C-SCANNER FOR SEI CERT RULES IMPLEMENTATION

BY
Wania Zafar Abbasi
Registration ID: 04071313035
Supervised By
Dr. Muddassar Azam Sindhu
Department of Computer Science
Quaid-i-Azam University
Islamabad

Acknowledgement

In the name of ALLAH, the Most Gracious, the Most Merciful. First of all, I thanks to Almighty Allah because of His kindness and grace for completion of my project.

I would like to express my sincere gratitude to project supervisor Mr. Mudassar Azam Sindhu for the continuous support of project, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of development of project and writing of this thesis document.

Wania Zafar Abbasi

Abstract

C scanner is a desktop-based application for recognizing vulnerabilities in C language. It has been developed in Netbeans IDE in Linux OS. It is targeted for students and organizations with C language.
This tools is very simple to use. The user inputs his code in C scanner and parse it. The code is parsed and tokens are generated. Then it generates Abstract Syntax Tree for visual inspection from the code and recognizes code according to rules defined by SEI CERT C Coding Standard.

Contents

List of Table

List of Figure

Chapter 1

Software Project Management Plan (SPMP)

1. Software Project Management Plan (SPMP)

1.1 Introduction

This is a C language scanning/checking tool. In this project, the user will input the code for recognizing insecure code according to secure coding standard CERT. The tool will generate tokens and also generate Abstract Syntax Tree for visual inspections. It will also identify potential vulnerability sections according to the rules provided by SEI CERT C Coding Standard.

1.2 Project Overview

The aim of this project is to develop a tool that recognize vulnerable code on the basis of rule provided by SEI CERT C Coding Standard. The tool also generates Abstract Syntax Tree for visual inspections.

1.3 Scope

The project is intended to provide a tool for students and organization through which they can see visual inspections of the code using abstract syntax tree and identify vulnerabilities and undefined behavior resulting from coding errors before software is deployed.

1.4 Project Deliverables

Project deliverables for this tool are Software requirement Specifications (SRS), Software Project Management Plan (SPMP), Software Design Description (SDD) and Software Test Documentation (STD).

1.5 Project Organization

1.5.1 Software Process Model

Waterfall process model will be used for the development of this project. This model is linear-sequential life cycle model. It is easy to manage because each phase must be completed before the next phase can begin and there is no overlapping in the phases. Waterfall model works well for smaller projects where requirements are very well understood. So I preferred to use this model.

1.5.2 Roles and Responsibilities

There is no division of roles and responsibilities.

1.5.3 Tools and Techniques

The tools and techniques used for this project are following:

Table 1-1: Tools and Techniques

Sr.	Tool	Purpose
1	MS Word	It is used for documentation purpose.
2	ProjectLibre	It is used to make project plan.
3	ArgoUML	It is used to make diagrams.
4	Antlr 4	It is used for lexer and parser generation.

1.6 Project Management Plan

The description of project management plan for this project. It explains how time and resources are managed throughout the life cycle of this project.

1.6.1 Tasks

There are two phases of project plan. First is the requirement and analysis and second is the design phase of this project. In requirements and analysis phase, the major tasks are to identify requirements, define use cases, develop analysis model, develop SRS and review SRS.

In the second phase, the major tasks are develop a design using Object Oriented Approach, design mode of user input, validate input, develop models and evaluate design.

Following figure 1.1 and figure 1.2 shows tasks.

1.6.2 Description

Following is the description of major tasks of both analysis and design phases.

Requirements and Analysis

- Identify requirements

  The main goal is to review case study and define requirements by meeting stakeholders.

- Define use cases

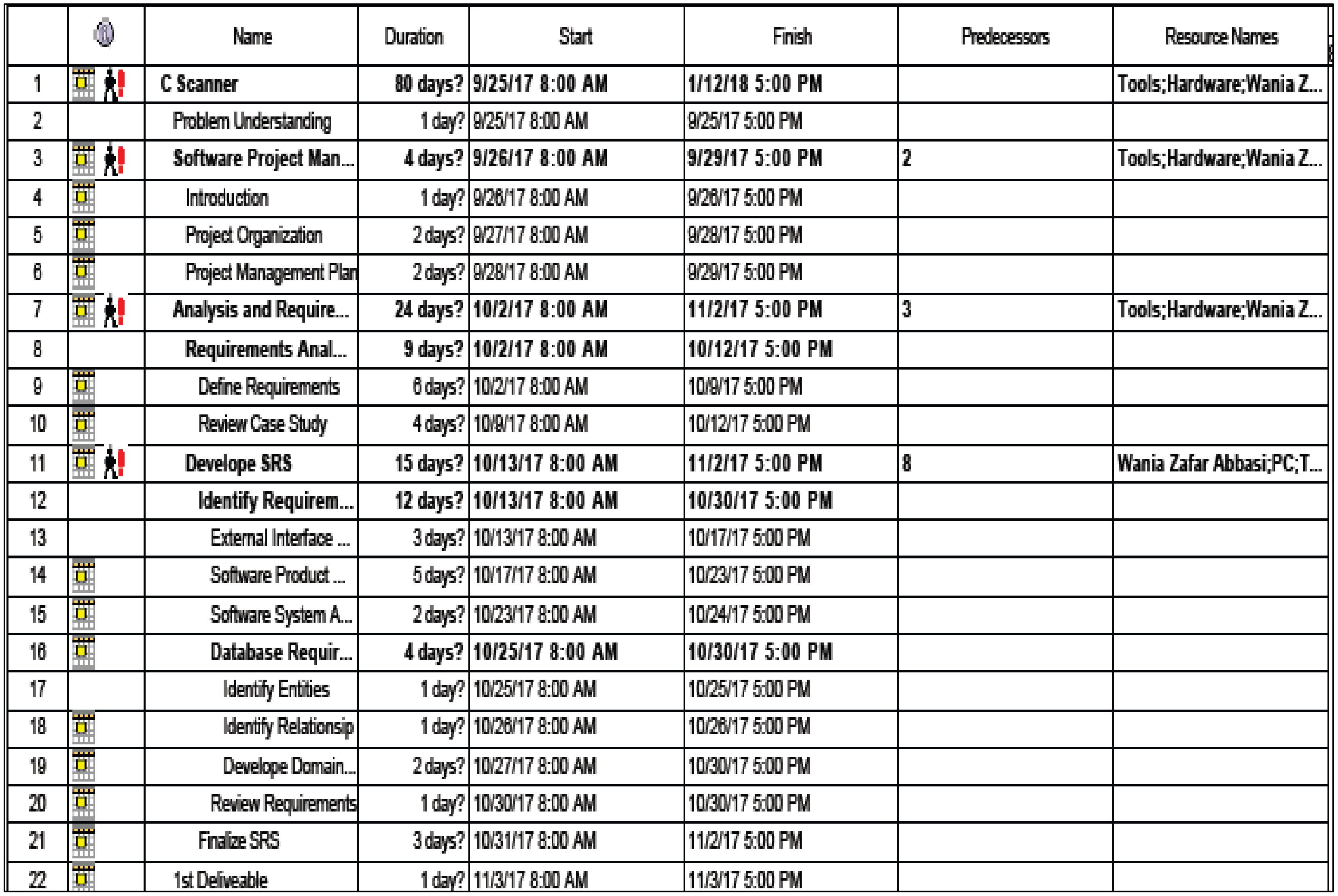
  Define use cases and make a use case diagram.

- Develop SRS

  Define functional and nonfunctional requirements and develop software requirement specification document. It includes all other details of product like scope, purpose and introduction.

- Review SRS

  Review software requirement specification document.

