Deliverable 1: Debugger Aid Desktop Application

### Group 4

Team Members: David Terry, Marcielly Nascimento, Kevin Racktoo, Lief Watts

[Project Description 3](#_bookmark0)

[SDLC 3](#_bookmark1)

[Project Scope 3](#_bookmark2)

[Project Schedule 4](#_bookmark3)

[WBS 4](#_bookmark4)

[Deliverables (Activity Graph) 5](#_bookmark5)

[Timeline (Gantt chart) 5](#_bookmark6)

[Team Organization 6](#_bookmark7)

[Team Hierarchy 6](#_bookmark8)

[Resource Allocation Chart 7](#_bookmark9)

[Technical Description 7](#_bookmark10)

[Standards and Procedures 7](#_bookmark11)

[Planning 7](#_bookmark12)

[A: Quality Assurance Plan 7](#_bookmark13)

[B: Configuration Management Plan 8](#_bookmark14)

[C: Documentation Plan 8](#_bookmark15)

[D: Data Management Plan 8](#_bookmark16)

[E: Resource Management Plan 8](#_bookmark17)

[F: Test Plan 8](#_bookmark18)

[G: Training Plan 8](#_bookmark19)

[H: Security Plan 8](#_bookmark20)

[I: Risk Management Plan 8](#_bookmark21)

[J: Maintenance Plan 9](#_bookmark22)

1. [Reference 9](#_bookmark23)
2. [Contributions 10](#_bookmark24)

# Project Description

The Debugger Aid Desktop application is a tool for novice programmers. It will help to locate common mistakes that novice programmers make to help them get through the creation of new code. The application will provide a clear and easy to follow process that will allow a user to efficiently debug new code.

# SDLC

We are going to use Phased Development because it allows for us to use

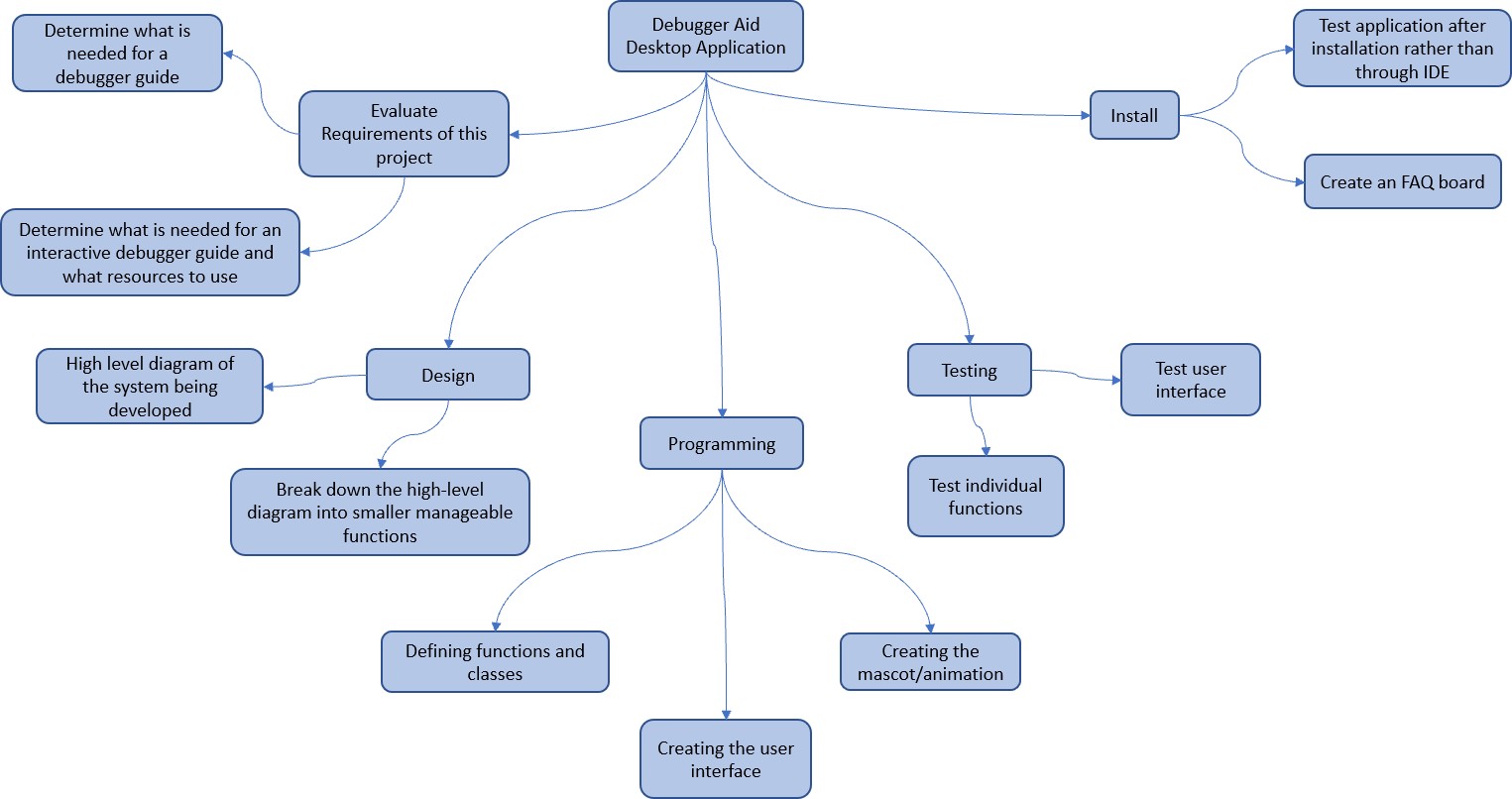
iterative development. We chose this SDLC because the software skills of the team and feasibility of the project will be hard to determine until programming the code, so it may need to be revisited and tweaked accordingly.

# Project Scope

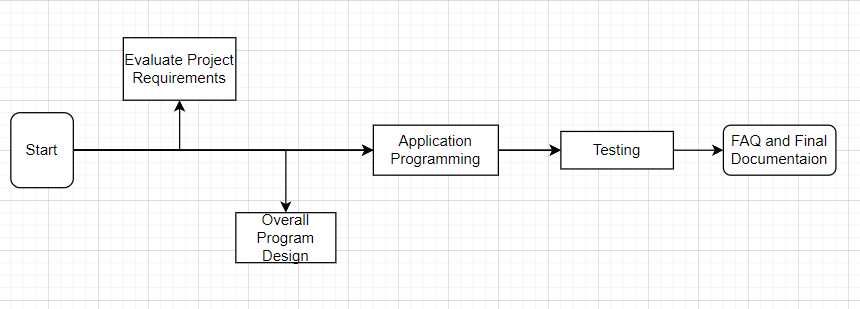
This project includes a Graphical User Interface (GUI) that will direct users through correcting common errors in programming. This program will not scan any other program. The software for this project will guide the user through a series of guided questions with an interactive list.

