

Human Computer Interaction



Instructor: Rajagopalan Jayakumar

SOEN 6751

Assignment 1

:Team Members:

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

Table Of Contents

1. Purpose of the system:	2
2. Scope of the system:	2
3. Requirement Gathering Technique:	3
4. Non - Functional requirements:	4
5. Functional requirements:	5
6. User Personas and their user characteristics:	5
7. Task Characteristics using use case diagrams:	8
8. Task Characteristics using Hierarchical task model:	14
9. Narrative UI Scenarios	15

1. Purpose of the system:

The SmartGCC is a smart graphical interface for program development in C++. It is designed for mainly three types of users: Novice programmers, Typical programmers and Expert programmers. It lets the users write, edit, compile, execute and facilitates other operations on its code.

The goal of this document is to define software requirements according to specifications that can fulfil the need of these users.

2. Scope of the system:

In the interface, depending on the user type, the programmers can perform different tasks. A novice programmer can write, compile, execute and debug his code. A typical programmer can not only perform the novice's tasks, but can also optimize and generate code. In addition to the novice's and typical programmer's tasks, the expert developer can use developer options and save his used options for future.

Selecting The StakeHolders:

Stakeholder characteristics:

The following characteristics are used to select stakeholders:

- Name and age of stakeholder
- Education level
- Programming expertise
- Frequency of programming
- List of features most widely used while programming

Primary Stakeholders of Smart GCC (benefit directly):

- High School Students (Novice): They would be using the product for programming tasks in C++ with the expertise that is relevant in their population.

- University Students (Typical): Students who have preliminary knowledge of programming languages, especially in C++, with a few academic projects under their belt.
- Developers of the Product (Expert): They are the direct users of Smart GCC. They would like to add features in the product which will help them code efficiently. They have the technical expertise necessary to advise executives on which features are feasible and how long each would take to build.

Secondary Stakeholders of Smart GCC(rely on the product) :

- Managers: They make the final decision about the timeline, budget, and scope.
- Business Analyst: They gather the requirements from the client, analyze and negotiate with them and finally help the developers in building the product of the client's expectation.

3. Requirement Gathering Technique:

The ***existing literature of the GCC manual*** was analyzed to recognise basic functionalities and features. A similarity with these features is being strived to maintain.

Additionally, ***interviews*** were held with people fitting with the persona descriptions to realize their needs and the kind of interfaces that they would prefer for highest user experience. The questions asked in the interview included an exhaustive list of features and functionalities as well as a confirmation of the workflow. This work flow is further formulated in the use cases later.

4. Non - Functional requirements:

- **Performance Efficiency** : The system is responsive in it's behaviour. Response comes within 1.0s It also proves to be stable under various conditions.
- **Usability**: The interface of the application is built following the 5 E's of User Experience, the interface also is easy to understand and is therefore easy to use, ease of use has been decided after taking various targeted user groups.
- **Reliability**: The aim of the system is to execute code in GCC, the built system produces correct results. The system is stable under various conditions.
- **Security**: The system collects no personal information and thus is secure, the system also never shares the written code content information to anyone. The system is thus secure in all environments.
- **Maintainability**: The system is highly maintainable. All the codes are well documented. While the process of development, proper naming convention of variables and functions have been used along with proper comments documenting the description of written methods. The system can also be modified and extended properly.
- **Installability** : The software can be easily installed and can be run very easily.

5. Functional requirements:

Actor Goal List:

Actor	Goal
Novice Programmer	Select user type, Write, compile, link and execute code.
Typical Programmer	Select user type, Write, compile, link and execute code, generate and optimize code.
Expert Programmer	Select user type, Write, compile, link and execute code, generate and optimize code, add and use developer options

6. User Personas and their user characteristics:

- Typical Programmer:

Stevenson Ross

Age: 35
Work: Software Quality Assurance Analyst
Family: Married
Location: Montreal, Canada
Character: The Authentic
Education: MNC
Native Language: English
Computer Knowledge: Good
Programming Knowledge Level: Intermediate

Bio

Steve is SQA analyst in a MNC. He does not codes very frequently as he uses the automated tools for the quality assurance purposes. He is an independent and reliable worker. He loves to code sometimes. He says his coding skills are intermediate as he does not codes frequently but did use to code in his bachelors. Steve is an ardent fan of Java Eclipse IDE and wants something that is close to it for C also.

