UML Diagram Generator from C++ Code

Final Project Report

Review-3

Version: 1.3

Date Created: 2021.06.02

Signatures

Date	Revision	Approved By
2021.05.20	1.2	Abhinav Dholi
2021.05.20	1.2	Nehul Jindal

List of Contributors

Name	Initials	Organization	E-Mail
Abhinav Dholi	AD	VIT Vellore	abhinav.dholi2019@vitstudent.ac.in
Nehul Jindal	NJ	VIT Vellore	Nehul.jindal2019@vitstudent.ac.in

Change History

Revision	Date	Description
1.0	2021.04.24	Initial Revision
1.1	2021.05.19	Second Revision
1.2	2021.06.02	Third Revision

Declaration

June 2021,

I hereby declare that the thesis entitled UML diagram generator from C++ submitted by us, for the award of the degree of Bachelor of Technology in Computer Science Engineering to VIT is a record of Bonafede work carried out by me under the supervision of Professor Swathi N.

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place: Vellore

Date: 02.06.2021

 Abhinav

 Nehul

Signature of the Candidate

ACKNOWLEDGEMENTS

In performing our assignment, we had to take the help and guideline of some respected persons, who deserve our greatest gratitude. The completion of this assignment gives us much Pleasure. We would like to show our gratitude to Ms. Swathi N, Course Instructor, VIT university for giving us a good guideline for assignment throughout numerous consultations. We would also like to expand our deepest gratitude to all those who have directly and indirectly guided us in writing this assignment.

In addition, a thank you to Professor Swathi N, who introduced us to the Methodology of work, and whose passion for the "underlying structures" had lasting effect.

Many people, especially our classmates and team members itself, have made valuable comment suggestions on this proposal which gave us an inspiration to improve our assignment. We thank all the people for their help directly and indirectly to complete our assignment.

CONTENTS

1. Introduction

At present policy to generate UML diagram when the user enters the source code in object-oriented programming language (C++) as input, in our approach to generate the UML diagram, class diagrams and sequence diagram, static analysis to record the classes and their relationships between classes are implemented in C++ through colon (:) with class name is recorded. Currently there are several modelling tools that include code generation options.

1.1 Objective

The purpose of the Software Design Specification is to describe the specific design of the "UML diagram generator from C++ code" software by VIT Vellore. The design specification includes an overview of the design along with software module decomposition. This document provides a detailed description of each software module's design. For each module, a user interface design and class diagram design is given. As well, a process description is described for each module. It is in the process description that the details of what logic will need to be implemented are given.

1.2 Scope

It is within the scope of the Software Design Specification to describe the specific system design of the UML diagram generator project. This would include user interface design, object-oriented class design, process design, and data design. Any specific detail that is needed about the standards or technology used to design the software are within the scope of this document. It is outside the scope of this document to describe UML diagram generator systems and technology or the general problem with unwanted UML diagram generator. It is also outside the scope of this document to describe in any detail at all how certain mentioned standards or technologies work and operate.

2. **Project Description and Goals**

We have designed a software which helps user getting the UML diagram of their desired code using the software. The software designed accepts the code in C++ language as input and check for the errors in the file and process the file after proper authorization. After that the user gets a UML diagram according to their code. It is basically a reverse engineering project which explains the extraction of UML diagram from C++ code.

The software helps authorizing the user as genuine user and cross check the username and password. Then it permits the genuine user and accepts .cpp file as input. It tells the user if any error is found in the code. Then it displays the UML diagram obtained and user can download the file in .svg format.

3. **Technical Specifications**

Technical specification involves the basic decomposition or breakdown of the software into working modules and processes or different output pages:

- Authentication Module
- Import Module
- Parser Module
- UML generator Module

3.1 Authentication Module:

3.1.1 Introduction

This module authenticates if the user is valid or not and gives access to the software. The software shows the login page to the user a demand the username and password and checks for the validation of the data. If the user is authentic then it gives the access to the user to input the code file and go one. But if the user details are wrong then the user gets a retry for the software.

3.1.2 Functional Requirements

Purpose: To check the user is genuine or not

Input: User enters the login credentials that are username and password

Processing: System runs a check on the credentials entered by the user

Output: If it is right then user is shifted to next process, else the user gets a retry.

3.1.3 Stimulus Response

User Action:

- User enters the login credentials

System action:

- It checks the login credentials
- System shows the result to the user

3.2 **Import Module:**

3.2.1 Introduction

Once the user is authorised to this window, the software allows the user to input the code file, choose the language and then demand for the UML diagram. The user is asked fo the confirmation of .cpp file and if the file is checked to be a .cpp file then it undergoes further processing, ese user gets a retry.

3.2.2 Functional Requirements

Purpose: Get the input C++ code file from the user for UML diagram generation

Input: The user can input the code file in pdf or cpp format.

Processing: The system traces the file from the location and load it in software.

Output: The user is asked to confirm it is a cpp file. If it is, then it gets loaded.

3.2.3 Stimulus Response

User Action:

- Open the file in the software
- Choose the language and file type, confirm the cpp file and move further

System action:

- The system accepts and read the file from the location.
- Checks the cpp file and response back to the user.

3.3 Parser Module

3.3.1 Introduction

The parser module searches for the error and separate code file according to different class and other basic format. It the code file contains error then the user gets a retry.

3.3.2 Functional requirements

Purpose: Getting to know if the code file contains error or not.

Input: code file is the input from the user

Processing: the system separates the basic factors of code and process them

Output: if the code has errored the user gets a retry, otherwise the code starts processing.

3.3.3

3.4 UML Generator Module

3.4.1 Introduction

The rectified and checked code file is processed into a UML diagram which is exported to user in .svg format.

3.4.2 Functional requirements:

Purpose: Executing the final process and generate the UML diagram

Input: The code file from the user is the input

Processing: the class and other attributes are processed into UML diagram

Output: The UML diagram is the output in .svg format

3.4.3 Stimulus Response:

User Action:

- Load the final output UML

System Action:

- Runs the process of conversion and project the final UML to the user.

