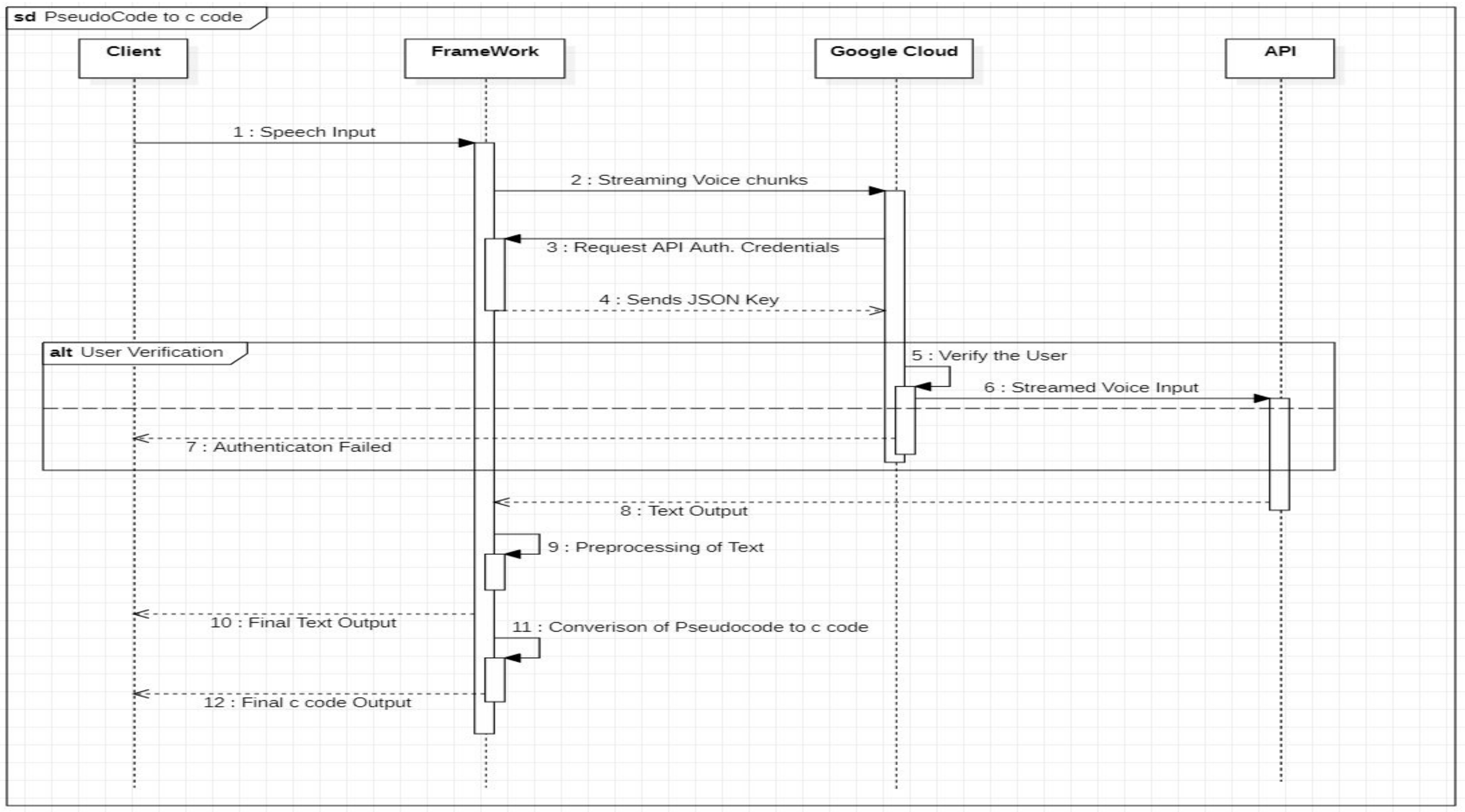
Design Description-III

Deployment Diagram



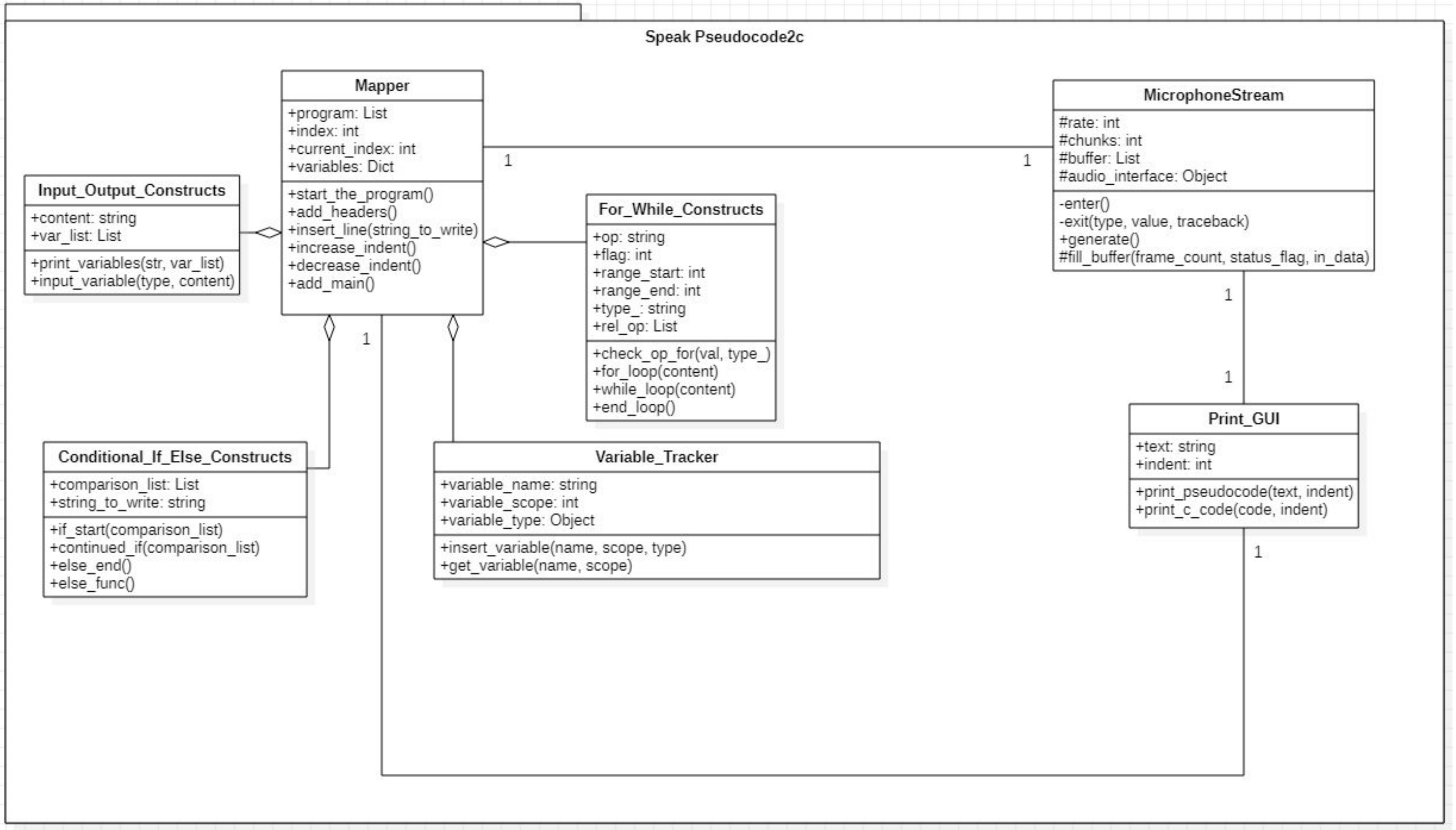
Design Description-IV

Sequence Diagram



Design Description-V

Master Class Diagram



Modules and Implementation Details-I

- Variable Tracker Module:
  - It keeps track of variable’s type and scope throughout the program for easy access to other modules and exceptions like Variable Already Declared and Variable Not Declared scenarios.

- Input/Output Module:
  - The module receives distinct types of inputs from the user i.e. declare, initialize, input (scanf) and output (printf). Uses an escape keyword for variables in output called “variable” followed by variable name.