Computer Knowledge

Basic Computers
Input/Output devices
Programming Languages

Motivations

Easy to Use
Efficient Instructions
Quick results
Understandable
Easy to learn

Goals

- Have a pleasant experience while working with C.
- The platform should support the II type of users so that working is fun rather than an overhead for the users.
- People who are comfortable working on different IDEs should also like to work on new tool.
- To have a quick recap of the the execution process, every time he tries to execute something.
- The features of the IDE can be categorized for the user type and the skill proficiency.


Frustrations

- Cannot remember the syntax
- It is hard to remember the execution sequence as he does not very frequently codes due to the nature of his job.
- Lesser online automated IDEs for execution.
- Not much help given by tools when a person who s not a frequent coder tries to code.

Personality

Independent Analytical Passive
Dependent Creative Active

Agreeable
Accessible
Dependable



"I want a quick recap of the execution process, as I know the basic but don't remind them very frequently."

- Novice Programmer:

James Christopher

Age: 15

Work: Student

Family: Unmarried

Location: Montreal, Canada

Character: The Authentic

Education: High School

Native Language: French

Computer Knowledge: Basic

Programming Knowledge Level:
Novice

Bio

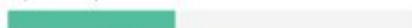
James is a high school student, interested in learning new programming languages. He has opted for 'Basic C' as an optional course in the upcoming semester. He has very basic knowledge about C language and wants to learn it in such an environment that can guide him with efficient instructions on how to run and compile his first program. He is not very fond of reading and wants quick and easy to remember instructions for his first program compilation.

Computer Knowledge

Basic Computers



Input/Output devices



Programming Languages



Motivations

Easy to Use



Efficient Instructions



Quick results



Understandable



Easy to learn



Goals

- Have a pleasant experience while learning C.
- The platform should support the novice users so that learning is fun rather than an overhead for the users.
- Language barriers should also be taken into considerations him being a French Canadian.
- To learn at least the basics of C language in one month.

Frustrations

- Programming Languages syntax.
- Many online and offline compilers are not student friendly.
- Incompatible compilers for his Windows Laptop.
- Learning C language seems to be an overhead rather than an interesting challenge with present tools.

Personality

Independent ☐ Dependent ☐

Analytical ☐ Creative ☐

Passive ☐ Active ☐

Agreeable ☐

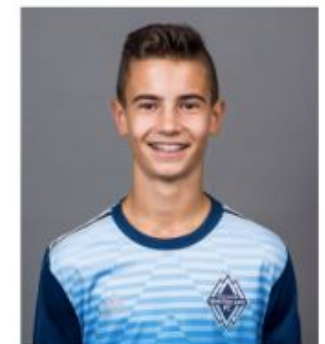
Accessible ☐

Dependable ☐

Agreeable

Accessible

Dependable



"I want to have a comforting and assuring experience while learning the C language."

- Expert Programmer:

Robert Camos

Age: 28

Work: Developer

Family: Unmarried

Location: Montreal, Canada

Character: The Authentic

Education: Master in Computer Science

Engineering

Native Language: English/French

Computer Knowledge: Good

Programming Knowledge Level:
Advanced

Bio

Robert is a developer in an MNC. He is an ardent coder in C language . He also enjoys working on Java, C sharp, Python and front end technologies. He is an independent and reliable worker. He really enjoys his work and may want to change the user friendly perspective for many execution tools available for C.

Computer Knowledge

Basic Computers



Input/Output devices



Programming Languages



Motivations

Easy to Use



Efficient Instructions



Quick results



Understandable



Easy to learn



Goals

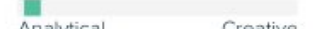
- Have a pleasant experience while working with C.
- The platform should support the all type of users so that working is fun rather than an overhead for the users.
- People who are comfortable working on different IDEs should also like to work on new tool.
- The look and feel could be more user oriented.
- The features of the IDE can be categorized for the user type and the skill proficiency.

Frustrations

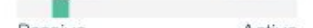
- Not enough platforms available for quick and smooth results
- Available resources are not very user friendly and old fashioned.
- Lesser online automated IDEs for execution.
- The absence of user friendly look and features makes it a lot boring.

