

WEB INTERFACE FOR THE PROGRAM ANALYSER SKINK

Arvin Matvosian

Bachelor of Engineering
Software Engineering



Department of Computing
Macquarie University

November 5, 2017

Supervisor: Associate Professor Franck Cassez



ACKNOWLEDGMENTS

I would like to acknowledge Associate Professor Franck Cassez for his guidance, understanding and flexibility throughout the course of my thesis project at Macquarie University.



STATEMENT OF CANDIDATE

I, Arvin Matvosian, declare that this report, submitted as part of the requirement for the award of Bachelor of Engineering in the Department of Software Engineering, Macquarie University, is entirely my own work unless otherwise referenced or acknowledged. This document has not been submitted for qualification or assessment at any academic institution.

Student's Name: Arvin Matvosian

Student's Signature: Arvin Matvosian

Date: November 5, 2017



ABSTRACT

Development of software in this day and age across a wide range of technologies, from embedded closed systems, to vital, complex, globally interconnected systems at some point or another suffer from unforeseen outcomes in the use of these systems. Software bugs and errors are an unfortunate reality in the software development process that are prone to exist due to the difficulty in producing perfect programs. Software Verification systems often become incredibly complex, require a large amount of time, resource and knowledge to install and operate. As such it is not always easily accessible to most developers who want to check their work. The goal of this project is to implement an easy to use web interface for an existing C program analysis tool called Skink, that does not require any installation or use of complex software or software verification knowledge, that will provide software verification feedback to the user.



Contents

Acknowledgments	iii
Abstract	vii
Table of Contents	ix
List of Figures	xiii
1 Introduction	1
1.1 Project Overview	2
1.1.1 Project Goals	2
1.2 Project Planning	3
1.2.1 Scope	3
1.2.2 Time	4
1.2.3 Cost	4
2 Background Literature and Related Works	5
2.1 Software Verification	5
2.1.1 Difference between Software Verification and Validation	6
2.1.2 Testing	7
2.2 Software verification systems used	7
2.2.1 Skink	7
2.2.2 SMT Libraries	8
2.3 Project languages used	10
2.3.1 Front-End	10
2.3.2 Back-End	11
2.3.3 File Types	11
3 Experimental Procedures	13
3.1 The System Environment	13
3.1.1 Software Installation	14
3.1.2 Installing The LAMP Stack	15
3.2 MySQL, Setting up the database	16
3.2.1 Experimental Procedures Summary	17

4	Results	19
4.1	Output	19
4.2	Ease of use	19
4.3	Steps to use the web interface system	20
4.3.1	Step 1: Login	20
4.3.2	Step 2: Prepare Program	20
4.3.3	Step 3: Run Analysis	20
4.3.4	Step 4: Analyse Results	20
5	Discussion	29
5.1	Designing the System	29
5.1.1	Requirements	29
5.1.2	User Requirements and defining the Use Cases	30
5.1.3	Use Cases	31
5.1.4	System Requirements	31
5.1.5	Considerations	31
5.2	High-Level Design	34
5.2.1	Initial System Design	34
5.2.2	PHP	34
5.2.3	C# .Net Core	35
5.2.4	Project Code	35
5.2.5	Visual Design	35
5.3	Final Design	36
5.4	Components	37
5.4.1	Front End	37
5.4.2	Back End	38
5.5	UML Diagrams	38
5.5.1	Behavioural UML Diagrams	38
6	Conclusion	43
7	Future Work	45
7.1	Analyser Analytics	45
7.2	Code Syntax Checking	45
7.3	Adding per user Code sample saving	45
8	Abbreviations	47
A	Project Plan	49
A.1	Project Gantt Chart	49

CONTENTS xi

B Project Code	51
B.1 HTML	51
B.2 CSS	55
B.3 JavaScript	62
B.4 PHP	72
C Meeting Attendance Form	73
Bibliography	73



List of Figures

2.1	Venn diagram showing relationship between Software Verification and Software Validation	6
2.2	Figure that is an overview of the process Skink uses to determine if a program is bug free or not.	8
2.3	An example of a SMT	9
2.4	An overview of how z3 solves SMT by test generation	10
4.1	Example Skink XML Output	21
4.2	Example Skink LL Output	22
4.3	Screenshot of the Web Interface system	23
4.4	Step 1: Screenshot of Login screen	24
4.5	Step 2: Screenshot of Analyser Screen	25
4.6	Step 3: Screenshot Running Analysis	26
4.7	Step 4: Screenshot of Analyser Results screen	27
5.1	An use case of the login for the system	32
5.2	A use case of running analysis on use code	33
5.3	A use case of logging out of the system	34
5.4	A use case of a privileged user adding/removing/viewing system users . . .	35
5.5	A high-level overview of how the system will be implemented	36
5.6	A screenshot of the web design colour palette	37
5.7	Login Activity Diagram	39
5.8	Skink Activity Diagram	40
5.9	The use case diagram of the system	41
A.1	Gantt Chart of expected Project Timeline	49



Chapter 1

Introduction

Developing any non-trivial program that will operate exactly as a developer intended is near impossible. The sheer complexities that arise as a program grows in size mean that it is very difficult for developers to check every possible outcome of a program to ensure that it behaves correctly. As humans we are prone to mistakes, couple this with the fact that there are often a team or organisation of developers working on different parts of a software system that need to interact and errors/unexpected behaviours are guaranteed.

Existing methods to reduce program errors

There are already development procedures and tools apart from software verification that help minimise these kinds of mistakes, such as:

- Pair Programming [1] - The practice of having two or more programmers working together on the same code can lead to quicker detection of errors and an overall faster development of a system, while improving the happiness of the developers.
- Linter code checking tools [2] - Linter code checking tools are often custom built code checking tools usually on a per language basis that check for typing/naming/structure errors in a developers code, and notify the developer to fix the mistakes or check the warnings in order to improve the general correctness of the code and reduce common errors to improve code quality.
- Code Review [3] - The process of having developed code being checked by an unbiased source to pick up on mistakes before code can be published/integrated into existing systems in order to reduce errors.

However these tools are only a way to mitigate some forms of bugs and errors in software, they do not actually test for correctness in the program.

The correctness of a program is the determination of how accurately a program meets its expected outcomes while taking into account all the different possible outcomes during runtime. A program can be considered correct if all its possible outcomes are expected results of the program.

Why develop this Web Interface Tool?

The purpose of this project is to help develop a more generalised way of accessing software verification for developers in an easy to use way. There are already some examples of this kind of tool, such as Whiley Web [4] that allow you to enter code in it's programming language and have it compile/run and return the result on your browser instead of installing all the dependencies required to compile and run the programs.

Being able to quickly check your code for correctness with no installation requirement is a huge advantage in writing better code and producing bug-free software systems. The main verification tool that will be used is the Skink C Analyser, however the system will be designed so that it is easy to add other tools for use. In the Experimental Procedures section there will be a detailed analysis of the benefits and downsides to this type of system.

1.1 Project Overview

In this section an overview of the project will be defined. The project plan and code can be found in the Appendix A and B. The overview of this project is to develop a web interface for a developing software verification system called Skink. The breakdown of this thesis document is as follows:

Chapter 2 is some background literature on the programs and systems used by the Skink analyser, and the tools used in developing the web interface.

Chapter 3 is the experimental procedures that outline the steps and processes used to completion of the project.

Chapter 4 is the results of the project.

Chapter 5 is the discussion chapter of the paper that will talk about an analysis of the project and why certain conclusions can be drawn from the project.

Chapter 6 is the conclusion of the project.

Chapter 7 is discussion of any future work that can be undertaken for this project.

1.1.1 Project Goals

The goal of this project is to have a user upload some program code, and receive a meaningful analysis of the code from the verification tool.

The operation of the web interface in the broad sense is as follows:

1. Take a users code input/parse and format it if necessary

2. Send it to the server that has the software verification systems running on it
3. Transform the input into the correct type for the software verification program
4. Run the Analysis
5. Gather and format the output
6. Send the output back to the user's computer
7. Display meaningful information from the output to aid a user to understand their code

The extended goal of this project is to generalise the web interface system so that it can work with any software verification system that can output analysis in the correct format, which will be discussed in a later section.

1.2 Project Planning

In order to complete this project it was necessary to determine the needs of the project and how to procure any necessary components in order to complete the project. In order to determine this it is important to set a scope for the project so there are some bounds on what is going to be accomplished.

In order to come to the conclusions of the type of system to use and tools required to build the project, you can find an analysis of the different options considered in the Experimental Procedures chapter of this thesis.

1.2.1 Scope

The base scope of this project is to deliver a working implementation of a web interface that can interact with some back-end system that interacts with the Skink program analyser. This means that the front-end of the system should be able to take user input such as code, parameters and other information, and process/send this information to the back-end that interacts with the software verification tool.

The back-end should then collect the result and send it back to the front-end where it can be displayed and interacted with the user.

Given the minimum requirements of the project it was easy to see that it is possible to generalise this implementation to target many different software verification systems. So the scope of the project already allows us to see that there must be some back-end system in place that will bridge the front-end to the software verification system.

The system set-up will be detailed later in the document, please see the Experimental Procedures chapter for more detailed information.

1.2.2 Time

The expected time period set for this project is beginning 31st of August until the 1st of November. A detailed view of the project timeline can be seen in the project Gantt chart in the Appendix.

1.2.3 Cost

The costs involved in this project primarily stem from the fees from renting a web server to host the back-end of the project.

Through consideration of options that can be found in the Experimental Procedures chapter, I decided to use a server hosting company called DigitalOcean through which I rented a server for 4 months at \$20 a month.

There were no other costs associated with this project as all used components were free to use under their respective licences.

Chapter 2

Background Literature and Related Works

This chapter of the thesis will provide background literature for the systems used in this project.

There are a couple of tools that are required for both the Skink Analyser to run and for the front and back-end systems to be built from that will be detailed in this section.

2.1 Software Verification

Being able to prove that a program will execute in a desired way is why there have been great strides in the design and development of software verification systems. Being able to mathematically or other prove a programs correctness means that we can guarantee a certain behaviour of the program. This is particularly useful property if we are looking to develop a program that has critical functions that can not behave unexpectedly, such as in aviation, medicine or nuclear science fields. Often the way these software verification systems work is through formal verification to try and prove the correctness of input programs by analysis of the program code under different circumstances to try and find a path through a program that may result in a bug or fault. A way this is can be done is by:

- Static program analysis [5] - Analysis of the source code of a program (without running the program) often by building a model of the program of its run time state and reasoning as to the possible outcomes of the model.

However these systems are also inherently not perfect, and most of the time can either identify a bug, find no presence of a bug or are unsure if a bug exists. In saying this however they are still vital in testing small components of systems to help find the errors that other methods may not find.

2.1.1 Difference between Software Verification and Validation

Software verification and Software validation are an overlapping idea [6]. Verification has the goal of proving the correctness of a system, whereas validation has the goal of asserting the functioning of a system.

The easiest way to understand the difference is to understand the following:

Validation: Is the system we are building correct?

Verification: Is the system being built correctly?

So in validation we are asking whether or not a system is being built to specification, whereas in verification we are asking whether or not it's implementation is correct. The following Fig.2.2 helps express the overlap between these concepts.

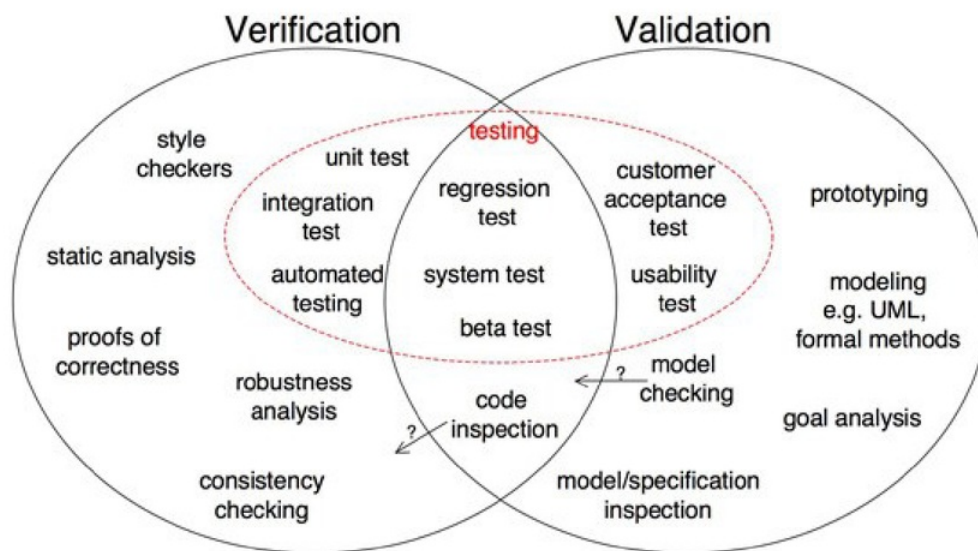


Figure 2.1: Venn diagram showing relationship between Software Verification and Software Validation

The goal of this project is not to validate a system, only to help in verifying that the correct outcome is generated from the input program.

Validation of a system is a concept that must be carried out by the developers or users of a system, this is most often done by a variety of testing methods.

2.1.2 Testing

Testing of a system is the detailed checking of a system against expected outcomes in order to understand if a system is performing as intended. Note that this does not mean that a system is behaving correctly, in that it is behaving without any faults or bugs.

In a sense software verification does implement some types of testing to find out if a system is correct. Some methods of testing might be creating and evaluating models of a system to expected outcomes and analysing the execution of a system with different inputs parameters or runtime conditions.

2.2 Software verification systems used

This section will outline the different existing tools and systems used in the creation of this project.

2.2.1 Skink

Skink is a static analysis tool that analyses LLVM-IR of program source code [7]. It checks whether or not a program can reach a designated point in the code to determine if the result of a program is correct, incorrect or inconclusive.

- Correct - The program will reach the designated block of code with the correct output.
- Incorrect - The program will not reach the designated block of code with the correct output.
- Inconclusive - The program will not reach the designated block of code or the status of the program can not be determined.

The approach Skink uses is to take:

“A program P that is abstracted into an automaton A that generates a language $L(A)$, first abstraction being the control flow graph of the program. Then the abstraction refinement loop is iterated until we find a bug or declare the program bug-free.” [8]

Which means it takes the input source code of the program and generates a custom language that can then be traced through it's theoretical execution until a bug is either found or not found.

Then with some analysis of the results it returns a result of correct, incorrect or inconclusive. Fig.2.2 visualises the process at a high level.