# Project Schedule

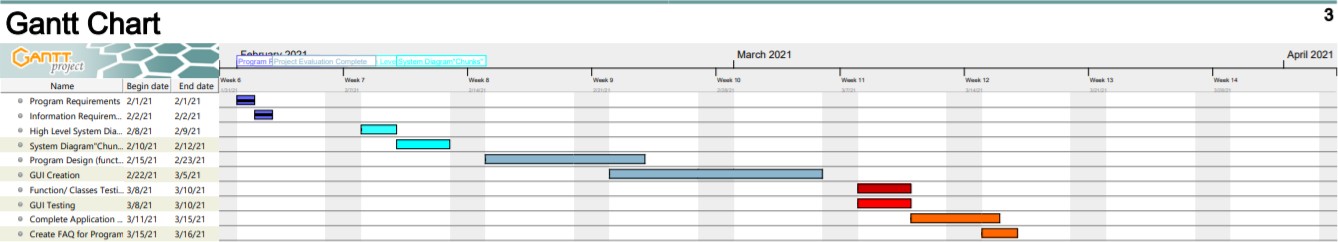
## WBS



## Deliverables (Activity Graph)



## Timeline (Gantt chart)

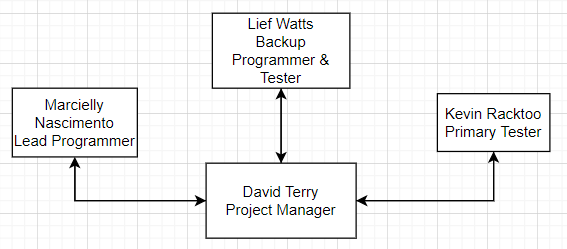


# Team Organization

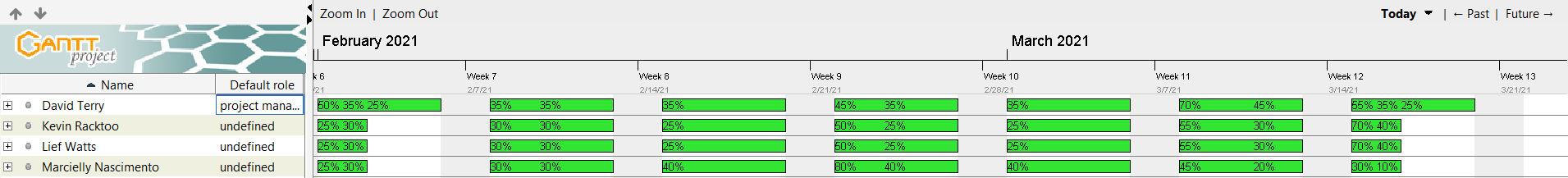
Group 4 consists of the following people:

|  |  |
| --- | --- |
| **Group Member** | **Work** |
| David Terry | Schedule meetings, ensure work is completed by any deadlines, work on weekly/bi-weekly deliverables, test software, edit content |
| Marcielly Nascimento | Design program, create content, work on weekly/bi-weekly deliverables |
| Lief Watts | Work on weekly/bi-weekly deliverables,  test software, edit content, generate and manage documents |
| Kevin Racktoo | Create content, edit content, work on weekly/bi-weekly deliverables, test software |

## Team Hierarchy



## Resource Allocation Chart



# Technical Description

The Debugging Assistant Tool will be a Windows program. A UI will be designed in order to make the program as user friendly as possible, since the target audience is novice programmers. The functionality requirement of the program is that it must be intuitive and simplistic in design and function. The performance requirement of the program requires a near instantaneous response time each time the user interacts with the program.

# Standards and Procedures

The programming language will be C++. The program will be created using Windows Form provided by Visual Studio. The program will implement Object Oriented programming. Peer review will be used to test the program to ensure it is user-friendly and to locate any errors, bugs, or faults.

# Planning

## A: Quality Assurance Plan

To ensure the program is at the standard it is expected to be, it will be peer reviewed by multiple different peer-reviewers. The peer reviews will focus on ensuring user-friendliness and ensuring the program is free of bugs or errors. This Quality Assurance Plan will be achieved by having the peer reviewer interact with the program without any directions or assistance from anything or anyone other than the program itself. This will ensure the peer reviewer will have to rely on the user-friendliness and intuitiveness of the program in order to successfully navigate the program.

## B: Configuration Management Plan

Configuration management and version control will be implemented with Git. Documentation will be done on Google Docs.

## C: Documentation Plan

Documents will be produced during each stage of the SDLC. As each team member uploads their assigned work, David Terry will update task completion in the appropriate record (e.g. deliverables, timeline, etc.).

## D: Data Management Plan

N/A No data will be collected and no external server will be used.

## E: Resource Management Plan

N/A application is downloadable and does not require server connectivity.

## F: Test Plan

This software for this project is broken up into two primary processes: the GUI and the list of questions and responses the program asks and provides users using the program. In scope testing includes the functionality of the GUI and the question/answer list the program provides users. Test methods will include an integration test and a complete system test once the program and GUI are complete. Any issues that arise as a result of these tests will be documented as will any revisions to the software for the project.

## G: Training Plan

N/A. An FAQ and popup descriptions will be provided to help new users learn this simple program.

## H: Security Plan

N/A No security plan is needed for the project since the program will be local and will not access external resources.

## I: Risk Management Plan

|  |  |  |  |
| --- | --- | --- | --- |
| **Risk Description** | **Probability of Occurrence (%)** | **Loss Size (days)** | **Risk Exposure (days)** |
| Lack of time to work due to other projects/class or work | 20% | 3 | 0.6 |
| Animation/mascot creation | 20% | 4 | 0.8 |
| Scope Creep | 15% | 1 | 0.15 |
| Insufficient time for professor feedback | 25% | 2 | 0.5 |

To manage the risks for this project, a combination of effective communication and research is required. This will allow for early completion of deliverables allowing more time for review by the professor.

