Personal Programming Assistant for the Physically Challenged

By

BSE 17-14

XCoder

DEPARTMENT OF NETWORKS

SCHOOL OF COMPUTING AND INFORMATICS TECHNOLOGY

A Project Report Submitted to the School of Computing and Informatics Technology

for the Study Leading to a Project in Partial Fulfillment of the

Requirements for the Award of the Degree of Bachelor of

Science in Software Engineering of Makerere University

Supervisor

Nsabagwa Mary

Department of Networks

School of Computing and Informatics Technology, Makerere University

mnsabagwa@cit.mak.ac.ug, +256-701-124388

May, 2017

# Declaration

We, group BSE 17-14, hereby declare that the work presented is original and it has never been submitted for an award to any university or institution of higher learning. We can confirm that where we have done consultations either from published material or the works of others, it has been attributed in this report.

|  |  |  |  |
| --- | --- | --- | --- |
| # | Names | Registration Number | Signature |
| 1 | WASSWA DERICK | 13/U/1317 |  |
| 2 | KWIZERA INNOCENT | 13/U/7234/PS |  |
| 3 | LUTALO MOSES | 13/U/7636/PS |  |
| 4 | KATENDE IVAN | 13/U/365 |  |

**Approval**

This project report titled Personal Programming Assistant for the Physically Challenged has been submitted for examination with my approval as the supervisor of group BSE 17-14.

**Signature:** ................................................... **Date:** ............................

**Supervisor:** Nsabagwa Mary

Department of Networks

School of Computing and Informatics Technology;

College of Computing and Information Sciences,

Makerere University

# Dedication

This project is dearly dedicated to our loving parents that have been a primary place of encouragement, provision and exemplary disciplines in most of life spheres, inspiring us to embrace traits such as determination and enthusiasm in all our endeavors. They have sacrificially added piles of deeds both financially and through other means for this project to come to fruition.

# Acknowledgements

We have the almighty God to thank and extend all the glory unto him, for he has granted us great health and knowledge and wisdom all through span of this project, which has enabled us to accomplish all the required deliverables.

We would like to extend our sincere appreciation each and every individual that contributed to the realization of this project, and a special recognition to management of NUDIPU and its staff particularly madam Agnes for the insights they availed to us in the preliminary phases of this project.

We extend out utmost gratitude to our supervisor, Madam Mary Nsabagwa whose selfless efforts; corrections, advice, and insights have been so beneficial to the completion of this project and ensuring that we produce work that is of a substantial quality.

To our fellow course mates, the projects coordinator Mr. Mwotil Alex, and the entire teaching staff of the College of Computing and Information Sciences, Makerere University, we owe a vote of thanks for their direct and indirect contributions to this project. The entire group of BSE 17-14, the utmost coordination and selfless efforts exhibited to the completion of this project has been commendable and deserves applause.

# Abstract

The physically handicapped are vulnerable due to any temporary or permanent loss or abnormality of a body structure or physiological function that are essentially sensory or the limbs making them restricted or unable to perform activities in a manner considered normal for a human being. Likewise, programmers sometimes encounter a syndrome called Restrictive Strain Injury (RSI) that causes wrist pains, which make them unable to do their work.

Some of the disabled people are unable to continue with their careers because of the disabilities. Despite their condition, the disabled persons and affected programmers are required to support themselves and their families. Therefore, we took a study to investigate this problem and suggest solutions that could be implemented to alleviate this problem. Therefore, we established that the physically disabled people especially those that lose their hands but with a sound mind can take on various none strenuous job opportunities to yield income to support themselves and their families such as jobs in the permanent loss or ICT industry such as computer programming.

Computer programming being an art of writing instructions into an editor that a computer can execute requires limited physical effort and hence a good candidate for persons with physical disabilities to take on as a profession with lots of different career options such as corporate product development, game development, embedded systems development, freelancing, custom product development among others, all of which are greatly rewarding.

We proposed and implemented a system that enable persons with physical disabilities and the already established programmers affected by wrist pains to write computer programs by issuing voice commands that are received and displayed with guided syntax. As these people take on programming as a profession, it will help to improve on their livelihood and standards of living.