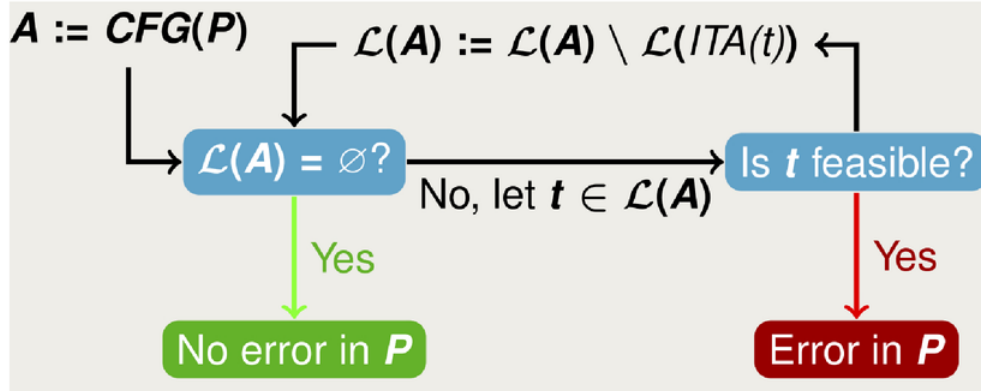


Figure 2.2: Figure that is an overview of the process Skink uses to determine if a program is bug free or not.

The systems used in creating and running the Skink system are:

- Clang - C based language front end for C, C++, Objective C and more [9]
- Scala - Functional Object-Oriented language that runs on the Java Virtual Machine [10]
- Sbt-rats parser generator
- Kiama Scala Library - Scala library for language processing [11]
- Scala SMT - Scala interface for SMT Solvers [12]

2.2.2 SMT Libraries

This subsection outlines the SMT libraries that Skink relies on when running analysis on program code.

Z3 Prover

Z3 is a SMT solver prover built by Microsoft Research available under the MIT licence [13]. A SMT is a type of decision problem for logical formulas and whether the formulas are satisfiable. See Fig.2.3 for an example of a SMT. It is a formula in first-order-logic that has the problem of determining if the formula is satisfiable. An overview of how z3 works can be seen in Fig.2.4

CVC4

CVC4 is another open source SMT solver that is used by the Skink system [14].

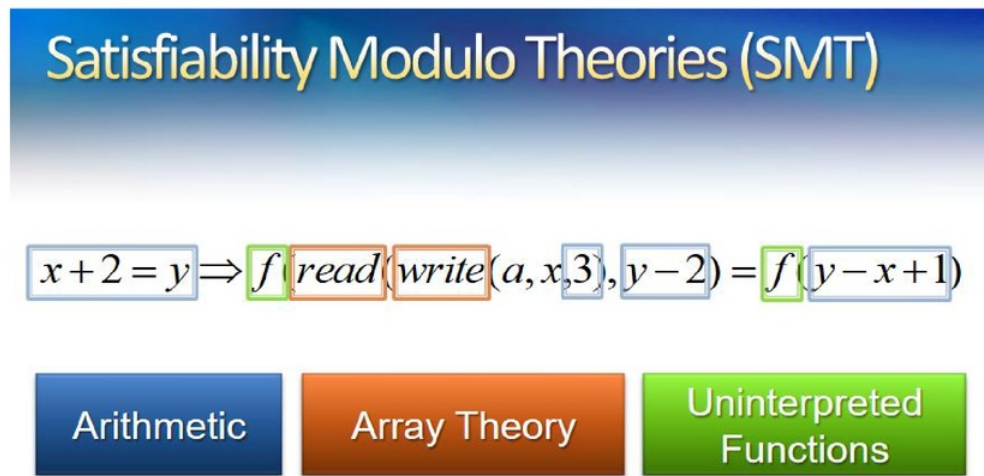


Figure 2.3: An example of a SMT

SMTInterpol

SMTInterpol is a java based SMT Solver library developed by the University of Freiburg, also used by Skink [15].

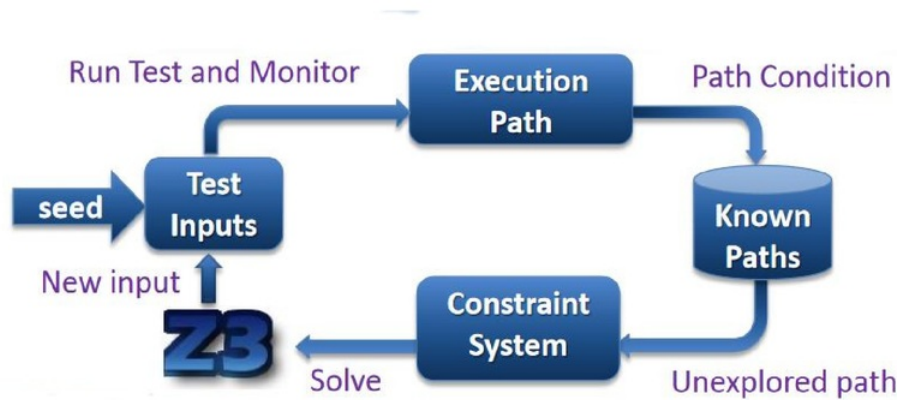


Figure 2.4: An overview of how z3 solves SMT by test generation

2.3 Project languages used

This section outlines the programming/markup languages used in this project in developing the web interface and back-end server code for the Skink Web Interface project.

2.3.1 Front-End

The front end of the web interface will be an online web page that can be accessed by a browser that can open a connection with the back-end of the system which in turn interfaces with the Skink Analysis system. The common tools for the web interface development are as follows in the sub-sections.

HTML

HTML is a markup language used to describe the structure of a text document most commonly used by web browsers to help understand rendering of components. Its purpose is to separate and give structure to different text such that they can be presented or manipulated in different ways, usually for styling.

CSS

CSS is a markup for applying styling to HTML elements. It comes with predefined styling parameters that can then be set to help style HTML components.

JavaScript

JavaScript is a weakly typed interpreted programming language. It is part of the three core technologies used in developing for the web browser. Its job is to provide executable code that can perform actions that are interactive for the user, such as communicating with other systems/computers, manipulating input and HTML elements. In this project its main purpose is to facilitate the parsing, sending, receiving and manipulating of user input/back-end output for the web interface system.

jQuery

jQuery is a JavaScript library that provides access to already built JavaScript functions for handling things like animation, events, and the HTML document traversal and manipulation. It is used in this project to save time in accessing already developed methods for searching and manipulating the HTML document.

AJAX

AJAX is a JavaScript library for running asynchronous code in the web browser. Usually for something that does not have an immediate response, such as connecting to a server sending queries and waiting for a response.

2.3.2 Back-End

PHP

PHP is a general use programming language that runs on servers that have its compiler installed. It handles a lot of back-end system work such as file manipulation, command line utilities, database access and any other function required on a computer. In a basic sense it is the system by which we can interface with

MySQL

MySQL is a type of relational database. The MySQL language is a specially designed syntax for running commands on a MySQL database. It is most commonly used to run queries against data in a database as well as Create/Remove/Modify data and tables in the database.

2.3.3 File Types

There are 4 main file types that are used in this project.

XML

XML stands for eXtensible Markup Language. It is a developed syntax that allows for structured text in a document in order to represent some data, that can be easily understood by humans and easily parsed by computers.

C Files

The C files are just the plain text C program text saved with the .c file extension.

Graphml

Graphml is an modified XML structure for a text file that allows it to represent graph structured data. It is one of the outputs the skink system generates from input code file.

ll File

The ll file in this project is one also generated by skink through the use of CLang LLVR-IM. It contains llvm data from the output of the c program from skink.

Chapter 3

Experimental Procedures

This chapter details the experimental procedures for this project.

3.1 The System Environment

Setting up the system environment requires an understanding of the requirements of the project as a whole and the component dependencies. The following table tries to summarise the dependencies of the system components. Nested components prefixed by – take their parents dependencies plus their own.

Components	Version	Dependencies
Front-end		
HTML	4/5	IE, Firefox, Chrome Web Browser
CSS	3	IE, Firefox, Chrome Web Browser
JavaScript	ECMAScript 5/6+	IE, Firefox, Chrome Web Browser
– jQuery	3.* +	“
– Ajax	1.0+	“
Back-end		
PHP	6.0/7.0 +	Windows/Ubuntu Operating System
Skink		Windows/Ubuntu Operating System, SMT Solvers
– Z3	-	“
– SMTInterpol	-	“
– CVC4	-	“
– Java	7/8	“
— Scala	2.* +	“

There was some consideration required when planning how to proceed with the project in relation to the set up of the back end system as well as the tools that would be used in the project.

Given the dependency table listed in Table.3.1, I decided to use a Linux based distribution,

Ubuntu version 16, due to the ease of installation of many components via the easy access to SSH command line utility and easy installation instructions that are detailed in the section below.

Another considerations was whether or not to rent a hosted server or to convert and use an existing system and turn it into a web host service. Researching some server hosting providers, initially starting a test server with domain name rental from goDaddy, I soon switched to a more easy to use server host solution: Digital Ocean.

The rented server configuration is as follows: Ubuntu 16.04.3 x64, 2GB Memory, 20GB SSD.

3.1.1 Software Installation

Z3

To install Z3, download either the compiled binary to a folder on your machine or un-compiled source from the Z3 github page and build it using the following on Linux using the command line:

```
$ python scripts/mk_make.py
$ cd build
$ make
$ sudo make install
```

CVC4

To install CVC4, follow the instructions below on Linux using the command line:

```
echo 'deb http://cvc4.cs.nyu.edu/debian/ unstable/'
>> /etc/apt/sources.list
echo 'deb-src http://cvc4.cs.nyu.edu/debian/ unstable/'
>> /etc/apt/sources.list
echo 'deb http://cvc4.cs.nyu.edu/debian/ stable/'
>> /etc/apt/sources.list
echo 'deb-src http://cvc4.cs.nyu.edu/debian/ stable/'
>> /etc/apt/sources.list
apt-get update
apt-get install -y --force-yes cvc4
```


Clang

To install Clang, follow the instructions below on Linux using the command line:

```
echo 'deb http://llvm.org/apt/trusty/ llvm-toolchain-trusty-3.7 main'
>> /etc/apt/sources.list echo
'deb-src http://llvm.org/apt/trusty/ llvm-toolchain-trusty-3.7 main'
>> /etc/apt/sources.list
apt-get update
apt-get install -y --force-yes clang-3.7 lldb-3.7
```

SMTInterpol

To install SMTInterpol, follow the instructions below on Linux using the command line:

```
wget --no-check-certificate
https://ultimate.informatik.uni-freiburg.de/smtinterpol/smtinterpol.jar
&& mv smtinterpol.jar /usr/bin/.
```

Java 8

To install Java 8, follow the instructions below on Linux using the command line:

```
wget --no-check-certificate
https://github.com/aglover/ubuntu-equip/raw/master/equip-java8.sh
&& bash equip-java8.sh
```

3.1.2 Installing The LAMP Stack

The LAMP stack is the Linux Apache MySQL PHP stack of programs that allow for a user to set up a web service on their machine.

PHP

To install LAMP, follow the instructions below on Linux using the command line:

```
Install Apache
$ sudo apt-get update
$ sudo apt-get install apache2
$ sudo apache2ctl configtest
$ sudo nano /etc/apache2/apache2.conf
Add server name and IP at the bottom of the above file and restart Apache
$ sudo systemctl restart apache2
```

Check for allowance of port 80/443 in the firewall

```
$ sudo ufw app info "Apache_Full"
```

```
Allow port 80/443
$ sudo ufw allow in "Apache_Full"
```

```
Install MySql
$ sudo apt-get install mysql-server
Set up some basic security
$ mysql_secure_installation
```

```
Install PHP
$ sudo apt-get install php libapache2-mod-php php-mcrypt php-mysql

$ sudo systemctl restart apache2
$ sudo apt-get install php-cli
```

Skink

The newer versions of Skink come with a bundled dependency list so that it is easier to install.

Download Skink, build the system using sbt(Scala Build Tools) build and then compile the system.

You should then be able to run skink.sh via command line like so:

```
"skink.sh -w -loc %TestFilePath %TestFileOutputPath" or
"./skink.sh -w -loc %TestFilePath %TestFileOutputPath"
```

3.2 MySQL, Setting up the database

Setting up the database for this project the main goal was to use it as an authentication method, however it is very easy to adapt and scale the database to support other functionality such as analytic metrics and other features as mentioned in the Future Work chapter.

To setup the MySQL Database, ensure it is installed and configured. To access the database I used a interface called PHPmyAdmin rather than directly working with the database through command line. Using this tool, I created a table for authentication that consists of the following fields and their data types:

- LoginId - VarChar
- LoginPassword - VarChar
- AccountCreated - TimeDate
- LastLogin - TimeDate

- LoginTokenId - Int
- LoginToken - Int

In order to authenticate users I would run the query of the username and password against this table and count the results that were found, if there was no results found then the login was invalid and the function would return false, and if there was a result found the user authenticate function would return true to the web system and it would then take action accordingly.

Using this kind of database makes it really easy to modify and adapt the data and tables to changes in the system design, by way of the PHPmyAdmin interface, it is also has the option to allow for database backups so that we do not lose data in the case of a catastrophic failure in the system.

3.2.1 Experimental Procedures Summary

This chapter summarised the experimental procedures in installing and running the system backend of this project. More in depth explanation of the web interface and design can be found in the Discussion section.

Chapter 4

Results

The results of this project are that as of this paper the Web Interface for Skink is up and running and working as intended. It satisfies the goals of this project.

1. Users are able to log into the site
2. Write some code into the code editor
3. Run an Analysis on that code
4. Receive a meaningful breakdown of the output of that code from the Skink Analyser

4.1 Output

The original output of the skink analyser for some code looks something like the following in Fig.4.1 and Fig.4.2. The web interface translates this into a more readable form as seen in Fig.4.3.

4.2 Ease of use

The ease of use of this system in comparison to the normal method of installing and running the software locally are readily apparent. The installation of the software and related systems, not to mention the difficulty of setting up the correct environment for the systems is already a great deal of work. The web interface system successfully removes all the installation and background knowledge required to develop such a system and makes it simple.

4.3 Steps to use the web interface system

This section will detail the steps to use the web interface system, each step is accompanied by a screenshot of what the interface looks like at that particular step.