## J: Maintenance Plan

This software will only be updated to fix errors reported by users, and to maintain compatibility with operating system updates. FAQs will be answered at set periods of time. These actions will only be performed during the Spring semester of 2021.

# Reference

Activity Graph with Examples

[https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-](https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-activity-diagram/) [activity-diagram/](https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-activity-diagram/)

C++ Tutorial - UI Application using Visual Studio <https://www.bogotobogo.com/cplusplus/application_visual_studio_2013.php>

Gantt and Resource Chart <https://www.ganttproject.biz/>

How the project will be programmed <https://www.bogotobogo.com/cplusplus/application_visual_studio_2013.php>

Software Development Risk Management Plan with Examples

[https://www.castsoftware.com/research-labs/software-development-risk-](https://www.castsoftware.com/research-labs/software-development-risk-management-plan-with-examples) [management-plan-with-examples](https://www.castsoftware.com/research-labs/software-development-risk-management-plan-with-examples)

WBS of Software Development Template <https://www.edrawsoft.com/template-wbs-of-software-development.html>

# Contributions

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Time** | **Percentage** | **Task** |
| David Terry | 6 hours | 43.0% | * Project Scope * Documentation Plan * Test Plan * Activity Graph * Project Team Organization * Final Review and Edits * Assist   + Gantt Chart   + Resource Allocation Chart |
| Marcielly Nascimento | 2 hours | 15.0% | * Technical Description * Standards and Procedures * Quality Assurance Plan * Configuration Management Plan |
| Kevin Racktoo | 3 hours | 21.0% | * Assist   + Gantt Chart   + Resource Allocation Chart |
| Lief Watts | 3 hours | 21.0% | * Project Description * WBS * Risk Management Plan * Maintenance Plan |

**Software Requirements**

**Specification**

**for**

**<Debugger Aid Desktop**

**Application>**

**Version 1.3 approved**

**Prepared by <Team 4>**

**<Team 4 Members:** David Terry**,** Lief Watts, Marcielly Nascimento, Kevin

### Racktoo**>**

**<February 15, 2021>**

# Table of Contents

[Table of Contents 1](#_TOC_250034)

[Revision History 1](#_TOC_250033)

1. [Introduction 2](#_TOC_250032)
   1. [Purpose 2](#_TOC_250031)
   2. [Document Conventions 2](#_TOC_250030)
   3. [Intended Audience and Reading Suggestions 2](#_TOC_250029)
   4. [Product Scope 2](#_TOC_250028)
   5. [References 2](#_TOC_250027)
2. [Overall Description 3](#_TOC_250026)
   1. [Product Perspective 3](#_TOC_250025)
   2. [Product Functions 3](#_TOC_250024)
   3. [User Classes and Characteristics 4](#_TOC_250023)
   4. [Operating Environment 4](#_TOC_250022)
   5. [Design and Implementation Constraints 4](#_TOC_250021)
   6. [User Documentation 4](#_TOC_250020)
   7. [Assumptions and Dependencies 4](#_TOC_250019)
3. [External Interface Requirements 5](#_TOC_250018)
   1. [User Interfaces 5](#_TOC_250017)
   2. [Hardware Interfaces 5](#_TOC_250016)
   3. [Software Interfaces 6](#_TOC_250015)
   4. [Communications Interfaces 6](#_TOC_250014)
4. [System Features 6](#_TOC_250013)
   1. [Assist in basic Debugging 6](#_TOC_250012)
   2. [Allows user to interact with the program with buttons on the UI 7](#_TOC_250011)
   3. [Teaches the user debugging skills 7](#_TOC_250010)
   4. [User-friendly and minimalistic UI 8](#_TOC_250009)
5. [Other Nonfunctional Requirements 8](#_TOC_250008)
   1. [Performance Requirements 8](#_TOC_250007)
   2. [Safety Requirements 8](#_TOC_250006)
   3. [Security Requirements 8](#_TOC_250005)
   4. [Software Quality Attributes 9](#_TOC_250004)
   5. [Business Rules 9](#_TOC_250003)
6. Other Requirements 9

Appendix A: Glossary 9

[Appendix B: Analysis Models 10](#_TOC_250002)

[Appendix C: To Be Determined List 11](#_TOC_250001)

[Contributions 11](#_TOC_250000)

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
| Individual Assignments | February 21, 2021 | Each team member worked on their assigned sections individually. | 1.1 |
| Original Submission | February 21, 2021 | Met as a group to assemble the complete document and clarify points raised by each team member on the various sections. | 1.2 |
| Fix Errors from feedback | February 28, 2021 | Fixing errors pointed out in feedback. | 1.3 |

1. **Introduction**

## Purpose

This stand-alone C++ software will help novice programmers locate and correct common mistakes in their code. This tool will provide a clear and easy-to-follow process that, using a Graphic User Interface (GUI), efficiently debugs new code. In addition to helping new programmers debug the code they have written; the Debugger Aid program (Dr. Whiskers ++) will help teach programmers “debugging” as a skill.

## Document Conventions

Each header and sub header are Times New Roman, font 14. All other text is Times New Roman, font 12. See section 6, appendix A for acronyms and other technical definitions.

## Intended Audience and Reading Suggestions

This document is intended for developers and testers. This SRS contains detailed information about the Dr. Whiskers ++ program and how it will run. The organization of the document is intended to give its reader a full understanding of the software requirements, starting with the conceptual aspects of the product and proceeding to the more technical details. Although this document is intended to be read in the order the sections are listed, different roles may find it more efficient to read specific sections in lieu of reading the entire document. For example, developers may focus on section 2 and section 4, whereas testers may focus on section 3.

## Product Scope

This product includes a GUI and animation that will direct users through correcting common errors in programming. For example, if a user leaves off a curly brace and they do not understand the error the compiler is giving them, they could use the Dr. Whiskers ++ program. The program will run through various scenarios that represent the most common mistakes that are made and show the user what needs to be done to correct the error and help the user develop debugging skills. Once the correction is implemented, the program will provide the user with a picture to show that they have corrected that particular error. More information regarding the scope and other details can be found in Deliverable 1: Debugger Aid Desktop Application by Group 4.

