Human Computer Interaction

Instructor: Rajagopalan Jayakumar

SOEN 6751

Assignment 3

:Team Members:

ID	Name
40071697	Bhavpreet Kaur
40080104	Subhodip Ray
40059367	Subhannita Sarcar
40088890	Nancy Goyal
40092781	Roohani Naik
40094053	Hesamedin Dadgar

Usability Testing

Usability Testing is a non-functional testing technique that provides as a measure to the ease of use by the end-users. Although it is difficult to evaluate and measure the usability of a prototype, we have measured two important aspects of each prototype - the user control and freedom of the prototype and the aestheticism and minimalism of the design. For each of these aspects, we have given a detailed description of the considerations used while gathering data, the results after collecting the data from an appropriate number of participants (a minimum of two) and further analysis and comments on the data.

Prototype 1 : Card Based Prototyping

1. User control and freedom

a. Data collection considerations

- Number of actions that later proved to be unwanted were repeated by multiple participants
- How users were able to undo an unwanted action that has already occurred
- Number of actions that later proved to be incomplete were repeated by multiple participants
- How users were able to edit an incomplete action that has already occurred
- Number of users who were able to complete the tasks successfully.

b. Testing

We collected the data by observing the reactions of each participant while running through each card of the Prototype 1. Three of our teammates served as the participants. It must be noted these participants were not the original architects of Prototype 1. User 1, User 2 and User 3 are novice, typical and expert users respectively.

Card	User 1 (novice)	User 2 (intermediate)	User 3 (Expert)
Home screen	Selects “new file”; Enters a file name in the text area and clicks on “create”	Selects “new file”; decides to go back, so clicks on “cancel”; Selects “open file” to browse the system’s file explorer in order to open the desired file	Selects “open file” to browse the system’s file explorer in order to open the desired file
The main	Starts modifying the	Starts modifying the code	Clears out everything in

editor page	code that is already written in the available template (“Hello world program”)	that is already written in the available template (“Hello world program”)	the text area and writes a code on his own
File menu	Selects “save” option that saves the written code under the filename he had earlier provided	Selects the “close” option that prompts him if he wants to save the written code. User chooses to save. He is further prompted if he wants to quit the application - user chooses to not quit and remains in the application	Selects the “new” option that prompts him to save the written code. User chooses to save. He can now browse the system’s file explorer in order to open the desired file. He chooses to “cancel” and goes back to the code he had written in the previous step
Options menu	Selects the “Change User mode” option where he is asked to choose a user type and click on “confirm”. User selects the “Typical User” option and clicks on “confirm”. Since the code is already saved, the application closes and relaunches - now with the interface for a typical user.	Selects the “generate code” options that gives him a sub-menu of options. He further selects the “if-Else if” option from the submenu and a template of the if-else if module is generated at the cursor of the text area. User did not actually need the if-else if module at that line of code, so he now has to manually cut and paste the code segment elsewhere.	Selects the “Change User mode” option where he is asked to choose a user type and click on “confirm”. However, the user now wants to go back to his default user type but sees no option for doing so.
Optimizing code under Options menu	Not available/ grayed out	The written code in the text area becomes sanitized, i.e., colour codes are implemented and indentations are maintained	
Debugging option	Not available/ grayed out	User clicks on the icon denoting a bug. A window pops up and user enters the debug option in the available text area and clicks on “Run” option	User clicks on the icon denoting a bug and a window pops up. User changes his mind and clicks on “cancel” option to go back to his previous state
Executing code	User clicks on the green arrow icon and the code written in the text area is compiled and executed. The results along with compilation errors (if any) are displayed in the command prompt window. A compilation in process can also be stopped by clicking on the stop icon.		

Adding and implementing developer option	Not available/ grayed out	User selects “add developer options” under the “Options” menu and a window pops up prompting him to enter the developer option in a text area. He changes his mind and clicks on “cancel” to close the window. He clicks on “Developer tools” and selects any of the available options to run the code.

c. Collection, Analysis, Presentation and Comments on data

The **Concurrent Think Aloud (CTA)** protocol was followed during this process since this is a task-oriented usability testing and unbiased thoughts are expressed during task performance. The data collection was done by **logging** these verbal thoughts and actions of the user in the above table. The data is analysed by **categorizing** into the following groups - Actions that later proved to be unwanted, Actions that later proved to be incomplete and Successful actions. They are further elaborated to include the problem(if any) the user encountered, how he recovered or did not recover from it (if there was any problem) and comments about the process.

Category	User	Card	Comments
Actions that later proved to be unwanted	User 2	Home screen	The user was able to click on cancel to go back to the previous step.
	User 2, User 3	File menu	The users are able to recover from their action (closing application or opening a new file) through the prompt that asks him to reconsider.
	User 3	Options menu	Problem encountered - user did not have a choice to go back. It is recommended to add a “Cancel” option in the

			“Change user type” window.
	User 3	Debugging option	The user was able to recover from their action by clicking on the “Cancel” option
Actions that later proved to be incomplete	User 2, User 3	Executing code	The users were able to stop an ongoing compilation and go back to editing the code.
Successful actions	User1, User 3	Home screen	The process was smooth as it retained familiarity with other popular applications
	User 1, User 2, User 3	Main editor page	User 1 and 2 had no complaints. User 3 would have liked the starting editor page as empty or be presented with options to layout the starting template.
	User 1	File menu, Executing code	The action was swift.
	User 1, User 2	Options menu	For user 1, the task was spontaneous and the interface was self-explanatory. However for User 3, he would have preferred to know how the “generate code” option worked before executing it.
	User 2, User 3	Optimizing code	The optimizer is able to detect tokens and follow indentation. However, it fails in advanced features such as detection of duplicate code.
	User 2	Debugging option	The user would have preferred to have a description of the debugging option