Modules and Implementation Details-II

- If… else conditional module:
  - Maps all the if, else, and else if statements by the user to the conditional statement structure in c. Has support for giving multiple relational operators in the condition.
  - Pseudocode :- if number less than ten

```
if number less than ten
print number less than ten
else
print number not less than ten
end if
```

  - Output :-

```c
if(number<10) {
        printf("number less than 10");
} else {
        printf("number not less than 10");
}
```

Modules and Implementation Details-III

- For Loop module:
  - Takes in multiple inputs from the user for all different types of implementations possible for “for” loop. Can have empty declarations and can handle increment and decrement of iterators easily. It handles declaration of the iterator if needed, or uses an iterator declared in the previous scope. For example:-
  - Pseudocode:- for i in range from 1 till number increment by 1.
  - Output:- for(int i = 1; i <= number; i++)

Modules and Implementation Details-IV

- While Loop module:
  - Maps all the while statements spoken by the user and handles any variable not declared errors by some default declaration of the variables used in the while loop. Also has support for multiple relational operators. Can use the break and continue statements to exit from the loop for some condition.

    For example:-
  - Pseudocode:-
    ```
    while i < 10
    end while
    ```
  - Source code:-
    ```
    int i = 0;
    while(i < 10)
    {
    }
    ```

Test Plan and Strategy-I

- Testing activities which were carried out along the timeline includes unit testing for each implementation module and the integration testing for the framework.

- The Functional Testing which we took care was unit testing and integration testing for the modules. A total of 95 test cases are written to cover the both. The Non-Functional Testing part is handled by the framework third parties, namely, Google Speech-To-Text and python modules which were used for the same.

Test Plan and Strategy-II

- The test environment was setup by using ***pytest*** for python, input and output test cases were handled by Rajdeep, if else constructs were handled by Srishti, while and for loops were handled by Raghav and Integration Testing was handled by Shaashwat.

- Using this approach we can handle any future errors with the ease of knowing which test assertions failed and for what reasons. Also, this being an automated test approach, it makes it easier for future testing of the framework for deployment.

- We wrote our own 30 programs and their pseudocode for testing and benchmarking purpose, 10 programs each for class 6, class 7, and class 8 with increasing difficulty.

Project Demonstration

- Demo will be presented now.