## References

* + - Deliverable 1: Debugger Aid Desktop Application by Group 4
    - Deliverable 1: C++ Tutorial – UI Application using Visual Studio

# Overall Description

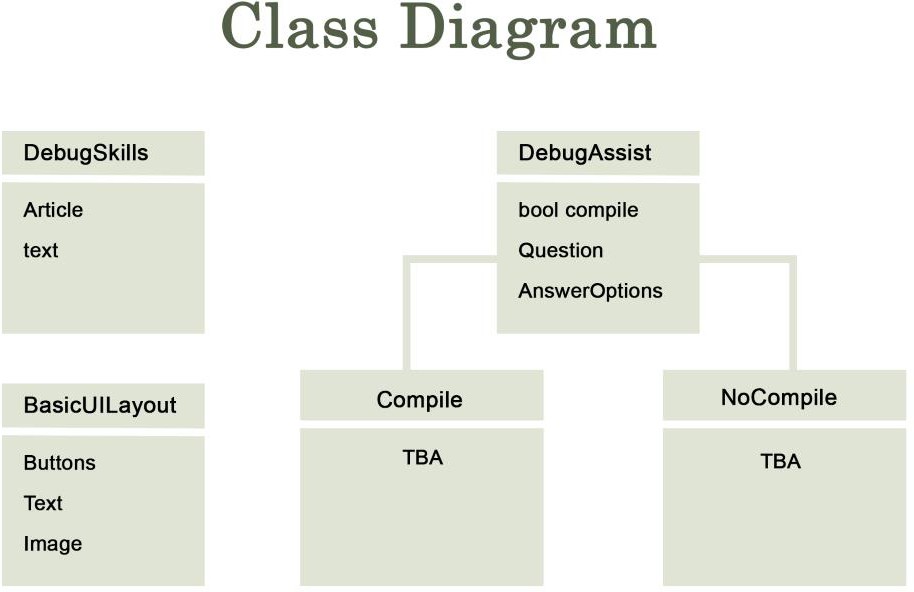
## Product Perspective

The product is new in that it does not follow from any pre-existing members of a product family. It is self-contained, as it is not replacing any pre-existing systems. The product is therefore not a component of any larger subsystems.

## Product Functions

Dr. Whiskers ++ with have the following Major functions:

* + - Assist in basic Debugging in C++.
    - Allows user to interact with the program with buttons on the UI.
    - Teaches users debugging skills.
    - User-friendly and minimalistic UI.





This class Diagram has classes DebugSkills, DebugAssist, BasicUILayout, Compile, and NoCompile. DebugSkills will correlate to the “Teaches users debugging skills” function. DebugAssist will correlate to the “Assist in basic Debugging in C++” function. BasicUILayout will correlate to the “Allows user to interact with the program with buttons on the UI” function and “User-friendly and minimalistic UI” function. Compile and NoCompile are child classes of DebugAssist.

## User Classes and Characteristics

Dr.Whiskers++ will only have one user class. This user class includes both the developers and programmers who use the application for assistance with debugging their program. Please see Appendix B for the Initial Process Flow diagram. The diagram models the user’s path through the program.

## Operating Environment

Dr. Whiskers ++ will run on 32-bit and 64-bit machines. These machines must run a Windows operating system, restricted to those still supported by Microsoft with security and compatibility updates. This includes Windows 8 and Windows 10. Necessary C++ dependencies must be installed for the software to run properly upon Windows. Microsoft’s .NET Framework (version 4.6.1) must be installed on the user’s machine for the program to correctly execute.

## Design and Implementation Constraints

The application is being created specifically for Windows operating systems. It will be self- contained and have no communication with external sources. The application will troubleshoot C++. The creation of the application will be done in C++ due to its familiarity with the developers. After the application is published/released, there will be no major support for the application other than a possible FAQ board.

## User Documentation

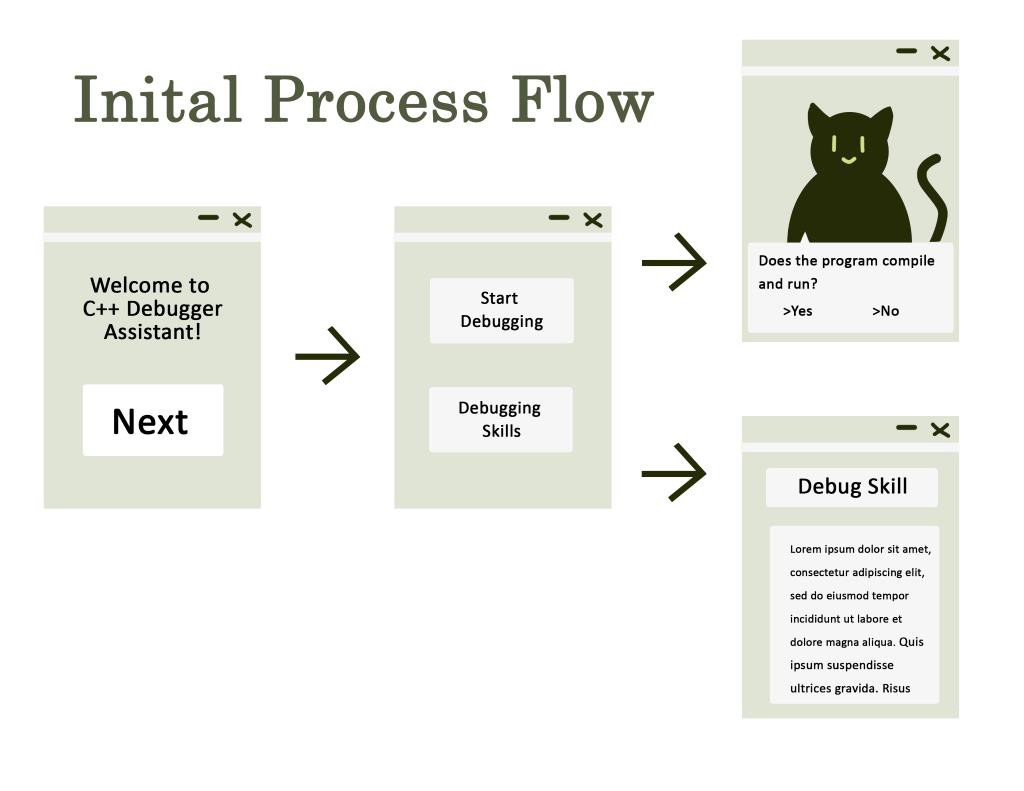
The only user documents created will be a Trouble shooting guide for installation and a user manual. This program is designed to be self-explanatory and thus not require major instructions or explanations, but the user manual will describe what the options and controls do if it is not clear from use.