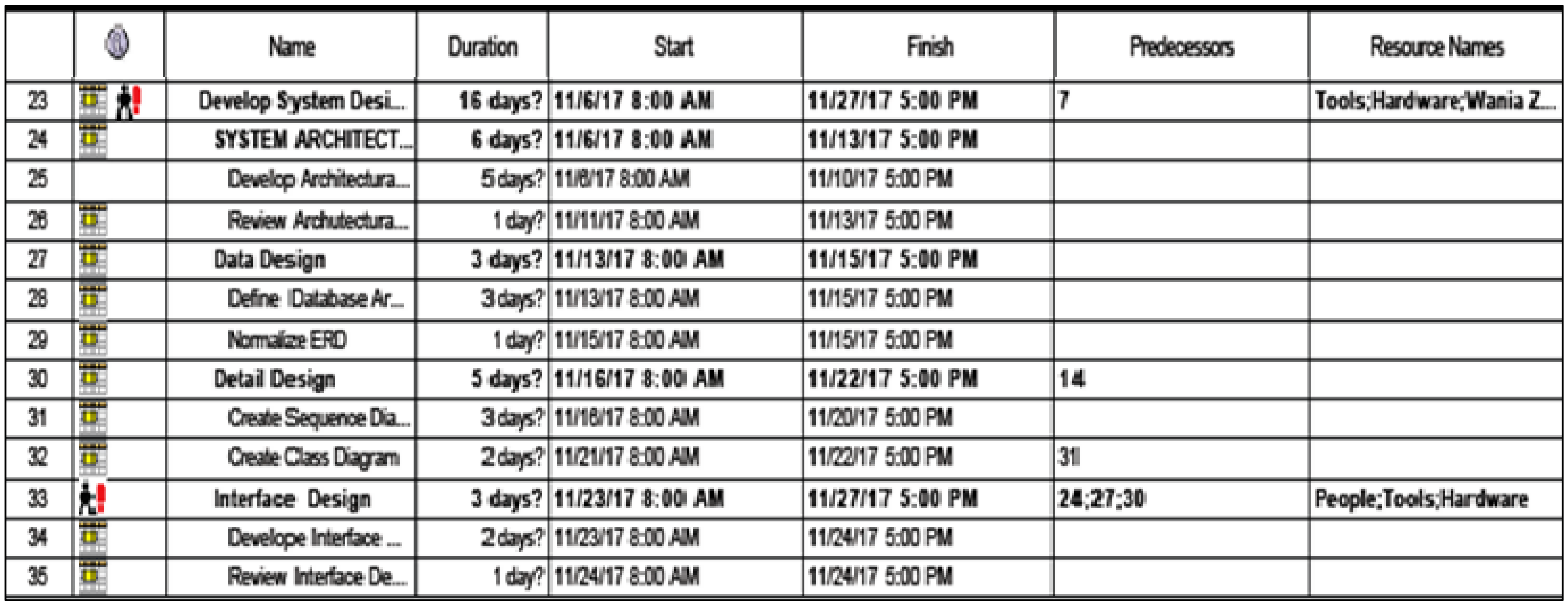


		Name	Duration	Start	Finish	Predecessors	Resource Names
1		**C Scanner**	**80 days?**	**9/25/17 8:00 AM**	**1/12/18 5:00 PM**		**Tools;Hardware;Wania Z...**
2		Problem Understanding	1 day?	9/25/17 8:00 AM	9/25/17 5:00 PM		
3		**Software Project Man...**	**4 days?**	**9/26/17 8:00 AM**	**9/29/17 5:00 PM**	2	**Tools;Hardware;Wania Z...**
4		Introduction	1 day?	9/26/17 8:00 AM	9/26/17 5:00 PM		
5		Project Organization	2 days?	9/27/17 8:00 AM	9/28/17 5:00 PM		
6		Project Management Plan	2 days?	9/28/17 8:00 AM	9/29/17 5:00 PM		
7		**Analysis and Require...**	**24 days?**	**10/2/17 8:00 AM**	**11/2/17 5:00 PM**	3	**Tools;Hardware;Wania Z...**
8		**Requirements Anal...**	**9 days?**	**10/2/17 8:00 AM**	**10/12/17 5:00 PM**		
9		Define Requirements	6 days?	10/2/17 8:00 AM	10/9/17 5:00 PM		
10		Review Case Study	4 days?	10/9/17 8:00 AM	10/12/17 5:00 PM		
11		**Develope SRS**	**15 days?**	**10/13/17 8:00 AM**	**11/2/17 5:00 PM**	8	**Wania Zafar Abbasi;PC;T...**
12		**Identify Requirem...**	**12 days?**	**10/13/17 8:00 AM**	**10/30/17 5:00 PM**		
13		External Interface ...	3 days?	10/13/17 8:00 AM	10/17/17 5:00 PM		
14		Software Product ...	5 days?	10/17/17 8:00 AM	10/23/17 5:00 PM		
15		Software System A...	2 days?	10/23/17 8:00 AM	10/24/17 5:00 PM		
16		**Database Requir...**	**4 days?**	**10/25/17 8:00 AM**	**10/30/17 5:00 PM**		
17		Identify Entities	1 day?	10/25/17 8:00 AM	10/25/17 5:00 PM		
18		Identify Relationsip	1 day?	10/26/17 8:00 AM	10/26/17 5:00 PM		
19		Develope Domain...	2 days?	10/27/17 8:00 AM	10/30/17 5:00 PM		
20		Review Requirements	1 day?	10/30/17 8:00 AM	10/30/17 5:00 PM		
21		Finalize SRS	3 days?	10/31/17 8:00 AM	11/2/17 5:00 PM		
22		1st Deliveable	1 day?	11/3/17 8:00 AM	11/3/17 5:00 PM		

Figure1.1: Requirement and Analysis

Design Phase

- Develop Design
  Develop architectural design and interface design using Object Oriented Approach.
- Design mode of user input
  Create mode of input layouts for this tool. Identify different inputs.
- Validate Input
  Validate input by checking existence
- Develop Designs
  Develop system sequence diagram and class diagram.
- Evaluate design
  Evaluate and verify design.



		Name	Duration	Start	Finish	Predecessors	Resource Names
23		Develop System Desi...	16 days?	11/6/17 8:00 AM	11/27/17 5:00 PM	7	Tools;Hardware;Wania Z...
24		SYSTEM ARCHITECT...	6 days?	11/6/17 8:00 AM	11/13/17 5:00 PM		
25		Develop Architectura...	5 days?	11/8/17 8:00 AM	11/10/17 5:00 PM		
26		Review Archutectura...	1 day?	11/11/17 8:00 AM	11/13/17 5:00 PM		
27		Data Design	3 days?	11/13/17 8:00 AM	11/15/17 5:00 PM		
28		Define Database Ar...	3 days?	11/13/17 8:00 AM	11/15/17 5:00 PM		
29		Normalize ERD	1 day?	11/15/17 8:00 AM	11/15/17 5:00 PM		
30		Detail Design	5 days?	11/16/17 8:00 AM	11/22/17 5:00 PM	14	
31		Create Sequence Dia...	3 days?	11/18/17 8:00 AM	11/20/17 5:00 PM		
32		Create Class Diagram	2 days?	11/21/17 8:00 AM	11/22/17 5:00 PM	31	
33		Interface Design	3 days?	11/23/17 8:00 AM	11/27/17 5:00 PM	24;27;30	People;Tools;Hardware
34		Develope Interface ...	2 days?	11/23/17 8:00 AM	11/24/17 5:00 PM		
35		Review Interface De...	1 day?	11/24/17 8:00 AM	11/24/17 5:00 PM		

Figure 1.2: Design Phase

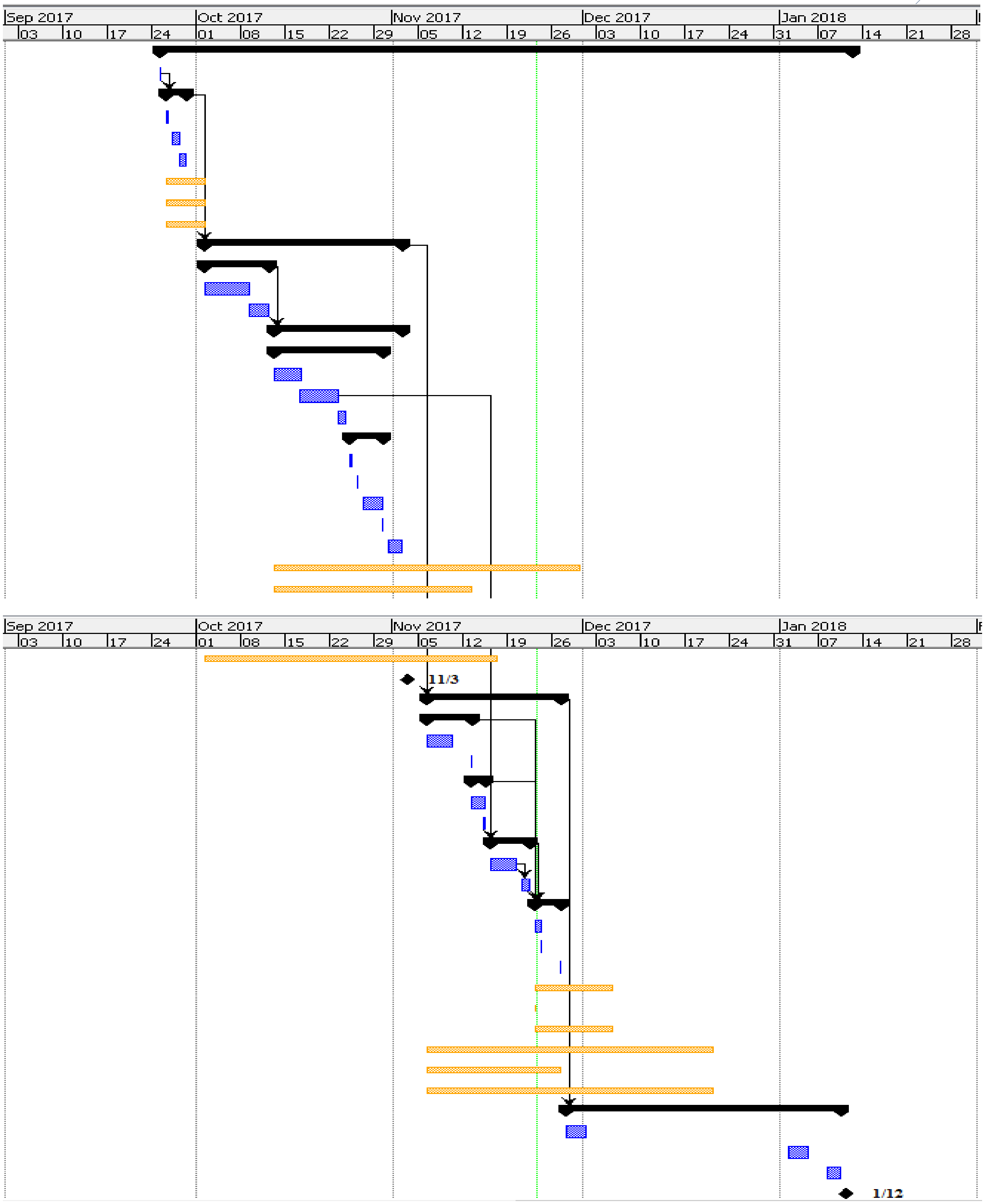


Figure 1.3: Gantt chart

1.6.3 Resources

Following are the resources needed.

- People
  - ➢ Wania Zafar Abbasi
  - ➢ Supervisor
- Software
  - ➢ MS Word
  - ➢ ProjectLibre
  - ➢ ROSE
- Hardware
  - ➢ PC

1.6.4 Deliverables and Milestones

Deliverable and milestones are shown in figure 1.1 and figure 1.2.

1.6.5 Dependencies and Constraints

Dependencies and constraints are shown in figure 1.1 and figure 1.2.

1.6.6 Risks and Contingencies

There are no risks and contingencies.

1.6.7 Assignments

Assignments are shown in figure 1.1 and figure 1.2.

1.6.8 Timetable

All time and dates are mentioned in above figure 1.1 and figure 1.2.

Chapter 2

Software Requirement Specification (SRS)

2 Software Requirements Specification (SRS)

2.1 Introduction

This chapter covers the software requirement specification of the tool. The purpose of this requirement specification and analysis is to clear the requirements of the tool and to decide what the tool should do and what the tool should not do.

2.2 Product Overview

C Scanner is a tool for recognizing vulnerabilities in the C language tool. This tool is basically a desktop-based application. This tool will be implemented using Netbeans IDE in Linux operating system. The tool will generates Abstract Syntax Tree for visual inspections from the code and also recognize vulnerable code according to the rules provided by SEI CERT C Coding Standard.

2.3 Definitions, Acronyms and Abbreviations

Table 2-1: Definitions, Acronyms and Abbreviations

AST	Abstract Syntax Tree
SEI CERT C Coding Standard	It provides rules for secure coding in the C programming language
CERT	Secure coding standard.

2.4 Specific Requirements

Following are specific requirements for interfaces like hardware or user or software.

2.4.1 External Interface Requirements

This section provides a detailed description of all inputs into and outputs from the system. It also gives a description of the hardware, software and communication interfaces and provides basic prototypes of the user interface.

2.4.2 User Interfaces

All interaction with user will be via GUI screen interface. It provides ease of use to users.

2.4.3 Hardware Interface

Hardware interfaces are not specified.

2.4.4 Software Interface

This tool is desktop-based application and it will be implemented in Linux, therefore, this system will be run on Linux operation system.

2.5 Software Product Features

Here is the list of the functions the tool must perform or must let the user perform.

- Input the code
  User clicks on the input section to input the code.
- Parse the Code
  The tool parses the code and generate tokens
- Generate Abstract Syntax tree
  The tool generates the Abstract Syntax tree for visual inspections.
- Recognize insecure code
  The tool recognizes vulnerabilities in the code according to the rules provided by SEI CERT C Coding Standard.