2. Aesthetic and minimalist design

a. Data collection considerations

- Amount of irrelevant data on card that reduces visibility of important information
- Were users able to comprehend the interface and its components as it is intended without additional help or information/user manual about them?
- Were users able to comprehend the interface and its components as it is intended through metaphors?
- Consistency and standardisation of metaphors throughout system

b. Testing

Card	User 1 (novice)	User 2 (intermediate)	User 3 (Expert)
Home screen	This task and its included sub-tasks (“new file”, “open file”, “help”) took almost no time and was processed with maximum efficiency for all users.		
The main editor page	Users take more time than other users to understand the interface, such as the area to write and edit code.	The users do not have to spend time in understanding the interface and start working/editing the code immediately.	
File menu	This task and its sub-tasks (“new”, “open”, “save”, “help”, “close”) take some time as there are subtasks involved which further prompt the user to reconsider or ask them for further instructions (such as selecting a file to open).		
Options menu	User completes the task of “choosing user type” effortlessly without any need for further instructions. However, he would have preferred a description of what entails with each user type before choosing one. This is the reason he may have chosen an incorrect user type.	This task and its included sub-tasks (“Generate code”, “Optimize code”, “Change user mode”) took less time for these users as compared to user 1.	
Optimizing code under Options menu	Not available/ grayed out	The written code in the text area becomes sanitized, i.e., colour codes are implemented and indentations are maintained. The task took almost no time and the option names are self-explanatory of how the result would look like.	
Debugging option	Not available/ grayed out	User clicks on the icon denoting a bug. At this point, he would have preferred a more detailed description of what is required of him, instead of simply asking for a debug option.	Since this user is familiar with expert concepts, he does not need more descriptions of the interface and likes the minimalist design.
Executing	This task requires almost no time as the execution starts as soon as the users click		

code	on the green arrow icon which is a perfect metaphor for execution.	
Adding and implementing developer option	Not available/ grayed out	User would have preferred to add more options and customizations in the “add developer options” section which is common in most sophisticated editors.

c. Collection, Analysis, Presentation and Comments on data

Category	User	Card	Comments
Actions where users failed to comprehend interface and metaphors due to Incomplete design	User 1	Main editor page	The user would have preferred tooltips to show how to start programming instead of letting him figure out himself.
	User 2	Debugging	The user would have preferred more descriptions of what is required of him
Actions where users failed to comprehend interface and metaphors due to insufficient help	User 1	Options menu	User would have preferred a description of each user type in order to be able to make the accurate choice.
	User 3	Developer options	User would have preferred more options and customization. The current interface seemed too simple.
Successful actions	User 1, User 2, User 3	Home Screen, File menu, Executing code	The interface was self-explanatory
	User 2, User3	Main editor page, Options menu, Optimizing code	The interface seemed to be sufficiently minimalistic with exactly the options to satisfy requirements

	User 3	Debugging option	The interface retained similarity with popular editors, and therefore, was satisfactory.

Prototype 2 : Sketching

1. User control and freedom

a. Data collection considerations

i. Ease of navigating around the application
ii. Ease in performing desired actions
iii. Ability to undo an unwanted action
iv. Stop an action triggered accidentally
v. Restoration of previous state in case of undo after undesired action

b. Testing

The prototype was tested by three of the group members acting as a novice, intermediate and expert developers

Card	User 1 (novice)	User 2 (intermediate)	User 3 (Expert)
User type	Chooses novice user type. The user was able to choose with no confusion	Chooses intermediate user type. The user was able to choose with no confusion	Choose expert user type. The user was able to choose with no confusion.
Create or open file	Decides to create new file in default location	Decides to create a new file on a location of his choice	Decides to open an existing file by browsing from his storage. Browses the file but decides to create and cancels the browsing window
Home screen	User sees a screen loaded with basic options	User sees a screen with basic + intermediate user options like optimization or generate code	User gets a screen containing basic+ intermediate + developer options

Save file	Clicks on the save button to save the file	Clicks on the button to save the file. Wanted to save a file in a new location but could not.	Uses keyboard shortcut Ctrl + S
Change user type	The user clicks on the change user button. A new window pops up. Selects Intermediate user radio button and clicks ok.	The user clicks on the change user button. A new window pops up. Selects the novice user radio button accidentally and clicks ok. He clicks the change user button again and chooses an expert user.	The user clicks on the change user button. A new window pops up. Changes his mind but cannot close the pop up. So selects the expert user radio button again and clicks ok.
Generate code snippets	The user wants to generate code snippets, but does not have an option. He selects generate code from all options to apply it to his profile	The user adds code snippets by selecting the option	The user adds code snippets by selecting the option but wants to use another. Uses Ctrl + Z for undo and chooses another option
Optimize code	User selects the optimize code option from all options and adds to his main screen. Then clicks on the options to format his code	The user clicks on the optimize button to format his code but does not like the result. Undo the action by using keyboard shortcut. The previous state is restored.	The user clicks on the optimize button to format his code.
Developer options	User adds developer options from the all options panel to his screen. Then selects it to use it during compilation		User selects it from main screen before compilation

c. Collection, Analysis, Presentation and Comments on data

Retrospective Think Aloud (RTA) protocol was implemented for this usability evaluation. This protocol was modified in the way that rather than recording the users on tape, they would have to verbalize their experience afterwards when asked about each functionality/screen one by one. This protocol would be useful and valid for this application because they would have to test only a few things and it would be easy to recall. Moreover, the users would not be under pressure to react at the same time of testing and would perform at the same pace as they usually would.