## Assumptions and Dependencies

The compiler used by the user will run the code perfectly every time. Users will be programming in C++. The application will be created using C++ as it is the most known among the programmers in the team. The UI will be simple with an added animated mascot. The animation, or images used for the mascot will be created separate from the application and added as files to be accessed in the program.

# External Interface Requirements

## User Interfaces



The UI will be minimalistic. The user will use buttons to interact with the program. The first screen will be a screen that welcomes the user and prompts a “next” button. The next screen will ask the user if they want to access the debugging assistant or learn about debugging skills. If they choose the debugging assistant, the UI will ask them a series of questions for which they can answer with provided buttons. The buttons needed will depend on the answer choices for each question the assistant will ask the user. The debugging skills choice will be a pop-up window with articles on different debugging methods. Error messages will only be needed if the program crashes.

## Hardware Interfaces

*N/A* The program is not accessing any outside hardware or resources.

## Software Interfaces

This product has few dependencies upon other software components. However, some software libraries must exist upon the system where the product is to be run in order to ensure that the product runs effectively. User interaction is also governed by a graphical user interface for which the appropriate software libraries must be present. These libraries include:

* + - Library element 1 - TBA
    - Library element 2 - TBA
    - Library element 3 - TBA

The interface through which the products communicate with these libraries is established by the compiler, so no external interfaces are required. The #include tags in the program files will indicate to the compiler that these software libraries are to be included in the compiled product.

The program also requires that the user install the .NET framework (version 4.6.1) for proper functionality.

## Communications Interfaces

*N/A* There are no communication interfaces. The program is self-contained after installation and does not communicate with outside resources.

# System Features

## Assist in basic Debugging

* + 1. **Description and Priority**

This feature involves asking the user questions to narrow down what the problem with their program could be. For example, the first question would be “Does your program compile and run?”, if they answer “no” then the issue is most likely a syntax error, if not, then the issue is most likely a mathematical or logical error. Dr. Whiskers

++ must use this method to aid the user in finding the problem and solution to their issue. This feature is of High priority.

* + 1. **Stimulus/Response Sequences**
       - Run Dr. Whiskers ++ > Opens program.
       - Click the “Next” button > Opens selection window.
       - Click “Start debugging” button. > Opens debugging assistant window.
    2. **Functional Requirement**

REQ-1: A UI that the user can interact with. (4.2)

REQ-2: The system responds to user interaction by displaying information pertaining to the issue being debugged.

REQ-3: Simple questions (typically “yes” or “no”) will be used to determine possible sources of error in the user’s program.

## Allows user to interact with the program with buttons on the UI

* + 1. **Description and Priority**

This feature involves creating a user interface that the user can use to interact with the program. The user will interact by clicking buttons on the UI and Dr.Whiskers++ will respond with text and images. This feature is of High priority.

* + 1. **Stimulus/Response Sequences**
       - Run Dr. Whiskers ++ > Opens program.
    2. **Functional Requirements**

REQ-1: A UI that the user can interact with. The UI will have buttons for user choices that will let the user navigate through the program. (4.2, Appendix B Initial Process Flow)

## Teaches the user debugging skills

* + 1. **Description and Priority**

This feature involves providing the user with a guide on methods to debug their program when the fault in the user's code is not obvious or easily found. This feature is of High priority.

* + 1. **Stimulus/Response Sequences**
       - Run Dr. Whiskers ++ > Open program.
       - Click the “Next” button > Opens selection window.
       - Click “Start debugging” button. > Opens debugging assistant window.
       - Get to a point where Dr.Whiskers++ determines a debugging skill is needed.

Alternatively:

* + - * Run Dr. Whiskers ++ > Open program.
      * Click the “Next” button > Opens selection window.
      * Click “Debugging Skills” button. > Opens debugging skills window.
    1. **Functional Requirements**

REQ-1: A UI that the user can interact with. The UI will have buttons for user choices that will let the user navigate through the program. (4.2, Appendix B Initial Process Flow)

## User-friendly and minimalistic UI

* + 1. **Description and Priority**

This feature involves designing the UI to be minimalistic and user friendly, since the target audience is novice programmers. The UI will have an appropriate amount of white space and a minimal number of buttons, text, images, and windows. This feature is of Medium priority.

* + 1. **Stimulus/Response Sequences**
       - Run Dr. Whiskers ++ > Open program.
    2. **Functional Requirements**

REQ-1: programming the UI to assist a user with debugging. (4.1)

REQ-2: A UI that the user can interact with. The UI will have buttons for user choices that will let the user navigate through the program. (4.2, Appendix B Initial Process Flow)

# Other Nonfunctional Requirements

## Performance Requirements

N/A. Dr. Whiskers ++ does not have specific performance requirements. This is a stand-alone program that will require very few system resources to run and will have a near instantaneous response time.

## Safety Requirements

While the intent of this product is to assist learning and novice programmers with problematic code, it’s important to remember that the product cannot predict all possible sources of error. It cannot anticipate every situation for which the programmer might be developing code. It is the explicit responsibility of the user of the product to ensure that, in following the advice and instructions delivered by the product, the user’s program does not cause injury or harm to individuals or property. While it is unlikely that problematic code can cause any real harm to the machine upon which the code is being run, this can be dependent upon the safeguards in place on the user’s machine. This, too, is the responsibility of the user.

## Security Requirements

*N/A* There are no security requirements because the system is self-contained and is not accessible from outside through communication.

## Software Quality Attributes

The application must be simple enough that ongoing support will not be needed. Ongoing support would include patching bugs or making changes due to user demand. It must also use minimal system resources to operate so as not to interfere with the IDE being used to develop a program. It will be able to operate on current windows operating systems windows 8 or later. The application needs to give correct feedback on troubleshooting user code.

## Business Rules

N/A Dr. Whiskers ++ is separate from any other software on the system and has no behind-the- scenes connectivity that might interfere with the rest of the system’s operations. Therefore, it does not have specific business rules that inhibit its use. The user will have complete control to run part or all of the program and can quit at any time. Since this program is running separately from any other program, it does not need to handle permissions from either a user or the system on which the program is being run.

1. **Other Requirements Appendix A: Glossary** Library:

A library, in this context, is a software library, or a collection of predefined software classes that can be used within another program.