4 Design Approach and Details

4.1 Design Approach & Methods

4.1.1 Architecture

Layered Architecture

• We have chosen manager model under centralized control model.

• The main reason why we are using the manager model as the parser module affects the import as well as the UML generation.

• Manager model applicable to concurrent system where one system component controls the starting stopping and coordination system of other system processes.

The architecture used is layered architecture with 4 modules.

4.1.2 **UML Diagram:**

umlGenerator

+_showPrivMembers
+_showProtMembers
+_showPubMembers
+_drawAssociations
+_drawInheritances

+addClass(self, aClass)
+_genFields(self, accessPrefix, fields)
+_genMethods(self, accessPrefix, methods)
+_genClass(self, aClass, withPublicMembers = False, withProtectedMembers = False, withPrivateMembers = False)
+_genAssociations(self, aClass)
+_genInheritances(self, aClass)()
+setDrawInheritances(self, enable)()
+setDrawAssociations(self, enable)()
+setShowPrivMethods(self, enable)()
+setShowProtMethods(self, enable)()
+setShowPubMethods(self, enable)()
+generate(self)()

Authentication

+password
+key

+checkPwd()

Import

+foldername
+filename

+findFilesInDir()
+FindFolder()
+FileTypeConfirm()

Parser

-parents
-privateFields
-privateMethods
-publicFields
-publicMethods
-protectedFields
-protectedMethods

+processClassField(cursor)
+processClassMemberDeclaration(umlClass, cursor)
+processClass(cursor, inclusionConfig)
+traverseAst(cursor, inclusionConfig)
+parseTranslationUnit(filePath, includeDirs, inclusionConfig)
+getId(self)
+addParentByFQN(self, fullyQualifiedClassName)