4.3.1 Step 1: Login

The first step is to login via the login screen, fill out the credential fields as seen in Fig:4.4 and click the login button. The Login request will then be processed and if correct credentials are given you will be redirected to the Analyser interface screen.

4.3.2 Step 2: Prepare Program

Once you have successfully logged in you will see the default interface for the Skink C program analyser as seen in Fig:4.5. Here you can edit the default test programs in the bottom code editor or clear the code and write/paste your own. You can also rename the file in the file name field. Once you have the desired program ready for analysing move on to Step 3.

4.3.3 Step 3: Run Analysis

Once you are ready click the Run Analysis button. Clicking this button will bring up a load screen 4.6 signaling that your analysis request has been sent and is being executed. Once the analysis is complete the load screen will disappear and you will be able to view the results of the analysis.

4.3.4 Step 4: Analyse Results

In Fig:4.7 we can see that the analysis has completed on our program and returned a Failed result, meaning that Skink has identified some error in our program. Skink has given us the error trace it used to lead to the error as signified by the "Step n" text printed in our code editor. We can also see the error block that the error was found in. Using the Step Function Button we can step through the returned trace to help us find the bug in our program.

```

    <default>false</default>
  </key>
  <key id="witness-type" for="graph" attr.name="witness-type" attr.type="string"/>
  <key id="sourcecodelang" for="graph" attr.name="sourcecodelang" attr.type="string"/>
  <key id="producer" for="graph" attr.name="producer" attr.type="string"/>
  <key id="specification" for="graph" attr.name="specification" attr.type="string"/>
  <key id="programfile" for="graph" attr.name="programfile" attr.type="string"/>
  <key id="programhash" for="graph" attr.name="programhash" attr.type="string"/>
  <key id="memorymodel" for="graph" attr.name="memorymodel" attr.type="string"/>
  <key id="architecture" for="graph" attr.name="architecture" attr.type="string"/>
  <key id="assumption" for="edge" attr.name="assumption" attr.type="string"/>
  <key id="assumption.scope" for="edge" attr.name="assumption.scope" attr.type="string"/>
  <key id="assumption.resultfunction" for="edge" attr.name="assumption.resultfunction" attr.type="string"/>
  - <graph edgedefault="directed">
    <data key="witness-type">violation_witness</data>
    <data key="sourcecodelang">C</data>
    <data key="producer">skink</data>
    - <data key="specification">
      CHECK( init(main()), LTL(G ! call(__VERIFIER_error)))
    </data>
    - <data key="programfile">
      /var/www/html/project/cScripts/59fd889f7411d-ecalikefalseunreachcall.c
    </data>
    <data key="programhash">3362facff0bdd522662ebe497649275edb121737</data>
    <data key="memorymodel">simple</data>
    <data key="architecture">32bit</data>
    - <node id="N0">
      <data key="entry">true</data>
      <data key="block">0</data>
      - <data key="node.src">
        int p1 = __VERIFIER_nondet_int(); // condition variable
      </data>
    </node>
    - <edge id="E0" source="N0" target="N1">
      <data key="edge.src">while(1) {</data>
      <data key="startline">8</data>
      <data key="endline">8</data>
    </edge>
    - <node id="N1">
      <data key="block">3</data>
      <data key="node.src">cond = __VERIFIER_nondet_int();</data>
    </node>
    - <edge id="E1" source="N1" target="N2">
      <data key="edge.src">if (cond == 0) {</data>
      <data key="startline">10</data>
      <data key="endline">10</data>
    </edge>
    - <node id="N2">
      <data key="block">6</data>
      <data key="node.src">if (p1 != 0) {</data>
    </node>
    - <edge id="E2" source="N2" target="N3">
      <data key="edge.src">if (p1 != 0) {</data>
      <data key="startline">14</data>
      <data key="endline">14</data>
    </edge>
    - <node id="N3">
      <data key="violation">true</data>
      <data key="block">8</data>
    </node>
  </graph>
</skinkml>

```

Figure 4.1: Example Skink XML Output

```

; Function Attrs: nounwind uwtable
define i32 @main() local_unnamed_addr #0 !dbg !6 {
  %1 = tail call @__VERIFIER_nondet_int() #2, !dbg !8
  %2 = icmp eq i32 %1, 0
  br label %3, !dbg !9

; @label:3:                                     ; preds = %6, %0
  %4 = tail call @__VERIFIER_nondet_int() #2, !dbg !10
  %5 = icmp eq i32 %4, 0, !dbg !11
  br i1 %5, label %7, label %6, !dbg !12

; @label:6:                                     ; preds = %3
  br i1 %2, label %3, label %5, !dbg !13, !llvm.loop !14

; @label:7:                                     ; preds = %3
  ret i32 0, !dbg !16

; @label:8:                                     ; preds = %6
  tail call void (...) @__VERIFIER_error() #4, !dbg !17
  unreachable, !dbg !17
}

declare i32 @__VERIFIER_nondet_int(...) local_unnamed_addr #1

; Function Attrs: noreturn
declare void @__VERIFIER_error(...) local_unnamed_addr #2

attributes #0 = { nounwind uwtable "correctly-rounded-divide-sqrt-fp-math"="false" "disable-tail-calls"="false" "less-precise-fpmad"="false" "no-frame-
pointer-elim"="false" "no-infs-fp-math"="false" "no-jump-tables"="false" "no-nans-fp-math"="false" "no-signed-szeros-fp-math"="false" "no-trapping-
math"="false" "stack-protector-buffer-size"="8" "target-cpu"="x86-64" "target-features"="+fxsr,+mmx,+sse,+sse2,+x87" "unsafe-fp-math"="false" "use-soft-
float"="false" }
attributes #1 = { "correctly-rounded-divide-sqrt-fp-math"="false" "disable-tail-calls"="false" "less-precise-fpmad"="false" "no-frame-pointer-elim"="false"
"no-infs-fp-math"="false" "no-nans-fp-math"="false" "no-signed-szeros-fp-math"="false" "no-trapping-math"="false" "stack-protector-buffer-size"="8" "target-
cpu"="x86-64" "target-features"="+fxsr,+mmx,+sse,+sse2,+x87" "unsafe-fp-math"="false" "use-soft-float"="false" }
attributes #2 = { noreturn "correctly-rounded-divide-sqrt-fp-math"="false" "disable-tail-calls"="false" "less-precise-fpmad"="false" "no-frame-pointer-
elim"="false" "no-infs-fp-math"="false" "no-nans-fp-math"="false" "no-signed-szeros-fp-math"="false" "no-trapping-math"="false" "stack-protector-buffer-
size"="8" "target-cpu"="x86-64" "target-features"="+fxsr,+mmx,+sse,+sse2,+x87" "unsafe-fp-math"="false" "use-soft-float"="false" }
attributes #3 = { nounwind }
attributes #4 = { noreturn nounwind }

!llvm.dbg.cu = !{!0}
!llvm.module.flags = !{!2, !4}
!llvm.ident = !{!5}

!0 = distinct !DICompileUnit(language: DW_LANG_C99, file: !1, producer: "clang version 4.0.0 (tags/RELEASE_400/final)", isOptimized: true, runtimeVersion:
emissionKind: LineTablesOnly, enums: !2)
!1 = !DIFile(filename: "/var/www/html/project/cScripts/596d98967411d-ecalihefalsreunreacchallo.c", directory: "/root/skink")
!2 = !{}
!3 = !{!42 2, !"Dwarf Version", 132 4}
!4 = !{!132 2, !"Debug Info Version", 132 3}
!5 = !{!"clang version 4.0.0 (tags/RELEASE_400/final)"}
!6 = distinct !DISubprogram(name: "main", scope: !1, file: !1, line: 3, type: !7, isLocal: false, isDefinition: true, scopeLine: 4, isOptimized: true, unit
!0, variables: !2)
!7 = !DISubroutineType(types: !2)
!8 = !DILocation(line: 5, column: 11, scope: !6)
!9 = !DILocation(line: 8, column: 2, scope: !6)
!10 = !DILocation(line: 9, column: 10, scope: !6)
!11 = !DILocation(line: 10, column: 12, scope: !6)
!12 = !DILocation(line: 10, column: 7, scope: !6)
!13 = !DILocation(line: 14, column: 7, scope: !6)
!14 = distinct !{!14, !9, !15}
!15 = !DILocation(line: 18, column: 2, scope: !6)
!16 = !DILocation(line: 20, column: 3, scope: !6)
!17 = !DILocation(line: 21, column: 9, scope: !6)

```

Figure 4.2: Example Skink LL Output

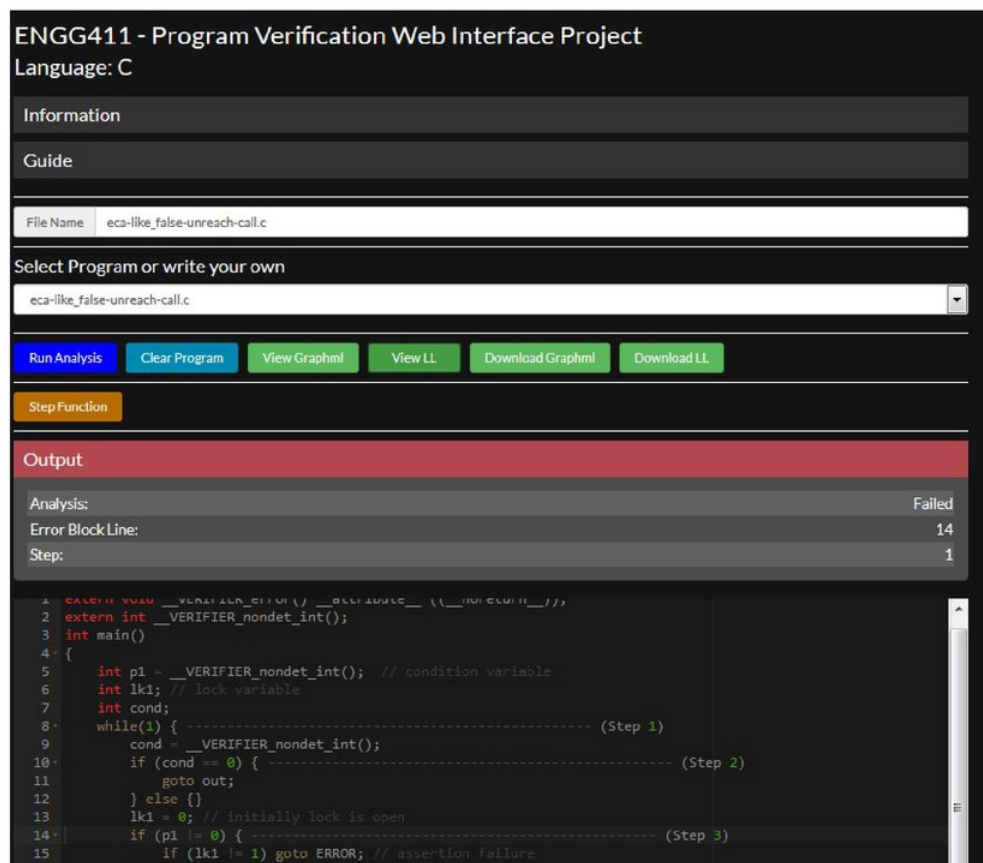
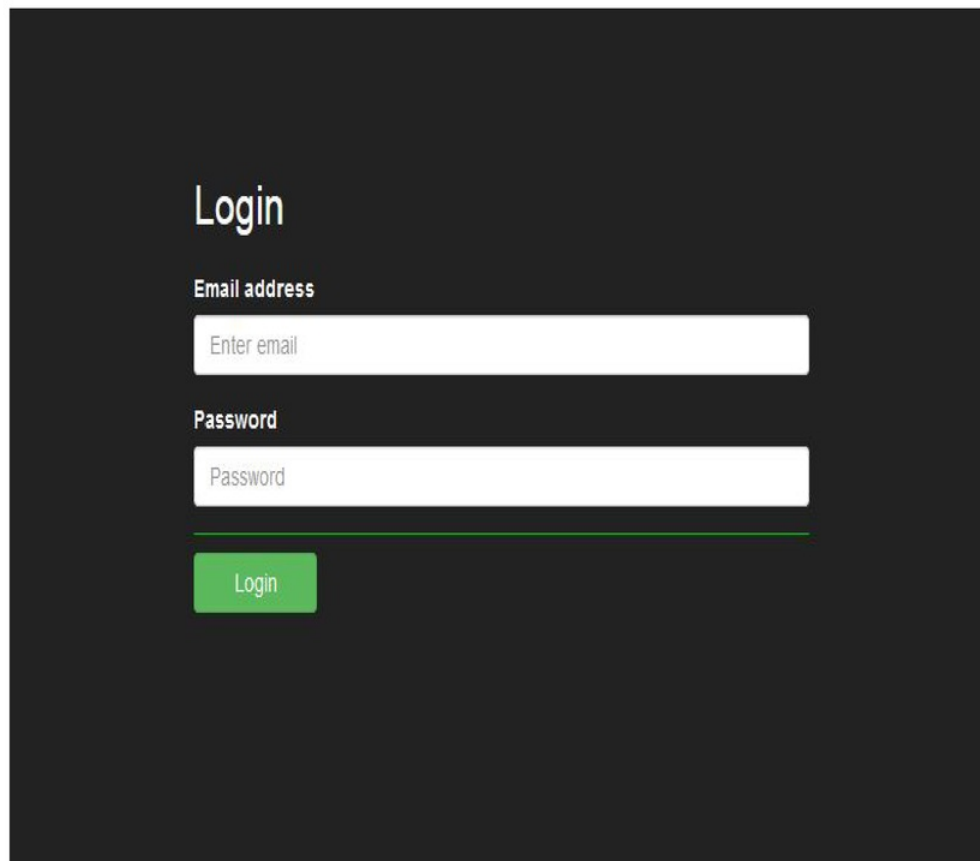