.NET Framework:

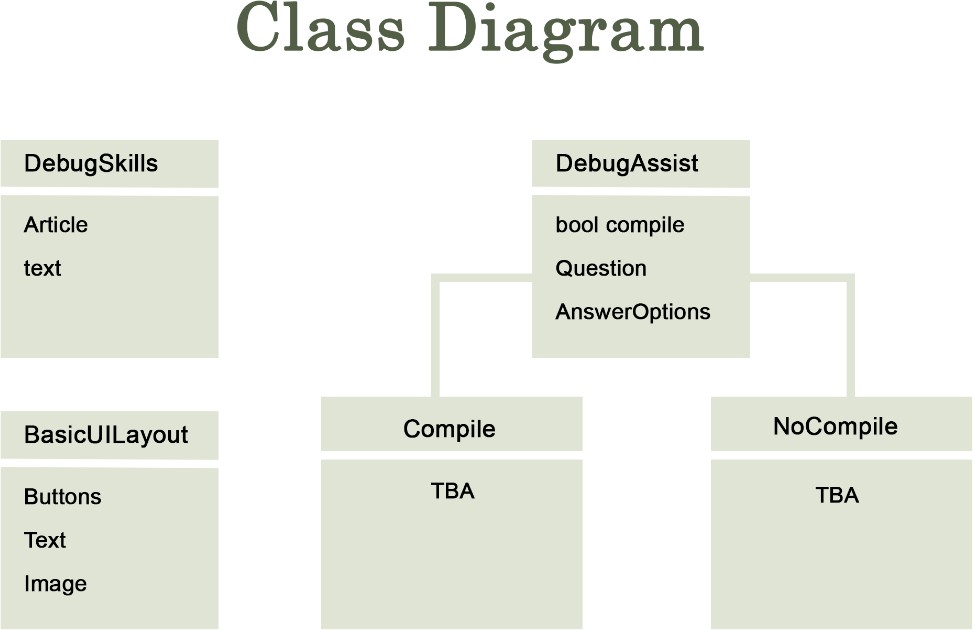
a framework used to create and run software applications. This allows for compatibility with multiple operating systems.

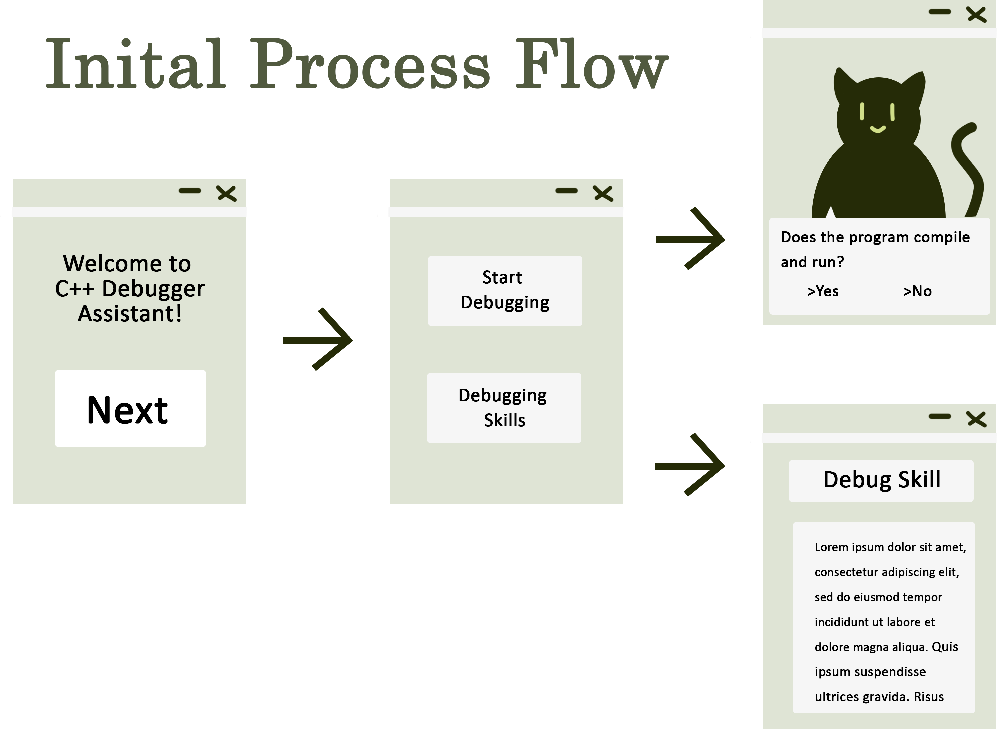
UI:

User Interface IDE:

Integrated Development Environment

**Appendix B: Analysis Models**





**Appendix C: To Be Determined List**

1. Library element 1 - TBA
2. Library element 2 - TBA
3. Library element 3 – TBA
4. Class Diagram classes “Compile” and “NoCompile”

**Appendix D: Contributions:**

Contributions

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** |  |  | **Task** |
| David Terry |  |  | 1. Sections 1.1 - 1.5 2. Sections 5.1 & 5.5 3. Appendix D: Contributions 4. Editing and Document Preparation |
| Marcielly Nascimento |  |  | 1. Sections 2.1 & 2.3 2. Section 3.1 3. Sections 4.1 & 4.2 4. Appendix B: Analysis Models |
| Kevin Racktoo |  |  | 1. Sections 2.1 & 2.4 2. Section 3.3 3. Section 5.2 4. Appendix A: Glossary |
| Lief Watts |  |  | 1. Sections 2.5, 2.6, & 2.7   1. Unless otherwise specified, all other N/A Sections 2. Section 5.3 & 5.4 |

