SOEN-6751 HUMAN COMPUTER INTERFACE DESIGN

Instructor Dr. Rajagopalan Jayakumar

SmartGCC Project

Team-12

Sr. No.	Name	Student ID	Email-id
1	Yash Golwala	40085663	golwalayash@gmail.com
2	Chirag Vora	40091454	chirag123192@gmail.com
3	Pooja Dhir	40104545	poojadhir1996@gmail.com
4	Harsh Divecha	40084737	harsh2826@gmail.com
5	Sanjana Udar	40094029	sanjana.udar@gmail.com
6	Jaswanth Banavathu	40080737	jaswanthbanavathu@gmail.com

Table of Contents

Table of Contents

Team-12 **1**

Objective **3**

Introduction **4**

Properties of Good Interface **4**

a. Easy to Learn and Use 4
b. Engaging 4
c. Effective 4
d. Efficient 4
e. Error Tolerant 4

Project: SmartGCC **5**

a. Easy to Learn 5
b. Engaging 6
c. Effective 7
d. Efficient 8
e. Error Tolerant 9

Technology Used **10**

Issues / Challenges faced **10**

Future Scope of this project **11**

Team Contribution for Report **12**

Team Contribution in Project **12**

Contribution Section of Each Member **13**

Bibliography **19**

Objective

This project is to design and implement a self-adjusting smart graphical interface for C++ program development using the GCC compiler (let's call it SmartGCC). Your interface is to be designed for three types of users: Novice user learning programming (need to use compiler options, linking options, execute options and debugging options), Typical Programmer (need to use code generation and code optimization options in addition to the options used by novice users) and Expert Developer (need to use the developer options in addition to the options used by typical programmers). Your Interface should have two windows, one showing the program the user is working on and the other showing the result/output from GCC, in addition to all the available options for the user. All the users will be prompted to select the appropriate user type and initializes the interface with the menus and commands for the selected user type. The interface for all types of users should also contain a menu item called "All Options" containing all the options that any user can select. Once the user selects the required options, SmartGCC should execute the program using the selected options by calling GCC through an appropriate command line and display the results. Your interface should be made self-adjusting in the sense that when a user performs a command/task from the "All Options" menu which is not included in the interface for his/her user type, the interface will include that command within appropriate menus from that time. Thus, eventually all the commands a user has used in the past will be available in the appropriate menus and can be used efficiently.

Introduction

User interface design can dramatically affect the usability and user experience of an application. The layout of a user interface design should also be clearly set out for users so that elements can be found in a logical position by the user. The interface design is a goal directed problem solving activity informed by intended use, target domains, materials cost and feasibility.

Properties of Good Interface

a. Easy to Learn and Use

Product should support both initial orientation and deep learning. It should be made in a way that it is easy to learn and use for first time users also. Focus on what is really important and do not distract the users from giving unimportant things on screen.

b. Engaging

This concerns how pleasant, satisfying, or interesting an interface is to use [3]. This means your interface can either encourage the use of the system OR can be a turn-off that makes your system painful to use [3]. Based on user type appropriate features must be kept making user's experience please and delightful. For example, expert user might not like to click many buttons for just simple execute command and would love to use short cut key allotted for that command.

c. Effective

This refers to the product's ability to complete the task in order to achieve goals set for that task. If not given proper focus on effectiveness on tasks, it can result into user's frustration and will lead to product's failure because of poor UX.