4.1.3 Use Case Diagram

Use Case Diagram

4.1.4.1 Import Module Detailed design

4.1.4.1.1 Design
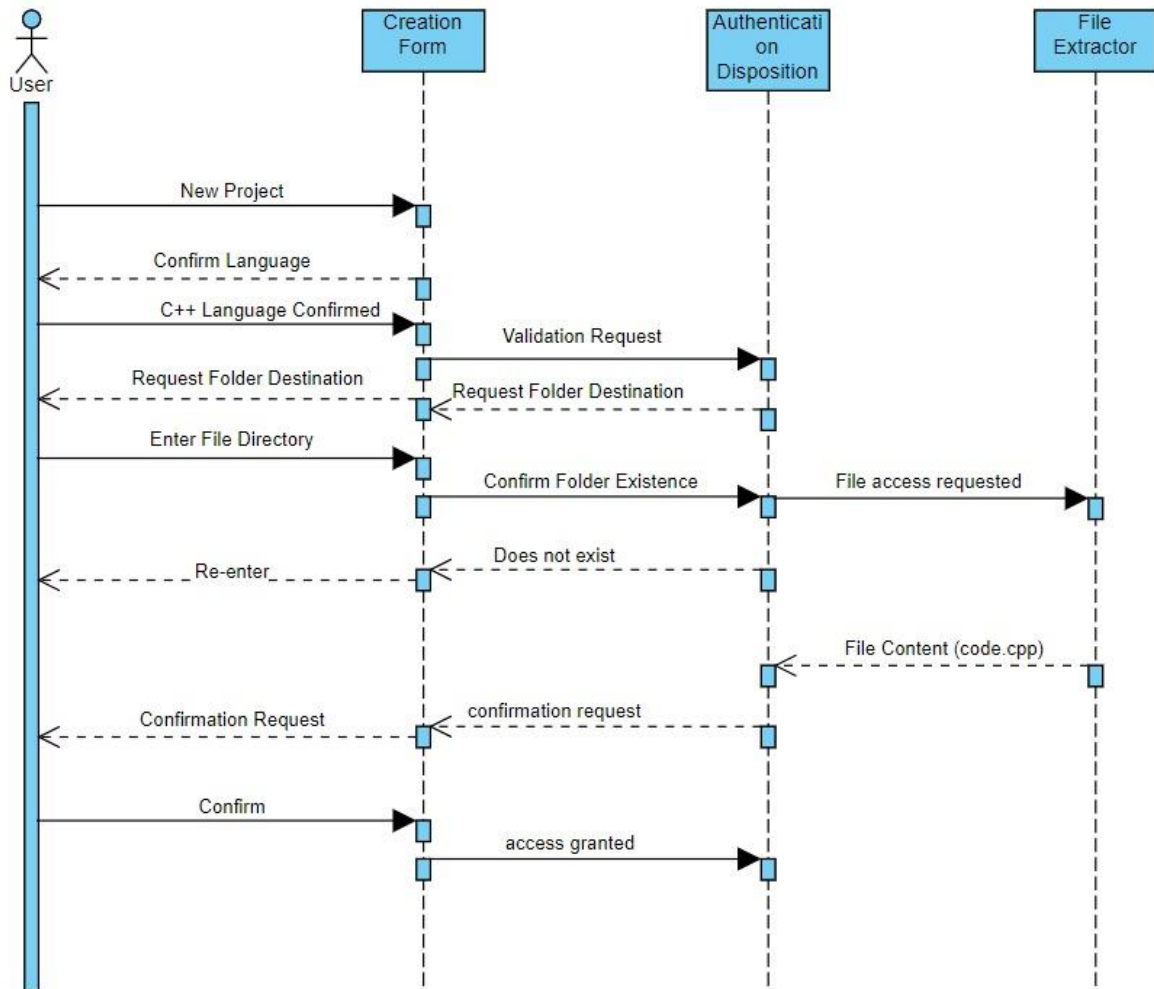
Figure 14, Import Module Diagram

4.1.4.1.2 Collaboration Diagram Design

Import Module
Collaboration Diagram

```
                        ┌──────────────┐
                        │     User     │
                        └──────────────┘
      Create New Project          Confirm Language
                        ┌──────────────┐
                        │   Creation   │
                        │     Form     │
                        └──────────────┘