Table of Contents

Declaration i

Dedication iii

Acknowledgements iv

Abstract v

List of Figures viii

List of Tables ix

Abbreviations/Acronyms x

Chapter 1: Introduction 1

1.1 Background and Scope of the project 1

1.2 Overview of the document 2

Chapter 2: System Specification 3

2.1 Version of requirement and Version Control 3

2.2 Input 3

2.3 Output 3

2.4 Functionality 4

2.5 Limitations and safety 4

2.6 Default settings 4

2.7 Errors and alarms 4

Chapter 3: Design Output 5

3.1 Implementation (coding and compilation) 5

3.1.1 Development tools 5

3.1.2 Languages 5

3.1.3 System Interfaces (UI) 5

3.2 Utilities for validation and testing 5

3.3 Documentation 6

3.5 Design details 6

Chapter 4: Inspection and Testing 7

4.1 Introduction 7

4.2 Test plan and performance 8

4.2.1 Test objective 9

4.2.2 Relevancy of tests 9

4.2.3 Scope of tests 9

4.2.4 Level of tests 9

4.2.5 Types of tests 10

4.2.6 Sequence of tests 11

4.2.7 Configuration tests 11

4.2.8 System Integration tests 11

4.2.9 Regression tests 12

4.2.10 Traceability tests 12

4.3 Precautions 13

Chapter 5: Installation and system acceptance test 14

5.1 Input files 14

5.2 Supplementary files 14

5.3 Installed components 14

5.4 Installed qualification 14

Chapter 6: Conclusion, Recommendations and Future Works 15

7. Appendices 16

7. 1 Appendix A: User Manual 16

7.2 Appendix B: Budget 18

7.3 Appendix C: QUESTIONNAIRE 18

7.4 Appendix D: Glossary 18

# List of Figures

# List of Tables

Table 1: Design details 7

Table 2: Inspection plan and performance 8

Table 3: Test plan and performance 9

Table 4: Sequence of tests 11

Table 5: Traceability tests 12

Table 6: Listing of the Installation and System Acceptance Test 14

Table 7: Installation Procedure Check 14

Table 8: System Acceptance Test 15

Table 9: Commands for creating and manipulating a file 17

Table 10: Commands for setting preferences 17

Table 11: Commands for writing a program 18

Table 12: Commands for running a project in the browser 18

Table 13: Budget 19

# Abbreviations/Acronyms

|  |  |
| --- | --- |
| RSI | Restrictive Strain Injury |
| ICT | Information and Communications Technology |
| HTML | Hypertext Markup Language |
| PwDs | People with Disabilities |
| NUDIPU | National Union of Disabled Persons of Uganda |
| CSS | Cascading Style Sheet |
| API | Application Programming Interface |
| PAPC | Programming Assistant for the Physically Challenged |

# Chapter 1: Introduction

## 1.1 Background and Scope of the project

Handicapped/Disabled people are individuals with temporary or permanent loss or abnormality of a body structure or function, whether physiological or psychological. Impairment is a disturbance affecting functions that are essentially mental (memory, consciousness) or sensory, head, the trunk or the limbs. This leads to restrictions or inability to perform an activity in the manner or within the range considered normal for a human being, mostly resulting from impairment. Hence the ultimate effects prevent persons with disabilities from fulfilling of one or several roles regarded as normal, depending on age, sex and social and cultural factors. Some people are born with disabilities while some become handicapped due to various reason accidents, RSIs, domestic violence, among others. Despite the situation in which they might be, they are obliged to work to survive and support their families just like any other person. Therefore physically disabled people especially those that lose their hands but with a sound mind can take on various none strenuous job opportunities to yield income to support themselves and their families with jobs such as computer programming in the ICT industry.

With the continuous presence of challenges and gaps in different fields say education, entertainment, banking, insurance, aviation, to mention but a few, people with potential to invent and innovate to leverage the existence of these gaps will be given chance to contribute to these fields irrespective of their abilities/disabilities. We proposed a personal assistant programming system that receives voice commands from a person and transcribe them into syntactical text hence the handicapped with skills and love for programming will be catered for.

This project is aimed at producing a personal programming assistant system to support the handicapped/disabled/physically impaired in participating in professional computer programming, academic practical sessions as regards computer programming. Emphasis will be put to web development for starters basing mainly on HTML programming language for witting to web pages that run in web browsers. People with physical disability/physically handicapped will be considered for the proposed system that is the lame, people difficulty in hand movement i.e. paralysis of the hand. The functional scope of the systems includes various functions that is creating new web page file, saving a web page file, speech recognition, syntactical textual corrections against correct programming grammar, custom code suggestions, deletion, speech to text, running web pages.