d. Efficient

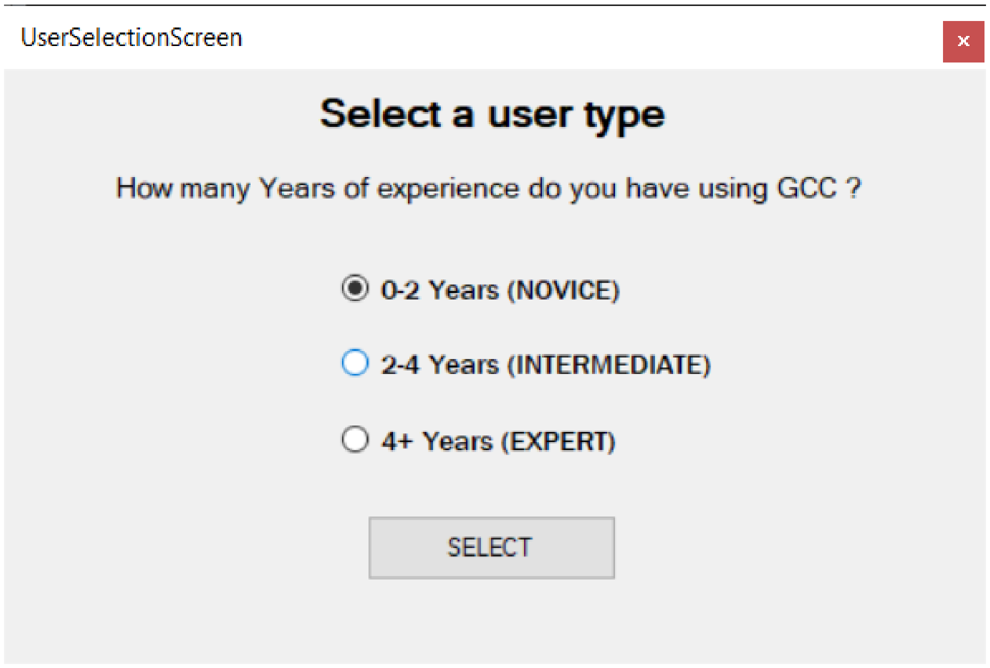
How accurately a task is completed to achieve its goals without generating false results or outputs irrespective of how old the product is. It should be maintained properly in order to make it efficient.

e. Error Tolerant

How well product can recover from an error or mistakes done. Product must be able to navigate back to its original state once a user gets stuck or selects a wrong option while using the product. This will also avoid most of the support calls made in the help window.

Project: SmartGCC

a. Easy to Learn



UserSelectionScreen

Select a user type

How many Years of experience do you have using GCC ?

[✓ 0-2 Years (NOVICE)]

2-4 Years (INTERMEDIATE)

4+ Years (EXPERT)

Figure (1)

To achieve this property in SmartGCC, conversational texts are used for making it easy for any user to use this product and can play around the UI. Precise and unbiased texts are selected for having conversing type interaction in UI as seen in figure 1. Use of good system metaphors can allow a user to learn about the application in an enhanced way without much help required for e.g. Radio Button used in the above figure depicts that user has to select only one option from the set of all available options. Also, to make it easy for the user, black bold dot is highlighted whenever he/she selects a option to avoid error as much as possible.