2.6 Use Case Diagram

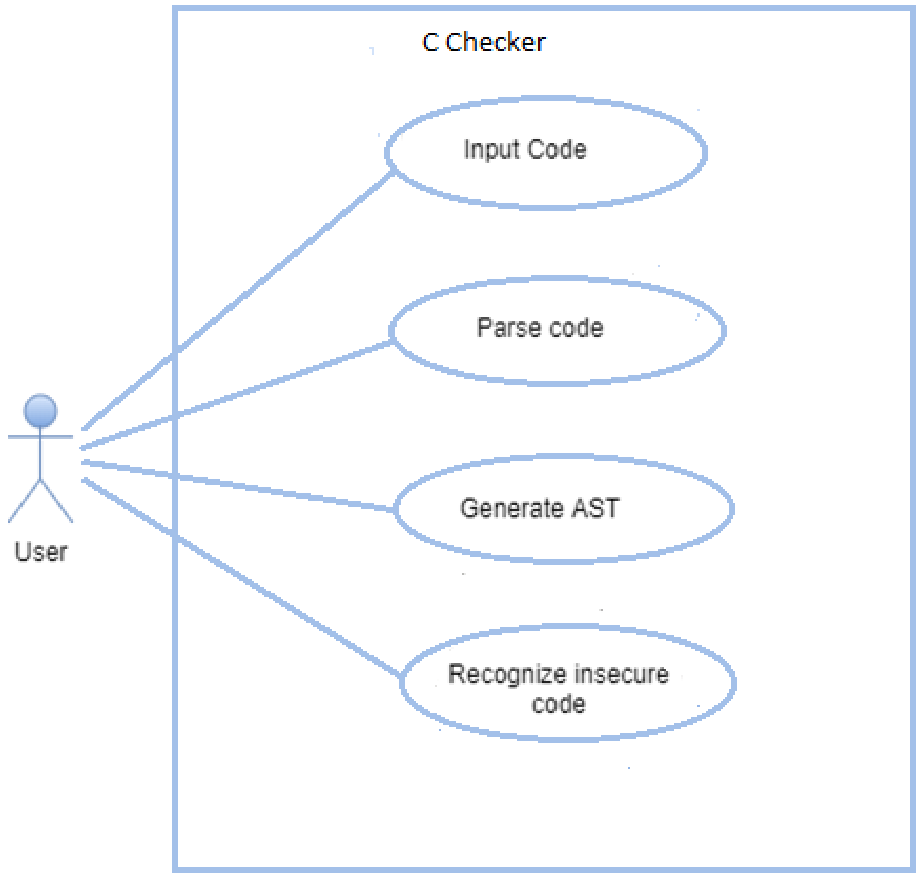


Figure 2.1: Use Case Diagram

2.7 Use Cases

Table 2-2: Use Case 1

UC-1 name	**Input the Code**
Primary actor	User
Stakeholder & Interest	User inputs code to identify vulnerabilities present in the code. User wants accurate, fast entry of code.
Pre-condition	1. Make sure the tool is installed correctly. 2. Make sure the tool is started correctly.
Post-condition	User has inputted the code.
Main Success Scenario	1. User inputs the code.
Alternate Flow	1(a). User doesn't input the code.
Frequency	Every time user starts the tool to recognize insecure code.

Table 2-3: Use Case 2

UC-2 name	**Parse the Code**	
Primary actor	User	
Stakeholder & Interest	User wants correct generation of tokens from the code.	
Pre-condition	Make sure user inputs the code.	
Post-condition	All the tokens has generated correctly.	
Main Success Scenario	<u>User</u> 1. User asks for parsing the code.	<u>System</u> 2. System parses the code and generates tokens.
Alternate Flow	2(a). The tool doesn't parse the code.	
Frequency	Every time user starts the tool to recognize insecure code.	

Table 2-4: Use Case 3

UC-3 name	**Generate Abstract Syntax tree**	
Primary actor	User	
Stakeholder & Interest	User wants to generate abstract syntax tree.	
Pre-condition	Code has parsed.	
Post-condition	Abstract Syntax tree has generated.	
Main Success Scenario	User 1. User asks to generate Abstract Syntax tree.	System 2. System generates Abstract Syntax tree.
Alternate Flow	None.	
Frequency	Every time user starts the tool to recognize insecure code.	

Table 2-5: Use Case 4

UC-4 name	**Recognize Insecure code**	
Primary actor	User	
Stakeholder & Interest	User wants fast, correct and accurate recognition of insecure code.	
Pre-condition	3.4 User inputted the code. 3.4 User parsed the code. 3.4 Abstract Syntax tree generated.	
Post-condition	Vulnerabilities from the code has identified.	
Main Success Scenario	User 3. User asks to recognize vulnerabilities from the code.	System 3. Vulnerabilities are identified from the code.
Alternate Flow	2(a) . There are no vulnerabilities in the code. 2(b) . All the vulnerabilities are detected. 2(c) . Vulnerabilities are detected when there are no vulnerabilities. 2(d) . Vulnerabilities are not detected when there are vulnerabilities.	
Frequency	Every time user starts the tool to recognize insecure code.	

2.8 Domain Model

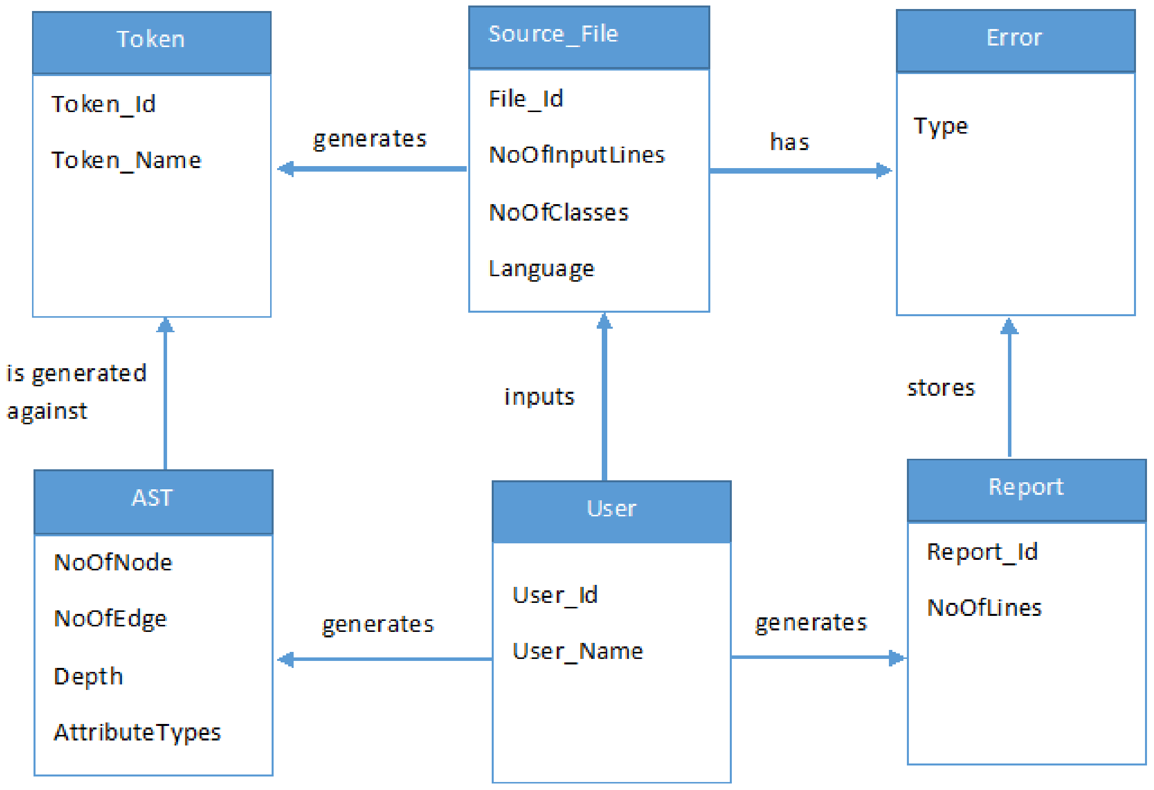


Figure 2.2: Domain Model

2.9 Software System Attributes

Software system attributes define overall factors that affect run-time behavior, design, and user experience. Here is detail of some software system attributes.

2.9.1 Reliability

The tool should be reliable and should never hang, other than as the result of an operating system error. There should be no occurrence of the failure. The tool should give the proper response to every query perform by user.

2.9.2 Availability

As tool will be used when installed on a pc in an operating system, and completely a desktop based application so it requires an operating system on which it is installed. Hence it will be available all the time to user whenever operating system is running on the user's pc and application is installed in it.

2.9.3 Security
There is no such security constraints. Because this project is a tool and in this tool there is not any confidential data to keep it secure.

2.9.4 Performance

The tool should support C language code with no performance penalty.

2.9.5 Maintainability
There should be aspect of maintainability for the tool. The tool should be easily extended. The code should be written in a way that it allows implementation of new functions.

2.10 Assumptions and Constraints
2.10.1 Development Languages and Tools
- Development tool
    - Netbeans IDE 8.1
- Programming language
    - Java

2.10.2 Operating System
The tool will be built on Linux by using Java as programing language and it will install and run on Linux.

2.11 Additional Requirements
There are additional requirements needed.

Chapter 3

System Design

3. System Design

3.1 Introduction

Software Design Description (SDD) is the representation of a software design to be used for communication design information to its stakeholders. It shows how the software system will be structured to satisfy the requirements. The SDD is performed in two stages. The first is a preliminary design in which the overall system architecture and data architecture is defined. In the second stage, that is the detailed design stage, more detailed data structures are defined and algorithms and codes are developed for the defined architecture

3.2 Purpose

The purpose of this chapter is to provide a description of the design of the tool to allow for software development to proceed with an understanding of what is to be built and how it is expected to build. The Software Design Description provides information necessary to provide description of the details for the software and system to be built.

3.3 Requirements Traceability Matrix

Requirement Traceability Matrix or RTM captures all requirements proposed by the client or development team and their traceability in a single document delivered at the conclusion of the life-cycle. The main purpose of Requirement Traceability Matrix is to see that all test cases are covered so that no functionality should miss while testing. It is used to track all the requirements and whether or not they are being met by the current process and design.

Traceability matrix of this system is shown below.

Table 3-1: Requirements Traceability matrix

Project Name	**C Scanner**				
Project Description	It is a C language Scanning Tool.				
Requirement Id	Requirement Name	Sequence Diagram	Test Case	Class Diagram	Interface
UC:1	Input Code	Yes	Yes	Yes	Yes
UC:2	Parse Code	Yes	Yes	Yes	Yes
UC:3	Generate AST	Yes	Yes	Yes	Yes
UC:4	Recognize Insecure Code	Yes	Yes	Yes	Yes

3.4 System Architectural Design

System Architecture Diagram is used to represent the components of system and interaction between them. Interaction between components of our system is shown in diagram. Double arrow line represents the interaction from both sides. Similarly single arrow represents one way interaction. A system architecture is a conceptual model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviors of the system. System architecture design of C Scanner is shown in figure below.