Figure 4.3: Screenshot of the Web Interface system



Login

Email address

Password

Login

Figure 4.4: Step 1: Screenshot of Login screen

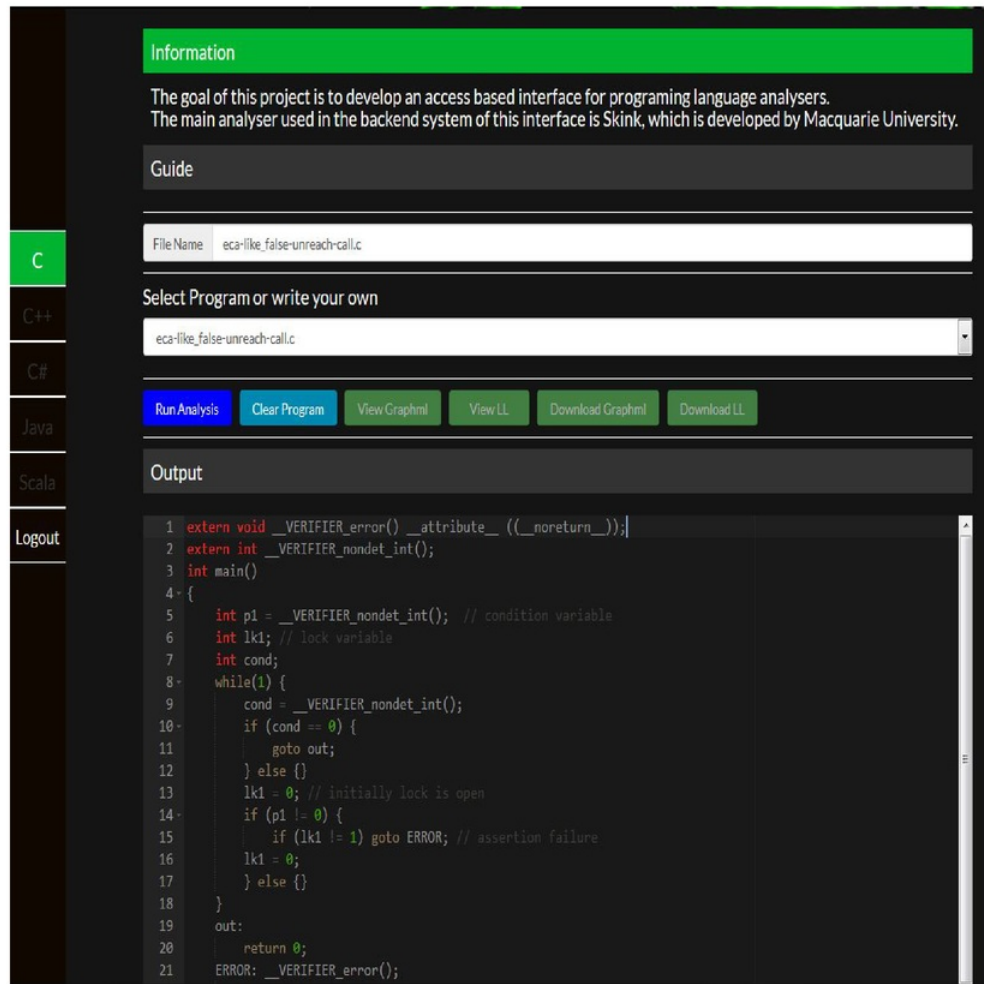


Figure 4.5: Step 2: Screenshot of Analyser Screen

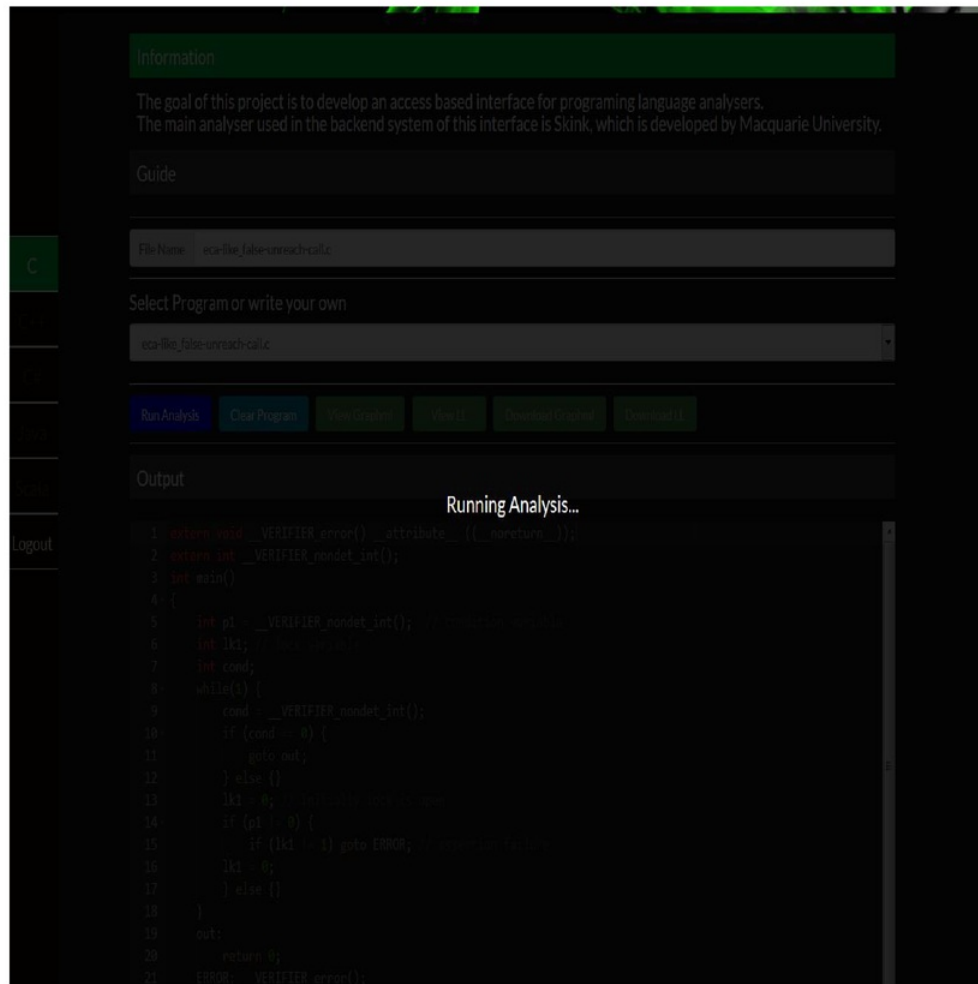


Figure 4.6: Step 3: Screenshot Running Analysis

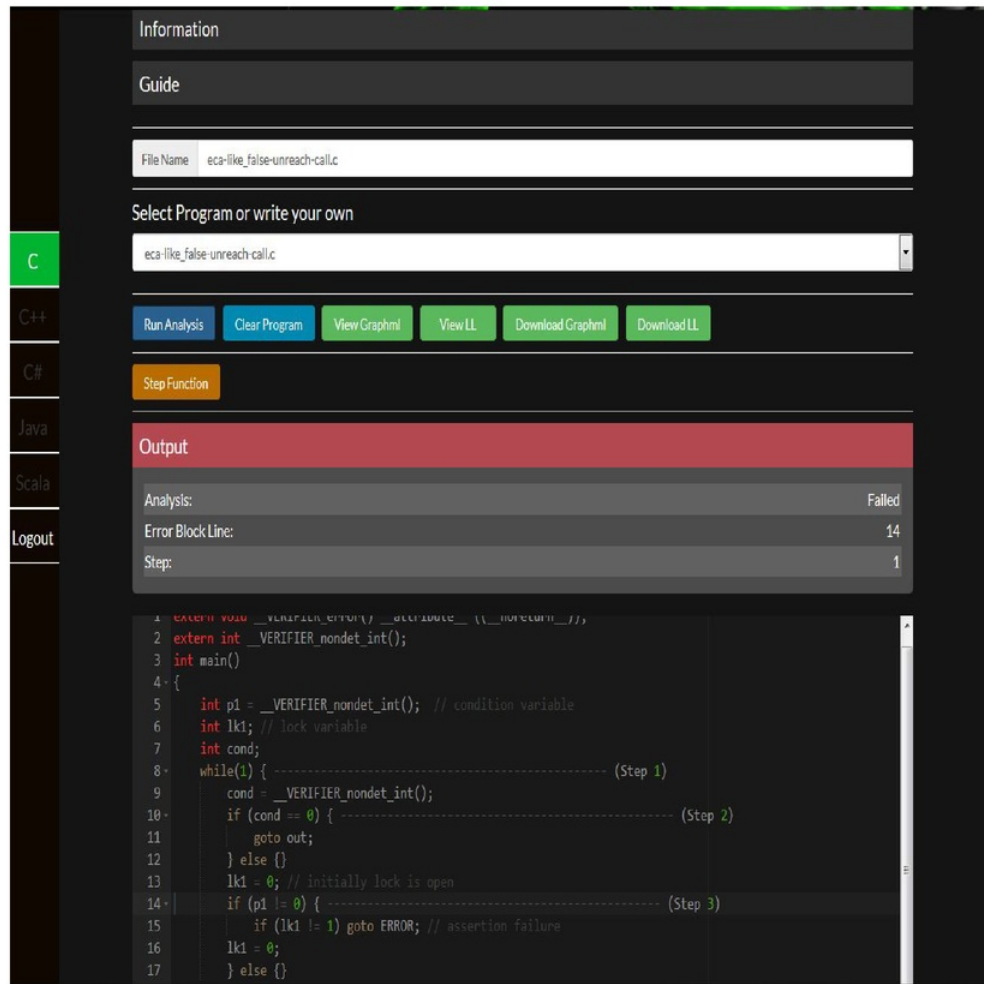


Figure 4.7: Step 4: Screenshot of Analyser Results screen

Chapter 5

Discussion

5.1 Designing the System

In this section I will detail the process that I took in designing the system. The first step to designing a software system is to understand the use and the context in which the system will be used. To do this I needed to do some background research into the field of Software Verification and use the meetings with my supervisor to understand what their use of the system would be as well as develop some project goals that I would use as a starting point in designing the system. The background literature for this project can be found in Chapter 2.

In order to begin designing the system first we need to document and understand the requirements of the system.

5.1.1 Requirements

In order to understand the requirements of the system it is required to understand the use cases of the system you want to design. Once you have a foundation with the Use Cases of the system you are then able to begin breaking down the use cases to understand the requirements of the whole system. The most important Use Case of the system was that it should be able to be completely operated by the user via a common web browser. A list of Use Cases can be found in the Use Cases subsection.

Requirements Elicitation

Requirements Elicitation is the process of gathering the requirements for a project. There are many popular methods for achieving this some of these ways are:

- Document Analysis and Background Research - This was one of the main methods that I used to get an understanding of the topics this project was about. Getting an understanding of the component systems in the project helped me reach a better understanding of the project goals by understanding how the systems interacted and their usage/platforms helped guide the design of the system environment.

- Interviews or Focus Groups - The meetings with my supervisor who is one of the main end users of the proposed system was a vital part in helping me shape my understanding of the requirements of this project. Asking questions about expected functionality and use cases helped me design the system to meet the goal of the project more accurately.
- Observation - Observation in this context means observation of existing old or similar systems to get a better understanding of the expected system and help develop requirements. In this project I observed and used the Wiley Web [4] system to get an idea of how they designed the front-end of their system.
- Survey/Questionnaire - This method is often used when a proposed system targets a large and often diverse user base. The use of surveys and questionnaires helps gather data that can then be analysed to get a better understanding of the user requirements of the system. This was not a viable option in this project as the user base for the main program analyser tool was small.
- Brainstorming - During the early stages of this project I used brainstorming to come up with and filter out ideas to narrow down the scope of the project at its initial stage in order to have a solid starting point for the project.

5.1.2 User Requirements and defining the Use Cases

This subsection will outline the Use Cases I developed and used in designing the system. The main goals and expected outcomes of this project were vague in the early stages of the project, due to the lack of solid goals and outcomes for the project, and as progress was made prototyping and developing the system it became more and more concrete. The main way this happened was through constant communication and feedback from the program supervisor, who was also an end-user tester of the system. The following will define the terms and structure of the use case documentation.

- Goal - The Goal describes the expected outcome of the use case.
- Primary Actor - The primary actor in this project is the sole end user that uses the user interface in the web interface system.
- Pre-Condition - The condition of the system prior to this use case.
- Post-Condition - The condition of the system after the use case.
- Failure Outcomes - What are the possible failure outcomes in this use case.
- Flow of Events - The flow of events the user takes.

5.1.3 Use Cases

Here are the two main use cases used to design the two main functional components of the system as seen by the user.

- The Login Event
- The Code Analysis Event

These two use cases give a good indication of the front-end requirements for the system.

For example we can extrapolate from 5.1, the login use case, that there needs to be some login page that restricts access to any other page on the site. But we don't want the user to keep logging in every time from a usability perspective so we need a way to identify if a user has a valid timed login and automatically redirect them to the where they want to go. From 5.2 we can get an idea of the users experience of the process of running analysis on their code.

These use cases do not give us the whole picture of what we need to design the system but they are a good start in understanding how it should work from a users perspective which will influence the design of the system.

5.1.4 System Requirements

This subsection details the system requirements of the system. As mentioned previously there are some basic requirement for some of the component systems such as Skink and the SMT solvers that will restrict the system environment. In particular for this project that happens to be the operating system that the back-end runs on. As mentioned because of this I have chosen to use an Ubuntu Operating system that is hosted and deployed on a Virtual machine by Digital Ocean.

5.1.5 Considerations

There are a few considerations to take into account in designing this system. They are not direct requirements but they affect the design of the system in some way or another.

Cross-browser Compatibility

The main considerations to account for cross-browser compatibility stem from the visual design of the web interface. Most of the modern popular browsers such as Edge, Chrome and Firefox, and unfortunately in some cases Internet Explorer, run on different core systems which means that in some cases they render and display web pages differently. This is because of how they understand and apply the stylesheet (CSS) for a particular page. For this project it turned out to not be a factor due to the design which is explained more in the Visual Design subsection of High level Design.

Use Case ID	UC-1
Goal	Log into Skink Web Interface System
Primary Actor	End User
Pre-Condition	<ol style="list-style-type: none"> 1. User has a valid login to the web interface 2. The user already has a valid login token as they navigate to the website
Post-Condition	<ol style="list-style-type: none"> 1. User has successfully logged into the web interface and can access the program
Failure Outcomes	<ol style="list-style-type: none"> 1. User cannot log into the system <ol style="list-style-type: none"> a. This may be due to incorrect login credentials or the user does not have an account 2. User cannot access the online web site <ol style="list-style-type: none"> a. The web site is down or unavailable
Flow of Events	
<ol style="list-style-type: none"> 1. User navigates to the Skink Web Interface 2. User types in credentials into the login fields 3. User clicks the login button 4. User then is successfully redirected, or the credentials are rejected/invalid, and the user stays on the login page 	
OR	
<ol style="list-style-type: none"> 1. User navigates to Skink Web Interface with valid login token 2. User is automatically redirected to the main Skink analyser page 	

Figure 5.1: An use case of the login for the system

Security

Security is a major concern for this project due to the fact that its main purpose is to send user code to the server and run analysis on it. Since this project aims to generalise the implementation of the Skink analyser to more systems it may be a problem if there are some systems that run the code server side.

