CEN4725/5726 Natural User Interaction

Course Project Final Paper

Group Members:
Aarti Kulkarni
Sweta Rathore
Akshay Sehgal
Shivam Agarwal

Section: Graduate

Voice Controlled Web Browser

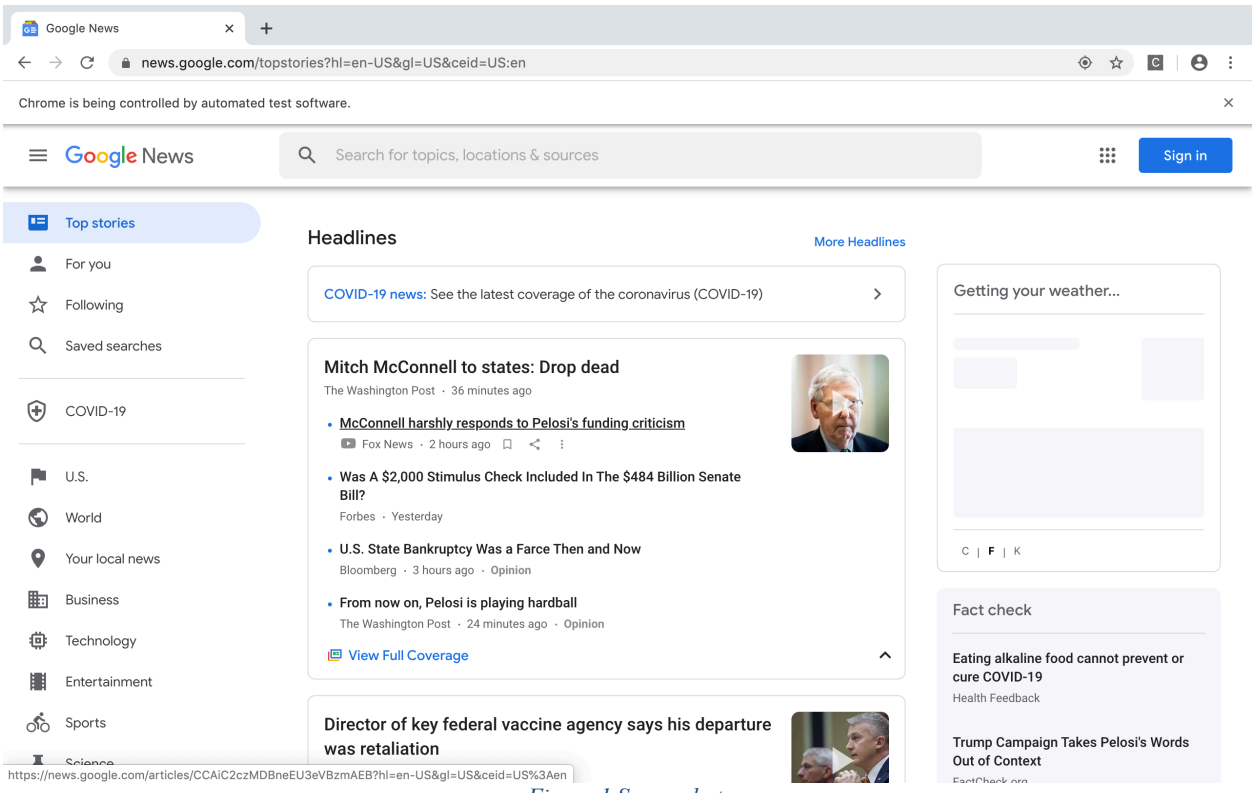

Figure 1 Screenshot

1 Iterative Design and Development Process

1.1 Initial Design

The initial design of the project included a system that is always listening to the user commands which are specific to the browser related tasks. The system shall provide a voice and sound feedback after executing the commands. The selection of commands and scenario building is done using the methods of Requirement gathering, persona and scenario creation and informal interviews with our friends. Affinity diagram and data analysis is done based on the responses gathered from the brainstorming sessions summarized below.

Decision of NUI Commands

The NUI commands and feature selection is decided after initial requirement gathering and data analysis. The focus is to create commands that are - easy to remember, justifies the task performed and are short and concise. For the feature selection, the focus is to set up commands for the task that are used quite often and also provides an excellent feedback to make the interaction more natural.