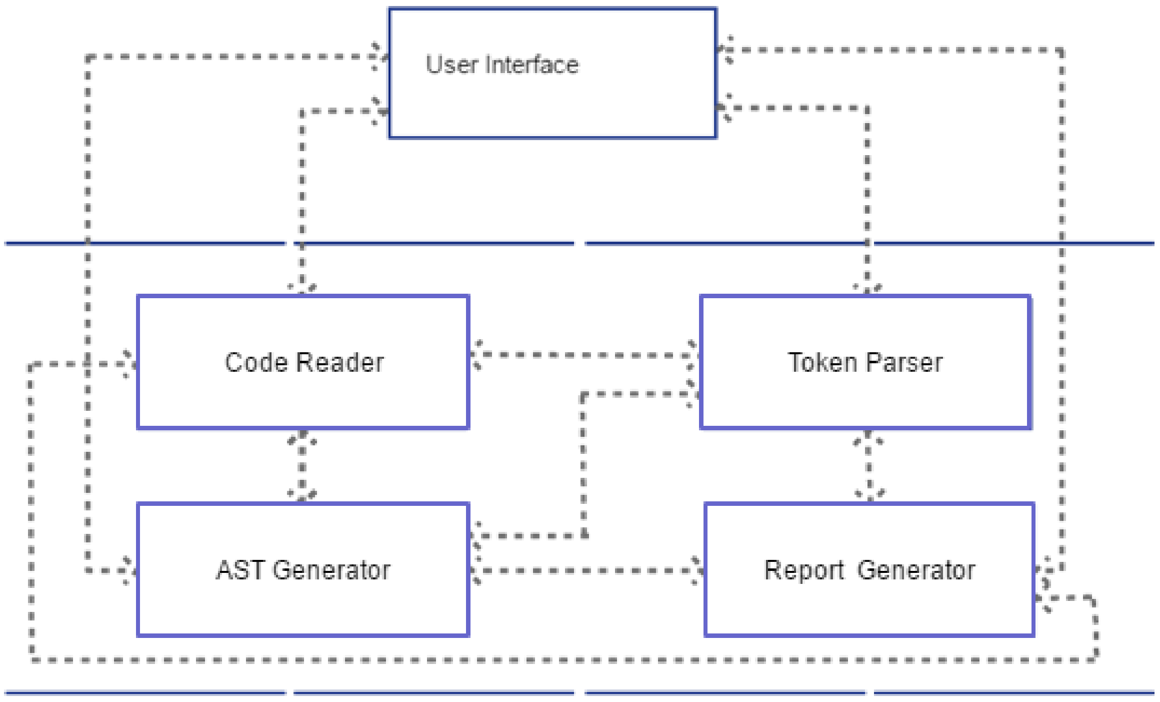


Figure 3.1: SYSTEM ARCHITECTURAL DESIGN

3.4.1 Chosen System Architecture

Chosen system architecture is 2-tier. Users can download and install this tool.

3.4.2 Discussion of Alternative Designs

An alternative design was to add database (for storing user's record) and to make it a 3-tier application.

3.4.3 System Interface Description

System interface describes the flow of resources. It is the logical characteristics of each interface between the software product and the hardware components of the system. Figure shows the software interface of the tool.

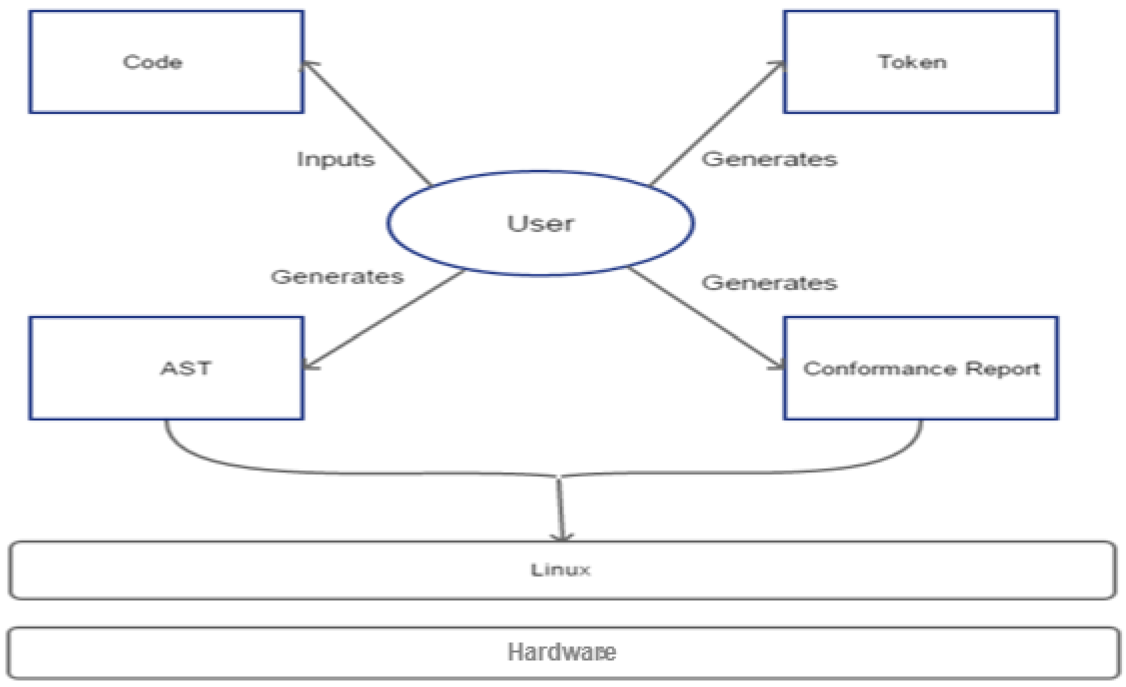


Figure 3.2: System Interface Description

3.5 Detailed Description of Components

Here is the detailed description of components of system architecture.

3.5.1 User

User inputs the code and asks for parsing the code to generate tokens. User asks to generate AST against tokens and asks to recognize vulnerabilities from the code. User finally asks for insecure code.

3.5.2 Token

User generates token by parsing the code. Tokens are generated against the inputted code.

3.5.3 AST

User asks to generate AST and AST is generated against the tokens.

3.5.4 Conformance Report

User asks to identify vulnerable code and vulnerabilities are checked according to the rules provided by SEI CERT C Coding Standard.

3.6 User Interface Design

User interface is the logical characteristics of each interface between the software product and its users. In this section user interface of tool is discussed.

3.6.1 Description of the User Interface

User can interact with the tool by using GUI screen interface of desktop. When user clicks on tool icon start page will appear. Start page has two buttons Input code and Exit. When user will click on input code a new screen will appear for entering the code. User will selects his/her own code and clicks parse button. Now next screen will appear where tokens are generated and to generate AST click on the generate AST button. When user will click on generate AST button a new screen will appear where AST will appear. To identify insecure code, User will click on the Identify Code button and then a new screen appear where vulnerabilities are identified.