This inherently causes a big problem for security as malicious code can easily steal private information off or cause damage to the system. Unfortunately the main way to stop this is to restrict access to the system and log usage of the system.

In order to prevent any catastrophic failure it is also recommended to keep up to date backups of the system image and/or code.

Use Case ID	UC-2
Goal	User clicks Run Analysis button
Primary Actor	End User
Pre-Condition	<ol style="list-style-type: none"> 1. There is some amount of code in the code text editor 2. The user is logged in 3. There is a filename given
Post-Condition	<ol style="list-style-type: none"> 1. The user ends up with the analysis output on screen 2. The user has the option to download or view the skink output files
Failure Outcomes	<ol style="list-style-type: none"> 1. The analysis crashes or hangs 2. The system returns incorrect data/values
Flow of Events	
<ol style="list-style-type: none"> 1. The user enters or pastes some code into the code editor. 2. The user has entered a valid file name. 3. The user clicks the Run Analysis button. 4. The interface view updates as the analysis is complete to show the results. 5. The user can then also view or download the skink output 	

Figure 5.2: A use case of running analysis on use code

System hand-off and takeover

Another consideration is the inevitable hand-off of or replication of the system after the completion of this project. The easiest method to tackle this apart from replicating the system using this document and provided code is to provide an image bundle of the server that can then be redeployed and configured.

Use Case ID	UC-3
Goal	User logs out
Primary Actor	End User
Pre-Condition	1. User is logged in
Post-Condition	2. The user is logged out so that another person cannot access the system
Failure Outcomes	1. User cannot be logged out and the system can be accessed by other person with access to the device
Flow of Events	
1. The user clicks the log out button 2. The server invalidates the user's login session token 3. The browser clears/invalidates the stored login token	

Figure 5.3: A use case of logging out of the system

5.2 High-Level Design

5.2.1 Initial System Design

There were two main contenders that the design would be implemented with on the server side (back-end), C# .Net Core or PHP.

For this project I have chosen to use PHP instead of C# .Net Core because it has all the functionality that is required for this project, as well as it being easily manageable in small files and very easy to install. C# .Net Core on the other hand requires installation of specific IDEs that support it as well as the compilers and runtime in order to get it working. It also comes with a lot of required files and folder structure that make the project much more complex.

Using PHP will allow me to have very small functional files that can quickly and easily be modified and do not require any compilation beforehand, making the development and testing cycle much faster. Another reason is that the back-end of the system is not large and a smaller footprint is easier to manage for the small system.

5.2.2 PHP

PHP is also a widely used stable library with many available packages for download. It does not come with a lot of the boilerplate and system structure that C# .Net requires

Use Case ID	UC-4
Goal	Add/Delete/View Users
Primary Actor	Privileged User
Pre-Condition	1. User has correct privilege
Post-Condition	2. User has successfully added/removed/viewed a user or users
Failure Outcomes	1. User fails to add/remove/view users of the system
Flow of Events	
1. User navigates to <u>users</u> page 2. User views users in a list 3. User selects add user and creates a user login OR 4. User selects a user and removes them from the system	

Figure 5.4: A use case of a privileged user adding/removing/viewing system users

and is easy to implement and use.

5.2.3 C# .Net Core

The C# .Net Core is a widely used stable software package developed by Microsoft that has inbuilt libraries to handle many aspects of web development as well as many tools and utilities for developing server side software. It can run cross-platform and has the ability to achieve almost anything that requires implementation.

Fig.5.5 gives a high-level view of the prototype design of the Web interface system.

5.2.4 Project Code

The project code in its current state can be found in Appendix B. This may not be a complete listing of all the code required to deploy the project as some of the PHP files contain security sensitive information and functions as well as database scripts.

5.2.5 Visual Design

For my design of the interface, it was a requirement for the interface to be easy to use and understand; this meant that I had to minimise the amount of on screen elements and

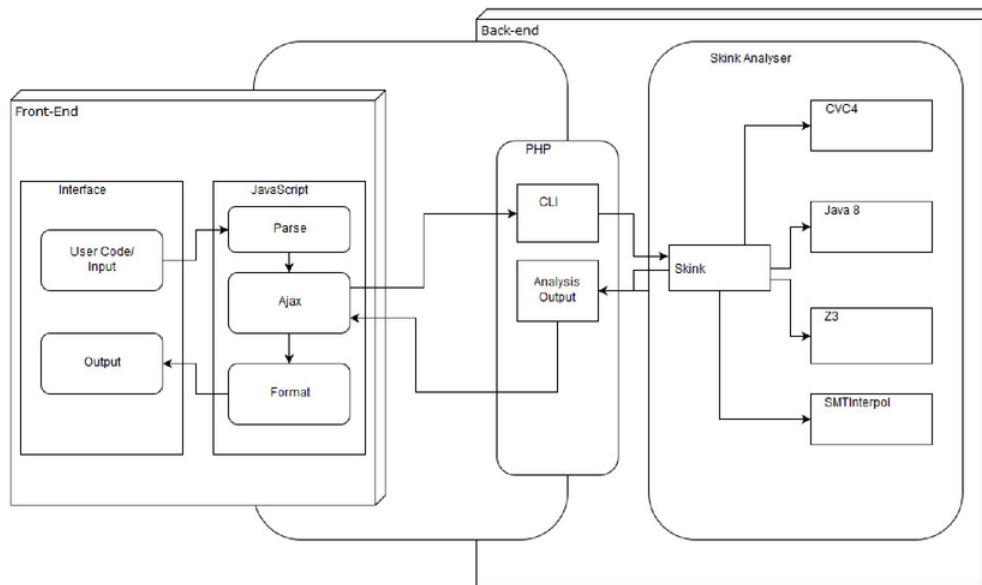


Figure 5.5: A high-level overview of how the system will be implemented

information presented to the user at any given time. My method for tackling this was to hide away any unnecessary information the user did not need to see.

This was accomplished by using accordion drawers that animate and hide information and buttons/elements that are disabled. In order to accomplish this I had to use a colour palette that allows for easy contrast. In this project I decided to use a dark colour for the theme with green/blue/red highlights and white text as seen in 5.6. This combination was chosen after iterations of tweaking. Now when the system is being used active items are clearly highlighted, important information is easily distinguished and disabled features are not able to be interacted with and greyed out.

In order to give the user visual feedback for their actions, in particular to understand that the analyser is running I implemented a loading screen that shows as the analyser is running and disappears when the results have returned.

5.3 Final Design

This section will detail the final design of the system using software UML diagrams.

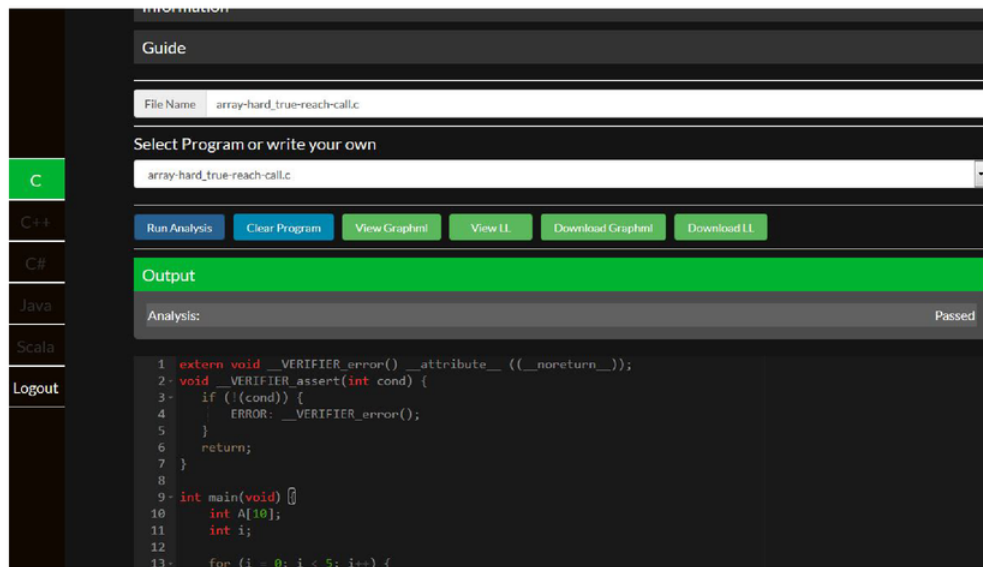


Figure 5.6: A screenshot of the web design colour palette

5.4 Components

5.4.1 Front End

The front end of the system consists of a login component and an analysis page component. Any request to the server calls a user authenticate function which checks for a valid user login token. If the token is not found or not valid the user is redirected to the login screen.

When the user successfully logs in a cookie is stored on their browser that expires in one hour. The same token is then stored against their login ID so that the server knows what to validate against when checking for user authentication.

The default behaviour after successful authentication is to redirect to the code analyser page. This page has been designed as generic as possible so that with the menu on the left you can easily switch between analysing tools using the same web elements, such as the code editor/filename/parameters.

The user can then enter their desired code and information and click a simple Run Analysis button that will send the analysis request to the server. When the web page receives the return success or error response it then proceeds to fetch the corresponding output files for the request.

The xml file is then parsed and the nodes that correspond to the success or failure of the analysis is found and displayed to the user. The output section turns red to indicate an error is found in the code or green if the analysis deems the code is correct.

The user can then step through the code lines that it returns (for the skink analyser)

to see the steps it took through the program.

The user can then modify their code/ clear it and run another analysis.

5.4.2 Back End

The back end of the system acts as a typical php web server that handles web requests to it. Every web request that is sent to it is authenticated to ensure the user has valid access to the system.

Then depending on the request that is send it will either redirect to the login page or the program analyser page. The program analyser page then can send an AJAX request to the server that bundles the code and relevant data from the interface and gives it to the analyser function. This function generated a c file from the data and passes that to the Skink system. The Skink system outputs a graphml file which I have modified to output to .xml instead, and a ll file. The function then sends a response to the web page with the unique id of the files so that the web page can then access those files, parse and display relevant information to the user.

5.5 UML Diagrams

This subsection contains two uml diagrams to further help convey the designs of the system.

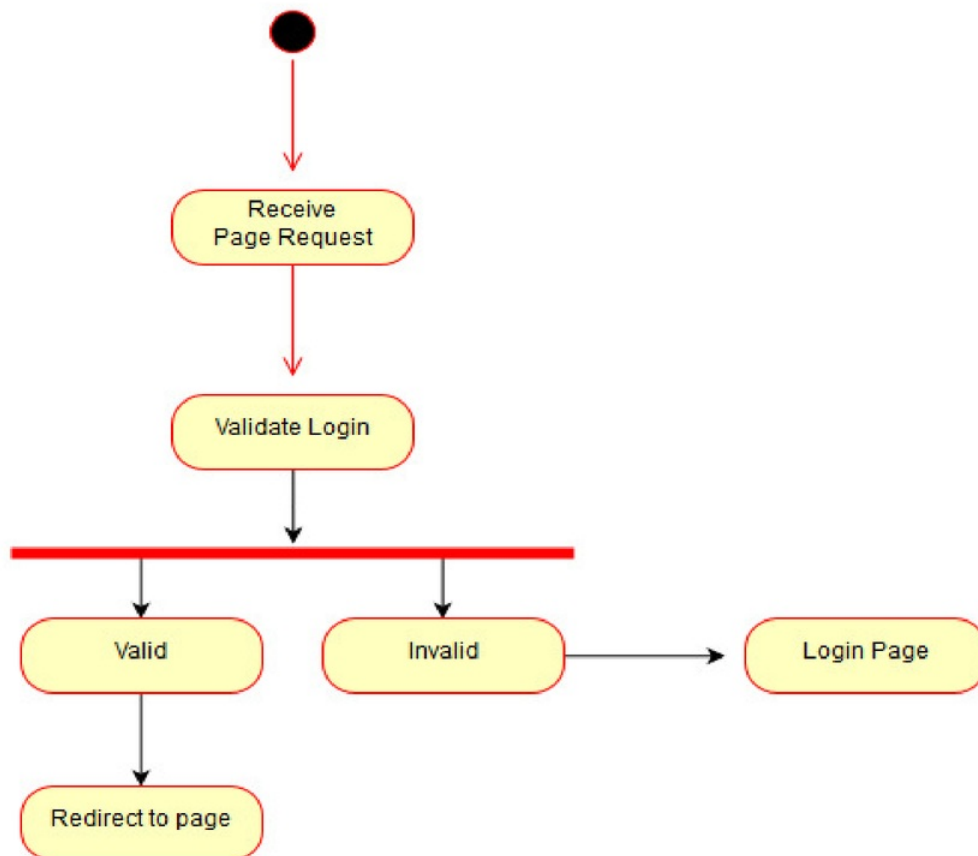
5.5.1 Behavioural UML Diagrams

The activity diagrams detail the two main activity processes the server completes currently.

Activity Diagrams

Use Case Diagram

This use case diagram 5.9 details the current basic usage that users can have of the system. This can be expanded upon after the end of the project in future work.