Category	User	Card	Comments
Successful Actions	User 1,2,3	User type	All users were able to successfully select a user type and proceed to the main screen
	User 1,3	Create or open file	Were successfully able to create a new file
	User 2	Create or open file	Was able to open a pre-existing file from his own location.
	User 1,2,3	Home screen	The screen looks met their expectations
	User 1,3	Save file	User 3 was expecting a save-as option to rename his file
	User 1,2,3	Change user type	User 3 found the cancel option missing in case he does not want to change type. He figured out that applying the same type again would give the same result.
	User 1,3	Optimize code	User 1 would like a tooltip that says what optimize code would do
Actions that did not produce desired results	User 2	Save file	User wanted to save on a new location but there was no prompt for the new location.
	User 2	Change user type	User was not asked if he was sure of changing the user type.
Missing actions that were needed	User 3	Rename file	Wants to rename a file
	User 2,3	Undo and redo button	While the users were comfortable using keyboard shortcuts, he suggested adding buttons too. User 2 used a keyboard shortcut to restore the previous state after optimization and User 3 wanted to undo his generated code.

2. Aesthetic and minimalist design
   a. Data collection considerations
      i. Is there any irrelevant or unneeded data on screen that reduces visibility?
      ii. Enough information to understand how to perform tasks
      iii. Consistent and common metaphors throughout the application
      iv. Are the metaphors used understandable and efficient?
   b. Testing

Card	User 1 (novice)	User 2 (intermediate)	User 3 (Expert)
User type	The user chose the novice type using the bar representation given besides it	The user wanted to choose expert user type but as he was not sure of his skill level and if he would be able to change afterwards, he chose intermediate	The user chose expert user by rating himself but wanted to know what extra features he would get compared to other types
Create or open file	The user was a bit confused at the start about the use of the pop up window. Would like a clearer representation.	The user successfully browsed and selected a location to create a file	The user interpreted from the note that it has to create or open a new file. He opened a file from a location of his choice.
Home screen	The user found the screen it satisfactory	The user found the home screen satisfactory. Though he would like the button on the icons more visible the font	The user found the home screen cluttered and would like some of the options he does not use often to have less visibility
Save file	The user was able to identify the save button due to its color	The user was able to identify the save button due to its icon	The user was able to identify the save button due to its color
Change user type	The user chose the novice type using the bar representation given besides it	The user wanted to choose the expert user type but he could not find out what extra features would that user type have.	The user chose an expert user by rating himself.

Generate code snippets	The user was surprised and happy to see the auto code generation	The user would like the values that are to be change in the default code snippet be of different color	The user would like keywords generated be represented in different color
Optimize code	The user was surprised and happy to see how code optimization works	At the start, the user was confused about how the code optimization would word. Would like a tooltip that would explain what it means	The user would like better metaphors to be used to represent code optimization.

c. Collection, Analysis, Presentation and Comments on data

Category	**User**	**Card**	**Comment**
Successful design	User 1	User type	Would have like more color
	User 2,3	Save file	User 3 would like different windows for creating a file and open an existing file
	User 1,2	Home screen	User 1 is satisfied, user 2 would like bigger icons
	User 1,2,3	Save file	Satisfied
	User 1	Generate code snippets	Satisfied with how it works
Incomplete design	User 2,3	User type	User 2 would like a note or help button to know if he can change user type afterwards. User 3 would like more details explaining the action.
	User 1	Save file	User was a bit confused at first. Would prefer having different interactions for both tasks.
	User 3	Home screen	The user found it cluttered
	User 2,3	Change user	User 2 would like a note or help

		type	button to know if he can change user type afterwards. User 3 would like more details explaining the action.
Inefficient metaphors	User 2	Home screen	Bigger icons
	User 2,3	Generate code snippets	Different colors would be better to represent keywords and variables
	User 2,3	Code optimize	Confusing on how it would work. A tooltip might help in understanding.

Prototype 3 : Scenario and Storyboard

1. User control and freedom

a. Data collection considerations

- Choosing the correct type of usr type to begin with the features provided for each user type.
- Recovery after committing an error.
- Number and type of errors committed by users while using the SmartGCC application.
- Time taken to compile the user code and display results.
- Options and tasks that the user was able to perform successfully.
- Common mistakes and issues repeated by all 3 types of users.

b. Testing

The data was collected while observing the user try and use the prototype, in the presence of all the group members. The test was carried out by 3 group members, who were not the architects for this prototype, where they role played the 3 levels of expertise.

Card	User 1 (novice)	User 2 (intermediate)	User 3 (Expert)
Welcome screen	Read the Instructions and move to sign up for the prototype.	Read the Instructions and move to sign up for the prototype.	Skim the Instructions and move to sign up for the prototype.
Signup Page	Enter the basic information and select novice user type.	Enter the basic information and select the intermediate user type.	Enter the basic information and select the expert user type.