3.6.2 Interfaces

Interface 1: Start Tool

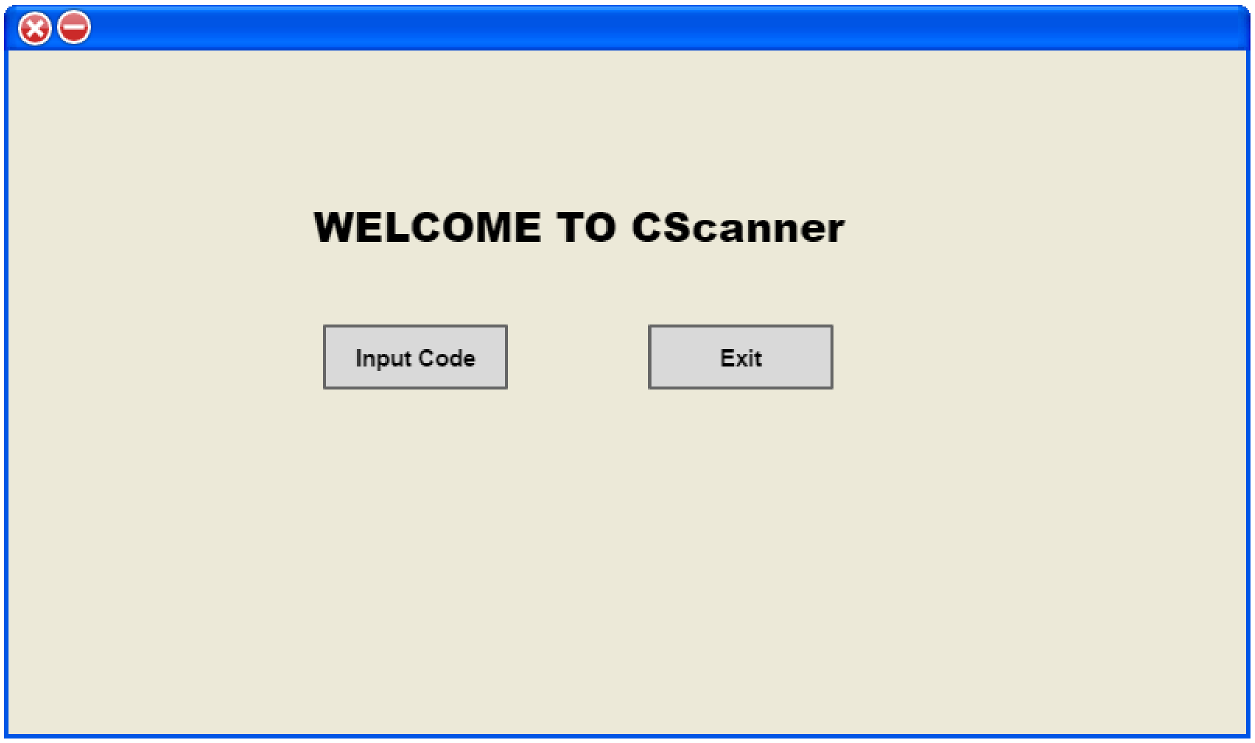


Figure 3.3: Start Tool

Interface 2: Input code

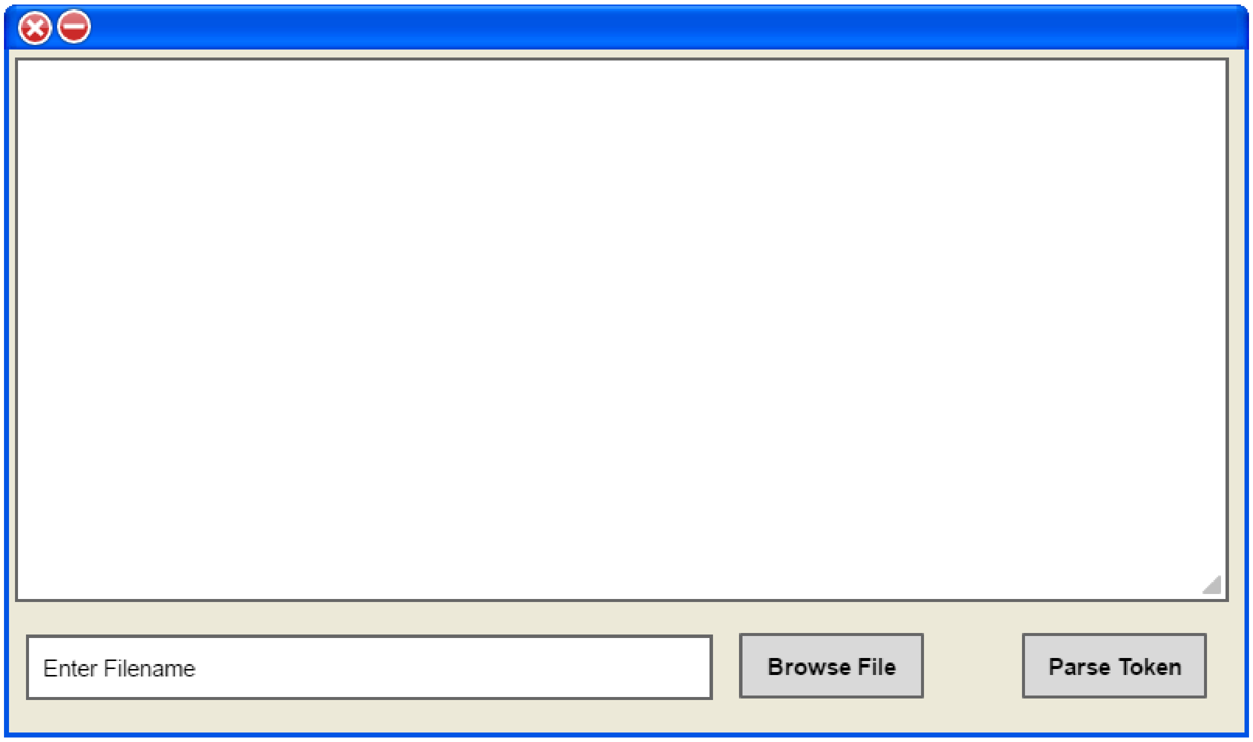


Figure 3.4: Input Code

Interface 3: Parse token

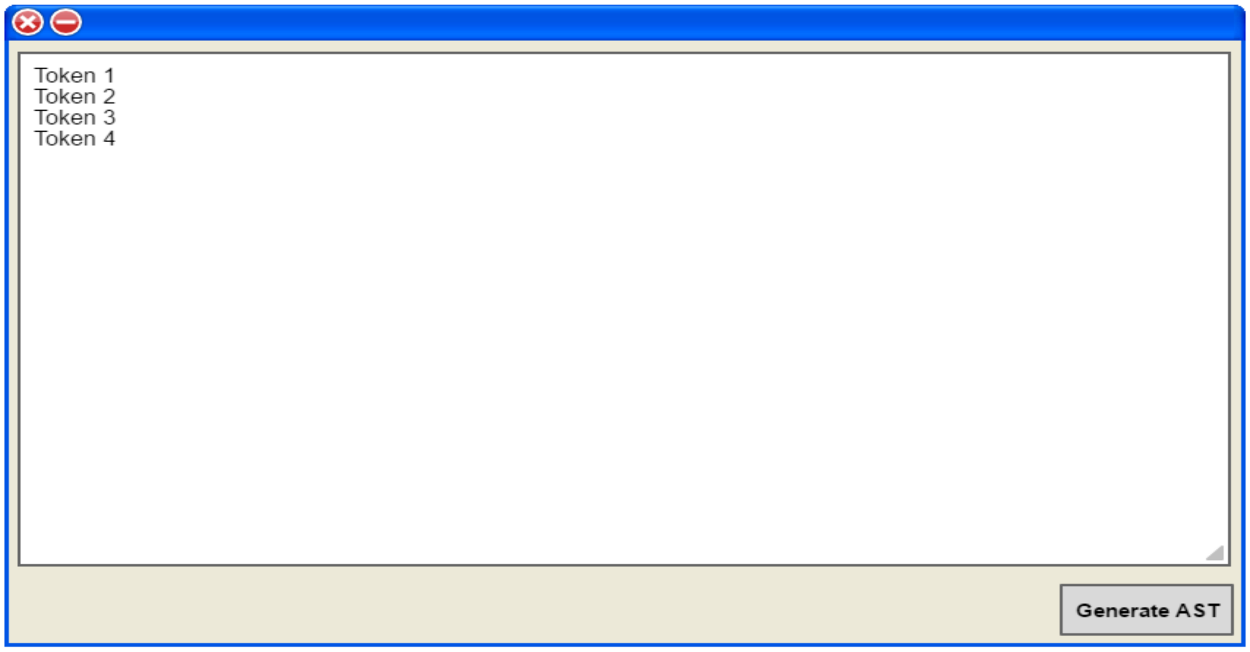


Figure 3.5: Parse Token

Interface 4: Generate AST

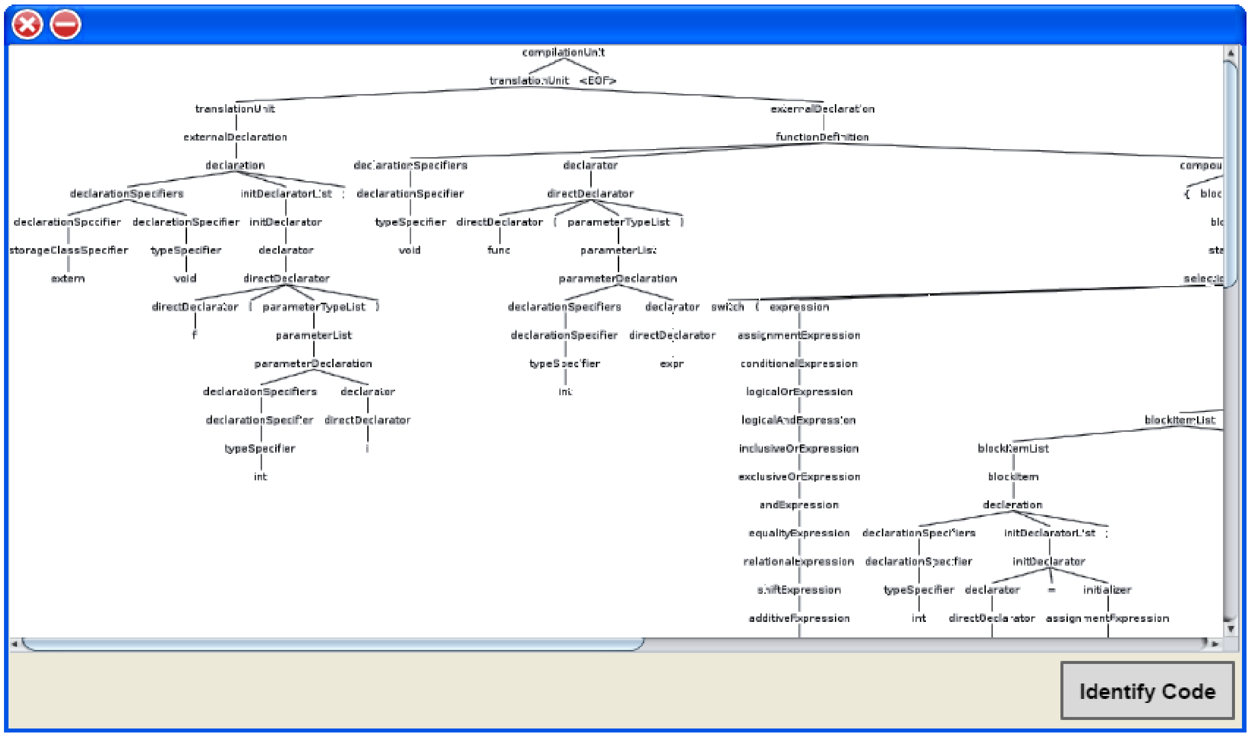
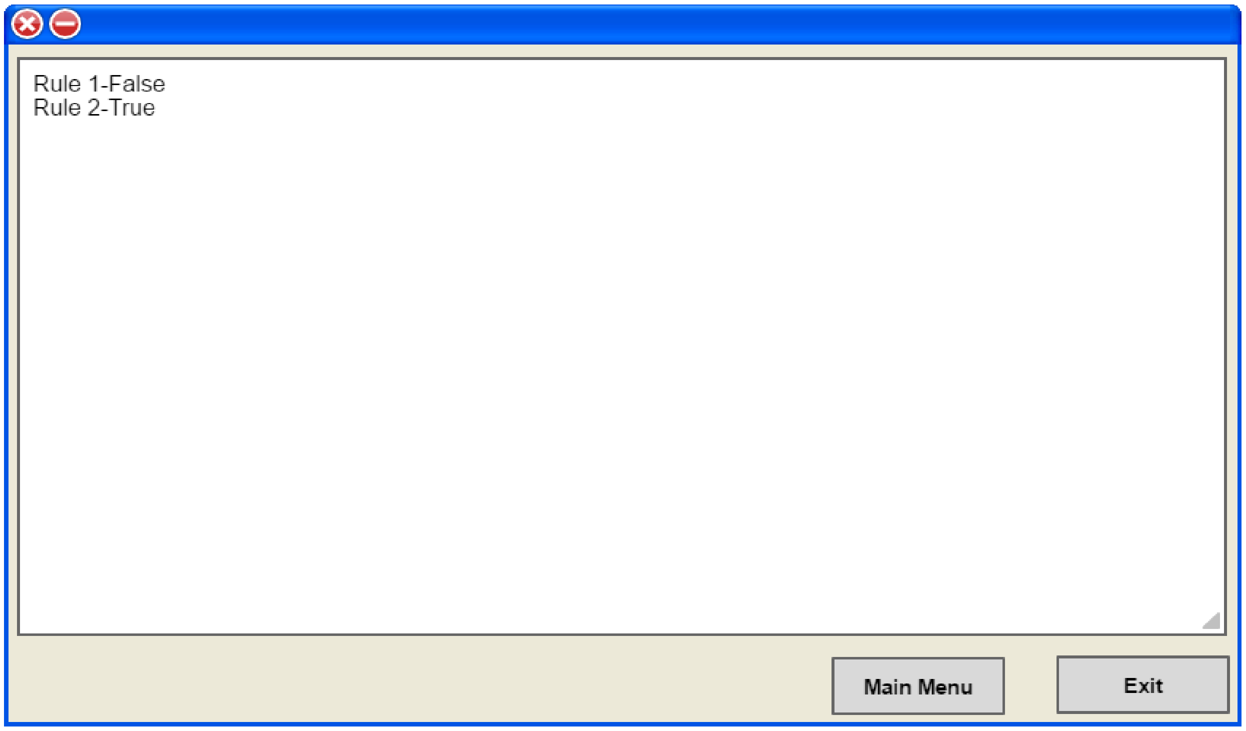


Figure 3.6: Generate AST

Interface 5: Identify Code



Rule 1-False
Rule 2-True

Figure 3.7: Identify Code

3.7 Sequence Diagram

Sequence diagram depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the system. A sequence diagram is an interaction diagram that shows how objects operate with one another and in what order. It is a construct of a message sequence chart. A sequence diagram shows object interactions arranged in time sequence.