## 1.2 Overview of the document

This document describes the implementation, testing and validation of findings for the system that is Programming Assistant for the Physically Challenged (PAPC), and it is sub-divided into the following sections.

**Section 1:** This section gives an overview of the document.

**Section 2:** This section describes and specifies the system completely and is the basis for the validation process.

**Section 3:** This section describes the development tools used and the programming procedures followed.

**Section 4:** This section describes the testing techniques used, their relevance to the project, levels of tests and the scope.

**Section 5:** This section details the validation of the installation processes of the system.

**Section 6:** This section describes the documentation of the services and support concerning maintenance, future updates, problems, solutions and the requested modifications.

**Section 7:** This section clearly states the conclusion to this project and recommendations.

# Chapter 2: System Specification

This project (Programming Assistant for the Physically Challenged) is a desktop application that receives voice commands from the user to write computer programs specially HTML language programs. The system requires an active Internet connection for the users voice commands to be captured accurately returning the corresponding text to the issued voice commands. Once the system is fully launched, the user can then start issuing commands to the system.

The acceptable commands are classified into three classes that coding commands, preference commands, and system commands. The system commands include commands that are used to create new files, saving a file to the file system, saving a users’ coding progress to the file. The coding commands include commands for write HTML code for example the commands for the headers, images, links, paragraphs, among others. The preference commands are commands used to set the bot speed, pitch, volume, and rate.

## 2.1 Version of requirement and Version Control

Draft Version

Proposed Version

Validated Version

Approved Version

## 2.2 Input

The system accepts various kinds of input for manipulation inclusive of; system, code, preferences, and clicks inputs/commands.

The command inputs are issued for operations such as creating a file, saving a file, moving the cursor, running a web page among others. Code inputs are responsible for specifying a particular HTML tag that the user wants to access for example the paragraph, header, and image tags. The preference inputs are used to specify the custom system preferences that yield to the users’ need for example the speed, voice, volume of the bot, the application theme. The click input is used to as an alternative to alert the application that the user wants to start issuing voice commands.

## 2.3 Output

The system has four main outputs to the user namely; preference status, code insertion result, command issuance result, and audio feedback from the bot.

The preference status displays the current preferences that the system is using in that moment. The code insertion result contains a code auto completion corresponding to a particular voice input that the system received from the user. The output of command issuance includes results such as saving a file, saving a users’ progress to the file, a cursor moved from one position to another position, and a browser running a web page from the application.

## 2.4 Functionality

**User requirements**

1. The system is convenient to use in terms of accessing all the system function for example it provides an easy to read menu to assist in accessing of the system functions, user guide manuals, user interface metaphors that the users are more familiar with such as menus, use of color to signify different things; red representing errors, and audio feedback to the users.
2. The system automatically saves the users work after specified insertions of code in the coding environment in order to avoid losing work that is not saved, and to reduce on the number of activities to be performed by the user, hence making the system easy to use.
4. The system auto-completes coding commands, hence saving the users time because they do not have to issue out a lot of commands to perform the functions they want.
6. The system is informative as regards errors encountered, directions or guidelines, and success messages.
8. There is a user guide manual that clearly documents what a user can do with the system that is; the system functions, modes of operation, expected results, user characteristics, and other system requirements.

**Reporting requirements**

The system highlights code segments with multiple colors with each color clearly distinguishing a particular group of code segment.

The system shall have audio feedback to the programmer in response to the some of the actions they perform or actions that they would like to perform. The audio messages direct users on what commands to issue next as well as guiding users of how to best use the system.

**System and Integration requirements**

XCoder runs on operating systems like Linux, windows and Unix with minimal requirements to host XCoder as; an input microphone, working speaker, at least 1GB of RAM, HDD space of about 1GB for program files, and dual core processor.

## 2.5 Limitations and safety

System requires an active internet connection, and if there is no internet connection, the voice recognition component shall not be able to capture the user input for feedback of recognized texts, hence no other activity shall be able to be performed.

There are no custom voices for the bot hence the system requires a users’ at most reliance on the primitive voices loaded with Web speech library.