b. Engaging

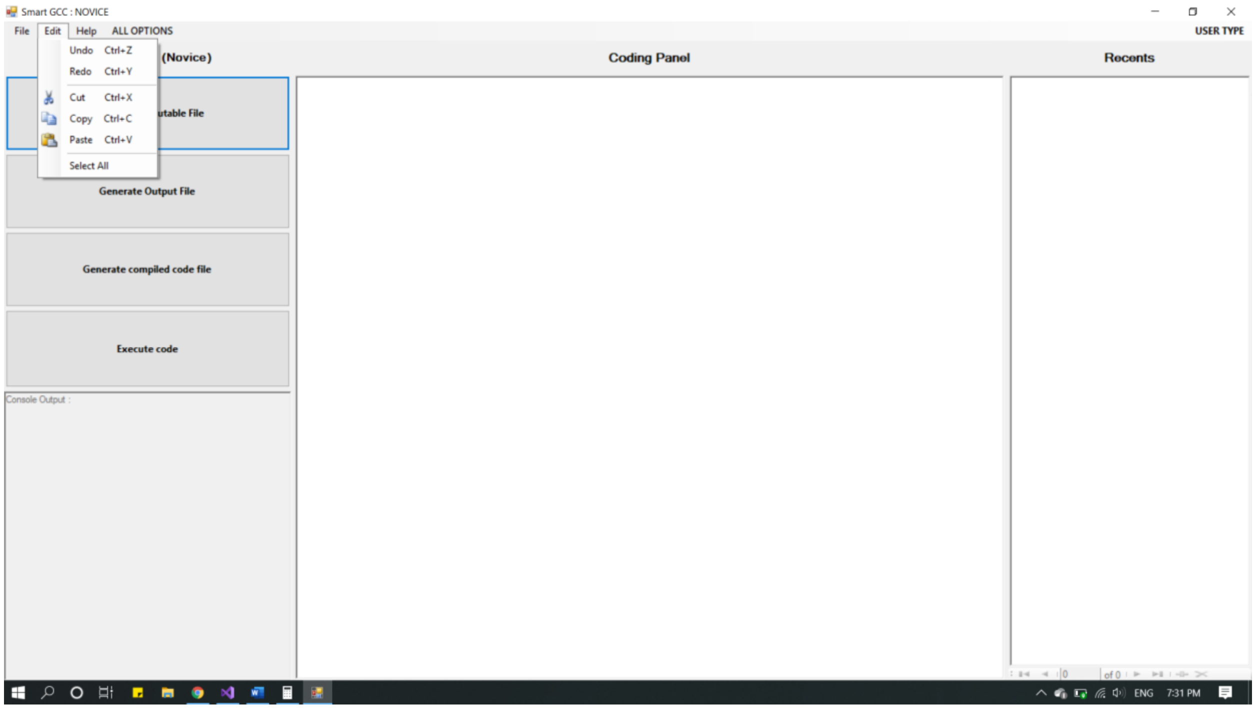
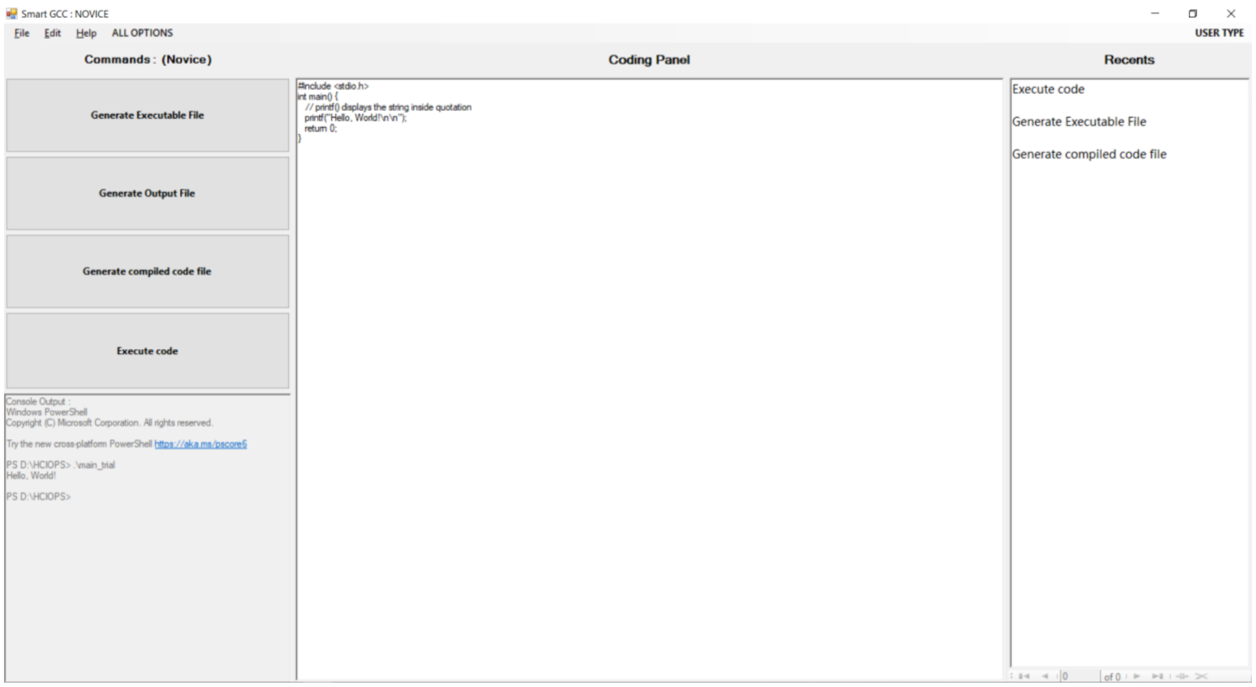


Figure (2)