    Enter File Directory        Confirm Folder
                                  Existence
                        ┌──────────────┐
                        │Authentication│
                        │ Disposition  │
                        └──────────────┘
         File Content              File Access Requested
                        ┌──────────────┐
                        │File Extractor│
                        └──────────────┘
```
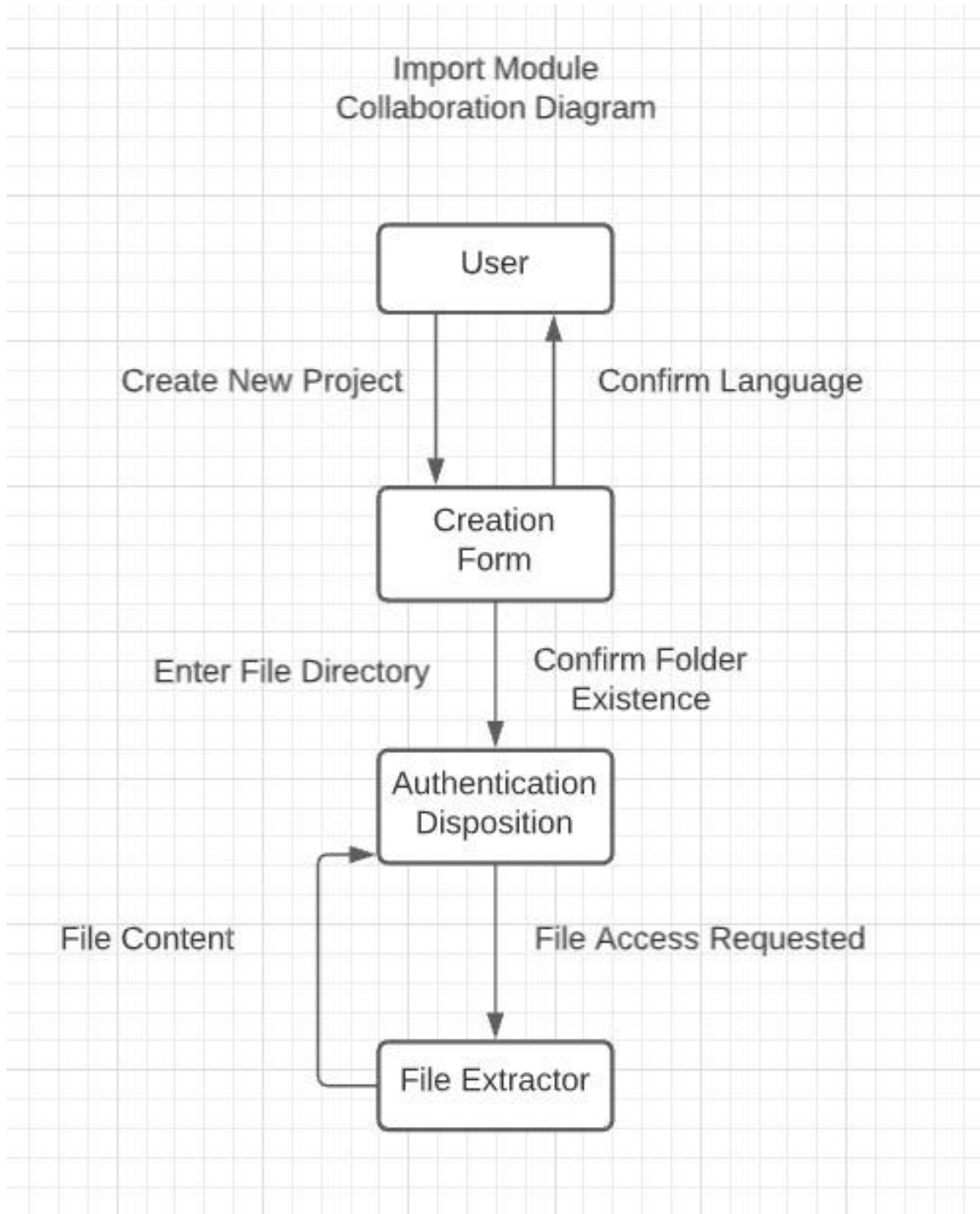
4.1.4.1.3 Activity Diagram
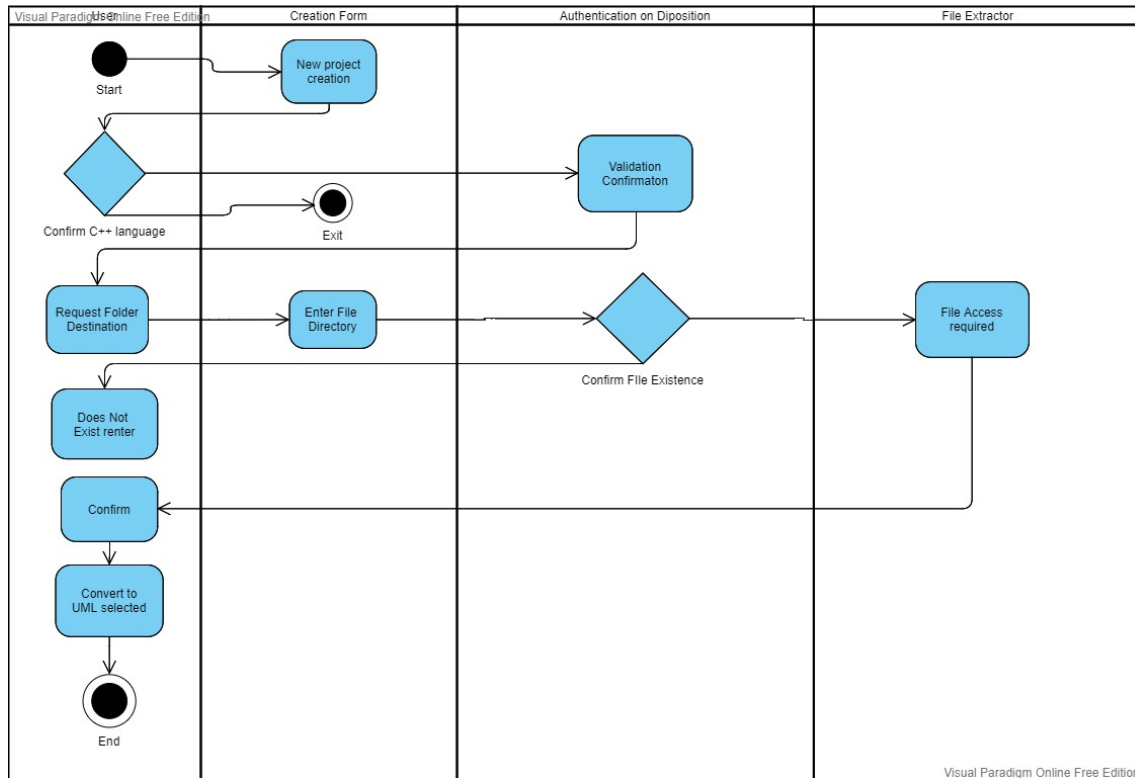
4.1.4.1.4   Design Description