Set of Features

The set of features are tabulated below with the set of commands used for performing these features after brainstorming and data analysis.

Task Performed	Command
Logging in facebook	Log in facebook
To hear a joke	Tell me a joke
Weather Update	Current weather in {cityname}
Maximizing and minimizing browser	Maximize/Minimize
Closing browser/tab	Close browser/window
Reading news	Read news
Switch tabs	Switch
Download songs	Download songs
Search content on google	Search

Brainstorming and Data Analysis

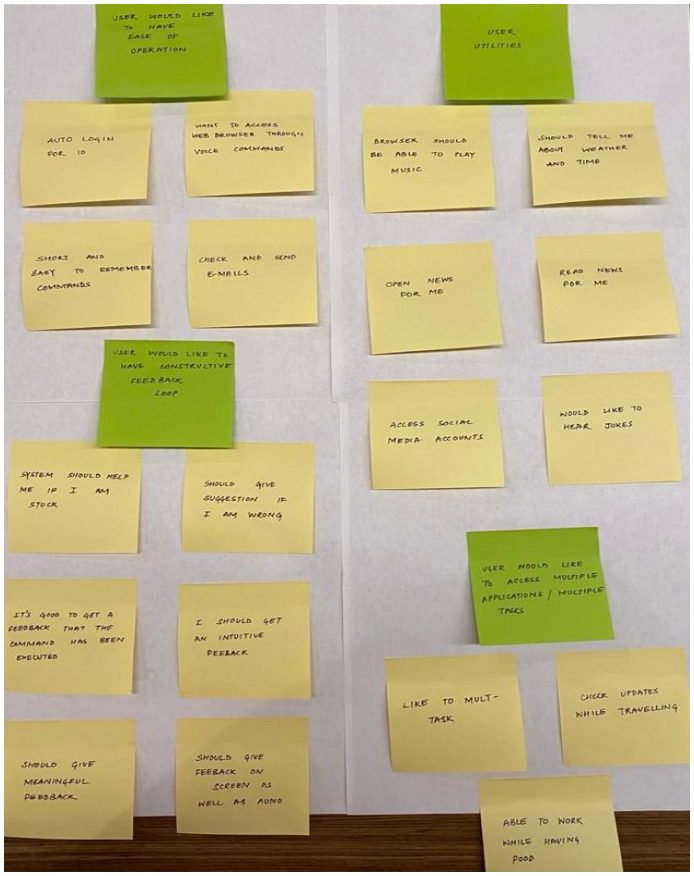

Figure 2 Data Analysis

Personas

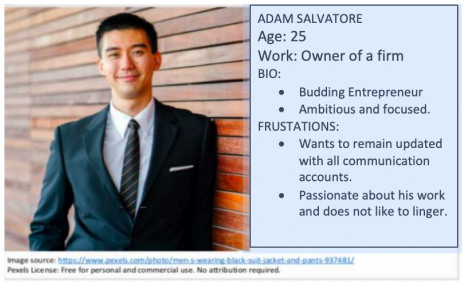

Figure 3 Persona 1

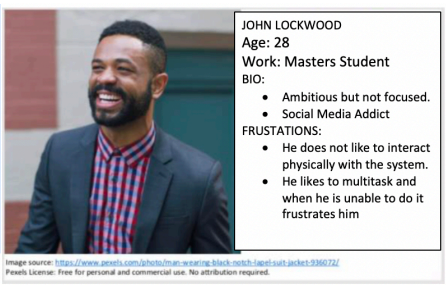

Figure 4 Persona 2

Scenarios

1. One evening Adam was cooking and he receives an important email from his boss. He wants to access the email immediately. However, his hands are occupied. He asked the system to login into the email for him by passing a voice command.
2. John is a master's student and is a social media addict. While he was working on his homework and is occupied by pen and paper, he wants to access his social media account.

He commands his web browser to open social media site and the browser opens it for him.

3. There a soccer tournament going on. Jay is a sports enthusiast. He wants to get the latest updates on the soccer game while he was playing table tennis. He asked the system to open her the news feed for him.