Figure 2 clearly depicts the keyboard shortcuts that have been integrated into our UI. After using this app for more than once, user might expect to do few things using key board shortcuts and in order to keep him using this UI proper consistent key-board shortcuts have been integrated in the menu tool pull-down strip. Also, the key board shortcuts used are not selected at random and are kept standard which are commonly followed across all applications to enhance the user experience goal maintain the consistency.

c. Effective



```c
#include <stdio.h>
int main() {
   // printf() displays the string inside quotation
   printf("Hello, World!\n\n");
   return 0;
}
```

```
Console Output :
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS D:\HCIOPS> .\main_trial
Hello, World!

PS D:\HCIOPS>
```

Execute code

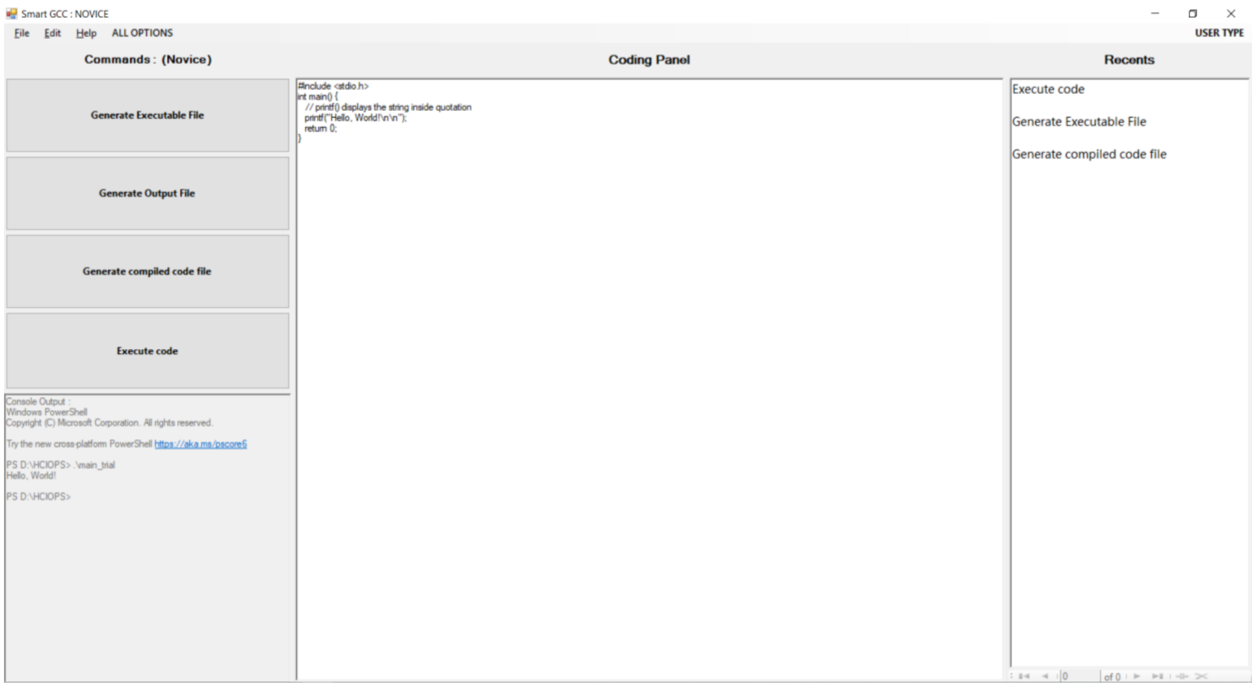
Generate Executable File

Generate compiled code file

Figure (3)

Effectiveness is evaluated on terms of how successfully a task can achieve the end goal after execution of that task. From figure 3, Task Effectiveness can be seen as user was successfully able to write the code and also is able to check the output of the code after successful execution of the command that was selected by the user from the given list of commands. Output displayed in a structured manner for not letting a user feel too new to this system and can relate it easily with previously used software.