The user creates a new project and enter the file location, provide access to the file. Then the software checks the file existence and then confirms the language of the code.

4.1.4.2   Parser Module Detailed Design

4.1.4.2.1   Design
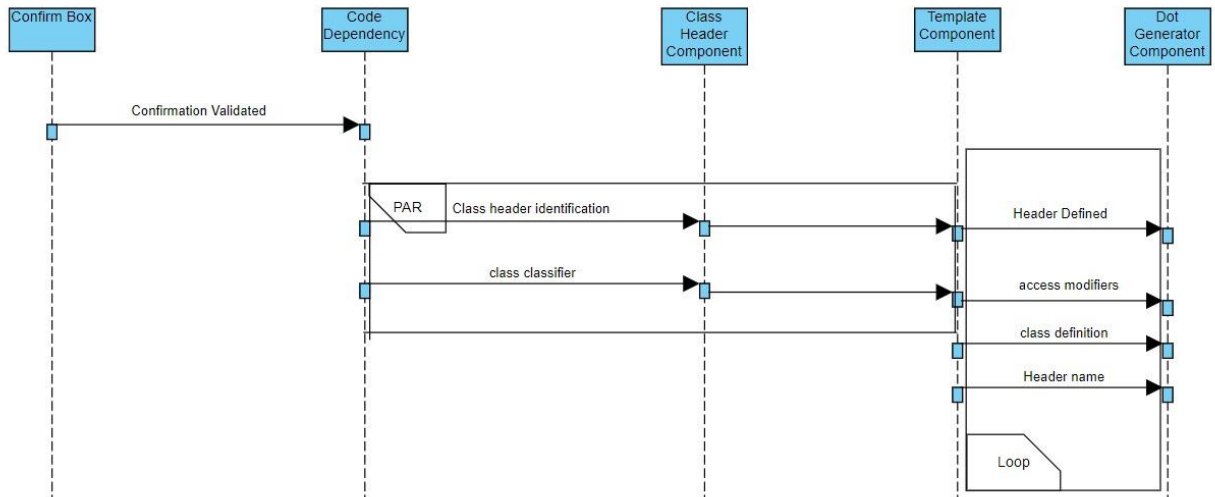
Figure 15, Parser Module Diagram

4.1.4.2.2 Collaboration Diagram Design
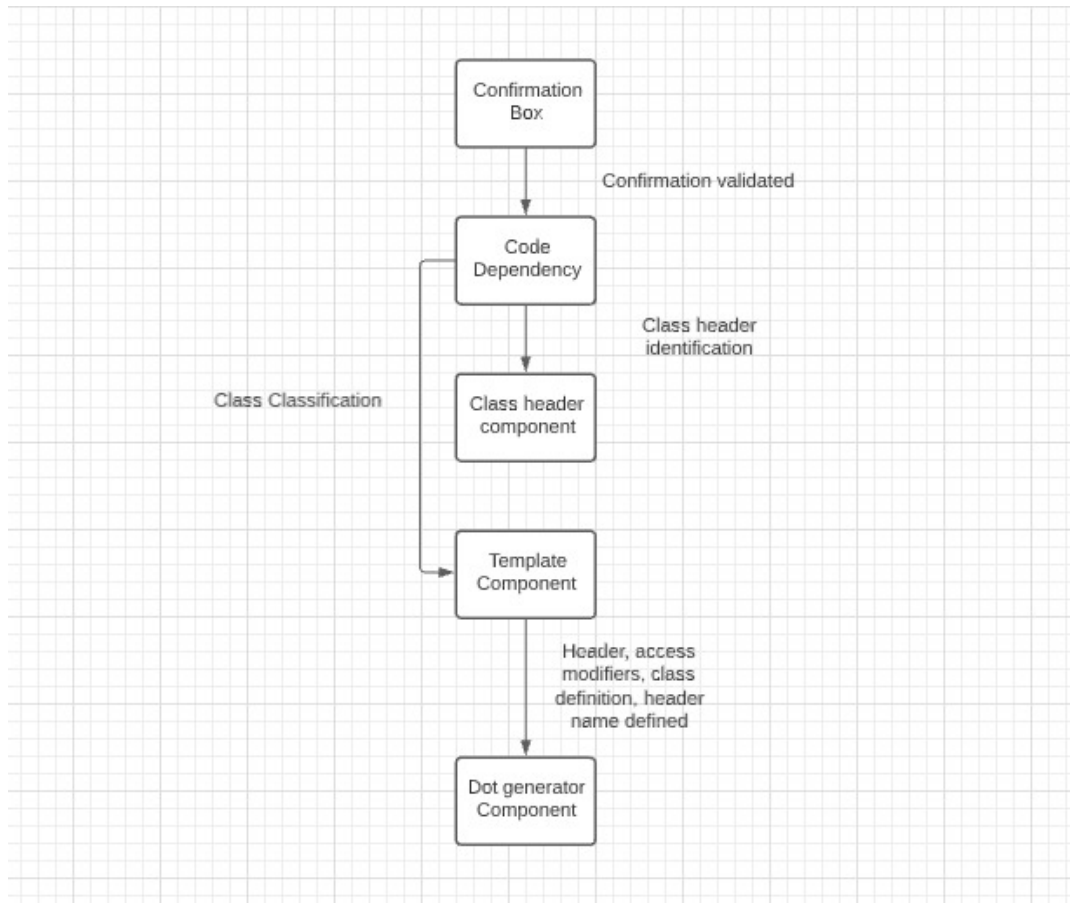
4.1.4.2.3   Activity Diagram
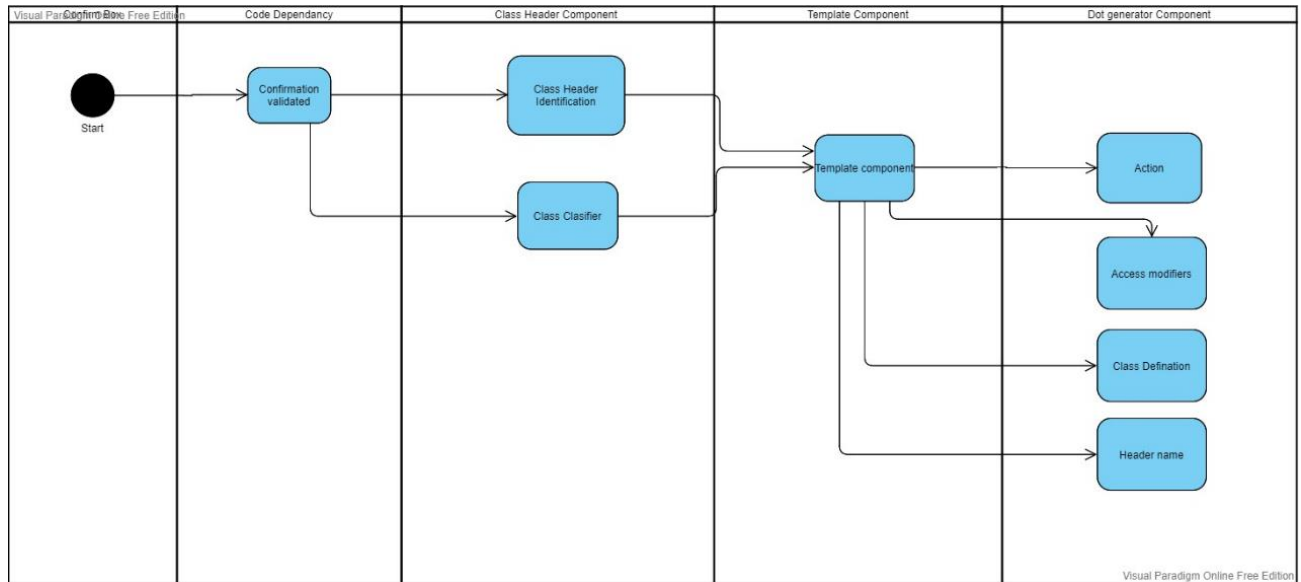
4.1.4.2.4   Design Description

After importin the file this module confirms the existence of file and language used in the code. Then the code is checked and compiled and errors are found if any. If code contains error, then the server will send response to the user. Else it will continue with the process of UML diagram generation.

4.1.4.3　　UML Generator Module Detailed Design

4.1.4.3.1　　Design

UML Module Generator
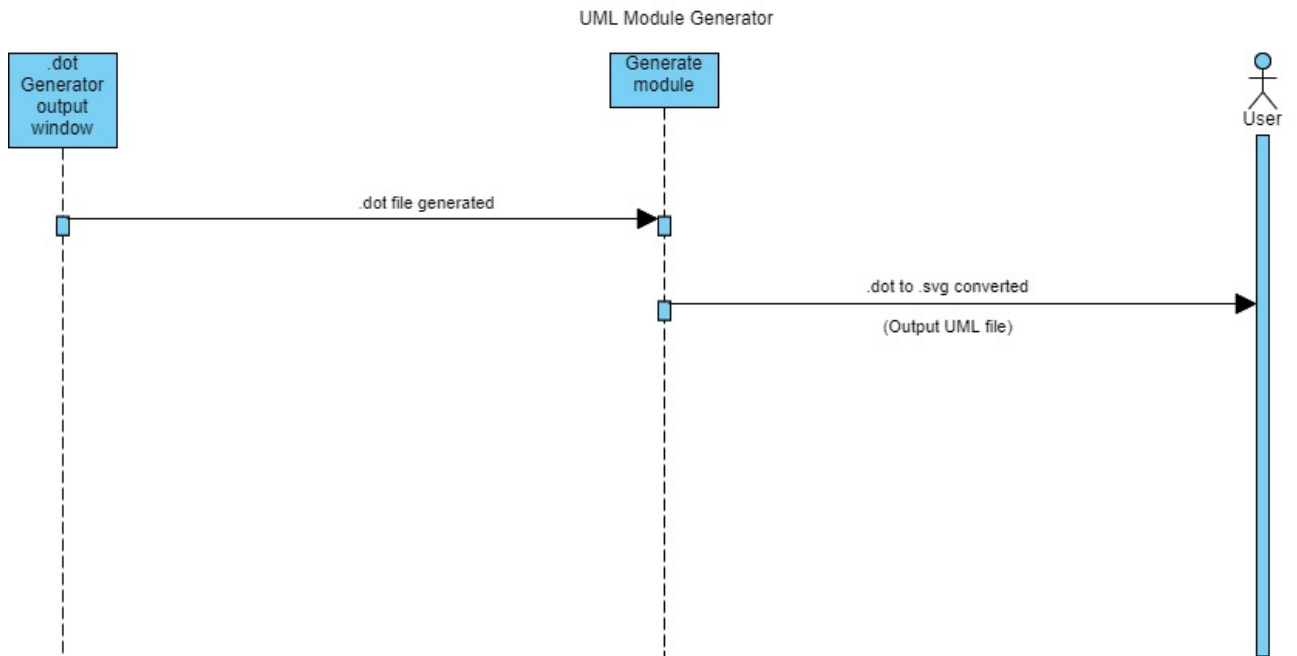
Figure 16, UML Generator Module Diagram