Home Page	Analyse all the options that are available in this skill level, i.e. novice. The top header shows all the options available to this user and also gives an option to change skill level. It also gives an option to add features that are not a part of this skill level.	Analyse all the options that are available in this skill level, i.e. intermediate. The top header shows all the options available to this user and also gives an option to change skill level. It also gives an option to add features that are not a part of this skill level.	Analyse all the options that are available in this skill level, i.e. expert. The top header shows all the options available to this user and also gives an option to change skill level. It also gives an option to add features that are not a part of this skill level.
Top Header menu	Selects the “Change User mode” option where he is asked to choose a user type and click on “confirm”. User selects the “Typical User” option and clicks on “confirm”. Since the code is already saved, the application closes and relaunches - now with the interface for a typical user.	Selects the “generate code” options that gives him a sub-menu of options. He further selects the “if-Else if” option from the submenu and a template of the if-else if module is generated at the cursor of the text area. User did not actually need the if-else if module at that line of code, so he now has to manually cut and paste the code segment elsewhere.	Selects the “Change User mode” option where he is asked to choose a user type and click on “confirm”. However, the user now wants to go back to his default user type but sees no option for doing so.
Optimizing code under Top Header menu	Not available	The written code in the text area becomes optimized, i.e., colour codes are implemented and indentations are adjusted for increased readability.	
Debugging option under Top Header	Not available	User clicks on the icon denoting a bug. A window pops up and user enters the debug option in the available text area and clicks on “Run” option	User clicks on the icon denoting a bug and a window pops up. User changes his mind and clicks on “cancel” option to go back to his previous state
Executing code	User clicks on the green icon of ‘RUN’ and the code written in the text area is compiled and executed. The results along with compilation errors (if any) are displayed in the side window for results. A compilation in process can also be stopped by clicking on the red stop icon.		
Adding and implementing developer option	Not available		User selects “add developer options” under the “Options” menu and a window pops up

		prompting him to enter the developer option in a text area. He changes his mind and clicks on “cancel” to close the window. He clicks on “Developer tools” and selects any of the available options to run the code.

c. Collection, Analysis, Presentation and Comments on data

Category	User	Card	Comments
Actions that later proved to be unwanted	User 1, User 2, User 3	Welcome screen	The Welcome Page aws not needed. The information about SmartGCC could have been given on Signup page or once the user logs in. It just has an extra click.
	User 1, User 2, User 3	File menu	The users are able to recover from their action (closing application or opening a new file) through the prompt that asks him to reconsider.
	User 3	Options menu	Problem encountered - user did not have a choice to go back. It is recommended to add a “Cancel” option in the “Change user type” window.
	User 3	Debugging option	The user was able to recover from their action by clicking on the “Cancel” option
Actions that later proved to be incomplete	User 2, User 3	Executing code	The users were able to stop an ongoing compilation and go back to editing the code.
Successful actions	User1, User 3	Home screen	The process was smooth and easy to understand.
	User 1, User 2, User 3	Editing Window on Home Page.	User 1 and 2 had no complaints. User 3 would have liked the starting editor page as empty or be presented with options to layout the starting template.

	User 1	File menu, Executing code	The action was swift.
	User 1, User 2	Options menu	For user 1, the task was spontaneous and the interface was self-explanatory. However for User 3, he would have preferred to know how the “generate code” option worked before executing it.
	User 2, User 3	Optimizing code	The optimizer is able to detect tokens and follow indentation. However, it fails in advanced features such as detection of duplicate code.
	User 2	Debugging option	The user would have preferred to have a description of the debugging option

2. Aesthetic and minimalist design

a. Data collection considerations

- Were users able to understand the interface and its component with their intended meaning ?
- Time to complete a task ?
- Number of users who were able to complete tasks successfully
- Was the user task completion according to the requirements mentioned in the requirement analysis phase ?
- Number and type of errors committed by the user.

b. Testing

Card	User 1 (novice)	User 2 (intermediate)	User 3 (Expert)
Welcome screen	It gives information about the SmartGCC, and if the user is interested in exploring the platform , it gives a green arrow button to continue.		
Home Screen	Users take more time than other users to understand the interface, such as the area to write and edit code.	The users do not have to spend time in understanding the interface and start working/editing the code immediately.	
File menu	This task and its sub-tasks (“new”, “open”, “save”, “help”, “close”) take some time as there are subtasks involved which further prompt the user to reconsider or ask them for further instructions (such as selecting a file to open).		

Options menu	User completes the task of “choosing user type” effortlessly without any need for further instructions. However, he would have preferred a description of what entails with each user type before choosing one. This is the reason he may have chosen an incorrect user type.	This task and its included sub-tasks (“Generate code”, “Optimize code”, “Change user mode”) took less time for these users as compared to user 1.	
Optimizing code under Options menu	Not available	The written code in the text area becomes optimized, i.e., colour codes are implemented and indentations are maintained. The task took almost no time and the option names are self-explanatory of how the result would look like.	
Debugging option	Not available	User clicks on the icon denoting a bug. At this point, he would have preferred a more detailed description of what is required of him, instead of simply asking for a debug option.	Since this user is familiar with expert concepts, he does not need more descriptions of the interface and likes the minimalist design.
Executing code	This task requires almost no time as the execution starts as soon as the users click on the green arrow icon which is a perfect metaphor for execution.		
Adding and implementing developer option	Not available		User would have preferred to add more options and customizations in the “add developer options” section which is common in most sophisticated editors.

c. Collection, Analysis, Presentation and Comments on data

Category	User	Card	Comments

Actions where user had difficulty to comprehend the interface components as the way they were intended	User 1	Home Screen	The user would have preferred tooltips to show how to start programming instead of letting him figure out himself.
	User 2	Debugging	The user would have preferred more descriptions of what is required of him
Actions where user was not able to complete task as per requirements	User 1	Options menu	User would have preferred a description of each user type in order to be able to make the accurate choice.
	User 3	Developer options	User would have preferred more options and customization. The current interface seemed too simple.
Successful actions	User 1, User 2, User 3	Welcome Screen, File menu, Executing code	The interface was self-explanatory
	User 2, User3	Home Screen, Options menu, Optimizing code	The interface seemed to be sufficiently minimalistic with exactly the options to satisfy requirements
	User 3	Debugging option	The interface retained similarity with popular editors, and therefore, was satisfactory.