The inbuilt microphones in computers cannot be entirely relied on. Therefore, one has to use an additional microphone peripheral in cases where the voice commands are not captured.

The system only recognizes/captures voice commands issued in the English language preferably with a British accent with an adequate accuracy.

## 2.6 Default settings

The bot has default parameters initialized and they include the language set to English (UK), the speed set to 1, the voice is of a male, and the pitch is set to 1. The user can override these bot default preferences by issuing voice commands to override the nature of voice from that of a male to that of a female.

The system has a default theme set as the night owl containing black for the coding panel and pale white for the side bar for files.

## 2.7 Errors and alarms

The errors are categorized as system and user input errors. System errors include examples such as the XCoder application trying to run using a port number being used by another application, inactive Internet connections. The user input errors include no voice command recognized, no result for issued command was recognized, success messages and status calls (web page loaded).

The bot issues all the system errors and messages to the user through audio messages to the users that help to point the user where attention is required.

# 

# Chapter 3: Design Output

## 3.1 Implementation (coding and compilation)

XCoder has been developed using JavaScript and python as the back-end scripting languages, node-webkit as the web technologies, HTML, CSS for designing the interfaces. The application has been developed in modules so as to break the tasks into small manageable works include the voice recognizer, syntax parser, transcriber, and the bot.

XCoder uses the computer’s inbuilt microphone or other voice input peripherals to capture the user voice commands. These commands are then be transcribed into text using the web speech API that resides on a local server that starts on program launch using the Connect and static server libraries. For cursor movement and position, the caret.js library was used.

The captured text is mapped to real syntactical statements using a dictionary of commands. The prepared dictionary of commands contains the text mapped to the syntax of the HTML language. In cases of syntactical errors, code highlighting, suggestions, short cuts, the dictionary is used to cast visual feedback to the user.

After coding, the user can then issue commands to save their progress to the file system and node JS and FS were used for managing the machines file systems. The application runs on all computers powered with operating systems such as Windows, Linux, and Macintosh using node webkit as the runtime environment for creating the desktop application.

## 3.1.1 Development tools

The primary development tool used is the sublime text editor, which was used for writing the system code. The Node Webkit tools were used as runtime environment tools for creating desktop applications that is an executable.

## 3.1.2 Languages

The primary languages used to develop the system are JavaScript and python as the back-end scripting languages, node-webkit as the web technologies, HTML, CSS for designing the interfaces.

## 3.1.3 System Interfaces (UI)

## 3.2 Utilities for validation and testing

There are several tools that were used for validation and testing including Google chrome development tools were used to test the voice recognizer and bot components to verify whether they perform their required tasks to completion.

Node webkit was used to create a runtime desktop application to validate that a usable application is created.

## 3.3 Documentation

One of the preliminary deliverables for this project was a requirements specification document that spearheaded in identifying and documenting the entire user needs. The document vividly presents an analysis and description user requirements, product functions, reporting requirements, and the system integration requirements.

Taking the requirements specification document as an input, a design specification document was prepared to enable the implementers (BSE 17-14) to easily implement the project as the project architecture, its components, and the development tools to be used were clearly specified. Therefore, following the blueprint provided by the design specification document, the system was implemented and deployed. Then follows the implementation report that documents implementation procedures followed, the tests carried out hence meeting the primary aim of producing a computer programming platform that enables people with disabilities to participate in computer programming.

## 3.5 Design details

|  |  |  |
| --- | --- | --- |
| Topics | Design Output | |
| Good programming practice | Source code is …  Modularized ☐ Encapsulated  Functionally divided ☐ Strictly compiled  Fail-safe (handling errors) | Source code contains … ☐ Revision notes  Comments  Meaningful names  Readable source code ☐ Printable source code |
| Windows Programming |  Interface implemented using standard windows elements  Interface implemented using self-developed windows elements ☐ Application manages single/multiple running instances | |
| Dynamic Testing |  All statements have been executed at least once  All functions have been executed at least once  All case segments have been executed at least once  All loops have been executed to their boundaries ☐ Some parts were not subject to dynamic test | |