d. Efficient



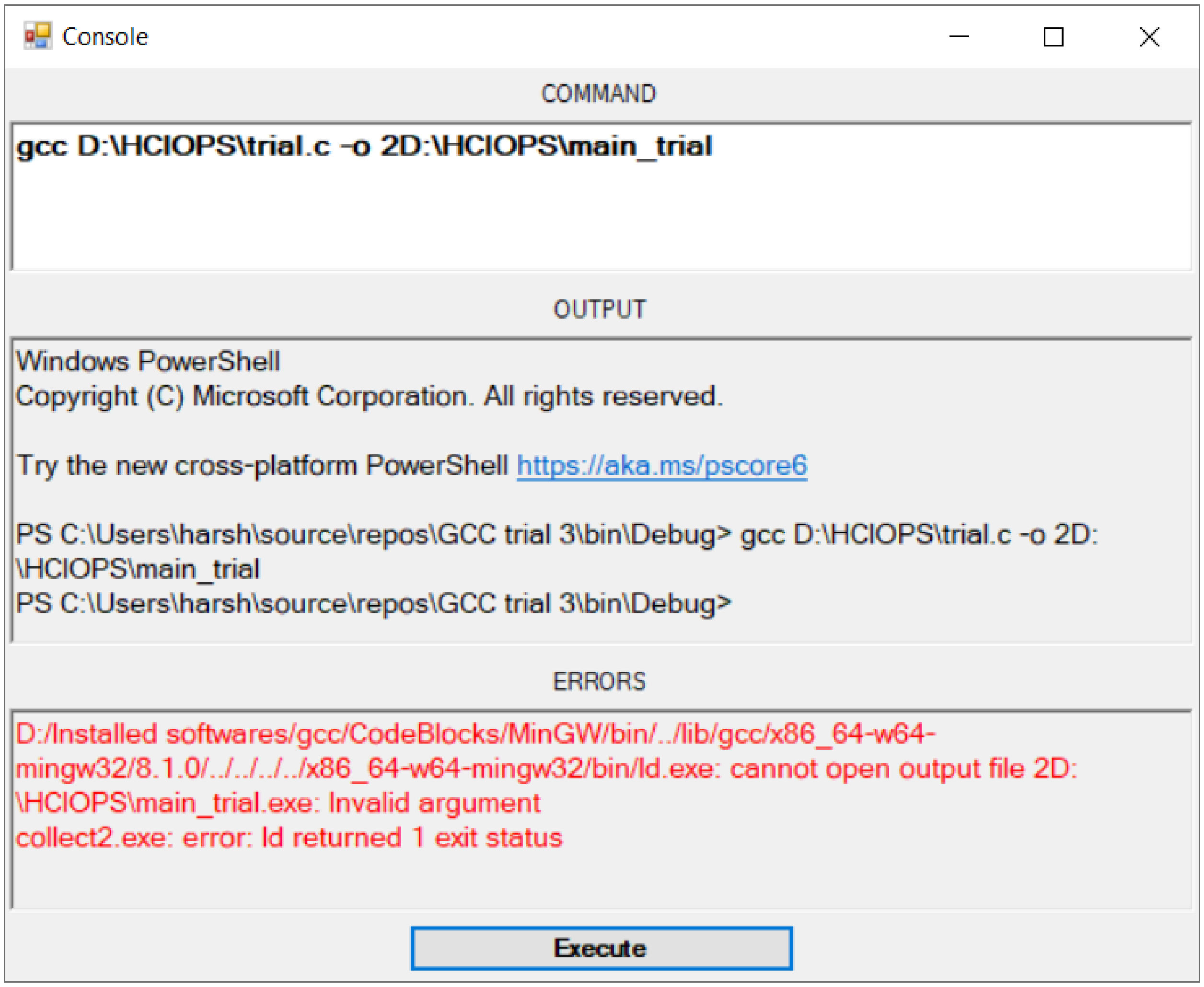
```
#include <stdio.h>
int main() {
   // printf() displays the string inside quotation
   printf("Hello, World!\n\n");
   return 0;
}
```

```
Console Output :
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS D:\HCIOPS> .\main_trial
Hello, World!

PS D:\HCIOPS>
```

Figure (4)

For having the efficiency in the task being executed, the commands integrated in the application were tested thoroughly. Upon generation of accurate results of every command, list of commands was prepared. The accuracy of the result can be seen in the output console window provided just below the list of commands on main screen. Also, without too much wrapping of commands task can be executed in no time and can display the result as efficiently as possible on any good computing system.