Analytical Evaluation

Prototype 1 : Card Based Prototyping

1. Consistency and Standards
    a. Data collection considerations
        - Checking the ability of the user to comprehend the interface components through gaining an understanding of previous components in this interface
        - Checking the ability of the user to comprehend the interface components through gaining an understanding of common standardised interface components
        - Interrelation between screens or ease of transition between screens
        - Ability to exit or terminate from any state

- Rate and degree of confusion in each task and the statistics of users going through the same confusion

b. Testing

Card	Inspector 1	Inspector 2
Home screen	User modes do not have any description that says what features are included in each user type	The symbol for “Open file” is not the common standardised metaphor used for this operation
The main editor page	No option to exit or terminate that is visible directly. User will have to go to File and then select Close	There is no view for the console unless the program is executed. This is a major diversion from other popular editors.
File menu	No option for user to “save as” a new file name	No keyboard shortcuts for the options under File menu. This reduces ease of transition.
Options menu	No option or indication to undo a task or restore previous state after selecting any of the sub options under Options menu	The interface is satisfactory and follows the standard set by the rest of the system
Optimizing code under Options menu	This option can include a preview feature to let the user see the difference between optimized and un-optimized code. If the code is already self-indented by the user, there seems to be no difference after selecting this option.	Colour coding of the program tokens should be built-in already and not a part of code optimization.
Debugging option	The term “debug options” is vague.	There is no option to save the debug configuration which is common in other popular editors.
Executing code	There is no option to save the run configuration which is common in other popular editors.	No indication of how to close the terminal (No “cross” sign).
Adding and implementing developer option	Confusion about what type of developer option needs to be added	Confusion about the workflow - there is no indication or tooltip to let the user know that a developer option needs to be first added and will then be made available for implementation

c. Collection, Analysis, Presentation and Comments on data

Category	Inspector	Card	Comments	Solution
Components with Inconsistencies within the system and/or as with industry standards	Inspector 2	Home screen	Lack of usage of most common metaphors	Use the appropriate metaphor
	Inspector 2	Main editor screen, Debug option	Lack of usage of most common design elements : “console” view, saving debug configuration	The right part of the screen can be used as a console (fixed). Include a dropdown menu beside debug icon and show saved debug configurations along with an option to add debug configuration
	Inspector 1, Inspector 2	File menu	Lack of usage of most common design elements : “save as”, keyboard shortcuts	Add a “save as” option under File menu that lets you save the same file again under a new filename. Include keyboard shortcuts along with their metaphors beside the corresponding options
	Inspector 1	Options menu, Executing code	Lack of usage of most common design elements : “undo” option, saving run configuration	Include an option to “undo” and “redo” under Options menu along with the arrowed metaphors. Include a dropdown menu beside run icon and show saved run configurations along with an option to add run configuration
Components that had an element of confusion	Inspector 1, Interface 2	Home screen, Debug option, Developer options	Components need more description	Include a short description of features available for each user mode as a hovering tooltip. Include more input fields while setting debug options and adding developer tools.
	Inspector 1 Inspector 2	Main editor screen Executing code	Lack of a clear termination option	Include a cross icon denoting close or cancel
	Inspector 1, Inspector 2	Optimizing code	No marked difference being presented as a result of this action	Improve the features for code optimization

Successful actions	Inspector 2	Options menu	Interface is satisfactory	No improvements necessary

2. Aesthetic and minimalist design

a. Data collection considerations

- Amount of unnecessary and irrelevant information on card that reduces visibility
- Data that needs further explanation for comprehension
- How complicated tasks have been broken down into steps
- Priority of the content and tasks need to be considered.

b. Testing

Card	Inspector 1	Inspector 2
Home screen	The interface seems minimalistic enough with the most important options outlined in the beginning	User modes do not have any description that says what features are included in each user type
The main editor page	The initial template has unnecessary information for users who will write their own custom code. A better design would be simply a commented code segment.	Many options are grayed out for novice and intermediate users. These options may simply be made invisible as it is reducing visibility of the other important features by adding visual clutter.
File menu	Lack of metaphors alongside the options under File menu	Alternating colours on each option of the dropdown menu would have made it more readable, especially if the number of options under File menu grows with future releases.
Options menu	The “generate code” option must be highlighted when it is selected to open the sub options.	Most of the options under Options menu except “Change user mode” are related to the program/code itself. Therefore the name “Options” is deceptive and vague and “Change user mode” must be relocated to some other place such as right beside the top-right label of the current user mode. This would have reduced the confusion of the user who wanted to change user mode too.
Optimizing code under Options	“Optimize code” is vague as it lacks a description of what	There is no indication to the existing point in the code where optimization can

menu	features are included in code optimization	lead to improvements.
Debugging option	The debug options lacks descriptions and is too simple	There is no option to include breakpoints.
Executing code	The interface seems minimalistic yet contains detail to the sufficient degree	Lack of verbose option while executing the code which shows how the entire process is broken down into parts.
Adding and implementing developer option	Lack of verbose option while executing the developer option which shows how the entire process is broken down into parts.	No confirmation message that clearly indicated when a developer option has been added