**Figure 5.7:** Login Activity Diagram

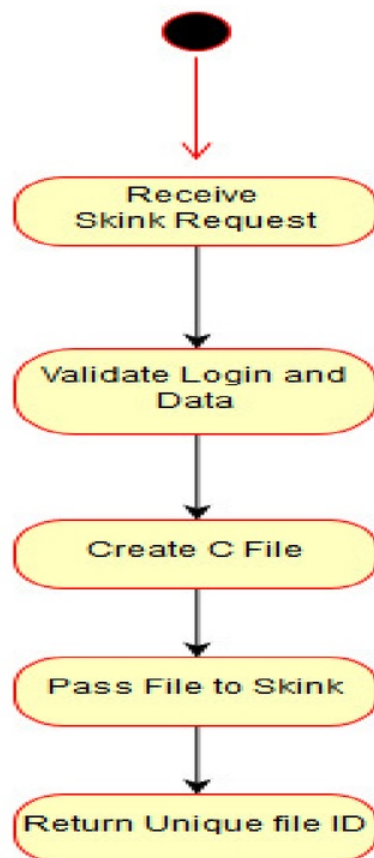


Figure 5.8: Skink Activity Diagram

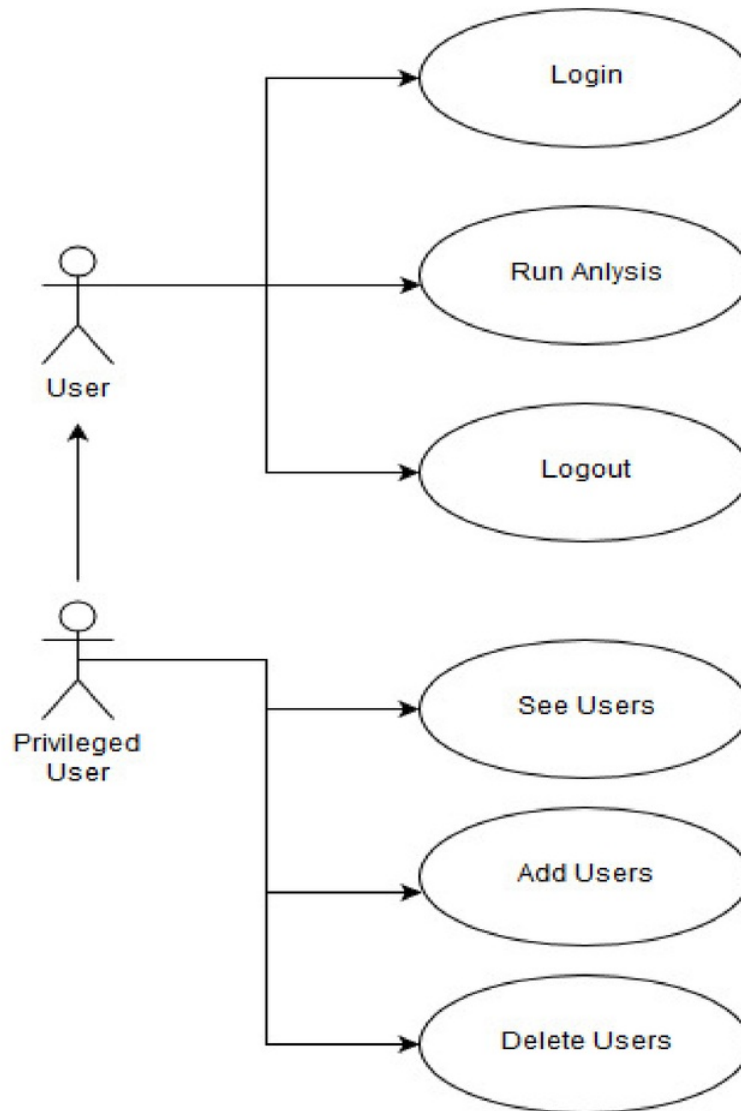


Figure 5.9: The use case diagram of the system

Chapter 6

Conclusion

In conclusion the designed and developed system is at a good, stable point that achieves the goals that were set out in this project. We are able to allow a user that only has access to a browser, validate their login, run analysis on their code, return and extract meaningful data from the output of the skink analyser without having the user install any program or have an understanding of the underlying system.

This project is important because it takes the field of software verification that is usually so niche and difficult for the average developer to access and makes it readily and easily available through a popular medium. This kind of system allows the spread of the field to more and more people which can in turn directly affect the funding and progress in development of the software verification systems.

Chapter 7

Future Work

In this section I will detail potential future work that can be implemented onto the current system.

7.1 Analyser Analytics

With the setup of the MySQL database it would be easy to add a table to log the analysis requests that are sent to the server. With added functionality we can determine the type of program sent, the analysis result, the time taken and other metrics to help the developers of the program analysis tool see and understand potential problems and issues with their system.

7.2 Code Syntax Checking

Currently there is no syntax checking in the online code editor to warn the user of syntax errors in their code. So the user will send their code for analysis and get an error result that does not indicate a bug but rather that their code is not correctly formatted. This was not in the scope of the project but it is a nice tool to have and its implementation may be trivial if there is an external javascript library that has this functionality.

7.3 Adding per user Code sample saving

If a user wants to test the analysis system using a complex piece of program code, currently they would need to copy and paste that code into the editor every time they start a new session with the interface. This can be implemented using per user directory storage or direct database text storage.



Chapter 8

Abbreviations

AJAX	Asynchronous Javascript and XML
CSS	Cascading StyleSheet
HTML	Hyper Text Transfer Protocol
IDE	Integrated Development Environment
LAMP	Linux Apache MySql PHP
LLVM-IR	LLVM - Intermediate Representation (LLVM is not an acronym)
PHP	PHP: Hypertext Preprocessor
SMT	Satisfiability Module Theories
SAT	Boolean Satisfiability Problem

Appendix A

Project Plan

A.1 Project Gantt Chart

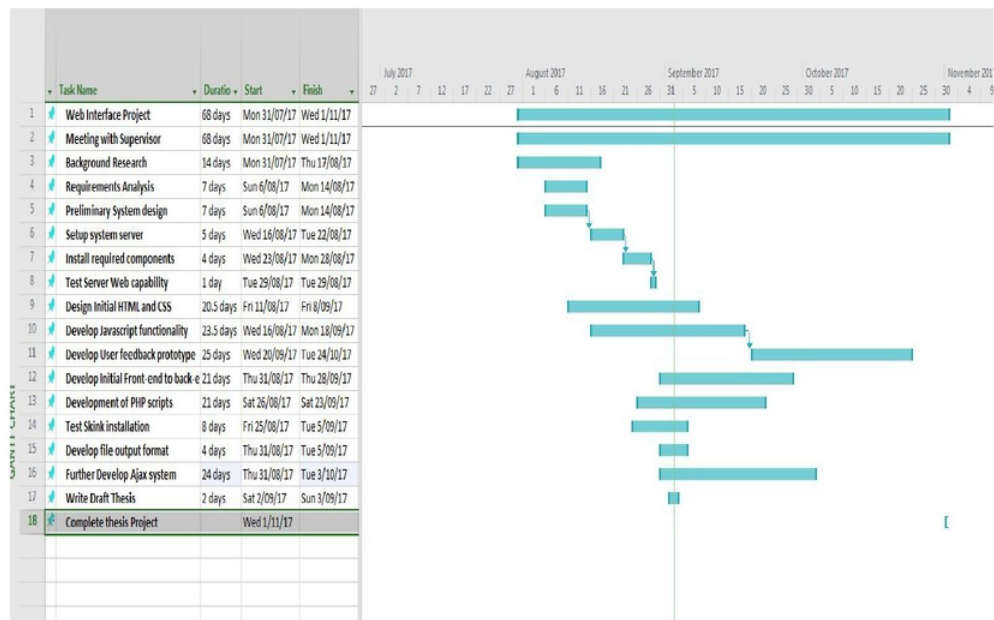


Figure A.1: Gantt Chart of expected Project Timeline

Appendix B

Project Code

B.1 HTML

```
1 <head>
2   <base href="<?php echo base_url(); ?>" />
3   <meta name="viewport" content="width=device-width"/>
4   <meta charset="utf-8"/>
5   <title>ENGG411 - C File Analysis</title>
6
7   <!-- Latest compiled and minified CSS -->
8   <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/
9     bootstrap/3.3.7/css/bootstrap.min.css"
10     integrity="
11       sha384-BVYiISiFeK1dGmJRAkycuHAHRg320mUcww7on3RYdg4Va+
12       PmSTsz/K68vbdEjh4u" crossorigin="anonymous">
13   <link href='http://fonts.googleapis.com/css?family=Lato&
14     subset=latin,latin-ext' rel='stylesheet' type='text/css'>
15
16   <link rel = "stylesheet" type = "text/css" href = "css/reset.css">
17   <link rel = "stylesheet" type = "text/css" href = "css/styles.css"
18   >
19 </head>
20
21 <body>
22   <div class="container">
23     <div class="header">
24       <h1>ENGG411 - Program Verification Web Interface Project</h1>
25       <h2>Language: <span id="chosenProgramLanguage">C</span></h2>
26
27       <div class="moreInfobtn inactiveBtn" data-toggle="collapse"
28         data-target="#extraInformation" aria-expanded="false"
29         aria-controls="collapseExample">
30         Information
31       </div>
32
33       <div class="collapse" id="extraInformation">
```

```

27     <div class="moreInfoContent">
28         <p>The goal of this project is to develop an access based
29         interface for programing language analysers.</p>
30         <p>The main analyser used in the backend system of this
31         interface is Skink, which is developed by Macquarie
32         University.</p>
33     </div>
34 </div>
35
36
37 <div class="guidebtn inactiveBtn" data-toggle="collapse"
38     data-target="#guideInformation" aria-expanded="false"
39     aria-controls="collapseExample">
40     Guide
41 </div>
42
43 <div class="collapse" id="guideInformation">
44     <div class="moreInfoContent">
45         <p>Site Usage:</p>
46         <div class="guideList">
47             <div><div class="listMarker"></div>Ensure a file name is
48             entered</div>
49             <div style="display: none;"><div class="listMarker"></div>
50             Include input paramaters (if any)</div>
51             <div><div class="listMarker"></div>Develop or Paste your
52             code snippet into the text editor</div>
53             <div><div class="listMarker"></div>Press Run Analysis and
54             follow prompts</div>
55             <div><div class="listMarker"></div>View results</div>
56             <div style="display: none;"><div class="listMarker"></div>
57             Step through program trace (optional)</div>
58         </div>
59     </div>
60 </div>
61
62 <div class="code-analyser-div">
63     <div class="code-analyser-info input-group">
64         <span class="input-group-addon" id="basic-addon1">File
65         Name</span>
66         <input id="file-name" type="text" class="form-control"
67         placeholder="eg. ProgramName" aria-describedby="
68         basic-addon1">
69     </div>
70     <div class="code-analyser-info input-group" style="display:
71     none">
72         <span class="input-group-addon" id="basic-addon1">Input
73         Parameters</span>
74         <input id="parameters" type="text" class="form-control"
75         placeholder="eg: -t 1 --example" aria-describedby="
76         basic-addon1">

```

```

62     </div>
63     <div class="form-group">
64     <label for="codeSelect" id="codeSelectLabel">Select Program or
        write your own</label>
65     <select class="form-control" id="codeSelect">
66         <option value="0">array-hard_true-reach-call.c</option>
67         <option value="1">array-hard_true-unreach-call.c</option>
68         <option value="2">array-sequence_true-unreach-call.c</option>
69         <option value="3">count-up-down_false-unreach-call.c</option>
70         <option value="4">count-up-down_true-unreach-call.c</option>
71         <option value="5">eca-like_false-unreach-call.c</option>
72         <option value="6">simple-if_true-unreach-call.c</option>
73     </select>
74 </div>
75 </div>
76
77
78
79
80     <div class="row control-box">
81         <div class="code-analyser-buttons col-md-12">
82             <div class="control-buttons">
83                 <button id="run-button" class="btn
                        btn-default btn-primary">Run Analysis</
                        button>
84                 <button id="clear-button" class="btn
                        btn-default btn-info">Clear Program</
                        button>
85                 <a class="dl-graphml-button-anchor" target="
                        _blank"><button class="dl-graphml-button
                        btn btn-default btn-success" disabled>View
                        Graphml</button></a>
86                 <a class="dl-ll-button-anchor" target="
                        _blank"><button class="dl-ll-button btn
                        btn-default btn-success" disabled>View
                        LL</button></a>
87                 <a class="dl-graphml-button-anchor" download
                        ><button class="dl-graphml-button btn
                        btn-default btn-success" disabled>
                        Download Graphml</button></a>
88                 <a class="dl-ll-button-anchor" download>
                        <button class="dl-ll-button btn
                        btn-default btn-success" disabled>
                        Download LL</button></a>
89             </div>
90             <div class="method-buttons collapse">
91                 <button id="step-function" class="btn
                        btn-default btn-warning">Step Function</
                        button>
92             </div>
93         </div>
94     </div>

```

```

95
96     <div class="outputCollapsebtn" data-target="#outputCollapse"
97         aria-expanded="false" aria-controls="">
98         Output
99     </div>
100
101 <div class="collapse" id="outputCollapse">
102     <div class="output">
103         <p id="analysisOutput" class="passed">Analysis: <span>Passed</span></p>
104         <p class="displayNone" id="errorLineP">Error Block Line: <span id="errorLineS"></span></p>
105         <p class="displayNone" id="errorStep">Step: <span id="currentFunc"></span></p>
106     </div>
107 </div>
108 <p id="xmlOutput" style="background: white; color: black; width:
109     100%; min-height:40px; display: none;"> </p>
110
111     <div class="code-analyser-text-box">
112         <div class="row">
113             <div class="col-12">
114                 <div id="editor"></div>
115             </div>
116         </div>
117
118         <div class="left-side-selector">
119             <div>
120                 <div class="left-side-selector-list">
121                     <p class="languageSelector active selected">C</p>
122                     <p class="languageSelector disabled"
123                         data-toggle="tooltip" title="Not Yet Implemented">C++</p>
124                     <p class="languageSelector disabled"
125                         data-toggle="tooltip" title="Not Yet Implemented">C#</p>
126                     <p class="languageSelector disabled"
127                         data-toggle="tooltip" title="Not Yet Implemented">Java</p>
128                     <p class="languageSelector disabled"
129                         data-toggle="tooltip" title="Not Yet Implemented">Scala</p>
130                 <div class="logout"><p>Logout</p></div>
131             </div>
132         </div>
133     </div>

```



```

133     <div id="loadScreen" class="displayNone"><div id="loadSpinner"><p>
        Running Analysis...</p></div></div>
134
135
136 <!-- jQuery -->
137 <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/
        jquery.min.js"></script>
138
139 <!-- Latest compiled and minified JavaScript -->
140 <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/
        bootstrap.min.js"
141         integrity="
            sha384-Tc5IQib027qvyjSMfHjOMaLkfuWVxZxUPnCJA712mCWNIpG9mGCD8wGNIC
            "
142         crossorigin="anonymous"></script>
143
144 <!-- ACE Code Editor-->
145 <script src="https://cdn.jsdelivr.net/ace/1.2.6/min/ace.js"></script>
        >
146
147 <script type = 'text/javascript' src = "<?php echo base_url(); ?>js/
        scripts.js"></script>
148 </body>