Walkthrough

- Code Walkthrough will be shown now

Results and Discussion

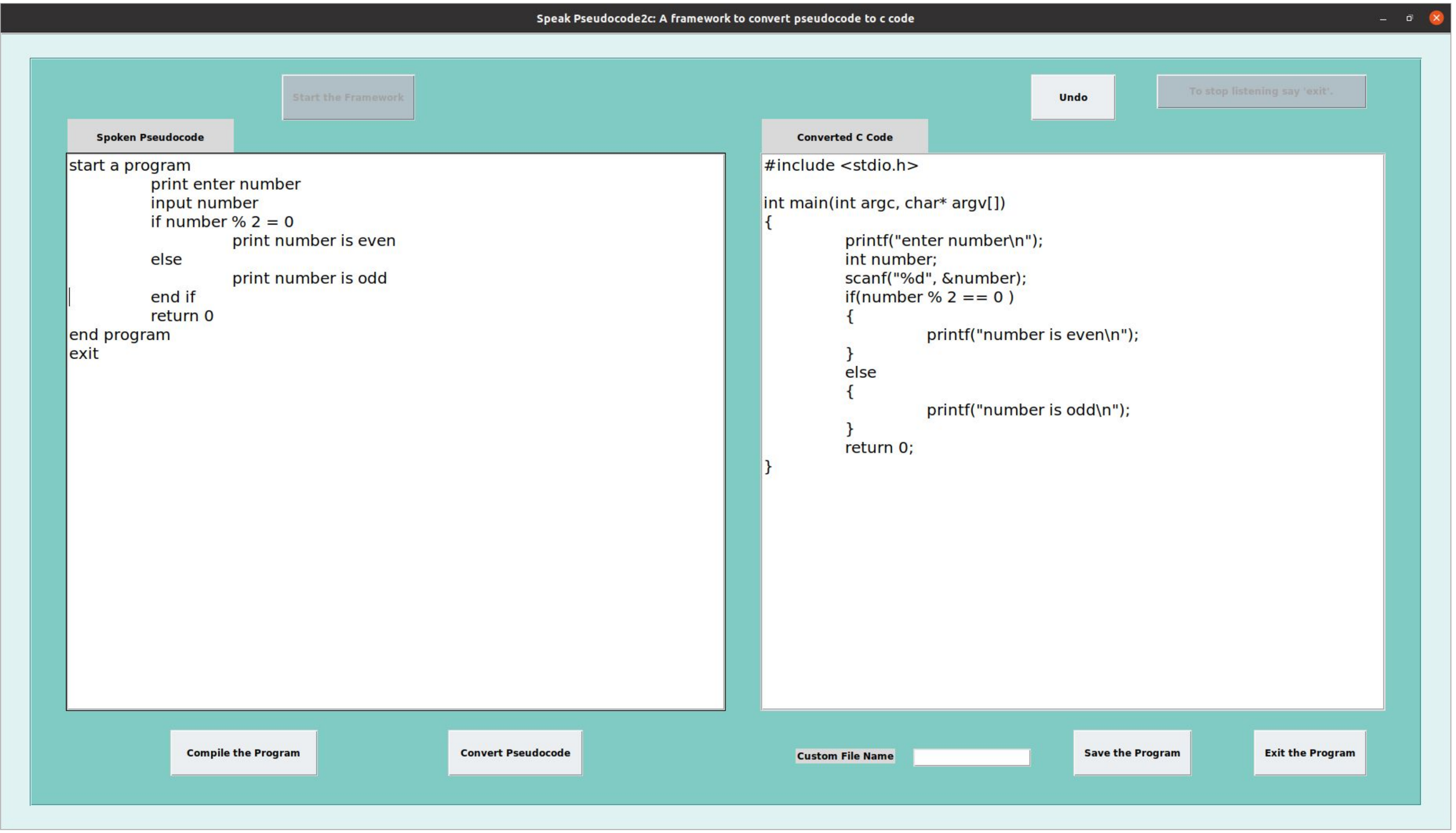
Our results for the project are as follows:-

- According to cxtoday[7], Google cloud speech-to-text scored a 79% accuracy in 2020.

- Due to the nature of the stubs approach and static mapping, the accuracy for the framework for converting speech to source code will also be 79%.

- The accuracy of the conversion, given that the user does not speak anything which is alien to the program and the conversion made by the engine is accurate, will be 100%.

Results and Discussion



```
start a program
        print enter number
        input number
        if number % 2 = 0
                print number is even
        else
                print number is odd
        end if
        return 0
end program
exit
```

```
#include <stdio.h>

int main(int argc, char* argv[])
{
        printf("enter number\n");
        int number;
        scanf("%d", &number);
        if(number % 2 == 0 )
        {
                printf("number is even\n");
        }
        else
        {
                printf("number is odd\n");
        }
        return 0;
}
```

Fig: This is the framework, currently converted for a program to print odd and even numbers. The left side represents the pseudocode spoken and the right side represents the converted source code in c.