c. Collection, Analysis, Presentation and Comments on data

Category	Inspector	Card	Comments	Solution
Components with visual clutter	Inspector 1	Main editor screen	Unnecessary starting code segment that increases work effort for users	Keep the starting code template as comments for flexibility
	Inspector 2	Main editor screen	Too many grayed out options	Remove grayed out options
	Inspector 2, Inspector 1	File menu, Options menu	Lack of asceticism	Implement alternation colours (gray and white) on the dropdown menus and highlight the menu (with blue) over which mouse is hovering. Once an option with another dropdown with sub options is selected, keep that option highlighted in blue.
	Inspector 2	Options menu	Incorrect categorisation	Rename “options” to “edit”; The option “Change user mode” should be implemented on clicking the label “User mode: Novice”
Components that need further explanation	Inspector 2	Home screen, Optimizing code, Executing	Lack of explanation of interface components and work-flow	Include tooltips to describe features available in each user mode. Once code optimization is implemented, highlight briefly (like a flashing) the lines of code that have been

		code		changed. Save the verbose log of code execution in a file.
	Inspector 1	File menu, Optimizing code, Debug options	Lack of metaphors or descriptions explaining the task	Include metaphors right beside each option. Include more input fields in the debugging window.
	Inspector 2	Debug options, developer options	Lack of essential components	Implement design elements to define debug components. Briefly flash in the footer a confirmation message saying that developer option has been implemented
	Inspector 1	Developer options	Lack of explanation of how a complicated task is broken down	Save the verbose log of code execution during the developer tool execution in a file.
Successful actions	Inspector 1	Home screen, Executing code	Interface is satisfactory	No improvements suggested

Prototype 2 : Sketching

- ### Consistency and Standards

To ensure that both the graphic elements and terminology are maintained across similar platforms. For example, an icon that represents one category or concept should not represent a different concept when used on a different screen

a. Data collection considerations
   - Checking the use of common terminology for the type of users (ex – Novice, Typical, Expert), buttons, actions (ex – run, compile, save) throughout the application
   - Checking the font of the header and other content on the screen to be consistent
   - Making sure the color of the common buttons like “OK” and “Cancel” to be the same to avoid confusion
   - Using approximately the same size for the window unless the content of the screen is a lot different.

- Checking the basic options on the screen like “Exit” button or “Tooltip” if present, should be consistent.
- Making sure consistent visual UI Elements throughout the application

b. Testing

Inspector 1

- **Screen 1 (Choosing the user type)**-
  i. Application name in the center of the window header could be placed on the left hand side of the header.
  ii. There is no Exit (cross) or cancel button to cancel the task.
  iii. There is no tooltip on the first screen as is there on the second screen
  iv. A note stating the scale description would be advised to put at the bottom of the screen
- **Screen 2 (Choosing the file location)**
  i. There is no Exit (cross) or cancel button to cancel the task.
  ii. Windows should have a title in accordance with the task performed by it.
- **Screen 3 (Main Screen)**
  i. Colour of the “open” button and “compile” button is almost similar.
  ii. Windows should have a title in accordance with the task performed by it.
  iii. There is no Exit (cross) or cancel button to cancel the task.
  iv. “All options” should be a label with colon to maintain consistency with second screen
  v. “Menu Bar” at bottom is not properly maintained with respect to colour and space.
- **Screen 4 (Change User Type)**
  i. Default user type should be selected to show the user what his/her current user type is.
  ii. Application name in the center of the window header could be placed on the left hand side of the header.
  iii. There is no Exit (cross) or cancel button to cancel the task.

Inspector 2

- **Screen 1 (Choosing the user type)**
  i. There is no option to terminate the window.
  ii. User would not know what features would they get upon choosing a user type
- **Screen 2 (Choosing the file location)**
  i. The note should be below the location rather than in the bottom of the window to be clear
- **Screen 3 (Main Screen)**
  i. Is the stop button for the run command or for the debug command?

ii. Lack of menu bar (File, Edit, View…) that contains comprehensive list of functions in the window

iii. Cluttered screen in case of expert users

- **Screen 4 (Change User Type)**

i. The screen does not show the current user type.

ii. There is no cancel or termination metaphor on the window.

b. Collection, Analysis, Presentation and Comments on data

Problem	Inspector	Comment	Solution/Comment Accepted
No Exit or cancel button in each screen	Inspector 1,2	There should be either exit or cross button to cross the window or cancel button	Cross button to terminate the window is accepted.
There is no task name of the window in the header.	Inspector 1	Application name should be cornered and the task of the window should be centered in the header	Header bars should be made common to all the screens with task names changing according to the action performed by the screen.
Screen 1 -no tooltip	Inspector 1,2	Consistency should be there to provide tooltip on every screen, if provided on one window.	Tooltip to be provided as is given on screen 2 to avoid confusion with the icons and better clarification
Lack of note providing description of the scale	Inspector 1	A note providing description of the scale with colors to distinguish between them	Scale is self explanatory with numbers marked on it. Providing a note can take unnecessary space.
Lack of description on user type	Inspector 2	A note on what the features would be included in the user type	A note would take a lot of place. Can put a “question mark” metaphor, that would show the description on a mouse hover
Confusion between “open” and “compile” button due to similar color	Inspector 1	Color of “open” and “compile” button should be different on screen 3	Could be avoided to change as the icon is totally different

Menu bar at bottom is not fitting with the rest of the UI design	Inspector 1	Needs proper spacing	“All Options” to be put in dropdown save space and the selected option to be added on the add button click.
	Inspector 2	Menu bar on the top of the window that gives all options	
No default option is selected on change user type screen	Inspector 1	Default radio button for user type should be shown selected	Current user type default option should be shown as selected
Current user type not shown on the screen	Inspector 2	The current user type should be displayed on the main screen and on change user type window	Current user type would be shown on the screen

- ## Aesthetic and minimalist design

Keep clutter to a minimum. All unnecessary information competes for the user's limited attentional resources, which could inhibit a user's memory retrieval of relevant information. Therefore, the display must be reduced to only the necessary components for the current tasks, whilst providing clearly visible and unambiguous means of navigating to other content.