e. Error Tolerant



```
gcc D:\HCIOPS\trial.c -o 2D:\HCIOPS\main_trial
```

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\harsh\source\repos\GCC trial 3\bin\Debug> gcc D:\HCIOPS\trial.c -o 2D:
\HCIOPS\main_trial
PS C:\Users\harsh\source\repos\GCC trial 3\bin\Debug>
```

```
D:/Installed softwares/gcc/CodeBlocks/MinGW/bin/../lib/gcc/x86_64-w64-
mingw32/8.1.0/../../../../x86_64-w64-mingw32/bin/ld.exe: cannot open output file 2D:
\HCIOPS\main_trial.exe: Invalid argument
collect2.exe: error: ld returned 1 exit status
```

Figure (5)

Considering the human error, it is of prime importance to incorporate a functionality for getting back to a normal state after selecting a wrong option or writing down a wrong command. SmartGCC allows a user to change user-type at any given point of time from the main screen if he/she selected a wrong user type on the welcome screen. Additionally, human beings can make a mistake while typing a command and SmartGCC over comes that error in a very simple way. As seen in figure 5, user enters wrong command attributes and tries to execute it, but Errors can be seen in Error logs tab in Console pane. Also, window does not shut down and allows user to change the command again and let user execute the command again after correcting it.

Technology Used

Language Used for Implementation: **C# 8.0 language**
C# used over java swing as it was more challenging to develop a desktop application using java swing. Also, C# has a good base library targeted to Windows-based applications. In addition to this, also the group members were familiar more with C#.

Framework: **ASP.NET 3.0**

Integration of GCC: **GCC compiler extracted from Code Blocks**
To use GCC in our Windows based Desktop application, GCC compiler module was extracted from the code blocks Binary of Windows.

Application IDE: **Visual Studio 2019**
Visual Studio IDE used to create SmartGCC application using ASP.NET and C#.

Issues / Challenges faced

- ASP.NET is only supported by the windows platform.
  We tried installing it in other platforms like Linux and MacOS but there was no support available.
- Difficulty while integrating GCC with ASP.NET.
  Since there is no direct support of GCC available, we extracted the GCC module from code blocks binary.
- Difficulty while implementing keyboard shortcuts for the frequent commands used in GCC.
  While writing the program, providing the keyboard shortcuts in the drop-down menu (example- CTRL+C for copying, CTRL+V for pasting, etc.) was very challenging.
- Responsive design was a challenge.
  It was difficult to decide which widget and tools to use for making the application responsive for better UX.

Future Scope of this project

1. User Interface Adjustment: As the Artificial Intelligence technology is rapidly evolving, the smart feature which can also be called as the self-adaptive nature of the compiler can assist the user in modifying the interface. The complier can take record of facial patterns of the user and can provide better suggestions in altering the color and theme of the interface according to the state of mind of user.
2. Error Handling: The complier can reduce the number of errors with respect the user's level of understanding and can simultaneously generate suggestions while writing the code, thus reducing the time involved in debugging of the program.
3. Compact Lite Version: The complier can itself be miniatured into a smaller lite version and through online Integrated Development Environments; the complier can be made compatible even in smart phones.
4. ChatBot Integration: A chatbot can be integrated into the design of the complier and this feature can act as an intelligent virtual assistant. The chatbot will help the user in providing recommendations according the user type and his problem-solving skills.
5. Enhanced Performance: With the help of quantum computing technology, the complier can work swiftly with enhanced speed and can gather suggestions from all over the internet in context of the problem and can provide solutions.

Team Contribution for Report

Member Name	Task Allocated
Yash Golwala, Harsh Divecha	Objective, Properties of good Interface & SmartGCC
Sanjana Udar , Jaswanth Banavathu	Introduction, Future Scope of this Project
Chirag Vora, Pooja Dhir	Technology Used, Issues/ Challenges Faced

Team Contribution in Project

Member Name	Task Allocated
Yash Golwala	Making Application Adaptive and Responsive, Panel 1 of main screen UI
Sanjana Udar	Panel 2 of main screen UI, Transferring screen from one Windows frame to another
Harsh Divecha	GCC Integration, Console Operation from within the App
Chirag Vora	User Type Selection Window , Reusable Template for button functionality, custom console
Pooja Dhir	Validation methods, Transferring data from one screen to another
Jaswanth Banavathu	Open Dialog, Save Dialog, Text Selection for Interaction Type

Contribution Section of Each Member

Contribution by Yash Golwala:

I was responsible for making this application adaptive and responsive based on user's choice for achieving high User Experience goals. Initially, all screen tabs were static and was able to achieve only the task efficiency goal but was unable to achieve Engaging goal. So, to achieve that I worked on making all of the tabs responsive and adaptive in order to have user's engagement. Offering tab resizable affordance to our Interface was a challenging task but after many trial and error and seeking the solution, I came to solution to make it responsive by using split window function available in Visual Studio. If this function is applied on the correct area and with high precision, it can help a user to achieve the UX goals. After coming to this solution, I developed the Panel I of the main screen of our SmartGCC UI.