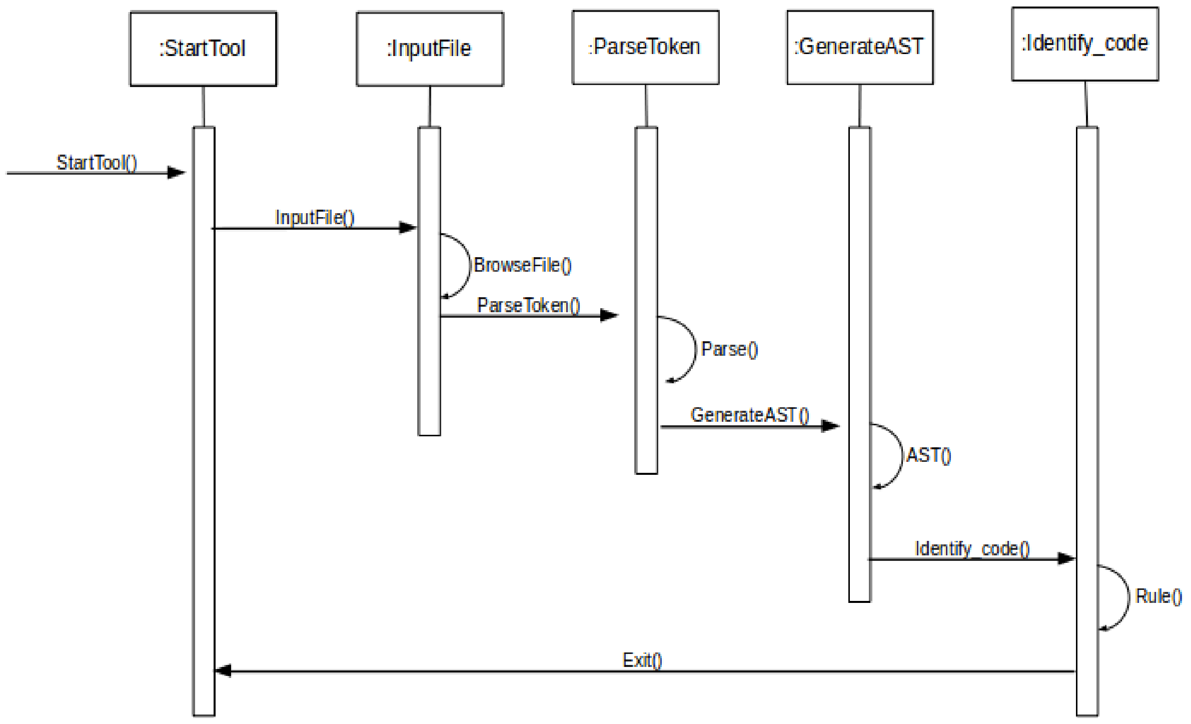


Figure 3.9: Sequence Diagram

3.8 Class Diagram

Class diagram shows the classes of the system, their interrelationships including inheritance, association and aggregation, operations and attributes of the classes.

Here is class diagram of C Checker

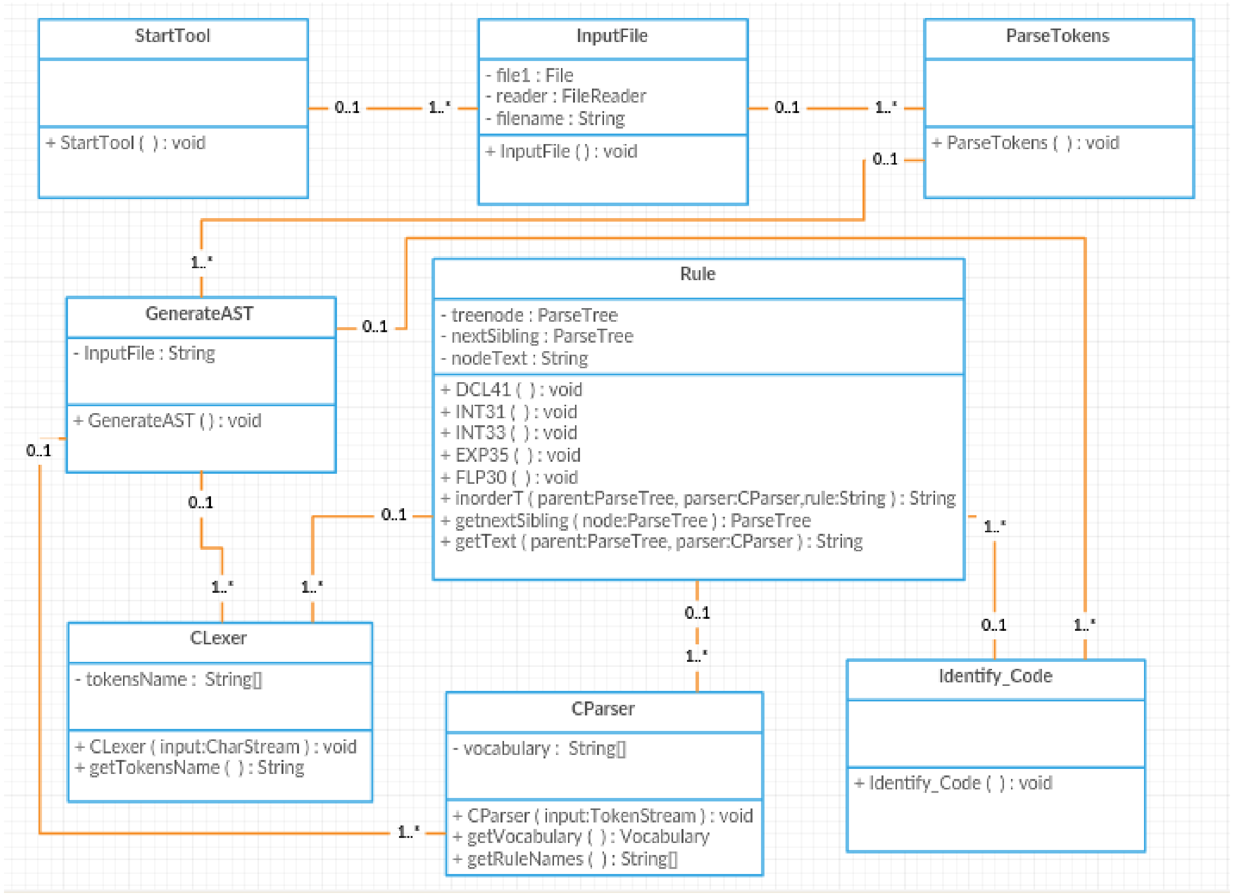


Figure 3.10: Class Diagram

Chapter 4

Implementation

4. Implementation

4.1 Introduction

Implementation is realization of technical specification or algorithm as a program, software component or other computer system through computer programming and development. In this chapter we provide you framework and language details of the tool. Screen shots are also given below

4.2 Framework selection

The tool is implemented in net beans framework using Antlr4 parser generator. Antlr4 helps to generate parser for C language.

4.3 Operating System selection

The Operating system preferred is Linux (Ubuntu 16.04). In future it can be for IOS and windows with little changes in code.

4.4 Language

Language used for tool is Java.

4.5 Tool

- Antlr-4: It is used to generate lexer and parser for C language using the grammar.
- Netbeans IDE 8.1: It is used for Implementation.

4.6 Implementation Detail

The main purpose of the tool is the implementation of the SEI CERT C language rules. The steps involve in the implementation of the rules are:

- Write grammar for the C language.
- Using Antlr4, lexer and parser are automatically generated from the grammar of the language.
- Lexer is used to generate tokens from the program.
- Parser is used to check the syntax of the program.
- Abstract Syntax Tree is used to visualize formal syntactic definition of the language.
- In-order traversal of AST is used to implement SEI CERT C language rules.

Antlr4:

Antlr 4 is a lexer and parser generator. This automatically generates lexer and parser from the grammar of the language. [10]

Lexer:

Lexer is used for lexical analysis of the program. In lexical analysis (scanning), we have a series of tokens where token is a name for a set of input strings with related structure. It maps characters of the source code into tokens and eliminate white spaces. [9]

Parser:

Parser is used for syntax analysis of the code. In syntax analysis (or parsing), we want to interpret what those tokens mean. A parser tries to map a program to the syntactic elements defined in the grammar. [9]

Abstract Syntax Tree (AST):

Antlr4 is used to generate AST of the program.

SEI CERT Rule:

These rules are used for secure coding in the C programming language and to develop safe, reliable and secure systems. These rules eliminate undefined program behaviors and exploitable vulnerabilities resulting from coding errors before the program is deployed. [11]