Deliverable III: Dr. Whiskers++

### Group #4

Team Members: David Terry, Kevin Racktoo, Marcielly Nascimento, Lief Watts

Contents

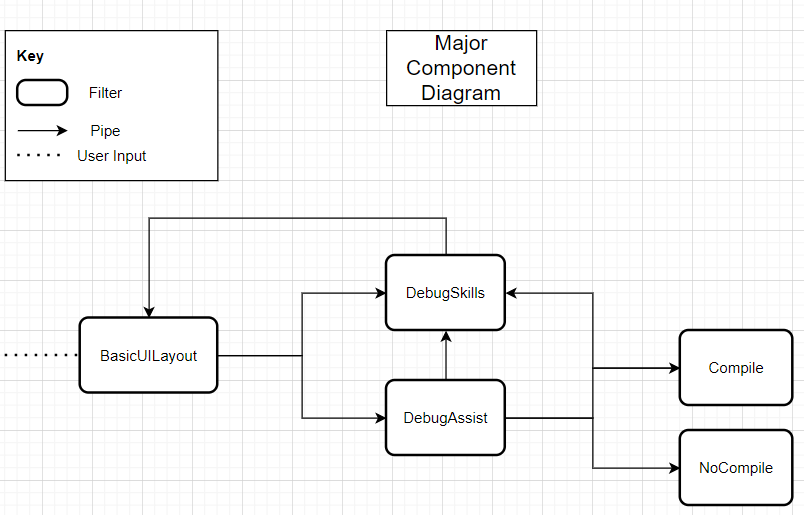
1. [Introduction 3](#_bookmark0)
2. [High-level Architecture 3](#_bookmark1)
   1. [Major Components 3](#_bookmark2)
   2. [System Interfaces 4](#_bookmark3)
   3. [Potential Design Issues 5](#_bookmark4)
3. [Lower Level Architecture 5](#_bookmark5)
   1. [UML Class Diagrams 5](#_bookmark6)
   2. [UML Message Sequence Charts 6](#_bookmark7)
   3. [UML Statechart Diagrams 9](#_bookmark8)
4. [References 10](#_bookmark9)
5. [Contributions 11](#_bookmark10)

# Introduction

This stand-alone C++ software will help novice programmers locate and correct common mistakes in their code. This tool will provide a clear and easy-to-follow process that, using a Graphic User Interface (GUI), efficiently debugs new code. In addition to helping new programmers debug the code they have written; Dr. Whiskers ++ will help teach programmers “debugging” as a skill.

# High-level Architecture

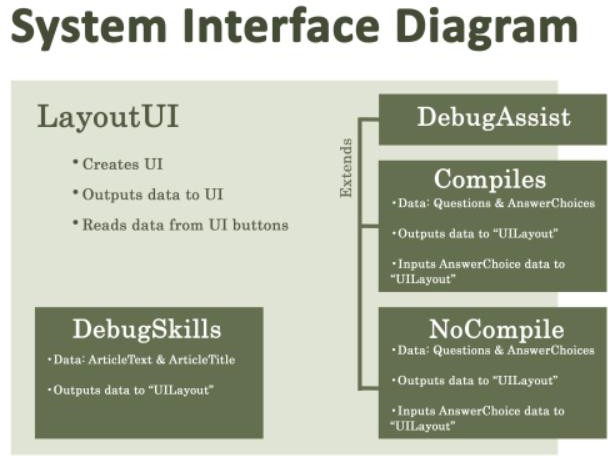
## Major Components



Dr. Whiskers++ has five major components: BasicUILayout, DebugSkill, DebugAssist, Compile and NoCompile. The BasicUILayout is the GUI that our users will use to access the program. From this GUI, the user will have the opportunity to proceed to DebugSkills and DebugAssist. The DebugSkills component is where we will store and provide information for our users to learn more about debugging. From DebugSkills, the user can navigate back to the BasicUILayout component. Our users will be able to access DebugAssist from BasicUILayout and the DebugAssist component is where our users will, via a Boolean question, access either the Compile or NoCompile components. The Compile components is where our users will access a reference sheet designed to help them debug their code when it is compiling and the NoCompile component of our process is where our users will access a reference sheet to help them debug their code if their program is not compiling.

Although a couple of the architectural styles apply for our project, our team has decided to use the Pipes and Filter style. We choose this architecture for Dr. Whiskers ++ because it is ideal for dividing a larger processing task into a sequence of smaller, independent processing steps (Filters) that are connected by channels (Pipes).

## System Interfaces



The System Interface Diagram depicts the basic functionality of the major components of the Dr.Whiskers++ program. The major components are the LayoutUI (aka BasicUILayout), DebugSkills, DebugAssist, Compile, and NoCompile. The LayoutUI creates and holds data regarding the UI. DebugAssist is the parent to Compile and NoCompile. It will create variables and methods for Compile and NoCompile. Compile and NoCompile will both use their Question and AnswerChoice data and LayoutUI to output questions and answer choices to the UI. DebugAssist will take the data ArticleText , ArticleTitle, and LayoutUI data to output debug assist data to the UI.

## Potential Design Issues

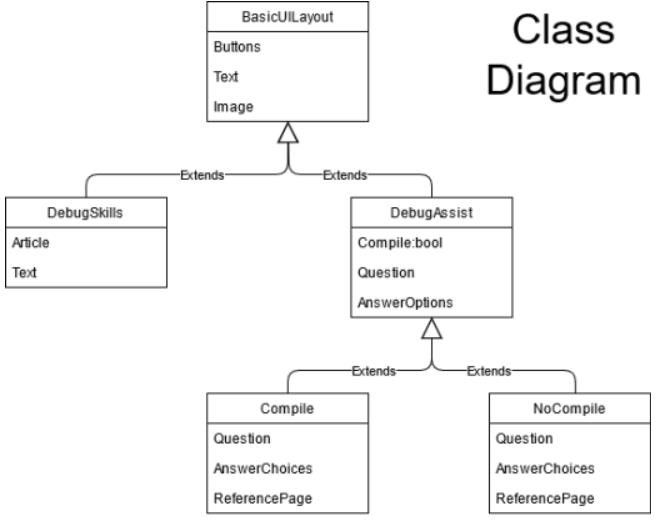
Dr. Whiskers++ is an Object-Oriented Program (OOP) and as such, is reliable, reusable (classes and objects), easy to test and maintain, and is secure. When considering the architecture to use, our team’s focus is on the following relevant issues: functionality, ease of use, reliability, reusability, maintainability, portability, security, and safety.

To develop Dr. Whiskers++, our team will use incremental prototyping. Incremental prototyping is useful for us because it allows for our team to test each major component of our project independently from one another and confirm that they are working prior to “putting them together”. This will also allow us to adopt a different design strategy, if the team finds one that is better suited for our program. Since our program is not overly large, we feel that any other type of prototyping is not useful at this time.

At this time, we do not foresee any technical difficulties in the either the design or build of Dr. Whiskers++. We are, fundamentally, only working with five classes where the “flow” of our program from start to finish is perfect for OOP as well as the architecture we are using. If we do encounter any issues, we expect to be able to resolve them via the incremental prototyping we are using for each class build.