a. Data collection considerations

- Remove unnecessary functionality, process steps and visual clutter.
- Make sure to show only relevant things on the main screen.
- Break down complicated processes into multiple steps and put it on a different screen.
- Priority of the content and tasks need to be considered

b. Testing

Inspector 1

- **Screen 1 (Choosing the user type)**-
  - Colour range could be provided to distinguish the scale values.
- **Screen 2 (Choosing the file location)**

    - Font size of the “note” at the bottom could be made smaller and star marked.
    - Instead of giving a “browse” button, an icon could be provided to give more space for the textbox.
- **Screen 3 (Main Screen)**
    - All options could be grouped in a dropdown and an add button to be added on the click of that button.
    - “Compile” button should be aligned before the “run” button to maintain a flow of the steps.
- **Screen 4 (Change User Type)**
    - Colour range could be provided to distinguish the scale values..
    - Instead of “OK” button, “update” button can be provided to make it clear that the user type has been updated.

Inspector 2

- **Screen 1 (Choosing the user type)**
    - Use another metaphor to describe skills rather than a bar
- **Screen 2 (Choosing the file location)**
    - Issue in the space visibility of the note. Could be put into a box that takes less space
- **Screen 3 (Main Screen)**
    - Rearrange windows and panels for the expert users as their screen might look cluttered
    - Cluttered screen in case of expert users
- **Screen 4 (Change User Type)**
    - Color coded user type and bars to make it look eye pleasing.

c. Collection, Analysis, Presentation and Comments on data

Problem	Inspector	Comment	Solution/Comment Accepted
Lack of color range to distinguish between the scale on first screen	Inspector 1,2	Color range as a legend should be provided for better clarification	To keep the minimalist design, color range can be avoided as the numbers on the scale are self explanatory.
Font size of note is too big for a note on second screen	Inspector 1	Font size of note should be according to the standards for a note(generally small)	Font size of note to be kept smaller than normal font.

Note should be put below the browse location button	Inspector 2	The note should be put below the location browse button to avoid confusion	Font size of note to be kept smaller than normal font and to be put in a box at the bottom of the window.
Browse button is taking more space on second screen	Inspector 1	Browse button can be replaced by browse icon	Browse button to be replaced by browse icon and a tooltip tip can be provided
Rearrange screen panels	Inspector 2	Rearrange screen panels for expert user	The screen panel would be fixed for all types of user because when a user updates to an expert user, they might get confused.

Prototype 3 : Scenario and Storyboard

1. Consistency and Standards

Ensuring that the symbols and colors represented in one screen should have the same signification and indication. Consistency is the most important aspect of a good user interface.

Data collection considerations

- To make sure that color usage is uniform across all screens.
- The font and size of the font should be the same across all the screens.
- The size of windows and the text orientation on all of the windows must always be the same.
- The size and color of the buttons should be the same for all windows.
- Standard buttons like save, close and exit should be according to the standard norms to avoid any confusion among users.

Testing

Inspector 1

- **Screen1 (Welcome Screen)**-
    - Name of the application in the top header of the window.
    - The body of the screen tells the user about the platform. How it is designed for users with different skill levels

    - The body of the screen also has a green arrow button that allows the user to sign up as per his/her intellect level.
    - The footer of this screen contains a video to show the demo of how the user can use the application as per his/her skill level.

- **Screen 2 (Signup Page)**-
    - Name of the application in the top header of the window.
    - The body of the screen contains a form that asks the user to fill up the basic information and the skill level, user wants to use the application with.
    - Successful completion of the form takes the user to the main screen with all the features for the specified skill level earlier.

- **Screen 3 (Home Screen)**-
    - Name of the application in the top header of the window.
    - The body of the screen is divided into three parts, with one horizontal division and one vertical division in the other half of the horizontally divided part.
    - Below the header is the first horizontal division with no vertical divisions in it. This part includes all the features for the user as per the skill level, including the file upload and change user option.
    - The other horizontal part with one vertical division has its left part for writing the code with a save and run button on top right corner of it. The other half is for displaying the result of executed code.
    - The windows are adjustable in size as per the user requirement, i.e. user can widen or shorten this three windows on the home screen

- **Screen 4 (Choosing the file location)**-
    - There is no Exit (cross) or cancel button to cancel the task.
    - Windows should have a title in accordance with the task performed by it.

- **Screen 5 (Change user type)**-
    - Default user type should be selected to show the user what his/her current user type is.
    - There is no Exit (cross) or cancel button to cancel the task.
      Collection, Analysis, Presentation and Comments on data

Inspector 2

- **Screen 1 (Choosing the user type)**-
    - The cancel button should be used
    - The level of skill and corresponding features should be shown to the user.
- **Screen 2 (Sign up page)**-

    - The file location identifier could be more user friendly as helping the user to locate files with correct extension, rather than telling it later that file can not be loaded.
- **Screen 3 (Home screen)-**
    - Screen is quite cluttered.
- **Screen 4 (Change User Type)**
    - The screen does not show the current user type.
    - There is no cancel button on the window.

Collection, Analysis, Presentation and Comments on data

Problem	Inspector	Comment	Solution
Welcome Screen	Inspector 1, Inspector 2	Welcome Screen is not needed	Welcome screen is an extra screen, though it is important to tell the user about the platform but that can be done while first time using the platform.
Signup Page	Inspector 1, Inspector 2	Signup Page need not to be that elaborate	The Signup Page should not include all that question. Just a simple login id and skill level are enough.
Home Screen	Inspector 1	Windows should be more adjustable for user choice.	Users should get to decide how to organize or place the windows on the screen.
Home Screen	Inspector 2	Quire cluttered	More space and less windows on one screen could be helpful.
File Menu/ Change User	Inspector 1, Inspector 2	Not present	Should be added to the windows.