4.1.4.3.2   Collaboration Diagram Design

UML Generator
Module collaboration
diagram

Dot
Generator
output
window

dot file generated

Generate
module

.dot to svg converted
(Output UML file)
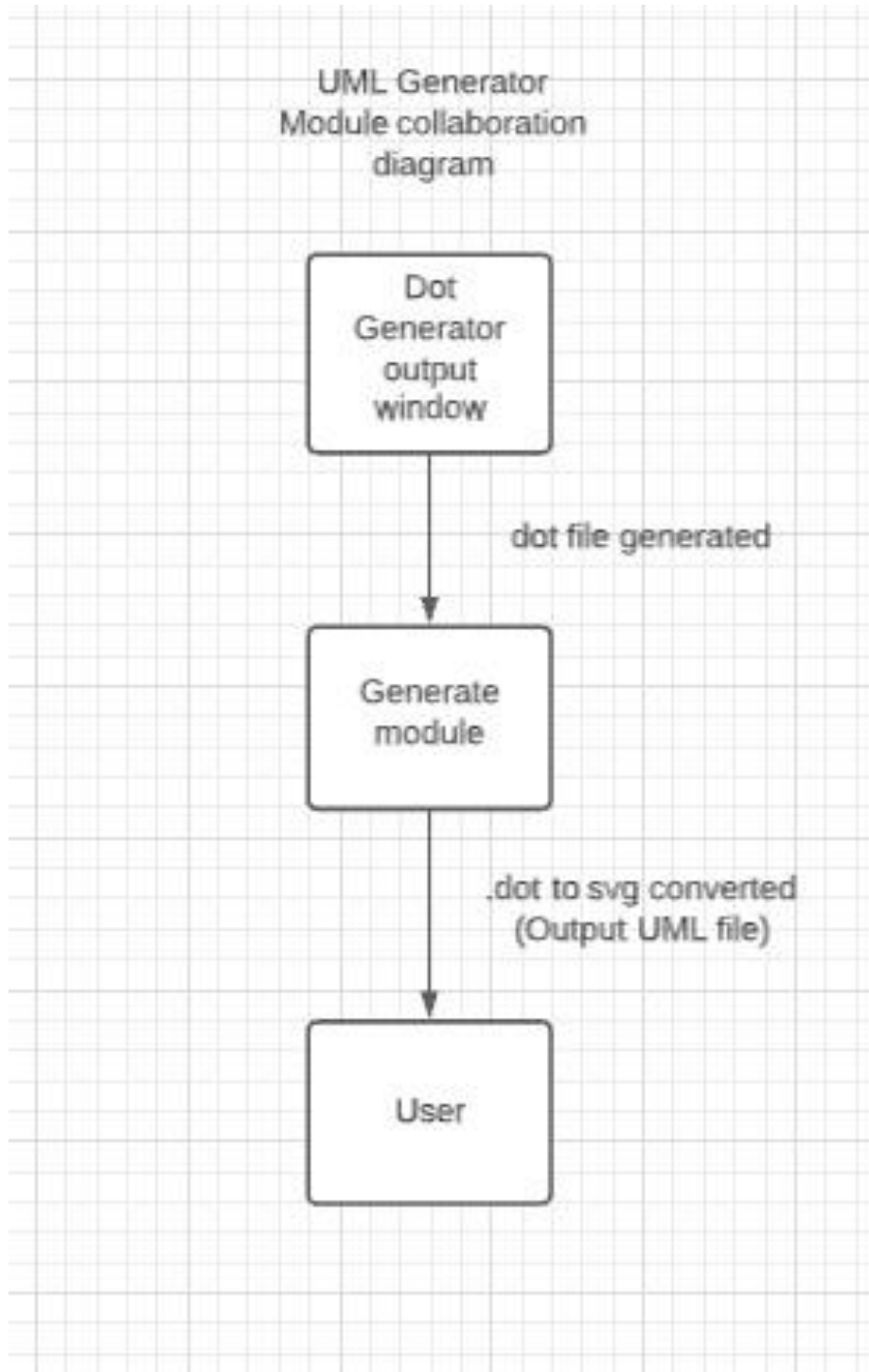
User