```

B.2 CSS

```

1  html {
2      background-color: #141414;
3  }
4
5  body {
6      padding: 0;
7
8      max-width: 1170px;
9      margin: 0 auto;
10     font-size: 20px;
11     color: white;
12     font-family: Lato;
13 }
14
15 /*----- Code Analyser Interface ----- */
16
17 .container {
18     width: 100%;
19     background-color: #141414;
20     max-width: 100%;
21     padding: 0;
22     margin-top: 30px;
23     padding-left: 70px;
24     padding-right: 70px;
25     height: 100%;

```

```
26 }
27
28 .col-lg-1, .col-lg-10, .col-lg-11, .col-lg-12, .col-lg-2, .col-lg-3,
    .col-lg-4, .col-lg-5, .col-lg-6, .col-lg-7, .col-lg-8, .col-lg-9,
    .col-md-1, .col-md-10, .col-md-11, .col-md-12, .col-md-2, .
    col-md-3, .col-md-4, .col-md-5, .col-md-6, .col-md-7, .col-md-8,
    .col-md-9, .col-sm-1, .col-sm-10, .col-sm-11, .col-sm-12, .
    col-sm-2, .col-sm-3, .col-sm-4, .col-sm-5, .col-sm-6, .col-sm-7,
    .col-sm-8, .col-sm-9, .col-xs-1, .col-xs-10, .col-xs-11, .
    col-xs-12, .col-xs-2, .col-xs-3, .col-xs-4, .col-xs-5, .col-xs-6,
    .col-xs-7, .col-xs-8, .col-xs-9 {
29     padding: 0;
30 }
31
32 .row {
33     margin: 0;
34 }
35
36 .header {
37     margin-top: 20px;
38 }
39
40 .header > h1 {
41     margin-bottom: 10px;
42     font-size: 28px;
43 }
44
45 .header > h2 {
46     margin-bottom: 20px;
47     font-size: 24px;
48 }
49 .moreInfobtn, .guidebtn {
50     border: 0;
51     background-color: #333333;
52     cursor: pointer;
53     padding: 10px;
54     margin-bottom: 10px;
55     transition: all 0.3s;
56 }
57
58 .outputCollapsebtn {
59     border: 0;
60     background-color: #333333;
61     padding: 10px;
62     margin-bottom: 10px;
63     transition: all 0.3s;
64 }
65
66 .form-group {
67     border-top: 1px solid white;
68 }
69
```

```
70 #codeSelectLabel {
71     margin-bottom: 10px;
72     margin-top: 10px;
73 }
74
75 .outputCollapsebtn {
76     margin-bottom: 0;
77     margin-top: 10px;
78 }
79
80 .inactiveBtn:hover {
81     background-color: #222222;
82 }
83
84 .moreInfoContent {
85     margin-bottom: 15px;
86     padding-left: 10px;
87 }
88
89 .languageSelector {
90     color: white;
91 }
92
93 .guideList {
94     margin-top: 10px;
95 }
96
97 .guideList > div {
98     padding-left: 20px;
99     margin-top: 5px;
100 }
101
102 .listMarker {
103     display: inline-block;
104     margin-right: 10px;
105     width: 10px;
106     height: 10px;
107     background-color: transparent;
108     border-top: 2px solid #00b432;
109     border-right: 2px solid #00b432;
110     transform: rotate(45deg);
111 }
112
113 .control-box {
114     margin-top: 20px;
115 }
116
117 .code-analyser-info {
118     font-size: 24px;
119     vertical-align: center;
120     margin-bottom: 10px;
121 }
```

```
122
123 .code-analyser-div {
124     width: 100%;
125     background-color: transparent;
126     margin-top: 20px;
127     border-top: 1px solid white;
128     padding-top: 10px;
129 }
130
131 .displayNone {
132     display: none;
133 }
134
135 .control-buttons {
136
137 }
138
139 .method-buttons {
140     border-top: 1px solid white;
141     padding-top: 10px;
142     margin-top: 10px;
143 }
144
145 .btn {
146     margin-right: 6px;
147     padding: 5px 15px;
148     min-width: 100px;
149 }
150
151 .btn-primary {
152     background-color: #0000ff;
153     border-color: #0000ff;
154 }
155
156 .btn-info {
157     background-color: #0088b1;
158     border-color: #0088b1;
159 }
160
161 .btn-warning {
162     background-color: #b96d00;
163     border-color: #b96d00;
164 }
165
166 .btn-danger {
167     background-color: #5eda00;
168     border-color: #5eda00;
169 }
170
171 .upload {
172     color: #ffffff;
173     margin-bottom: 10px;
```

```
174 }
175
176 .code-analyser-buttons {
177     padding-top: 10px;
178     padding-bottom: 10px;
179     border-top: 1px solid white;
180     border-bottom: 1px solid white;
181 }
182
183 .code-analyser-text-box {
184     margin-top: 20px;
185 }
186
187
188 #editor {
189     max-width: 100%;
190     width: 100%;
191     min-width: 100%;
192     min-height: 450px;
193     max-height: 770px;
194     font-size: 16px;
195     margin-bottom: 50px;
196 }
197
198 .output {
199     background-color: #4c4c4c;
200     border-radius: 0 0 5px 5px;
201     padding: 15px;
202     padding-left: 15px;
203     padding-right: 15px;
204     font-size: 16px;
205     line-height: 150%;
206     margin-bottom: 10px;
207 }
208
209 .passed {
210     background-color: #00b431;
211 }
212
213 .unknown {
214     background-color: #b48d3d;
215 }
216
217 .failed {
218     background-color: #b44851;
219 }
220
221 .output > p {
222     padding: 2px;
223 }
224
225 .output > p > span {
```

```
226     float: right;
227 }
228
229 .output p:nth-child(odd){
230     background-color: #616161;
231 }
232
233 /* --- Buttons --- */
234 #upload-button {
235     margin-right: 20px;
236 }
237
238 /* ----- Left side selector ----- */
239 .left-side-selector {
240     position: fixed;
241     left: 0;
242     top: 0;
243     width: 70px;
244     height: 100vh;
245     background-color: #0f0700;
246     padding-top: 200px;
247 }
248
249 .left-side-selector-list {
250     position: relative;
251     width: 100%;
252     height: auto;
253     list-style: none;
254     text-decoration: none;
255 }
256
257 .left-side-selector-list p {
258     height: 50px;
259     font-size: 20px;
260     line-height: 50px;
261     border-bottom: 1px solid white;
262     border-top: 1px solid white;
263     z-index: 2;
264     text-align: center;
265     vertical-align: middle;
266     background-color: transparent;
267     transition: background-color 0.3s;
268 }
269
270 .active {
271     cursor: pointer;
272     color: white !important;
273 }
274
275 .success {
276
277 }
```

```
278
279 .unknown {
280
281 }
282
283 .error {
284
285 }
286
287 .left-side-selector-list .active:hover {
288     background-color: #76c960;
289 }
290
291 .disabled {
292     cursor: default;
293     color: #313131;
294     transition: all 0.3s;
295 }
296
297 .disabled:hover {
298     background-color: #444444;
299 }
300
301 .left-side-selector-list p.selected {
302     background-color: #00b431;
303 }
304
305
306 /* top header */
307 .fixed-top-header {
308     width: 500px;
309     height: 50px;
310     position: fixed;
311     left: 50%;
312     transform: translateX(-50%);
313     top: -28px;
314     background-color: #990000;
315     cursor: pointer;
316     transition: all 0.3s;
317     color: white;
318     border-radius: 0px 0px 5px 5px;
319 }
320
321 .top-zero {
322     top: 0;
323 }
324
325 .logout {
326     cursor: pointer;
327     transition: all 0.3s;
328 }
329
```

```
330 .logout:hover {
331     background-color: red;
332 }
333
334 .logout > p {
335     text-align: center;
336     font-size: 18px;
337 }
338
339 .drop-down {
340     transition: all 0.3s;
341 }
342
343 .drop-down:hover {
344     background-color: red;
345 }
346
347 .drop-down > p {
348     text-align: center;
349     vertical-align: middle;
350 }
351
352 #loadScreen {
353     position: fixed;
354     top: 0;
355     left: 0;
356     width: 100%;
357     height: 100vh;
358     background-color: rgba(0,0,0,0.85);
359     z-index: 10000;
360 }
361
362 #loadSpinner {
363     position: fixed;
364     top: 50%;
365     left: 50%;
366     transform: translate(-50%,-50%);
367     color: white;
368     font-size: 22px;
369 }
```

B.3 JavaScript

```
1 var currentLine = 0;
2 var functions = [];
3
4 var outputReady = false;
5
6 var templateCode = [
7     "extern void __VERIFIER_error() __attribute__ ((__noreturn__));
      \n" +
```



```

8      "void __VERIFIER_assert(int cond) {\n" +
9      "    if (!(cond)) {\n" +
10     "        ERROR: __VERIFIER_error();\n" +
11     "    }\n" +
12     "    return;\n" +
13     "}\n" +
14     "\n" +
15     "int main(void) {\n" +
16     "    int A[10];\n" +
17     "    int i;\n" +
18     "\n" +
19     "    for (i = 0; i < 5; i++) {\n" +
20     "        A[i] = i;\n" +
21     "    }\n" +
22     "\n" +
23     "    __VERIFIER_assert(A[4] == 4);\n" +
24     "}"
25 ,
26 "extern void __VERIFIER_error() __attribute__ ((__noreturn__)); \n"
27 " +
28 "int main(void) {\n" +
29 "    int A[10];\n" +
30 "    int i;\n" +
31 "    for (i = 0; i < 5; i++) {\n" +
32 "        A[i] = i;\n" +
33 "    }\n" +
34 "    if (A[4] != 4) __VERIFIER_error();\n" +
35 "}"
36 ,
37 "extern void __VERIFIER_error() __attribute__ ((__noreturn__)); \n"
38 " +
39 "int main(void) {\n" +
40 "    int A[10];\n" +
41 "    int i;\n" +
42 "    for (i = 0; i < 5; i++) {\n" +
43 "        A[i] = i;\n" +
44 "    }\n" +
45 "    if (A[4] != 4) __VERIFIER_error();\n" +
46 "}"
47 ,
48 "extern void __VERIFIER_error() __attribute__ ((__noreturn__));\n"
49 " +
50 "unsigned int __VERIFIER_nondet_uint();\n" +
51 "int main()\n" +
52 "{\n" +
53 "    unsigned int n = __VERIFIER_nondet_uint();\n" +
54 "    unsigned int x=n, y=0;\n" +
55 "    while(x>0)\n" +
56 "    {\n" +
57 "        x--;\n" +
58 "        y++;\n" +
59 "    }\n" +

```

```

57     " if (y == n) __VERIFIER_error();\n"+
58     "}\n"
59 ,
60     "extern void __VERIFIER_error() __attribute__ ((__noreturn__));\n"
61     +
62     "unsigned int __VERIFIER_nondet_uint();\n"+
63     "int main()\n"+
64     "{\n"+
65     "  unsigned int n = __VERIFIER_nondet_uint();\n"+
66     "  unsigned int x = n, y = 0;\n"+
67     "  while(x>0)\n"+
68     "  {\n"+
69     "    x--;\n"+
70     "    y++;\n"+
71     "  }\n"+
72     "  if (y != n) __VERIFIER_error();\n"+
73     "}\n"
74 ,
75     "extern void __VERIFIER_error() __attribute__ ((__noreturn__));\n"
76     +
77     "extern int __VERIFIER_nondet_int();\n"+
78     "int main()\n"+
79     "{\n"+
80     "  int p1 = __VERIFIER_nondet_int(); // condition variable\n"+
81     "  int lk1; // lock variable\n"+
82     "  int cond;\n"+
83     "  while(1) {\n"+
84     "    cond = __VERIFIER_nondet_int();\n"+
85     "    if (cond == 0) {\n"+
86     "      goto out;\n"+
87     "    } else {\n"+
88     "      lk1 = 0; // initially lock is open\n"+
89     "      if (p1 != 0) {\n"+
90     "        if (lk1 != 1) goto ERROR; // assertion failure\n"+
91     "        lk1 = 0;\n"+
92     "      } else {\n"+
93     "        out:\n"+
94     "        return 0;\n"+
95     "      }\n"+
96     "    }\n"+
97     "  }\n"+
98     "  ERROR: __VERIFIER_error();\n"+
99     "  return 0;\n"+
100    "}\n"
101 ,
102     "extern void __VERIFIER_error() __attribute__ ((__noreturn__));\n"
103     +
104     "int __VERIFIER_nondet_int();\n"+
105     "int main ()\n"+
106     "{\n"+
107     "  int i = __VERIFIER_nondet_int(), j = __VERIFIER_nondet_int();\n"

```

```
105     " if (i > j)\n"+
106     " if (i > j) __VERIFIER_error();\n"+
107     "}"
108 ];
109
110 var currentFunc = 0;
111
112 var editor = ace.edit("editor");
113 var languageSpan = $("#chosenProgramLanguage");
114 var methodButtons = $(".method-buttons");
115 var outputCollapsible = $('#outputCollapse');
116 var outputSection = $('#div.outputCollapsebtn');
117 var codeChanged = true;
118
119 $(document).ready(function () {
120
121     initialiseEditor();
122     methodButtons.collapse("hide");
123     $("#file-name").val($(this).find(":selected").text());
124
125     $(''[data-toggle="tooltip"]').tooltip();
126
127
128     $(".drop-down").on('click', function() {
129         $(".fixed-top-header").toggleClass("top-zero");
130     });
131
132     $(".logout").on('click', function() {
133         var url = "http://vindleweb.co/index.php/Login_Controller/logout";
134
135         $.ajax({
136             type: "post",
137             url: url,
138             dataType: 'json',
139             success: function(response, status) {
140                 if(response == true) {
141                     window.location.href = 'http://vindleweb.co/';
142                 } else {
143                     //console.log("invalid");
144                 }
145             },
146             error: function(status) {
147                 window.location.href = 'http://vindleweb.co/';
148             },
149         });
150     });
151
152     $('.left-side-selector-list > p').on('click', function() {
153         if($(this).hasClass('disabled')){
154             return;
155         }
```

```

156     let selectedMode = $('.selected');
157     selectedMode.removeClass('selected');
158
159     $('html, body').animate({scrollTop: 0}, 300);
160     let selected = $(this);
161     selected.addClass('selected');
162     setSelectedLanguage(selected);
163 });
164
165 $('div.moreInfobtn').on('click', function() {
166     $(this).toggleClass('inactiveBtn');
167     $(this).toggleClass('passed');
168 });
169
170 $('div.guidedbtn').on('click', function() {
171     $(this).toggleClass('inactiveBtn');
172     $(this).toggleClass('passed');
173 });
174
175 // -----
176 // Editor Functionality -----
177 // -----
178
179 $('#run-button').on('click', function() {
180     var fname = $('#file-name').val().replace(/^[a-zA-Z0-9]/ig, "");
181     if(fname == "") {
182         alert("No filename, set filename and try again");
183         return;
184     }
185
186     if(codeChanged) {
187         currentLine = 0;
188         functions = [];
189         $("#loadScreen").removeClass("displayNone");
190         codeChanged = false;
191         $(this).prop('disabled', true);
192         outputSection.removeClass("failed");
193         outputSection.removeClass("passed");
194         hideOutputButtons();
195         $(".dl-graphml-button").prop('disabled', true);
196         $(".dl-ll-button").prop('disabled', true);
197         $("#clear-button").prop('disabled', true);
198         $("#codeSelect").prop('disabled', true);
199         $('html, body').animate({scrollTop: $(document).height()},
200             300);
201
202         var fname = $("#file-name").val().replace(/^[a-zA-Z0-9]/ig, ""
203             );
204         var parameters = $('#parameters').val();
205         var pCode = editor.getValue();
206         //console.log(pCode);
207         var url = "http://vindleweb.co/index.php/Login_Controller/