Table 1: Design details

# Chapter 4: Inspection and Testing

## 4.1 Introduction

Inspection plan and performance

|  |  |  |
| --- | --- | --- |
| Topics | Inspection plan and performance | Date/Initials |
| Design output |  Program coding structure and source code  Evidence of good programming practice  Design verification and documented reviews  Change-control reviews and reports  Comments: |  |
| Documentation |  System documentation  Test results  User manual  Content of user manual approved Comment: test cases were recorded at all times |  |
| Software development environment |  Data integrity  File storage  Access rights  Code protection  Installation kit, replication and distribution Comment: |  |
| Result of inspection |  Inspection approved |  |

Table 2: Inspection plan and performance

## 4.2 Test plan and performance

|  |  |  |  |
| --- | --- | --- | --- |
| Item to test | How to test | Expected Output | Approved? |
| E.g.  Can the system create a new file? | Issue command to create new file. | New file is created | No |
| Can the system save a file? | Issue command to save progress to a file or saving a file to a file system. | Users’ progress is saved to a file or file is saved to the file system. | No |
| Can the system capture voice commands? | User issues voice commands and have the system capture the issued voice command. | Transcribed text corresponding to the issued voice command. | Yes |
| Does the system issue correct textual syntax? | Map the transcribed text with the contents of the prepared dictionary. | Insertion of the proper HTML syntax match. | Yes |
| Does the system make auto code completions? | Issue a coding command | Completed HTML code segment | Yes |
| Does the system support highlighting code? | Issue command to highlight code segment | HTML code segment highlighted | No |
| Does the system delete the specified code? | Issue command to delete a code segment | HTML code segment deleted | No |
| Can the system load web pages in the browser? | Issue command to run the profile files in the browser | Web page running in the browser | No |

Table 3: Test plan and performance

### 4.2.1 Test objective

Testing was done to discover defects, which may be found while developing the system, to gain confidence in the levels of quality, to make sure, that all user needs are satisfied and to finally prevent defects. A sequence of steps were followed that is say; launching the application, provide some test case data, then verify whether the anticipated results are obtained.

### 4.2.2 Relevancy of tests

The tests helped to ensure that user requirements are met that is to say the users can operate without any complaints.

They helped to discover errors in the system since a number of defects were reported per test case.

Tests helped to prevent defects or faults that may easily occur.

The tests enabled quality enhancement as most defects were identified and fixed before the system is released for production stage.

### 4.2.3 Scope of tests

Component level. Testing was done on different components of the project to ensure that the system requirement specification are satisfied that is to say a good test coverage was done to test the application completely.

Volume. All the projects’ components and corresponding features for example classes, method functions and their respective variables were tested to ensure that proper functionality of a given component is delivered.

### 4.2.4 Level of tests

**Component testing.** The project had all its components including; voice recognizer, transcriber, syntax parser, and bot tested in isolation. This helped to identify the residual defects in each of the components and have them removed before they can be integrate as a whole.

**Integration testing.** The project had all its components combined together as a whole after each and every component had been tested. This ensured that the components are grouped into an aggregate delivering output as an integrated system ready for system testing.

**System testing.** For system acceptance, various test were carried out to ensure that the requirements specifications are met.

### 4.2.5 Types of tests

Functionality. The product functions such as capturing user voice input, creating a file, opening a file, saving to a file, setting preferences, reporting errors and messages to the user, highlighting code segments, deletion of code segments, code segments auto completion were tested against to check whether they were implemented to completion.

Boundaries. The use of the XCoder application was tested against the input value in can accept to ensure conformance of the system to the specified scope of accepting only HTML language tags, preferences commands and alternatives such as clicks.

Usability. User friendliness was checked and application flow was tested to ensure that new users could understand the application easily that is to say proper user manual was provided for whenever a user would like to consult it.

### 4.2.6 Sequence of tests

|  |  |  |  |
| --- | --- | --- | --- |
| **Test cases** | **Test procedure** | **Test data** | **Expected results** |
| Voice capturing | Issues voice commands and check whether the system captures the issued voice command. | Voice command | Text corresponding to the issued voice command |
| File creation | Issue create new file command, then specify the file name | File name | File name created in XCoder |
| Opening a file | Issue open file  “*file name*” command | File name | File is opened in XCoder |
| Saving a file | Issue save file  “*file name*” command | File name | File is saved to the file system |
| Text transcription | Issue coding command | Command and the corresponding value you want e.g. paragraph | Paragraph is written in the coding area |
| Syntax parsing | Issue coding command | Text | Corresponding HTML syntax for the written text |
| Error reporting | Create successful operations or anomalies | Text to return to the bot | Bot speaking back to the user |

Table 4: Sequence of tests

### 4.2.7 Configuration tests

The finished desktop application was tested to ensure that it runs on operating systems such as windows, Unix, and Linux with an active Internet connectivity required for the web speech library to return the text corresponding to the captured user voice input.