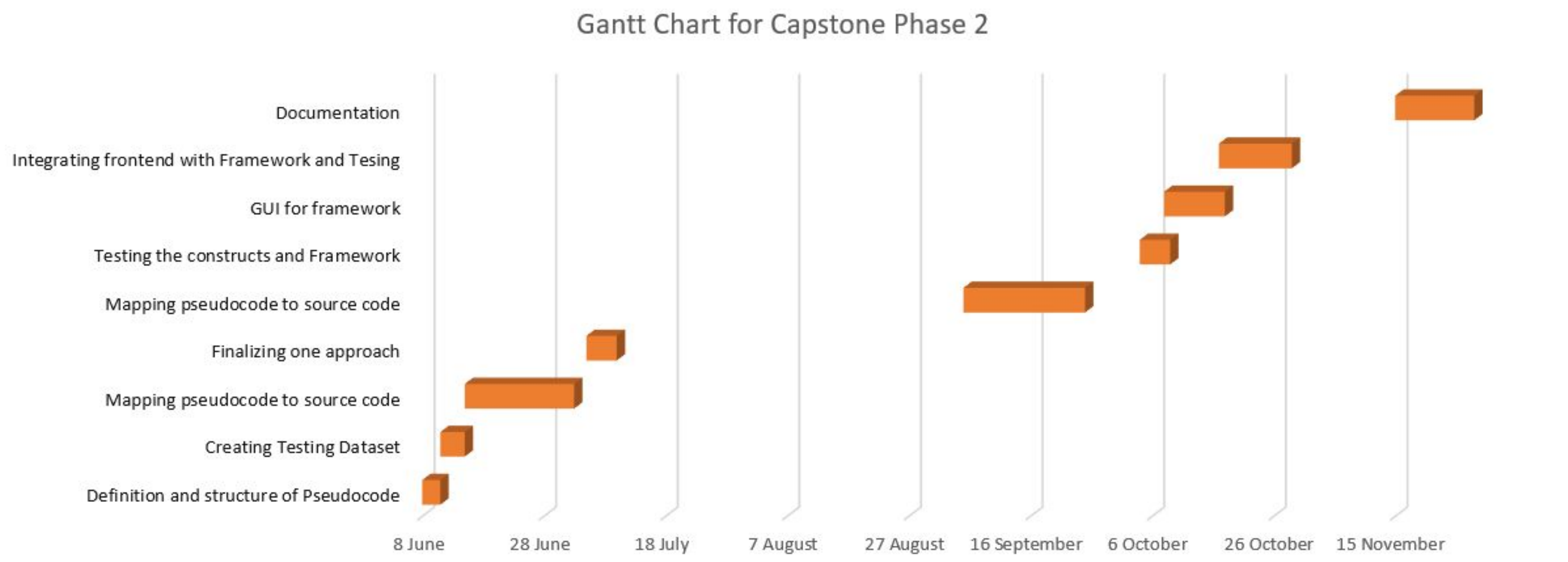
Results and Discussion

- The framework is currently tested on a set of 30 programs chosen for beginner levels i.e. class 6th, 7th, and 8th, 10 programs each.

- This accuracy comes with some assumptions that the user is following a