4.7 Screen Shots

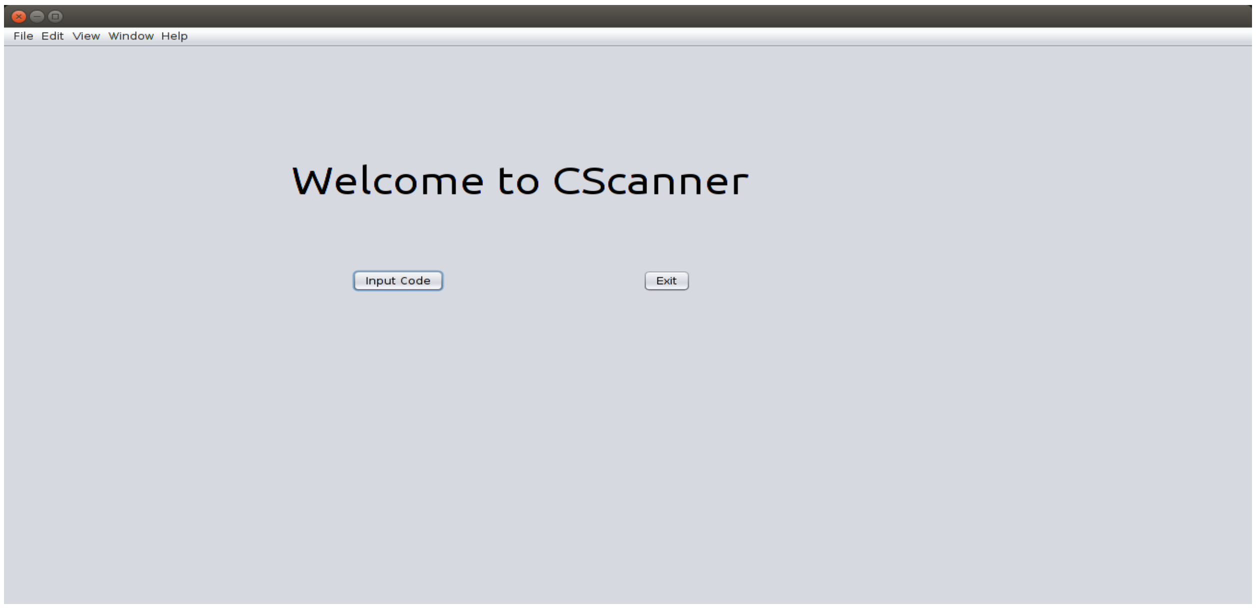


Figure 4.1: Main Menu

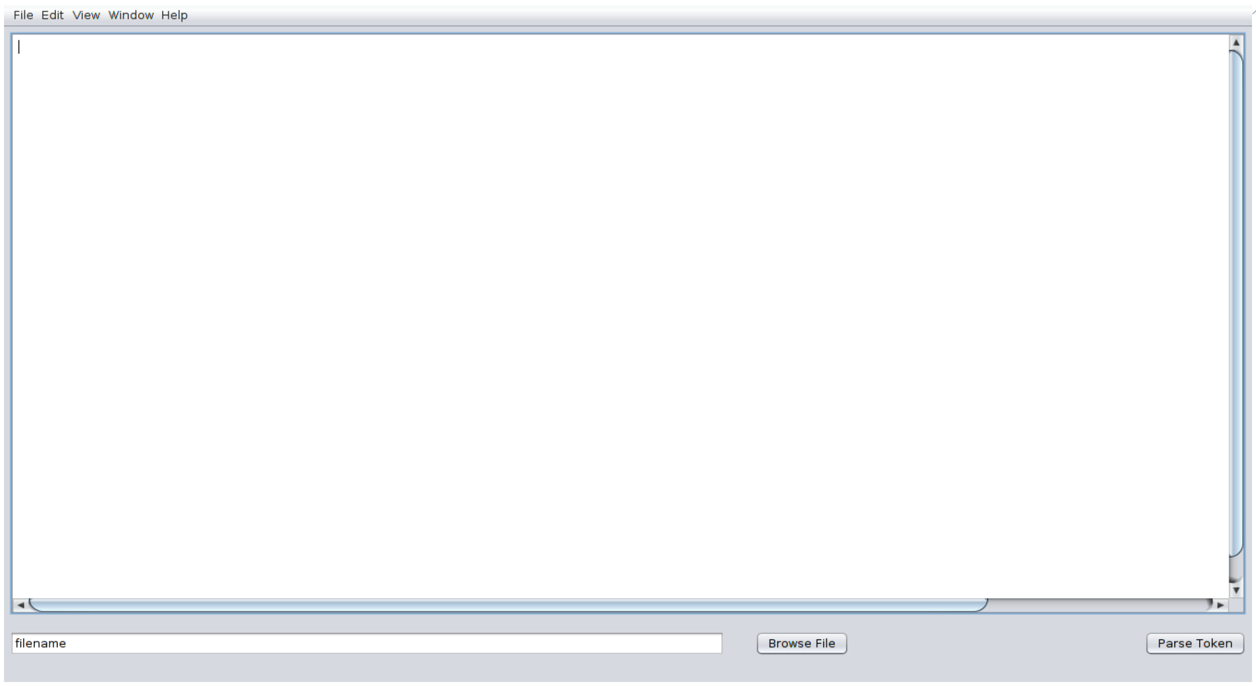


Figure 4.2: Input File

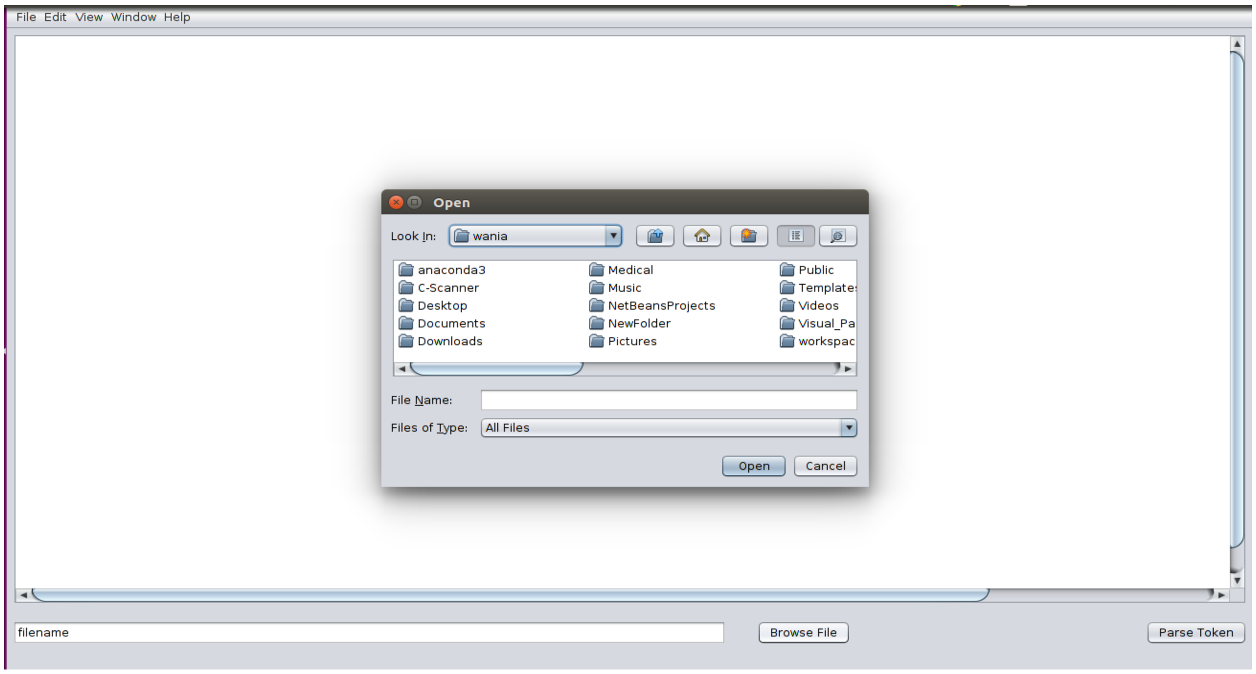
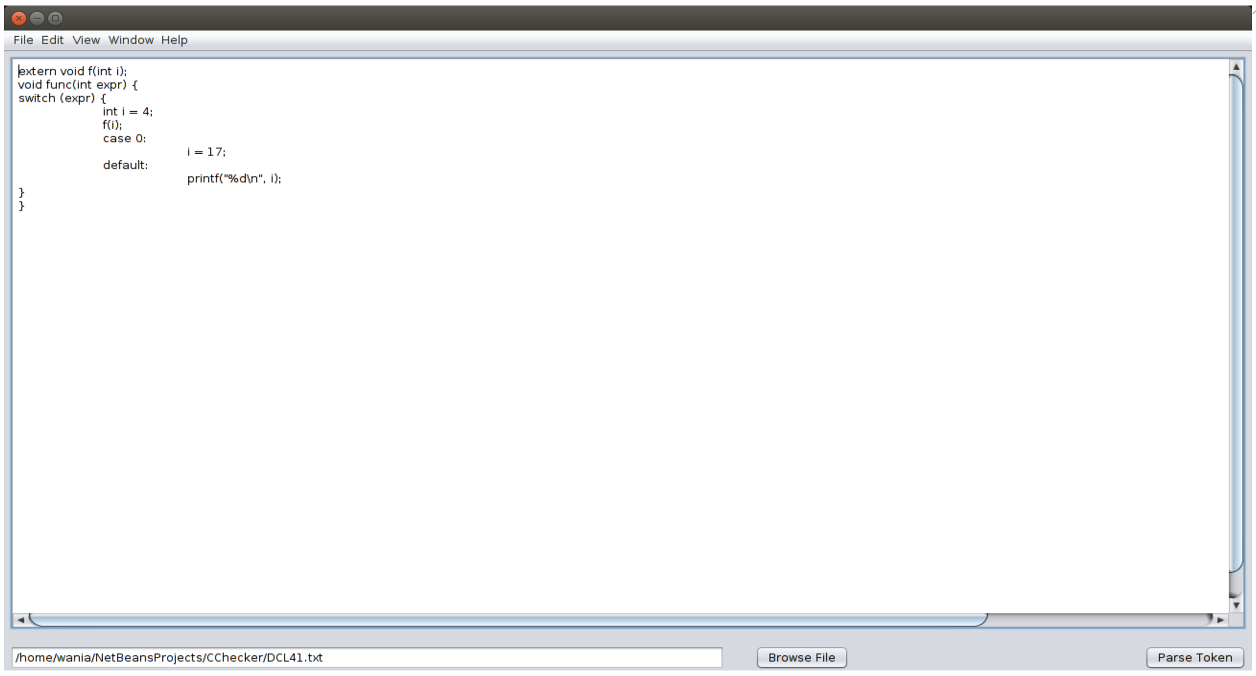


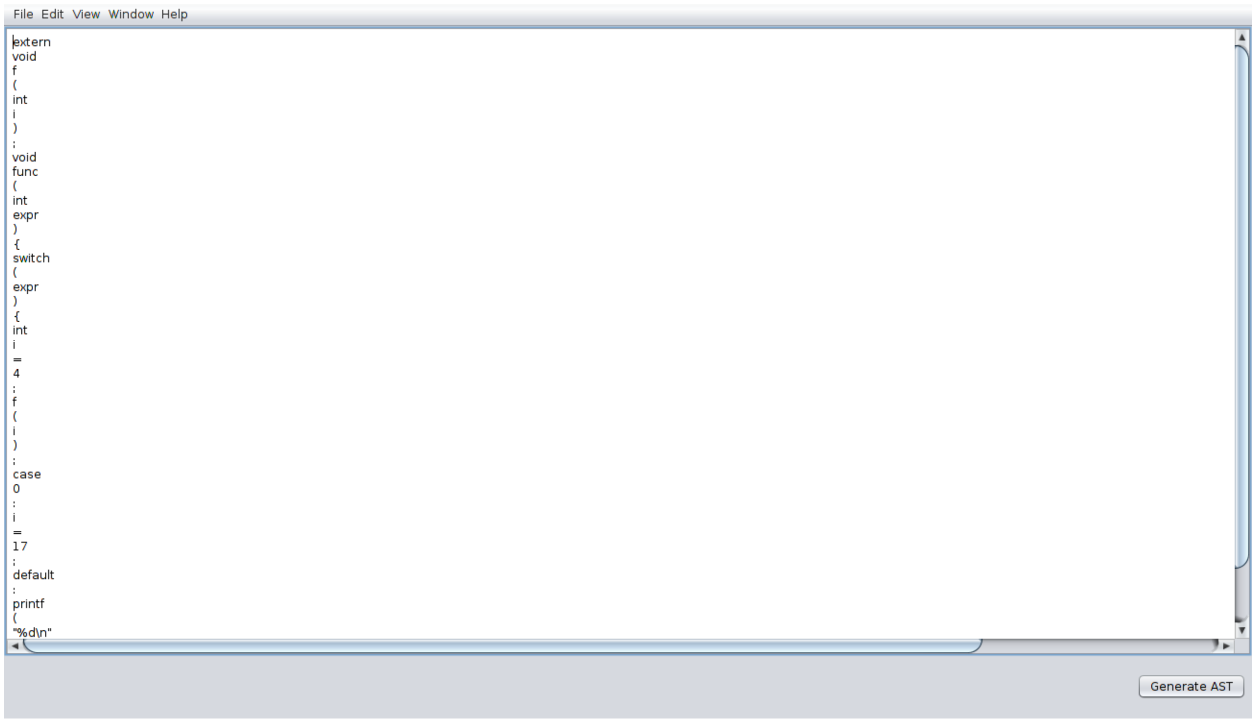
Figure 4.3: Browse File



```
extern void f(int i);
void func(int expr) {
switch (expr) {
		int i = 4;
		f(i);
		case 0:
			i = 17;
		default:
			printf("%d\n", i);
}
}
```

```
/home/wania/NetBeansProjects/CChecker/DCL41.txt
```