### 4.2.8 System Integration tests

This is the testing performed on the application under test, to verify the entire application works as per the requirements.  Critical Business scenarios were tested to make sure important functionalities in the application works as intended without any errors.

### 4.2.9 Regression tests

This testing was performed each time a new build is deployed for testing which components contains defect fixes and new enhancements, if any.

Regression Testing was done on the entire application and not just the new functionalities and defect fixes.  This testing ensured that existing functionalities works fine after defect fix and new enhancements are added to the existing application. Test cases for new functionalities were added to the existing test cases and executed.

### 4.2.10 Traceability tests

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test cases** | **Components** | | | |
| Voice recognizer | Transcriber | Bot | Syntax Parser |
| Capture user command | ✔ |  |  |  |
| Create new file |  |  |  | ✔ |
| Save file |  |  |  | ✔ |
| Open file |  |  |  | ✔ |
| Transcribe text |  | ✔ |  |  |
| Parse the text |  |  |  | ✔ |
| Highlight code segment |  |  |  | ✔ |
| Delete code segment |  |  |  | ✔ |
| Code auto completions |  |  |  | ✔ |
| Run web page in browser |  |  |  | ✔ |
| Error and alarm reporting |  |  | ✔ |  |

Table 5: Traceability tests

## 4.3 Precautions

**Anomalous conditions**

There are two forms of anomalous conditions that can occur that is;

In situations when there is no active Internet connection and the user wants to issue voice commands that would never be recognized.

In cases where the system user is not audible and their accent is not accurately recognized to the default language set to the web speech recognized language as the British English.

**Precautionary steps taken**

Informing the system users about the necessity of an active Internet connection prior to using the system to issue commands in the user manual.

# Chapter 5: Installation and system acceptance test

## 5.1 Input files

## 5.2 Supplementary files

There are no any supplementary files that are required for the system to run.

## 5.3 Installed components

The system files are submitted to the node webkit runtime environment to create an executable (.npm file) that is installed on to the computer.

## 5.4 Installed qualification

Listing of the installation and system acceptance test

|  |  |
| --- | --- |
| **Topics** | **Installation Summary** |
| Installation method | ☐ Automatic – Installation kit located on the installation media.   Manual – Copy and paste from the installation media. |
| Installation media | ☐ Diskette.  ☐ CD-ROM.   Source disk folder.  ☐ Download from internet |
| Installed files | 1. HTML files 2. Javascript files 3. CSS files 4. NPM files |

Table : Listing of the Installation and System Acceptance Test

Installation Procedure Check

|  |  |  |
| --- | --- | --- |
| **Topics** | **Installation procedure** | **Date/Initials** |
| Authorization | Everyone that has a computer with specifications in the system integration requirements, an up to date browser, and an active internet connection can install the application and run. |  |
| Installation test |  Tested and approved in a test environment   Tested and approved in actual environment   Completely tested according to test plan |  |

Table : Installation Procedure Check

System acceptance test

|  |  |  |
| --- | --- | --- |
| **Topics** | **System acceptance test** | **Date/Initials** |
| Test environment |  The actual operating environment  ☐ A true copy of the actual environment  ☐ External environment |  |
| Test performance |  Installation and version  ☐ Startup and shutdown  ☐ Selected or critical requirement   Selected input   Selected outputs   Selected functionality   Performance vs. user guides  Comment: Quick responses |  |
| User level test |  Tested on operator user level  ☐ Tested on super-user level   Tested on system administrator level  ☐ Tested on overall system manager level  ☐ Education and training documented   System user manual available |  |
| Result of testing |  Testing approved  Comment: Users perform tasks effectively |  |

Table : System Acceptance Test

# Chapter 6: Conclusion, Recommendations and Future Works

**Conclusion.**

We have learnt to greatly view voice recognition as a new and easy input method for applications on top of insertions through peripherals such as keyboard, and mouse.