4.1.4.3.3   Activity Diagram
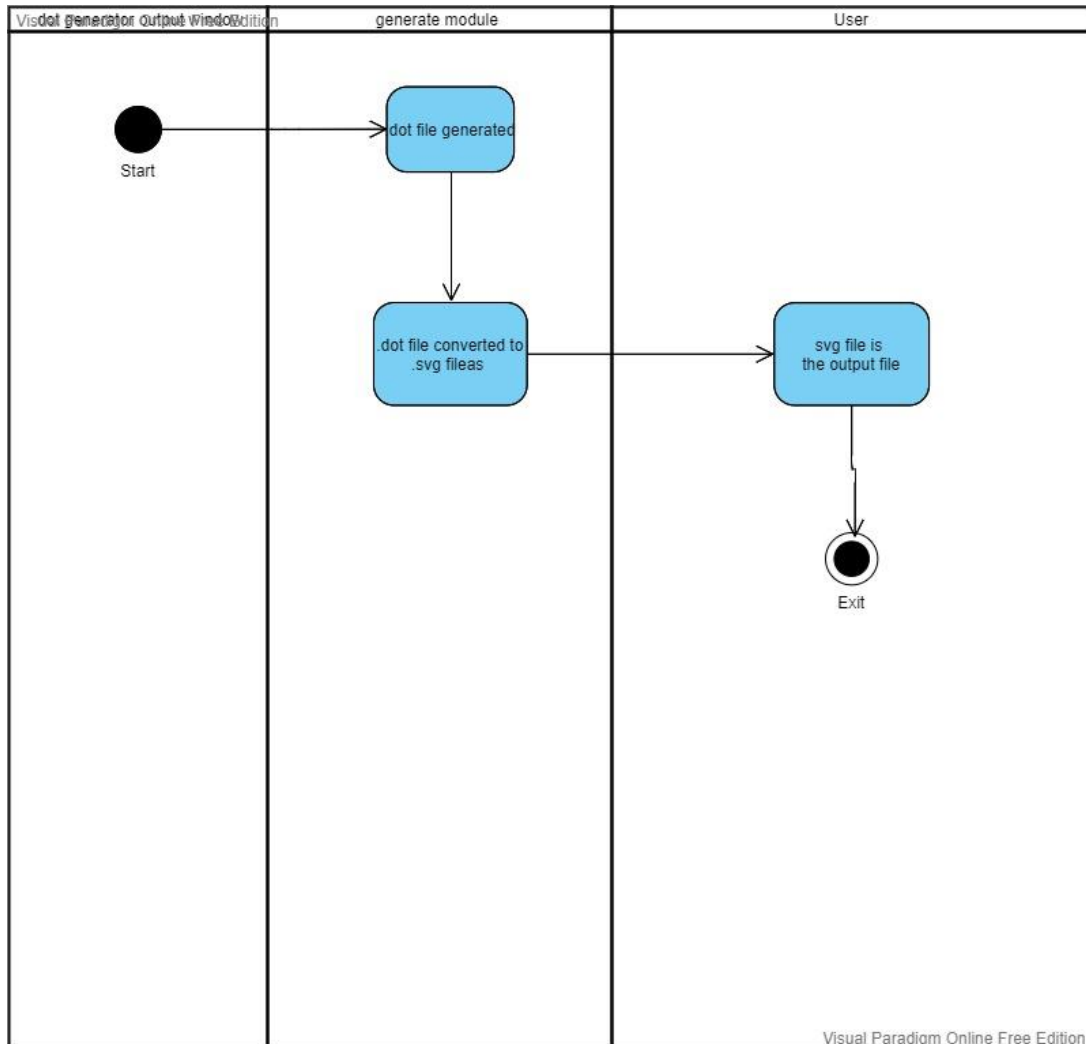
4.1.4.3.4   Design Description

When the user clicks on the UML diagram generation then the C++ code is parsed. All the classes, variables, methods, access specifiers, constructors are fetched and the code checks for any relationship between classes, variables and methods. Then the .dot file is generated and this file is converted into svg format and provide to the user.

4.3 Code Snippets:

Frontend Code:

4.3.1 **Login Page**
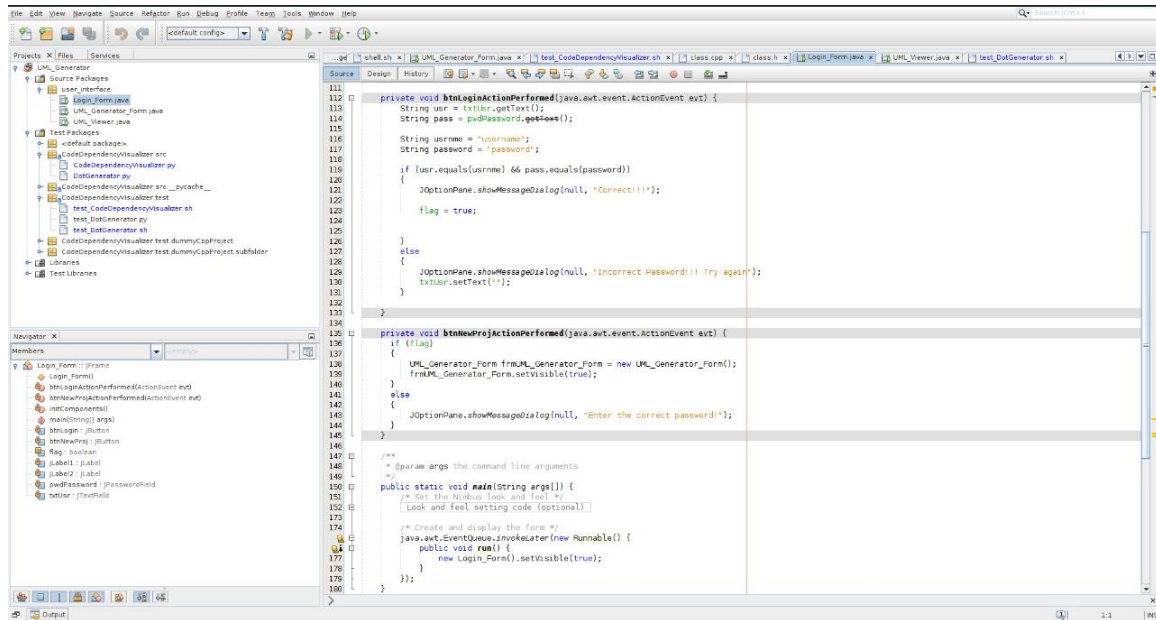
This process allows the user enter their username and password. The software fetches the information and checks it with the database and the user gets checked and authenticated. And marks the user as genuine or not.