certain set of rules, like giving variables meaningful names which can be spoken and recognised easily.

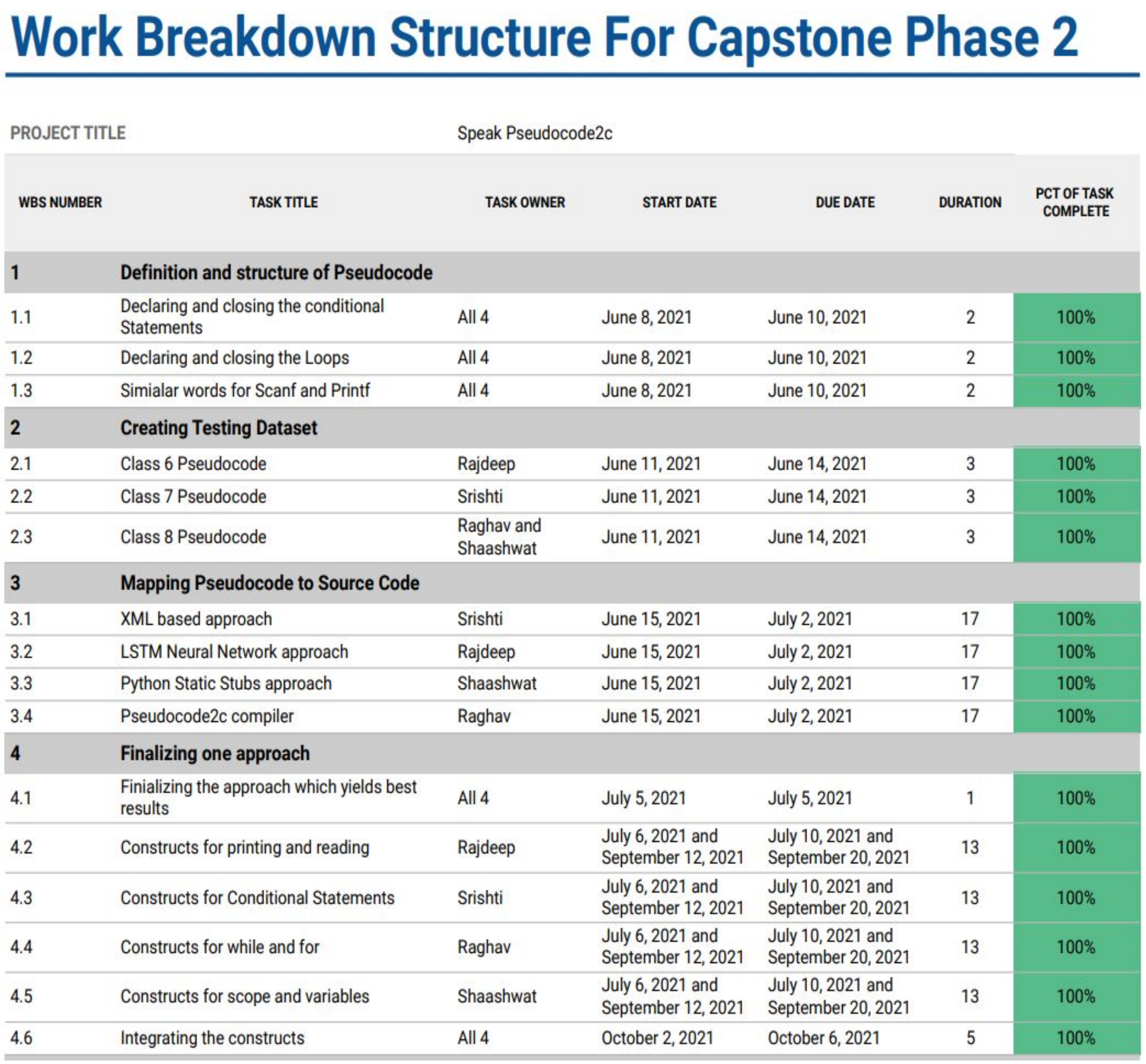
- The framework also includes tools for undoing wrong translations made by speech or otherwise, which makes it more robust to errors.