**Recommendation.**

As a new input method that has registered tremendous acceptance reports from its users, voice recognition should be embraced and included in the practicals carried out in lecture settings. It shall open up the students’ eyes to a new input method to enrich users choices and also help support people that cannot type. Hence, more students shall then build applications/systems that support every person inclusive of those that have physical impairments and hence cannot use their hands to type.

Encourage enrollment of people with disabilities to study IT related courses specifically programming by developing more tools that enable to issue voice commands to write programs, type notes, emails, coursework, among others.

Encourage organisations that care for people with disabilities to embrace and advocate for usage and development of assistive technologies such as XCoder to enable full inclusion of people with disabilities into executing activities like other people do.

**Future works.**

We hope to diversify the project scope to cover more language and software programs for example mobile applications (android using Java, Windows applications using C#, and IOS applications), desktop application etcetera.

We look forward to exhaust the commands required to program standard web applications from fewer HTML language tags to all tags supported by the language, inclusion of CSS, bootstrap, adding external libraries.

# 7. Appendices

## 7. 1 Appendix A: User Manual

**Introduction**

The XCoder application is a system developed to enable the physically challenged people especially those that are to able to use the keyboard to write computer programs (HTML programs) by issuing voice commands that are then transcribed following the HTML language syntax.

**Getting started**

Make sure that your computer has an active Internet connection, then load the application from the computer’s program files and execute it. The server starts shortly after executing the application, then let the user wait until when the XCoder development environment launches. After it has fully launched, the user can then start issuing commands to create a new file to work with, save it and start issuing commands to write code in the open file. Commands are taken a chronological order that is (1) issue command, (2) clear the previous command, (3) then issue next command.

A summary of the recognizable commands is documented in the table below showing their classifications that is commands for:

**Creating and manipulating a file**

|  |  |  |
| --- | --- | --- |
| **Activity** | **Command** | **Description** |
| Creating a file | Create new file |  |
| Opening a file | Open file “*filename*” |  |
| Saving a file | Save file “*filename*” |  |
| Saving code progress to a file | Save progress to “*filename*” |  |

Table 9: Commands for creating and manipulating a file

**Setting preferences**

|  |  |  |
| --- | --- | --- |
| **Activity** | **Command** | **Description** |
| Bot voice preference | Voice *“1 or 2”* |  |
| Bot volume preference | Volume “*1 … 10*” |  |
| Bot rate preference | Rating “*1 … 10*” |  |
| Bot pitch | Intensity “*1 … 10*” |  |
| Bot theme preference | Theme “*Default or Light*” |  |

Table 10: Commands for setting preferences

**Writing a Program**

|  |  |  |
| --- | --- | --- |
| **Activity** | **Command** | **Description** |
| Issuing coding commands | Coding Meta  Coding Link  Coding h1, h2, h3  Coding Nav  Coding UL  Coding Li  Coding a  Coding Button  Coding Span  Coding Image  Coding Title  Coding DIV  Coding Article  Coding Section  Coding Input  Coding Footer  Coding Paragraph  Coding Text  Coding Line Break  Coding Address  More … |  |
| Moving a cursor | Command Cursor  “*left or right*” |  |
| Highlighting a segment of code | Command Highlight “*line*” |  |
| Deleting a segment of code | Command Delete “line” |  |
| Deleting a character backwards | Command Delete |  |
| Deleting a character forward | Command Forward |  |

Table 11: Commands for writing a program

**Running a project in the browser**

|  |  |  |
| --- | --- | --- |
| **Activity** | **Command** | **Description** |
| Running the project in the browser | Command Run Project |  |

Table 12: Commands for running a project in the browser

## 7.2 Appendix B: Budget

|  |  |  |  |
| --- | --- | --- | --- |
| **Item** | **Quantity** | **Unit Price (UGX)** | **Total Price (UGX)** |
| Supplementary Microphone | 1 | 65,000 | 65,000 |
| Stationery | Several | 80,000 | 100,000 |
| Transport |  | 60,000 | 60,000 |
| Logistics |  | 100,000 | 50,000 |
| Miscellaneous |  | 50,000 | 50,000 |
| TOTAL |  |  | 325,000 |

Table 13: Budget

## 7.3 Appendix C: QUESTIONNAIRE

The contents of the user interviews are attached at the end of the document.

## 7.4 Appendix D: Glossary

**Bot** - A piece of software designed to complete a minor but repetitive task automatically or on command.

## 