Storyboards

1. For Scenario 1

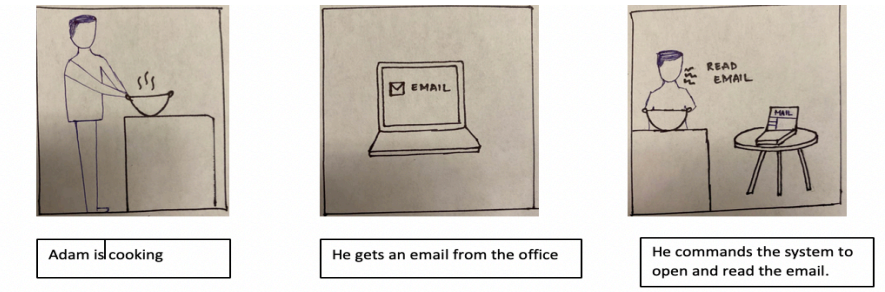

Figure 5 Scenario 1

2. For Scenario 2

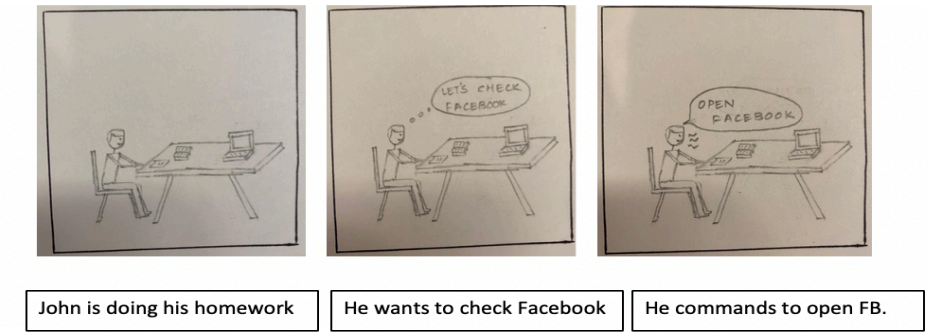

Figure 6 Scenario 2

3. For Scenario 3

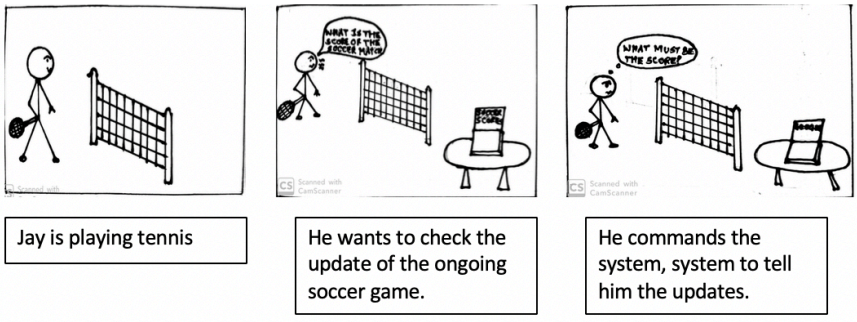

Figure 7 Scenario 3

1.2 Developing the First Prototype

The development process followed by the team is Incremental development model.

Team Members Contribution

Akshay Sehgal – Managed the project timelines, developed the system code and did testing and bug tracking. Also, refined and drafted the final report version from the data collected and presented by other teammates. Implemented all changes to second prototype and created videos of live demo and presentation for submission.

Shivam Agarwal: Handled the management for the development of the project. Contributed in coding and designing of the application and did the data analysis. Made the presentation for the application.

Sweta Rathore: Did the design and analysis for the presentation report and conducted the testing for the application. Helped in making the presentation.

Arti Kulkarni: Drafted the initial version of the report. Conducted informal interview and collected data points

Version Control: Git, **Bug Tracking**: Trello, **Debugging**: Command Prompt

Third Party APIs and Frameworks

Language:	**Toolkits Used**	**Library Used**
Python	Selenium browsers, Chrome web driver	Speech Recognition and other utility libraries

System Architecture Diagram:

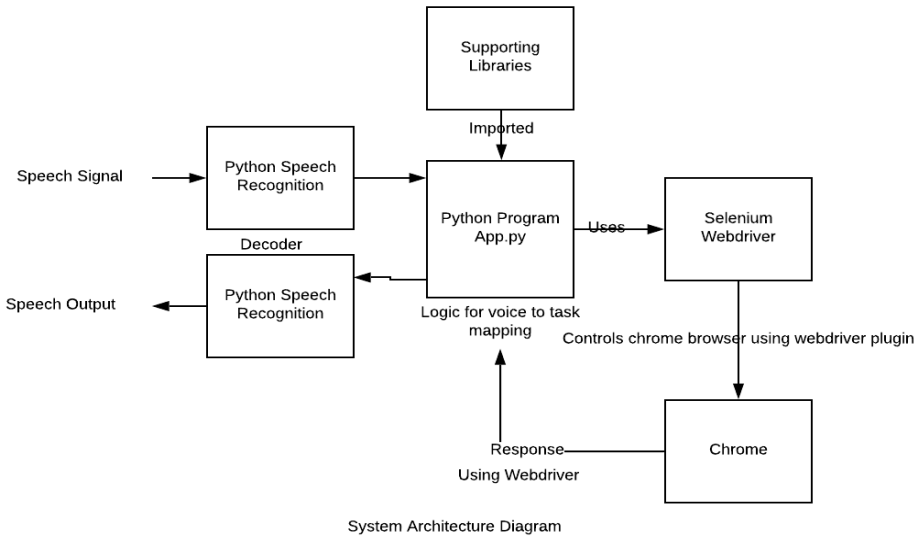

Figure 8 Low Level System Architecture

1.3 Developing the Second Prototype

The second prototype is enhanced by making changes to the system build in the first deliverable. The changes are documented below with the rationale and design considerations. The team followed the same Incremental development process with implementing one change per iterative cycle.

1.3.1 Change 1

Original Functionality: No feedback to the user while processing the voice input

New Functionality: The system is enabled to generate beep sound on success and error. One beep signal 'successful processing' and 2 beeps signal unable to understand the command.

Idea: The idea came from the feedback during mid submissions where it was observed that there is a slight lag i.e. for the time system takes to process the speech input. This did confuse the user about the present state of the system.

Implementation: Used beepy library in python to enable this functionality.

1.3.2 Change 2

Original Functionality: Missing Chaining of commands

New Functionality: The chaining of events is implemented for opening and logging in Facebook.

Idea: After analyzing the feedback from the users, the team felt a need to look for a way to chain commands for some functionality.

Implementation: This is done by creating a set (data structure) in python and storing keywords from first command. After successful completion of command, the set is cleared.

1.3.3 Change 3

Original Functionality: This change is a new functionality

New Functionality: In order to add new functionality, the team worked on implementation for downloading things from the internet. Right now, it is implemented only for songs.

Idea: Frequent file downloading by users, led us to think about implementing this feature for available library.

Implementation: This is done using YouTube plugin for python and also uses web scraping to enable the reading of links and download them.

1.3.4 Change 4

Original Functionality: User has to open browser if he wants to open any website or log in Facebook.

New Functionality: The system is enables to detect whether the browser is open or close or it there is tab opened and act accordingly.

Idea: This was the part of the feedback received after the classroom review and would have been a hassle to the user.

Implementation: This is done using flags for browser and tabs in the python code which are set and cleared as appropriate.

1.3.5 Change 5

Original Functionality: Only close browser functionality was available.

New Functionality: The system is enabled to switch tabs, presently working for only two tabs at a time. Also, close window functionality is added for closing tabs in addition to close browser.

Idea: The idea was to add to the basic sets of commands that are used quite often by the user and we found tab switching and individual tab closing would be useful.

Implementation: The code is modified to keep track of active and update it on closing and switching.

Design Consideration for Changes

1. Major design consideration was to select the feedback for successful receiving the input and on error prompt. The single beep for success and double beep for error are placed into the design to enhance the user feedback loop.
2. Additional changes are made to save the downloaded songs into a local directory.
3. Appropriate follow up questions are designed for chained commands.