Schedule



	Definition and structure of Pseudocode	Creating Testing Dataset	Mapping pseudocode to source code	Finalizing one approach	Mapping pseudocode to source code	Testing the constructs and Framework	GUI for framework	Integrating frontend with Framework and Tesing	Documentation
Start Date	8 June	11 June	15 June	5 July	5 September	4 October	8 October	17 October	15 November
🟧 Duration	3	4	18	5	20	5	10	12	13

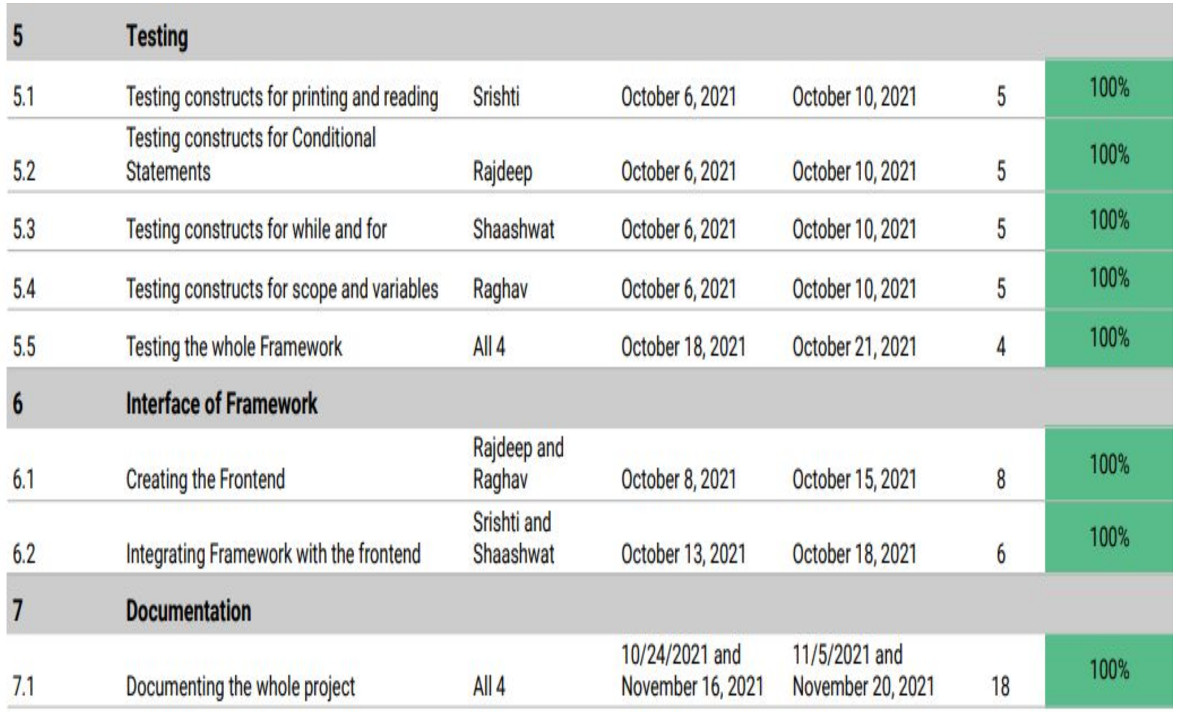
Schedule



Work Breakdown Structure For Capstone Phase 2

PROJECT TITLE Speak Pseudocode2c

WBS NUMBER	TASK TITLE	TASK OWNER	START DATE	DUE DATE	DURATION	PCT OF TASK COMPLETE
1	**Definition and structure of Pseudocode**					
1.1	Declaring and closing the conditional Statements	All 4	June 8, 2021	June 10, 2021	2	100%
1.2	Declaring and closing the Loops	All 4	June 8, 2021	June 10, 2021	2	100%
1.3	Simialar words for Scanf and Printf	All 4	June 8, 2021	June 10, 2021	2	100%
2	**Creating Testing Dataset**					
2.1	Class 6 Pseudocode	Rajdeep	June 11, 2021	June 14, 2021	3	100%
2.2	Class 7 Pseudocode	Srishti	June 11, 2021	June 14, 2021	3	100%
2.3	Class 8 Pseudocode	Raghav and Shaashwat	June 11, 2021	June 14, 2021	3	100%
3	**Mapping Pseudocode to Source Code**					
3.1	XML based approach	Srishti	June 15, 2021	July 2, 2021	17	100%
3.2	LSTM Neural Network approach	Rajdeep	June 15, 2021	July 2, 2021	17	100%
3.3	Python Static Stubs approach	Shaashwat	June 15, 2021	July 2, 2021	17	100%
3.4	Pseudocode2c compiler	Raghav	June 15, 2021	July 2, 2021	17	100%
4	**Finalizing one approach**					
4.1	Finialazing the approach which yields best results	All 4	July 5, 2021	July 5, 2021	1	100%
4.2	Constructs for printing and reading	Rajdeep	July 6, 2021 and September 12, 2021	July 10, 2021 and September 20, 2021	13	100%
4.3	Constructs for Conditional Statements	Srishti	July 6, 2021 and September 12, 2021	July 10, 2021 and September 20, 2021	13	100%
4.4	Constructs for while and for	Raghav	July 6, 2021 and September 12, 2021	July 10, 2021 and September 20, 2021	13	100%
4.5	Constructs for scope and variables	Shaashwat	July 6, 2021 and September 12, 2021	July 10, 2021 and September 20, 2021	13	100%
4.6	Integrating the constructs	All 4	October 2, 2021	October 6, 2021	5	100%



5	**Testing**					
5.1	Testing constructs for printing and reading	Srishti	October 6, 2021	October 10, 2021	5	100%
5.2	Testing constructs for Conditional Statements	Rajdeep	October 6, 2021	October 10, 2021	5	100%
5.3	Testing constructs for while and for	Shaashwat	October 6, 2021	October 10, 2021	5	100%
5.4	Testing constructs for scope and variables	Raghav	October 6, 2021	October 10, 2021	5	100%
5.5	Testing the whole Framework	All 4	October 18, 2021	October 21, 2021	4	100%
6	**Interface of Framework**					
6.1	Creating the Frontend	Rajdeep and Raghav	October 8, 2021	October 15, 2021	8	100%
6.2	Integrating Framework with the frontend	Srishti and Shaashwat	October 13, 2021	October 18, 2021	6	100%
7	**Documentation**					
7.1	Documenting the whole project	All 4	10/24/2021 and November 16, 2021	11/5/2021 and November 20, 2021	18	100%

https://drive.google.com/file/d/1QfiM1EBrrJqjeonc8VeoM7LhQq1ez5_J/view?usp=sharing