2. Aesthetic and minimalist design

Including all the important aesthetic aspects of the user interface design and avoiding all the unnecessary clutter to make a meaningful application that is easy to understand and use.

Data collection considerations

- To make sure that all the clutter is removed.
- Unnecessary steps should be avoided.
- To maintain color and button symmetry across all the windows.
- All the important features should be accessible from the home screen window.

- Steps should be simple and easy to learn suiting all kinds of users.

Testing

Inspector 1

- **Screen 1** (**Signup Page**)
  - The signup page is not needed, can start without it.
- **Screen 2 (Home Screen)**
  - Screen could be more colorful and easy to use.
- **Screen 3 (Changing the user type)-**
  - Using it is kind of complicated as it loads the application again.

Inspector 2

- **Screen 1 (Signup Page)-**
  - The signup page is not actually needed, only a unique email d is enough along with the skill level.
- **Screen 2** (**Choosing the file location**)
  - The drag-drop of the file is a far more easy option to use, so rather than having to select the file from the menu, so the user should actually be able to drag and drop the file.
- **Screen 3 (Home Screen)**
  - It is kind of cluttered, could possibly decrease the clutter and make separate windows, so that the platform does not scare the users
- **Screen 4 (Changing the user type)-**
  - Rather than having to change the user type from a different window, the user should be given a checkbox option on the home screen, so that whenever the user tries to change the user type he/she can just uncheck the current type and check the other one.

Collection, Analysis, Presentation and Comments on data

Problem	Inspector	Comment	Solution
Signup Page	Inspector 1, Inspector 2	Unnecessary Questions	Only unique email id and skill level are enough for sign up.
Home Screen	Inspector 1, Inspector 2	Cluttered	Adjustable windows would be helpful for the user to organize his/her workspace.
File Menu	Inspector 2	File locator is a bit cluttered and extra effort.	Drag and Drop option should be included

Change User Menu	Inspector 1, Inspector 2	Changing user takes extra time as application load once again	Checkbox for each user type should be given on the home screen.

Comparison between usability testing and analytical evaluation of Aesthetic and Minimalist design

	Usability Testing	**Analytical Testing**
Similarities	Both techniques are used to measure or evaluate user experience derived from the product, i.e. visual clutter, consistency in design throughout the system and prioritizing components.	
	Both techniques are useful in detecting usability problems such as use of inefficient and non standard metaphors to support actions	
Differences	Performed by non-professional users with limited experience in UI/UX in a real-world setting. This helps in revealing real problems that might be faced by other users too. For example, it is seen that an user may make an incorrect choice of user type if descriptions or a brief illustration is not given about each user type.	Performed by professionals with experience in heuristic evaluation. This reveals potential problems in the design. For example, inspectors have suggested removing the grayed out options for more minimalist design as they foresee this as a likely problem in future releases where the number of grayed out options may increase.
	Users have specific tasks to perform and go through the design according to the work-flow established by the prototyping technique (card by card or going through the scenario). They would focus only on evaluating the aesthetic value of elements provided on the screen. Eg. The users may not keep notice the varying size of the screens	Inspectors have an orthodox list of tasks to follow that is used to look at the system holistically. They can provide suggestions for the improvement or addition to the existing design. Eg. As inspectors check standards, they would check whether the window size is appropriate, the size of pop ups is similar etc.

Choice of final prototype and justification

The **Prototype 1** has been selected as the final prototype and a base conceptual model to build on by borrowing features from Prototype 2.

Features borrowed from Prototype 2 -

- The welcome screen denoting the skill level of each user type. This gives the user more information that aids in the accurate choice of user type. Therefore, the final welcome screen should include that of Prototype 2 in place of the top three boxes in Prototype 1. Additionally, the options of opening, creating new files and the help section should remain as it is in Prototype 1 as it prioritized the most important components.
- The main editor page should have a left section for the code and a fixed right section for the terminal to display results and errors as in Prototype 2. This removes the need for including a “close” icon to close the terminal during program execution and implements better ease of transition.
- The “Change user mode” should be implemented when the user clicks on the user label in the top right corner. However, the screen shown as a result of this should be the one illustrated under Prototype 2.

Features improved after Usability and Analytical evaluation -

- Include metaphors for the options under File menu and Options menu for enhanced user experience
- Make the grayed out options invisible to reduce visible clutter
- Include an “undo” and “redo” options with metaphors under Options menu for better user control
- Rename the “Options” menu to “Edit” to maintain consistency with the existing editor and better define the options. This would also necessitate relocating “Change User mode” to another place in the interface as it does not fall in this category
- Highlight the option when mouse hovers over it

Responsibility and task division in team

Module	**Team member**
Usability Testing Prototype 1	**Novice**:External User, **Intermediate**:Nancy Goyal, **Expert**: Roohani Naik
Analytical Evaluation Prototype 1	Hesamedin Dadgar, Bhavpreet Kaur
Usability Testing Prototype 2	**Novice**:External User, **Intermediate**: Hesamedin Dadgar, **Expert**: Bhavpreet Kaur
Analytical Evaluation Prototype 2	Subhannita Sarcar, Subhodip Ray
Usability Testing Prototype 3	**Novice**:External User, **Intermediate**:Subhannita Sarcar, **Expert**:Subhodip Ray
Analytical Evaluation Prototype 3	Nancy Goyal, Roohani Naik