# Lower Level Architecture

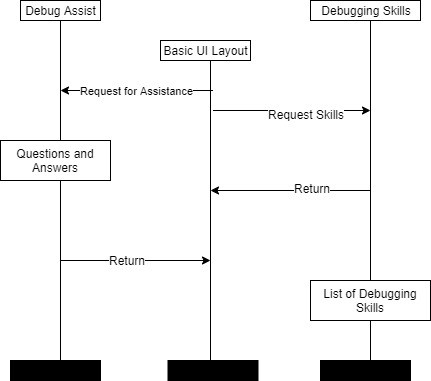
## UML Class Diagrams



The Dr. Whiskers ++ applications is largely GUI-based, with the primary BasicUILayout class providing the basis for all windows that will follow. The DebugSkills and DebugAssist menus both inhereit directly from BasicUILayout, but supplement the features provided by the parent class with additional variables. The DebugSkills class includes an "Article" variable which contains or links to resources which have pertinent information to assist the user in understanding a programming subject matter. DebugAssist has introduces a boolean variable to track whether the user's program compiles, a "question" variable to store the question that will be asked of the user, and the "AnswerOptions" variable that can be used to display potential responses to the question asked. Both Compile and NoCompile are classes which inherit from DebugAssist, as they, too, asks questions of the user and present options. However, they also give the option to link to a DebugSkills reference page.

## UML Message Sequence Charts

#### Message Sequence Diagram 1:



##### Classes

* + - Debug Assist – The class allows users to go through questions to help with debugging.
    - Basic UI Layout – This starts a starting screen and pulls information from the two classes to put into visual displays.
    - Debugging Skills – This class starts the process of getting the skills displayed.

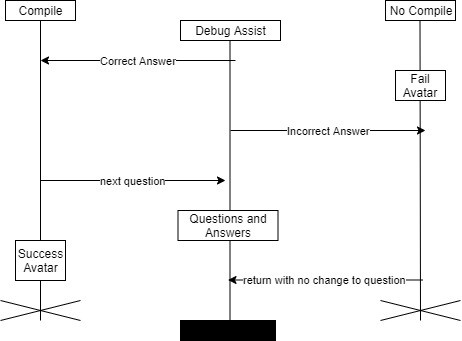
##### Data Storage

* + - Questions and answers – This data block that holds all the questions and answers that will guide a user through debugging.
    - List of debugging skills – This data block holds a list of debugging skills.

##### Communication

* + - Request for assist – This sends a request to Debugging Skills for its information/list.
    - Request skills – This sends a request to Debug Assist that starts the questioning process.
    - Return – This is a request to return to a previous class.

#### Message Sequence Diagram 2:



##### Classes

* + - Compile – This class performs the action of displaying the Success Avatar and signaling a move to the next question.
    - Debug Assist – The class allows users to go through questions to help with debugging.
    - No Compile – This class performs the action of showing the Fail Avatar and returning to the same question.

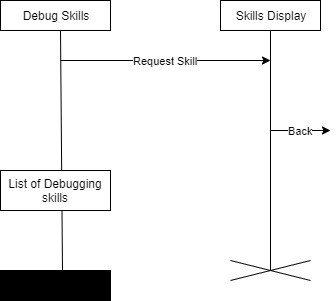
##### Data Storage

* + - Success Avatar – A jpeg or mp4 of or avatar in a success action
    - Questions and Answers – This is the list of questions and answers.
    - Fail Avatar – A jpeg or mp4 of or avatar in a fail action

##### Communication

* + - Correct Answer – This is initiated on a positive answer and request the Success Avatar.
    - Incorrect Answer – This is initiated on a negative answer and request the Fail Avatar.
    - Next Question – On the request the Success Avatar, this signals a move down the list of questions to the next one.
    - Return with no change to question – On request of the Fail Avatar, it returns to the same question.

#### Message Sequence Diagram 3:



##### Classes

* + - Debug skills – This class starts the process of getting the skills displayed.
    - Skills display – This places the skills requested into a visual layout.

##### Data Storage

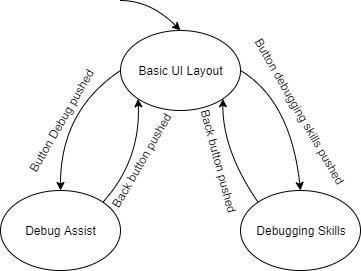
* + - List of Debugging Skills – This data block holds a list of debugging skills.

##### Communication

* + - Request Skill – Requests a skill from the list.
    - Back – This is a request to return to a previous class.

## UML Statechart Diagrams

#### State Diagram 1:



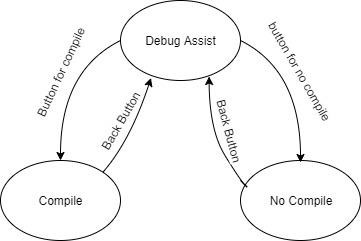
##### States

* + - Basic UI Layout – The starting point with the initial screen and navigation buttons.
    - Debug Assist – The class that guides users the list of debugging questions.
    - Debugging Skills – This is a very basic class that shows all the skills needed for debugging for beginners.

##### Interactions

* + - Button debug pushed – This action causes the application to move from its start state to the Debug Assist state.
    - Button debugging skills pushed – This action causes the application to move from its initial state to the Debugging Skills state.
    - Back button pushed – This action causes the application to move to its previous state.

#### State Diagram 2:



##### States

* + - Debug Assist – The class that guides users the list of debugging questions.
    - Compile – This is the state moved to if a positive answer is input.
    - No Compile – This is the state moved to if a negative answer is input.

##### Interactions

* + - Button for compile – This moves the application from the Debug Assist state to the Compile state.
    - Button for no compile – This moves the application from the Debug Assist state to the No Compile state.
    - Back button – Returns the application to a previous state.

# References

Pfleeger, S. & Atlee, J. (2010). *Software Engineering: Theory and Practice (Fourth Edition).* Upper Saddle River, NJ: Pearson Higher Education.

Skiles, J. (March 8, 2021). *Designing the Modules* (PowerPoint Slides). Retrieved from <https://floridapolytechnic.instructure.com/courses/5121/files/2816032/download?wrap=1>

# Contributions

|  |  |  |
| --- | --- | --- |
| **Name** | **Percentage** | **Task** |
| Team Member |  |  |
| David Terry | 38% | Introduction, Major Components Diagram and description, Potential Design Issues, document preparation and document editing |
| Kevin Racktoo | 18% | UML Class Diagram and description |
| Lief Watts | 21% | UML Message Sequence Charts and description, and UML Statement Charts and description |
| Marcielly Nascimento | 23% | System Interfaces Diagram and description |