Figure 4.4: Display Code



```
extern
void
f
(
int
i
)
;
void
func
(
int
expr
)
{
switch
(
expr
)
{
int
i
=
4
;
f
(
i
)
;
case
0
:
i
=
17
;
default
:
printf
(
"%d\n"
```

Figure 4.5: Parse Token

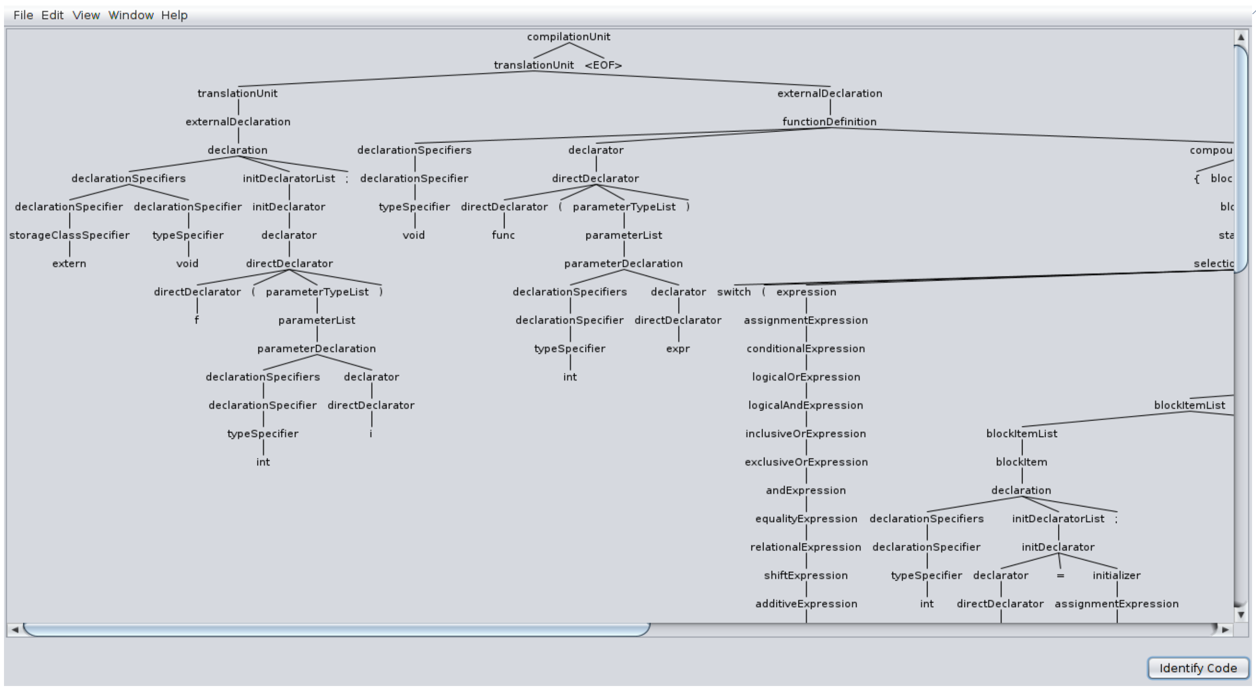
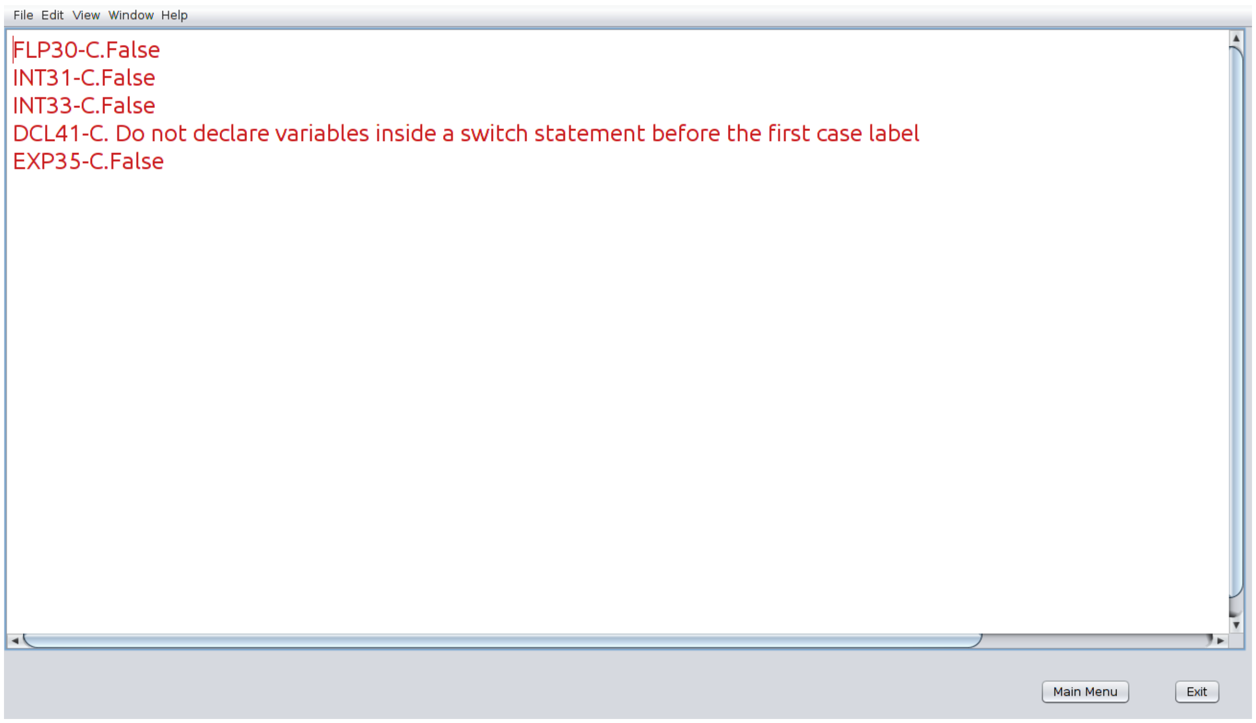


Figure 4.6: Generate AST



```
FLP30-C.False
INT31-C.False
INT33-C.False
DCL41-C. Do not declare variables inside a switch statement before the first case label
EXP35-C.False
```

Figure 4.7: Identify Code

Chapter 5

Testing

5 Testing

5.1 Introduction

Testing is the process of evaluating a system or its component(s) with the intent to find whether it satisfies the specified requirements or not. In simple words, testing is executing a system in order to identify any gaps, errors, or missing requirements in contrary to the actual requirements. According to ANSI/IEEE 1059 standard, Testing can be defined as - A process of analyzing a software item to detect the differences between existing and required conditions (that is defects/errors/bugs) and to evaluate the features of the software item.

5.2 Test Approach

User acceptance testing (UAT) also called beta testing or end user testing, consist of a process of verifying that a solution works for the user. It is not system testing (ensuring software does not crash and meets documented requirements), but rather is there to ensure that the solution will work for the user i.e. test the user acceptance the solution (software vendors often refer to as Beta testing).
This should be undertaken by a subject-matter expert (SME), preferably the owner or client of the solution under test, and provides a summary of the findings for confirmation to proceed after trail or overview. In software development, UAT as one of the final stages of a project often occurs before a client or customer accepts the new system. Users of the system perform test in line with what occur in real life scenarios.

5.3 Testing Objectives

For checking whether the requirement in SRS are fulfilled or not we have to make test on these cases.
UAT has following objectives.
User Acceptance test make test case against the requirements.
User Acceptance test check actual function, input, expected result, actual result, procedure to make test case, pass/fail status against each test case.

5.4 Test Plan

5.4.1 Features to be tested

Features to be tested are all according to user prospective. For example

- Input code.
- Parse Token.
- Generate AST.
- Identify Code.

5.4.2 Features not to be tested

Features not to be tested are from the developer's point of view. For example

- How much power is used by processor?
- How much memory is consumed by the tool?
- Software risk factor.

- Maintainability.

5.5 Testing Tools and Environment

As this is beta testing (testing by the user) so no specific tools and environment is required.

5.6 Test Cases

A test case is a test set, which has a set of test data, preconditions, expected results and post conditions, developed for a particular test scenario in order to verify compliance against a specific requirement. Test Case acts as the starting point for the test execution, and after applying a set of input values, the application has a definitive outcome and leaves the system at some end point or also known as execution post condition

Following are the test cases of the tool.

Test Case 1:

Table 5-1: Test Case 1

Purpose	**Input Code**
Setup	1. Open the tool. 2. Click input code.
Instruction	Input code by entering source code or by browsing source file.
Expected Result	Code inputted.
Actual Result	Code inputted.

Test Case 2:

Table 5-2: Test Case 2

Purpose	**Parse Token**
Setup	1. Open the tool. 2. Click input code. 3. Input code by entering source code or by browsing source file.
Instruction	Click Parse Token.
Expected Result	Token generated.
Actual Result	Token generated.

Test Case 3:

Table 5-3: Test Case 3

Purpose	**Generate AST**
Setup	1. Open the tool. 2. Click input code. 3. Input code by entering source code or by browsing source file. 4. Click Parse Token.
Instruction	Click Generate AST.
Expected Result	AST generated.
Actual Result	AST generated.

Test Case 4:

Table 5-4: Test Case 4

Purpose	**Identify Code.**
Setup	1. Open the tool. 2. Click input code. 3. Input code by entering source code or by browsing source file. 4. Click Parse Token. 5. Click Generate AST.
Instruction	Click Identify code.
Expected Result	All Insecure code sections are identified.
Actual Result	All Insecure code sections are identified.

References:

[1] Software Project Management Plan IEEE 1058-1998.

[2] Chapter 10. Software Requirement Specifications Software Engineering A lifecycle approach by P. Mohapatra

[3] Chapter 6. Use cases and Functional Requirement Applying UML and Patterns by Craig Larman

[4] Chapter 13. Design Concepts and Principles, Software Engineering A Practitioner's Approach by R.S. Pressman

[5] https://en.wikipedia.org/wiki/Traceability_matrix

[6] https://en.wikipedia.org/wiki/C_(programming_language)

[7] Chapter 8. Software Testing Ian_Sommerville _Software_Engineering_9th_edition

[8] SEI CERT C Coding Standard: Rules for Developing Safe, Reliable, and Secure Systems 2016 Edition

[9] A Practical Approach to Compiler Construction-Springer International Publishing (2017)

[10] https://en.wikipedia.org/wiki/ANTLR

[11] https://en.wikipedia.org/wiki/CERT_C_Coding_Standard