```

```
206         parseCode";
207     var graphmlFileUrl = ".xml";
208     var llFileUrl = ".ll";
209     codeChanged = true;
210
211     $.ajax({
212         type: "post",
213         url: url,
214         data: {filename: fname, code: pCode, params: parameters},
215         dataType: 'json',
216         success: function(response, status) {
217             //console.log("Error");
218             //console.log(status);
219             //console.log(response.responseText);
220             var fileLoc = response.responseText;
221             //console.log(fileLoc);
222             var graphmlFile = "/cScripts/" + fileLoc + "-" + fname + ".c" + graphmlFileUrl;
223             var llFile = "/cScripts/" + fileLoc + "-" + fname + llFileUrl;
224             var nodeXPath = "//node";
225             var edgeXPath = "//edge";
226
227             $(".dl-graphml-button").prop('disabled', false);
228             $(".dl-ll-button").prop('disabled', false);
229             $("#clear-button").prop('disabled', false);
230             $("#codeSelect").prop('disabled', false);
231             $(".dl-graphml-button-anchor").attr("href", "http://vindleweb.co"+graphmlFile);
232             $(".dl-ll-button-anchor").attr("href", "http://vindleweb.co"+llFile);
233
234
235             var xhttp = new XMLHttpRequest();
236             xhttp.onreadystatechange = function() {
237                 if (this.readyState == 4 && this.status == 200) {
238                     //console.log(xhttp.responseXML);
239                     var xmlresponse = xhttp.responseXML;
240                     //var nodes = $(xmlresponse).find("node");
241                     //console.log(nodes);
242                     /*
243                     var text = "";
244                     for (var i = 0, len = nodes.length; i < len; i++) {
245                         text += nodes[i].textContent + "<br />";
246                     }
247                     $("#xmlOutput").html(text);
248                     console.log("\n");
249                     */
250                     showResult(xmlresponse, nodeXPath);
251                 }
252             };
253         }
254     });
```

```

253     //console.log(graphmlFile);
254     xhttp.open("GET", "http://vindleweb.co"+graphmlFile, true)
255     ;
256     xhttp.send();
257     $("#run-button").prop('disabled', false);
258     $("#loadScreen").addClass("displayNone");
259 },
260 error: function(response, status) {
261     //console.log("Error");
262     //console.log(status);
263     //console.log(response.responseText);
264     var fileLoc = response.responseText;
265     //console.log(fileLoc);
266     var graphmlFile = "/cScripts/" + fileLoc + "-" + fname + ".
        c" + graphmlFileUrl;
267     var llFile = "/cScripts/" + fileLoc + "-" + fname +
        llFileUrl;
268     var nodeXPath = "//node";
269     var edgeXPath = "//edge";
270
271     $(".dl-graphml-button").prop('disabled', false);
272     $(".dl-ll-button").prop('disabled', false);
273     $("#clear-button").prop('disabled', false);
274     $("#codeSelect").prop('disabled', false);
275     $(".dl-graphml-button-anchor").attr("href", "http://
        vindleweb.co"+graphmlFile);
276     $(".dl-ll-button-anchor").attr("href", "http://vindleweb.co
        "+llFile);
277
278
279     var xhttp = new XMLHttpRequest();
280     xhttp.onreadystatechange = function() {
281         if (this.readyState == 4 && this.status == 200) {
282             //console.log(xhttp.responseXML);
283             var xmlresponse = xhttp.responseXML;
284             //var nodes = $(xmlresponse).find("node");
285             //console.log(nodes);
286             /*
287             var text = "";
288             for (var i = 0, len = nodes.length; i < len; i++) {
289                 text += nodes[i].textContent + "<br />";
290             }
291             $("#xmlOutput").html(text);
292             console.log("\n");
293             */
294             showResult(xmlresponse, nodeXPath);
295         }
296     };
297     //console.log(graphmlFile);
298     xhttp.open("GET", "http://vindleweb.co"+graphmlFile, true)
        ;

```

```

299         xhttp.send();
300
301         $("#run-button").prop('disabled', false);
302         $("#loadScreen").addClass("displayNone");
303     },
304     });
305
306
307     }
308     });
309
310     function showResult(xml, path) {
311         var txt = "";
312         if (xml.evaluate) {
313             //console.log("in show result");
314             //console.log(xml);
315             //console.log(path);
316             //console.log("\nevaluating\n");
317
318             var nodes = xml.evaluate("/*", xml, null, XPathResult.ANY_TYPE,
319                                     null);
320             var result = nodes.iterateNext();
321             //Find pass/fail
322             var nodes = $(result).find("node");
323             //console.log(nodes);
324             var passed = $(nodes).eq(nodes.length-1).prop('outerHTML');
325             //console.log(passed);
326             passed = $(passed).find("data[key='violation']").html();
327             //console.log(passed);
328
329             if(passed==="true")
330             {
331                 // Get Failure Edge/s
332                 functions = [];
333                 var edges = $(result).find("edge");
334                 var getSteps = $(result).find("edge");
335                 console.log(edges);
336                 for(var i = 0; i < edges.length; i++)
337                 {
338                     var tempEdge = $(edges).eq(i).prop('outerHTML');
339                     var tempEdgeLine = $(tempEdge).find("data[key='startline']").
340                         html();
341                     console.log(tempEdge);
342                     console.log(tempEdgeLine);
343                     functions.push([tempEdgeLine, tempEdgeLine, 1]);
344                     //editor.insert({row: tempEdgeLine, column:
345                         editor.session.getLine(row).length + 1}, "(Step " +
346                         functions[currentFunc][0] + ")");
347                     var customPos = { row: tempEdgeLine-1, column:
348                         editor.session.getLine(tempEdgeLine-1).length + 1}
349                     editor.session.insert(customPos, "
350                         ----- (Step "

```

```

        + (i+1) + "));
345     }
346     console.log(functions);
347     //console.log(edges);
348     var errorLine = $(edges).eq(edges.length-1).prop('outerHTML');
349     //console.log(errorLine);
350     errorLine = $(errorLine).find("data[key='startline']").html();
351     //console.log(errorLine);
352     if(errorLine === "" || errorLine == null) {
353         errorLine = "Error line not found, check Graphml";
354         if(!$("#errorStep").hasClass("displayNone")) {
355             $("#errorStep").addClass("displayNone");
356         }
357     }
358     $("#errorLineS").html(errorLine);
359     editor.gotoLine(errorLine);
360     } else {
361         passed = "false";
362     }
363     }
364     showOutputButtons(passed);
365     $("#xmlOutput").html(passed);
366     //console.log(passed);
367     }
368
369     function showOutputButtons(pass) {
370         if(pass === "false") {
371             outputSection.removeClass("failed");
372             methodButtons.collapse("hide");
373             outputCollapsible.collapse("show");
374             outputSection.addClass("passed");
375             $("#analysisOutput span").html("Passed");
376             if(!$("#errorLineP").hasClass("displayNone"))
377             {
378                 $("#errorLineP").addClass("displayNone");
379             }
380             if(!$("#errorStep").hasClass("displayNone")) {
381                 $("#errorStep").addClass("displayNone");
382             }
383         } else {
384             outputSection.removeClass("passed");
385             methodButtons.collapse("show");
386             outputCollapsible.collapse("show");
387             outputSection.addClass("failed");
388             $("#analysisOutput span").html("Failed");
389             $("#errorLineP").removeClass("displayNone");
390             if(functions.length > 1) {
391                 $("#errorStep").removeClass("displayNone");
392                 $("#currentFunc").html(currentFunc+1);
393             }
394         }
395     }

```



```
396
397
398 function hideOutputButtons() {
399     methodButtons.collapse("hide");
400     outputCollapsible.collapse("hide");
401     outputSection.removeClass("passed");
402 }
403
404 $('#clear-button').on('click', function () {
405     $('#dl-graphml-button').prop('disabled', true);
406     $('#dl-ll-button').prop('disabled', true);
407     editor.setValue("");
408     codeChanged = true;
409     if(codeChanged) {
410         //console.log("in function");
411         methodButtons.collapse("hide");
412         outputCollapsible.collapse("hide");
413         outputSection.removeClass("passed");
414         outputSection.removeClass("failed");
415     }
416 });
417
418 $('#step-function').on('click', function () {
419     stepFunction();
420 });
421
422 // -----
423 // -----
424 // -----
425
426
427 $('#codeSelect').change(function(){
428     var value = $(this).val().replace(".c", "");
429     setEditorCode(value);
430     $('#file-name').val($(this).find(":selected").text());
431     methodButtons.collapse("hide");
432     outputSection.removeClass("failed");
433     outputSection.removeClass("passed");
434     hideOutputButtons();
435 });
436 });
437
438 function setSelectedLanguage(selected){
439     $(languageSpan).html(selected.html());
440 }
441
442 function initialiseEditor() {
443     editor.setTheme("ace/theme/clouds_midnight");
444     editor.getSession().setUseWrapMode(true);
445     editor.getSession().setMode("ace/mode/c_cpp");
446     editor.setHighlightActiveLine(true);
447     editor.setValue(templateCode[0]);
```

```
448 editor.clearSelection();
449 }
450
451 function setEditorCode(value) {
452     editor.setValue(templateCode[value]);
453 }
454
455 function clearContents(element) {
456     element.value = "hwe";
457 }
458
459 function stepFunction() {
460     if(currentFunc < functions.length) {
461         editor.gotoLine(functions[currentFunc][0]);
462         $("#currentFunc").html(currentFunc+1);
463         currentFunc++;
464     } else {
465         currentFunc = 0;
466         editor.gotoLine(functions[currentFunc][0]);
467         $("#currentFunc").html(currentFunc+1);
468         currentFunc++;
469     }
470 }
```

B.4 PHP

The php code can be found and read on the server when the system is handed over to Macquarie University as it has security requirements.

Appendix C

Meeting Attendance Form

Consultation Meetings Attendance Form

Week	Date	Comments (if applicable)	Student's Signature	Supervisor's Signature
1	3/8/17	Project Initiation via email	<i>Am</i>	<i>X</i>
2	10/8/17	Meeting. Discuss project requirements/goals	<i>Am</i>	<i>X</i>
3	17/8/17	Progress update via email prototype front-end	<i>Am</i>	<i>X</i>
4	24/8/17	Backend discussion meeting	<i>Am</i>	<i>X</i>
5	31/8/17	Draft paper and progress on design via email	<i>Am</i>	<i>X</i>
6	7/9/17	Meeting. Update on project progress	<i>Am</i>	<i>X</i>
7	14/9/17	Email Update Authentication	<i>Am</i>	<i>X</i>
8	20/9/17	Email Update	<i>Am</i>	<i>X</i>
9	28/9/17	Email Update	<i>Am</i>	<i>X</i>
10	5/10/17	Email Update	<i>Am</i>	<i>X</i>
11	12/10/17	Demo of system meeting (update)	<i>Am</i>	<i>X</i>
12	19/10/17	Email Update Working Prototype	<i>Am</i>	<i>X</i>
13	26/10/17	Email Update added final requirements	<i>Am</i>	<i>X</i>
14	2/11/17	Final Meeting Wrapping up project	<i>Am</i>	<i>X</i>

Bibliography

- [1] L. Williams, R. R. Kessler, W. Cunningham, and R. Jeffries, “Strengthening the case for pair programming,” *IEEE Software*, vol. 17, pp. 19–25, Jul 2000.
- [2] “Atomlinter - example of an code editor with linter support.” <https://atomlinter.github.io/>.
- [3] S. McIntosh, Y. Kamei, B. Adams, and A. E. Hassan, “The impact of code review coverage and code review participation on software quality: A case study of the qt, vtk, and itk projects,” in *Proceedings of the 11th Working Conference on Mining Software Repositories*, MSR 2014, pp. 192–201, ACM, 2014.
- [4] D. D. J. Pearce, “Whiley web programming language.” <http://whileylabs.com/>.
- [5] A. Møller and M. I. Schwartzbach, “Static program analysis,” 2012.
- [6] “Software validation and verification - overview.” <http://www.easterbrook.ca/steve/2010/11/the-difference-between-verification-and-validation/>, Nov 2010.
- [7] F. Cassez, A. M. Sloane, M. Roberts, M. Pigram, P. Suvanpong, and P. G. de Aledo, *Skink: Static Analysis of Programs in LLVM Intermediate Representation*, pp. 380–384. Berlin, Heidelberg: Springer Berlin Heidelberg, 2017.
- [8] “Skink functioning overview.” <http://science.mq.edu.au/~fcassez/software-verif.html>, 2009.
- [9] “Clang - c language frontend for llvm.” <https://clang.llvm.org/>.
- [10] “Scala - the java based functional programming language.” <https://www.scala-lang.org/>.
- [11] “Kiama - scala library for language processing.” <https://bitbucket.org/inkytonik/kiama>.
- [12] “Scala smt - scala library for parsing smt-lib.” <https://github.com/regb/scala-smtlib>.
- [13] N. B. Leonardo de Moura, “Z3: An efficient smt solver.” https://nikolajbjorner.github.io/slides/Z3_System.pdf, 2008.

- [14] “Cvc4 - an open source smt solver library.” <http://cvc4.cs.stanford.edu/web/>.
- [15] “Smt interpol - java based smt solver.” <https://ultimate.informatik.uni-freiburg.de/smtinterpol/>.