2 Final Architecture

2.1 System Architecture

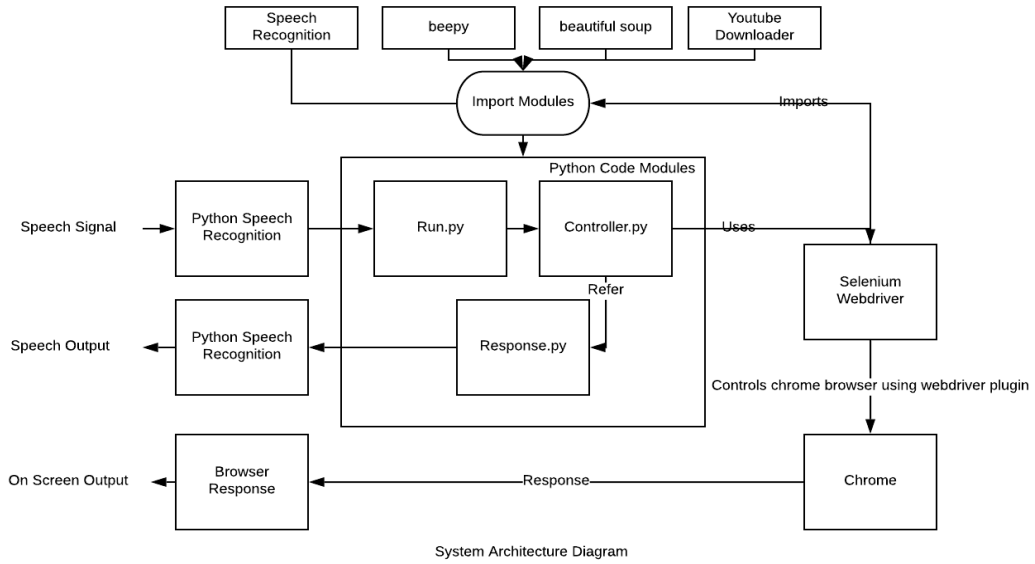

Figure 9 High Level System Architecture

2.2 Code Modules

1. Run.py

   This file contains all the standard imports and contains the continuous speech input loop, that is constantly hearing and mapping words with the utility/task functions in the Controller.py.

Name	Input Parameter	Description
`assistant`	`command (generated from myCommand() in Controller.py)`	`This method provides mapping with the words recognized and the corresponding controller method to be called. Also contains all the utilities for joke, YouTube downloading, weather, time and some other utilities.`

2. Controller.py

   This file contains all the methods for the utilities and actions performed based on the speech input and the mapping defined in Run.py file.

Name	Input Parameter	Description
`myCommand`	`None`	`Creates speech specifics to recognize audio input and`

		generate text response as command.
Maximize	None	Maximizes the browser window
Minimize	None	Minimizes the browser window
Search	None	Helps to search from google
openWebPage	name	Enable opening of webpages with name as input for webpage specifying
newsRead	None	Opens google news and read it to the user
close	Value	Close the browser or window/tab
loginFacebook	None	Helps in logging into facebook
scrollBrowser	None	Results in scrolling the webpage
switchTab	direction, flag	Results in switching of tab based on the parameters
openChrome	None	Open chrome browser
Switch	None	To switch tabs on chrome
setParent	None	Use to switching to enable the system know the active tab
updateParent	None	Results in toggling of parent and switching the tabs. Presently work for two tabs.

3. Response.py

   This file contains all the methods for different responses. The responses can be added/modified and used in all other files.

   Methods

Name	Input Parameter	Description
browserAssistantResponse	audio	Final response to the user using speech recognition python.
ErrorPrompt	None	Contain error description. Calls browserAssistantResponse
initalPrompt	None	Contain initial prompt description. Calls browserAssistantResponse
browserNotOpen	None	Contain browser not open description. Calls browserAssistantResponse

2.3 Third-Party Tools/APIs

- Speech Recognition
- YouTube downloader
- Selenium web driver
- Beautiful Soup
- Beepy

2.4 Source Code

Repository Link: https://github.com/sehgal-akshay/VoiceControlledWebBrowsers

3 Future Work

Future work would comprise of implementing this project with dialog flow as it would enable easy training and enable system to process a wide variety of input from the user. Other iteration can be to implement more selenium features with the aim of achieving maximum automation capability controlled through voice. Some of the functionalities can be extended to a wider use i.e. the download feature, web scraping, tab switching and chaining of commands can be made more efficient to enable wider functionality.