Panel I consist of list of commands initialized upon user type selection and Shell output tab integrated with UI.

Steps to achieve Responsiveness in Panel I:

- Vertical Split of the one main container of the main screen into two parts i.e. left and right.
- Horizontal Split in Left container in two parts i.e. List of commands and for the output shell.
- Then to make the parts resizable, use of percentage was made instead of one static fixed size of any tab.
- For making grid structure for the list of commands, table was used to integrate the commands in a structured fashion.
- Proper Visibility and affordance are added with precision to help user know that tabs can be resized as per the user's choice.

This process served as a template for making all tabs adaptive throughout the development of our application to achieve Usability goals and the Design Goals.

Contribution by Chirag Vora:

I was responsible for making a reusable template which will be consumed for implementing different functionalities over buttons to bring effectiveness in this application. In order to achieve that, I created a template for button click which can be used to incorporate different functionalities over multiple button being called.

Steps involved in creating this template:

- Creating a check which is nothing but a validation to check if that functionality needs to be performed before meeting certain criteria or not.
- Storing a command in a String object.
- Then, passing that command to a runCmd() function to execute that command.
- The output is stored back to an object and the command which was executed will be added to the recent section of the application using addToRecents() function.
- Then, we will be checking if the output contains error, we will show a popup using the inbuild function Show() so that the users knows what went wrong during execution.

Further, I was responsible for making User Type selection window by making it easy to learn for achieving high User Experience goals. Conversational texts are used for making it easy for any user to use this product and can play around the UI.

Steps to achieve User type selection window:

- The user type selection window consists of a table layout panel which can be used to add rows and columns to the table, edit the table or remove something from it.
- The tables were used in order to keep it aligned as the application needs to be responsive and the alignment will be distorted if the inbuilt options like dock or anchor were used without a table.
- The radio buttons are used in order to restrict the user to select only one type out of 3 user types.

Custom Console window:
As we have already achieved resizable affordance to our Interface and also the responsiveness by using split container function of visual studio, it was easier to make the custom console window for achieving UX goals.

The window contains three sections:

- Command- This section is used by the users to write their own commands if the user doesn’t want to select from the given command options. The section is resizable.
- Output- This section will generate the output of the executed program and is also resizable making the application adaptive and engaging.
- Errors- This section will give the errors if any, once the program is executed. This section is also resizable.

- Execute button- This button is responsible for executing the command if a user enters the command in the Command section of the console window.

These processes served as a template for making the application adaptive, engaging, effective and easy to learn, throughout the development of our application to achieve Usability goals and the Design Goals.

Contribution by Sanjana Udar:

I was responsible for the smooth transitioning from one Windows frame to another which was a crucial aspect for the Usability of the application. After a bunch of trial and errors which included the window being able to transition to another but an image of the previous window would get stuck on the new window with which the user could not interact, I was finally able to get to a simpler solution for the transitioning using the Show() and Hide() methods and passing the data from the previous window in the object of the new window which support data transitioning from one screen to another. This keeps the application effective, efficient and engaging.

Also, I created the UI for Panel 2 of the main screen in our Smart
GCC system.

To create Panel 2, me and my colleague did an intense research on how to achieve responsiveness.

I followed the steps listed below to create the panel:
- Vertical Split of the 2nd half of the main screen into two parts i.e. left and right.
- Then to make the parts resizable, I used percentage instead of one static fixed size of the tab.
- Proper Visibility and mental model are added to help user know that tabs can be resized as per the user's choice.

Contribution by Pooja Dhir:

My contribution was to make validation methods and transferring of data from one screen to another, to accomplish error tolerance and efficiency.