Personality

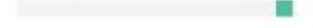
Independent. Dependent



Analytical Creative



Passive Active



Agreeable

Accessible

Dependable

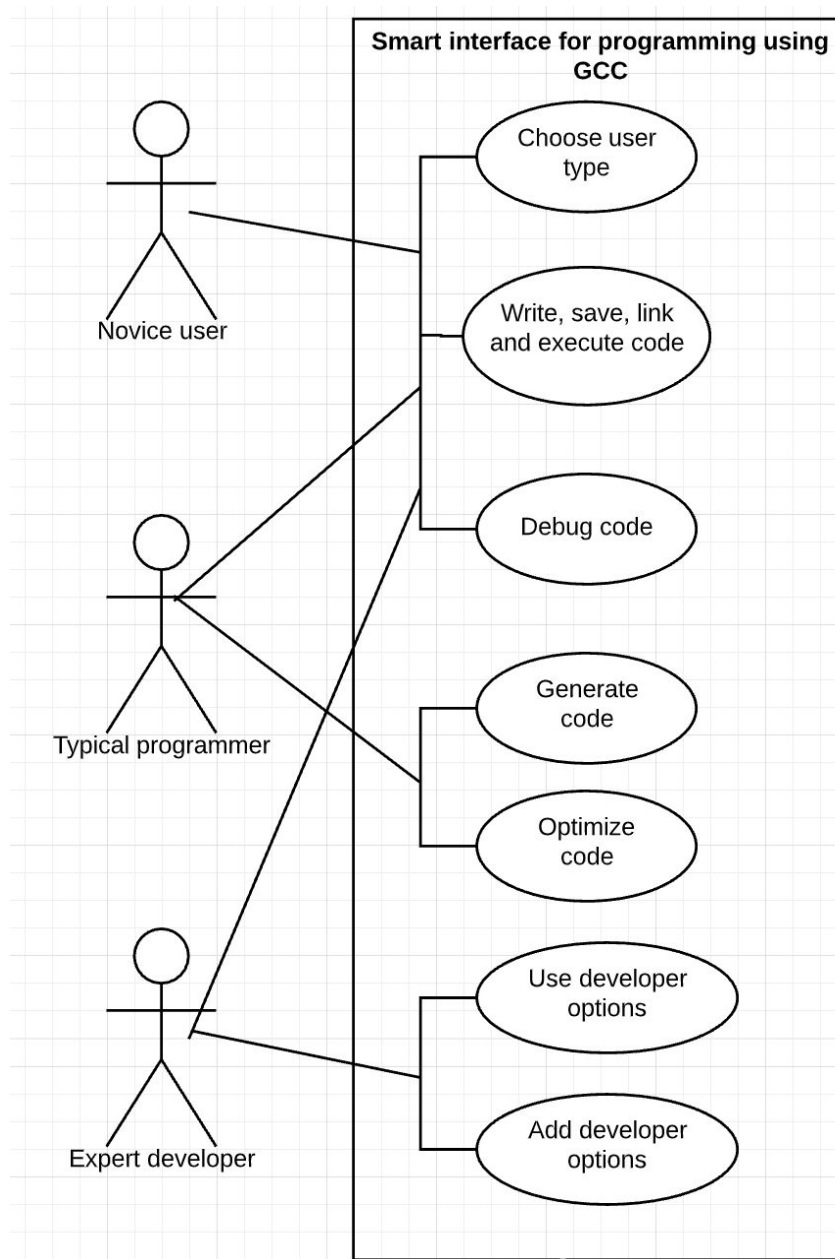


"I want a fast and quick solution to the execution process of C, if it can be more user friendly."

7. Task Characteristics using use case diagrams:

Use case model:

Use cases are helpful in explaining the task characteristics of a system.



Use cases:

Use Case: UC01 Choose user type

Actors: Novice user, Typical programmer, Expert developer

Pre-condition:

- Actor must be able to launch the application

Post-condition:

- The user type is selected

Success Scenario:

1. The user choices (novice, typical and expert) are displayed along with descriptions
2. The actor selects the user type depending on his/her programming expertise

Use Case: UC02 Write, save, link and execute operations

Actors: Novice user, Typical programmer, Expert developer

Pre-condition:

- Actor must have selected the user type

Post-condition:

- For write: the code written by actor must be displayed
- For save: the code written by actor must be saved
- For link: the object files are linked to a single executable file
- For execute: the executable file is compiled

Success Scenario:

1. The actor writes lines of code which become visible on the code panel in real time
2. The actor uses the save feature and the lines of code are saved so that it may later be restored from the saved version
3. The actor uses the execute feature
4. The lines of code are compiled using the compiler
5. The success message with output is displayed on the console
6. The linker links the object files to a single executable file
7. The linker displays a success message on the console

Extensions:

5a. If the code has any errors, error message containing the error description and the line of code where error is encountered is displayed on console

7a. If the linker discovers any errors (such as multiple files under same name), an appropriate error message is displayed on console

Use Case: UC03 Debug code

Actors: Novice user, Typical programmer, Expert developer

Pre-condition:

- Actor must have written code that is displayed on the code panel

Post-condition:

- The code written by actor must be compiled and identified errors can be removed

Success Scenario:

1. The actor defines breakpoints in the lines of code
2. The actor uses the debug feature
3. The compiler executes the code between the breakpoints
4. The call stack and associated error messages or output are displayed in the console

Use Case: UC04 Generate code snippet

Actors: Typical programmer, Expert developer

Pre-condition:

- Actor must be able to launch the application

Post-condition:

- A generate code snippet is displayed on the code panel

Success Scenario:

1. The actor uses the generate code feature
2. The actor selects the type of code snippet to generate
3. The appropriate code snippet is displayed on the code panel

Use Case: UC05 Optimize code

Actors: Typical programmer, Expert developer

Pre-condition:

- Actor must have written code that is displayed in the code panel

Post-condition:

- The written code is optimized

Success Scenario:

1. The actor uses the optimize code feature
2. The code optimizer applies certain rules such as coding conventions to refactor the code
3. The refactored code with syntax based colouring is displayed in the code panel

Use Case: UC06 Add Developer options

Actors: Expert developer

Pre-condition:

- Actor must have written lines of code that is displayed, saved and compiled

Post-condition:

- The results of the developer option selected is displayed on the console

Success Scenario:

1. The actor selects a developer option
2. The appropriate settings of debug dump for future compilation is saved
3. A confirmation of the selected developer option and results are displayed on the console

Use Case: UC07 Use Developer options

Actors: Expert developer

Pre-condition:

- Actor must have written lines of code that is displayed, saved and compiled

Post-condition:

- The implemented developer option is saved for future use

Success Scenario:

1. The actor writes the code to be executed when the custom developer option is implemented
2. The above code is compiled and a success message is displayed on console
3. The actor saves the above code under the alias of the developer option shortcut

Extensions:

- 2a. If the code written by actor has errors, an appropriate error message is displayed on the console

User scenarios:

- Novice Programmer:

James is a high school student who just started learning C language and he is struggling with learning and using different syntax for coding with the current compiler. He has to be able to set the compiler's configuration before starting to code which allows the compiler to recognize him as a novice user and hide functionalities that are not useful for

a beginner. He wants to write simple code and run it and be able to save his code. He would like the program to help him with writing commands as he is not used to coding frequently so keyword suggestions and color code highlights would be a great asset for him to learn C easier. And James would like a debug function that allows him to check his work and whether or not it needs any modification and to see where he coded incorrectly.

- Typical Programmer:

Jenna is a Masters student at Caltech University in California, used to programming and often codes for her courses and projects. She would like an IDE that allows her the basic functions of writing, saving, compiling and debugging. Since her projects are very long and time-consuming, she wants the compiler to help her write hundreds of lines of code by generating basic parts of code, so that she can fill the rest. In addition, she would like to optimize parts of her code to get the best outcome in the shortest possible time because she is usually on a deadline for her project submissions. Her work has to have specific standards, so optimization is an important feature for her.

- Expert Programmer:

David is a skilled programmer at a software development company in Montreal. He would like all the basic functionalities in a compiler. Since his projects often require changing some specific parts of code or compiler, he would like to have developer options that allow him access to higher functions within the compiler which is not necessary for novice and typical users. For convenience, he would prefer to save those options for future use so that he won't need to add them next time for his work.

Essential Use cases:

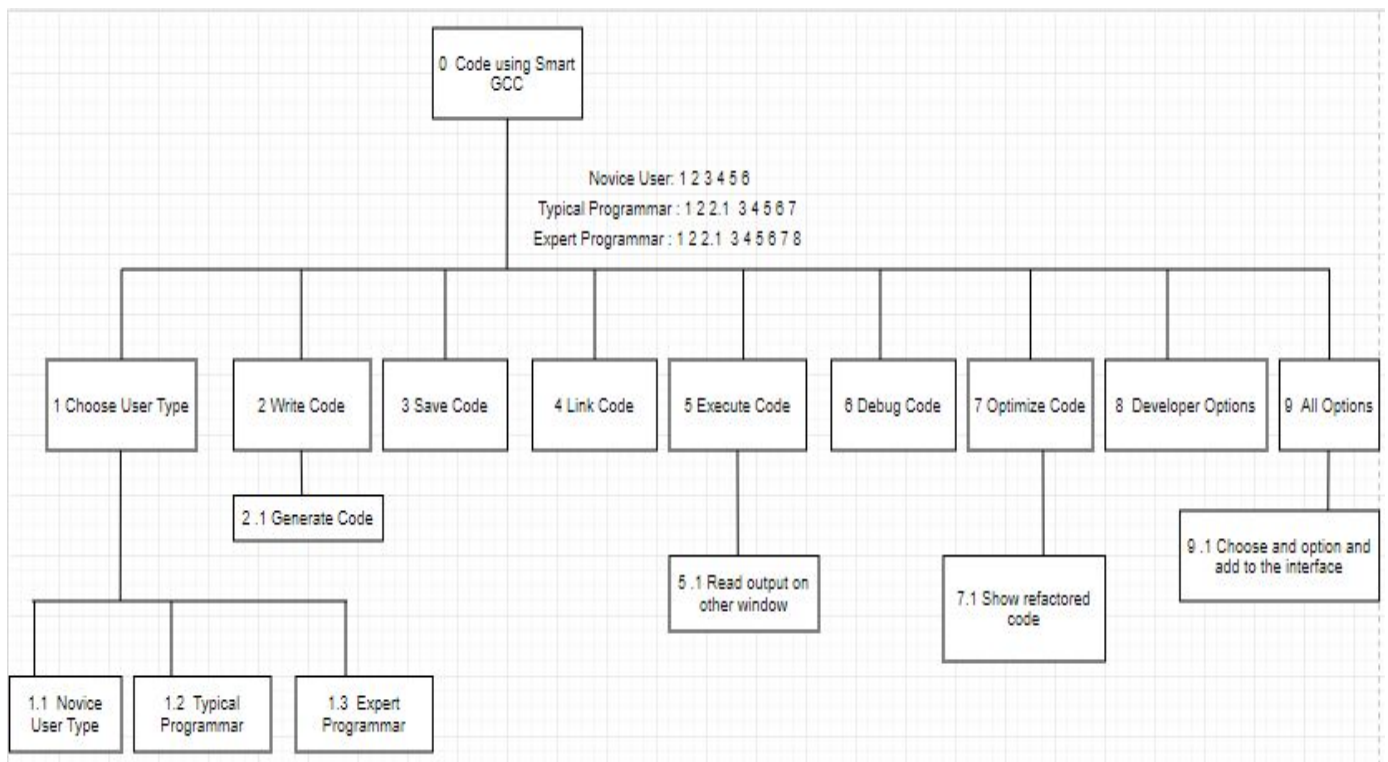
Essential use case is an abstract way for describing a use case to capture the intention of the user and system's response independently.

	USER INTENTION	SYSTEM RESPONSIBILITY
1.	Code using Smart GCC	Provide user type options
2.	Select user type	Open interface based on user type
3.	Write code	Display written code
4.	Save code	Save written code

5.	Execute code	Display result
6.	Debug code	Display result upto debug point
7.	Generate code snippet	Show selected code snippet
8.	Optimize code	Display refactored code
9.	Add developer option	Display confirmation of added option
10.	Use developer option	Display result based on chosen option

8. Task Characteristics using Hierarchical task model:

In Task Hierarchical Analysis (HTA), the tasks are broken into subtasks. It shows the order in which tasks should be performed. These diagrams help in recognising tasks and its subtasks as pictorial presentation are easy to understand.



9. Narrative UI Scenarios

A batch of Engineering students from an esteemed Engineering college enter the career fair and campusing flair of the year. A company with vacancies in an IT profile comes to the same campus with entry level portfolios. They allowed students from all engineering departments to enter the first round of questionnaires. Three friends- Amar, Akbar and Anthony decide to sit for this exam. Amar is an expert developer from the computer science department whereas Akbar is from the Electrical Department and Anthony is from the Civil Department. Evidently, Amar has considerable experience with programming throughout his four years of engineering. Akbar has had no formal training in programming, however, he has pursued a significant number of personal projects where he learnt coding on his own. Anthony is only entering the question round simply through his friends' persuasion and has no interest in programming. Anthony's only experience in programming is whatever he learned in high school. The company has a custom made UI where the students have to write a code and compile and execute them to get the desired results. The console is simple with no command line prompt and the result displayed is a very simple message - either a success or error message. Amar is visibly disappointed as he would like to investigate further and know the breakpoints of his code. He would also like to know the call stack and explore through the command line. Akbar is also used to more features such as code colouring according to syntax, automatic generation of code snippets and is not highly impressed with this UI. Anthony, being the least experienced, is not used to the most obvious icons and would love to have a tooltip for every icon as a means to guide him.

The proposed interface for code editor strives to resolve these issues and cater to the different requirements of the three different types of coders with varying levels of expertise.