Documentation

Evidences and status of the below documents:

- Project report finalized by Guide - DONE and hard copy submitted
- IEEE Draft - https://drive.google.com/file/d/1qztBlN_RR-TbNJmoDvnRCqn8k3XqQYbq/view?usp=sharing
- Video.
- Github repository link - https://github.com/shaashwatjain/Capstone-Project
- A3 Poster - https://drive.google.com/file/d/1ifWuOgE0WC0qjR8FZVVrMRXTIrycqgQL/view?usp=sharing
- All artifacts of your project uploaded in the CSE Project repository - By 15th December.

Lessons Learnt

- The use of a cloud speech-to-text converter increases accuracy but has a tradeoff as it introduces a dependency. In the future to switch to an offline version for the same.
- We are successfully able to convert pseudocode to c language source code.
- Ability of using multiple words and interpreting them using natural language processing.
- This technique can be easily implemented for other programming languages as well.

Conclusion and Future work

Conclusion:

- The framework successfully converts natural language pseudocode to formatted c source code with 100% accuracy.

- User can Compile and Save the generated program through the voice input.

- Added functionality for a narration/commenting system, convert written pseudocode button is there to directly convert the whole pseudocode to c code at one go and an undo button for a robust framework and handling errors in speech.

Conclusion and Future work

- The framework is beneficial to use especially for beginners who want to learn the C language. Speak Pseudocode2c can be accessed by the users through a well formed GUI which provides real time line to line updation of voice input.

- The user can save the generated program with any desired file name. Even if no file name is given, it will be saved with a default name and can be accessed by the user later in time if needed.

- When the framework is started, a set of rules are displayed to brief the user about the guidelines and limitations of the framework for smooth usage throughout.

Conclusion and Future work

Future Work:

- Functionality can be extended to include support for strings and other data types.

- Presently, the framework supports C language only. The framework can be extended for other languages as well.

- The framework can be modified to make the process completely offline as translating and retrieving data can add latency and doing so will remove the internet dependency as well.

- Functionality which provides text to speech conversion can be added. This could be helpful for the user as they can rectify if speech to text translation is correct or not.

References

- [1] P. Sirikongtham and W. Paireekreng, "Improving speech recognition using dynamic multi-pipeline API," 2017 15th International Conference on ICT and Knowledge Engineering (ICT&KE), Bangkok, Thailand, 2017, pp. 1-6, doi: 10.1109/ICTKE.2017.8259624.

- [2] T. Dirgahayu, S. N. Huda, Z. Zukhri and C. I. Ratnasari, "Automatic translation from pseudocode to source code: A conceptual-metamodel approach," 2017 IEEE International Conference on Cybernetics and Computational Intelligence (CyberneticsCom), Phuket, Thailand, 2017, pp. 122-128, doi: 10.1109/CYBERNETICSCOM.2017.8311696.

- [3] Alhefdhi, Abdulaziz & Dam, Hoa & Hata, Hideaki & Ghose, Aditya. (2018), “Generating Pseudo-Code from Source Code Using Deep Learning”. 21-25. 10.1109/ASWEC.2018.00011.

References

- [4] Liu Haowen, Li Wei and Long Yin, "An XML-based pseudo-code online editing and conversion system," IEEE Conference Anthology, China, 2013, pp. 1-5, doi: 10.1109/ANTHOLOGY.2013.6784920.

- [5] Imam, Ayad & Alnsour, Ayman. (2019). The Use of Natural Language Processing Approach for Converting Pseudo Code to C# Code. Journal of Intelligent Systems. 29. 1388–1407. 10.1515/jisys-2018-0291.

- [6] Abdalghani Abujabal, Judith Gaspers, “Neural Named Entity Recognition from Subwords Units” 2019 INTERSPEECH Graz, Austria (2019).

- [7] “How reliable is speech-to-text in 2021?”Mar 2021. [Online]. Available:https://www.cxtoday.com/speech-analytics/how-reliable-is-spe ech-to-text-in-2021/#:~:text=Asperbenchmarkspublishedin,scoredaslightly better84%.

Thank You