Discussing the Validation methods, we had two scenarios where user might commit errors:

1) If the user attempts to execute the code keeping the coding panel empty then an error message ought to be created to advise him that he needs to write the code in a coding panel or upload the existing file.

2) If the user is attempting to execute the code without saving what he has written in the coding panel then an error message should be produced to remind him that he has to save the content first.

In order to prevent errors and helps the user recover from mistakes, validations methods are put to achieve error tolerance.

Furthermore, I was looking for transferring the data from one screen to another.

Whenever a user changes the user type, the recent command list needs to be updated to the new screen and in no time. We used Array List for that whenever the user clicks on any command that will be added to the Array List. But the most challenging part is to execute that used commands for different screens depending on the type of user-selected at that time. So, I created a constructor for every screen and passed the Array List in that and initialized the recent commands in it. I used "For loop" for giving extra space between the commands which will be appearing in recent commands list to provide the clarity between them. So now at whatever point a user will select the user type all the commands which he used before will be appearing in his recent command window. In this way, we accomplished an efficiency of the task which assists with completing it quickly.

Contribution by Harsh Divecha:

I was responsible for integration of GCC in our UI and making console operations to work from within the application to achieve Users' Satisfaction goals.

Initial research on GCC integration:

My task was to integrate functionality into the system i.e. integration of GCC compiler. This was a major challenge for the team as the gcc project for windows as of time was on hold and there were no servers available for direct download and integration in windows. If we selected Linux systems, then then windows desktop app production was not possible. So, I had to do research on this topic and found a workaround, which was to find the compiler integration into a c++ based IDE. The solution was to install codeblocks and create an environment variable for the gcc module in codeblocks IDE tool suite.

Windows global command tool for gcc execution:

For gcc command execution I had to find a way to activate and control command line tool from within the ASP code. This task took majority of the time as the problem I faced was of same category but in different forms, that was to execute a command, get the output and close the console window without appearing in front of the user.

My task was to integrate in a way that we could:

1. Execute commands via the buttons provided for user.
2. Execute commands given as an input by the expert user through the custom console created for intermediate and expert user.

Solved it by gaining access over the command line system process as a process thread in ASP.net framework.

Contribution by Jaswanth Banavthu:

My role was to design the functionality for opening and saving a file in smart GCC compiler and to select the appropriate text for a particular feature for each specific user type. While the project was in inception phase, there were several ambiguities in selecting the appropriate technology for developing the interface as well as for this very specific functionality of file selection. I faced several challenges in restricting the file format type, in reading and writing data from file and various difficulties in making the window fluid.

With the help of my team members, and through rigorous research, we collectively learned a new technology i.e., C# and came up with the solution. We used Microsoft Visual Studio as our IDE and this IDE has various pre-installed plug in services. Though these services, we overcame all of the challenges, finally fulfilling the five essential E's and even the usability goals of an interface. I even worked with many interface interactions types to select the relevant conversation text for each user type for our compiler.

This particular feature satisfies all the five E's needed for the compiler thus increasing the overall functionality. It is very easy to learn to how to select a file and save it for every specific user type and this feature is even engaging to every user because of the embedded keyboard shortcuts in the compiler. Effectiveness is observed as the file selection is accurate and the file selection process is very efficient as the result is very definite. Finally, the compiler is error tolerant which means necessary support is provided if the user makes mistakes in selecting the file as there is restriction in file format selection. Thus, by satisfying all five essential E's, the compiler is characterized as a good interface for any specific user type and fulfills all the usability goals needed for the overall compiler.

Bibliography

[1] R. Jayakumar, "Lecture Slides Interaction Design Principles"

[2]"Pidoco - The Rapid Prototyping Tool", *Pidoco.com*, 2020. [Online]. Available:
https://pidoco.com/en/help/ux/user-interface-design

[3]"5Es of usability", *testkeis*, 2020. [Online]. Available:
https://testkeis.wordpress.com/2008/07/14/5es-of-usability/