4.3.2 **File loading/Home Page**
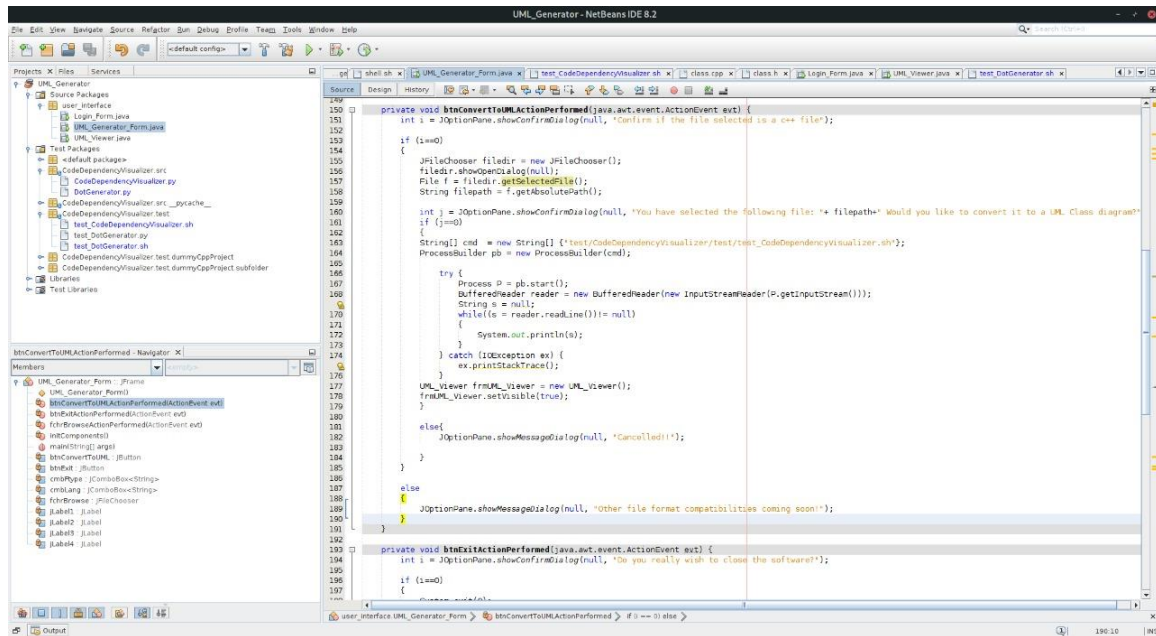
After checking the accessibility, the user is provided with the option to import the code file from the location where file is stored. Then the file is opened and the program checks and marks the language of the file. If the language obtained is C++, then the file will be added otherwise file gets rejected.

4.3.3 **Parser Module**

In this process reads the input code and separates all the classes, variables, methods, access specifiers, constructors are fetched and the code. It marks them as private, public or protected. It leads to association or inheritance. It checks for the errors if there. Then it passes the separated part forward for UML generation.

4.3.4 **Output/ UML Generator**

The software executes the code and generates the UML diagram in .dot file. This .dot file is converted into .svg file. This svg file is provided as output to the user for their respective code.

4.3.5 **Shell Scripting**

This is the script which is being triggered by the covert UML button inside the UML Generator form.
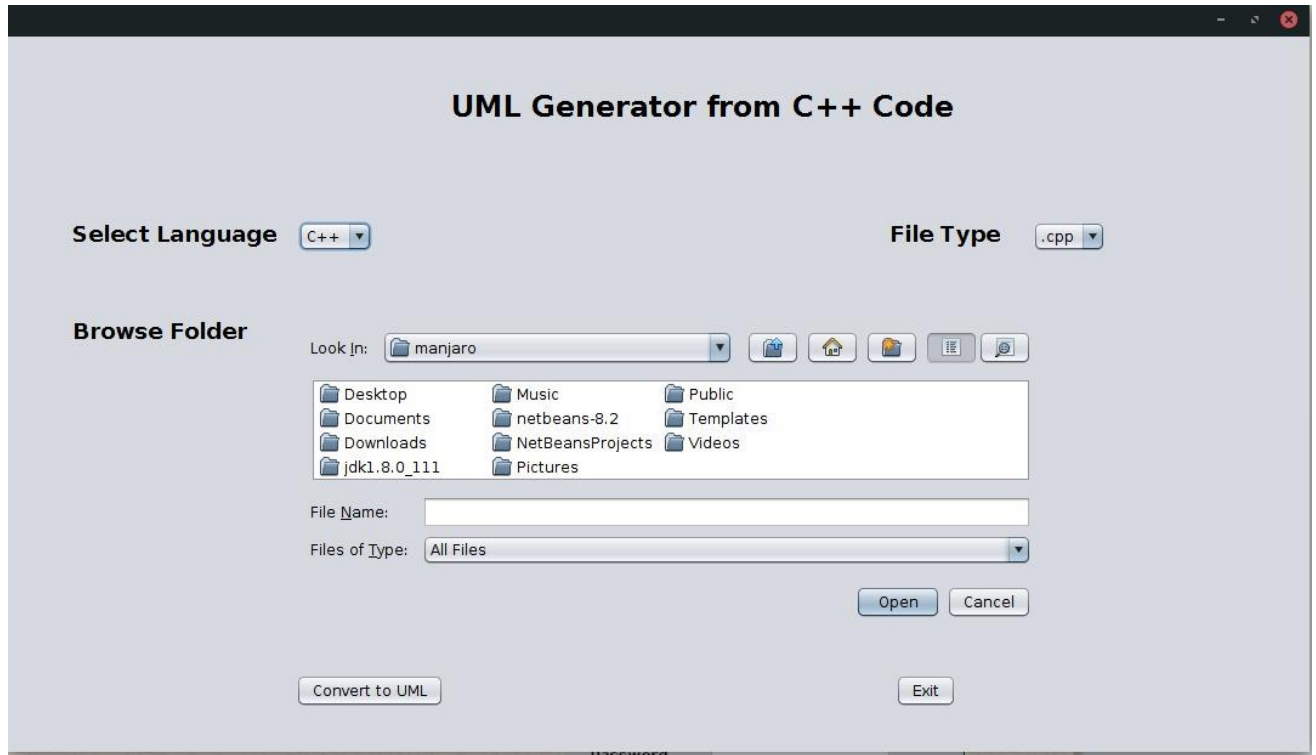
5   Project Demonstration

5.1   User Interface
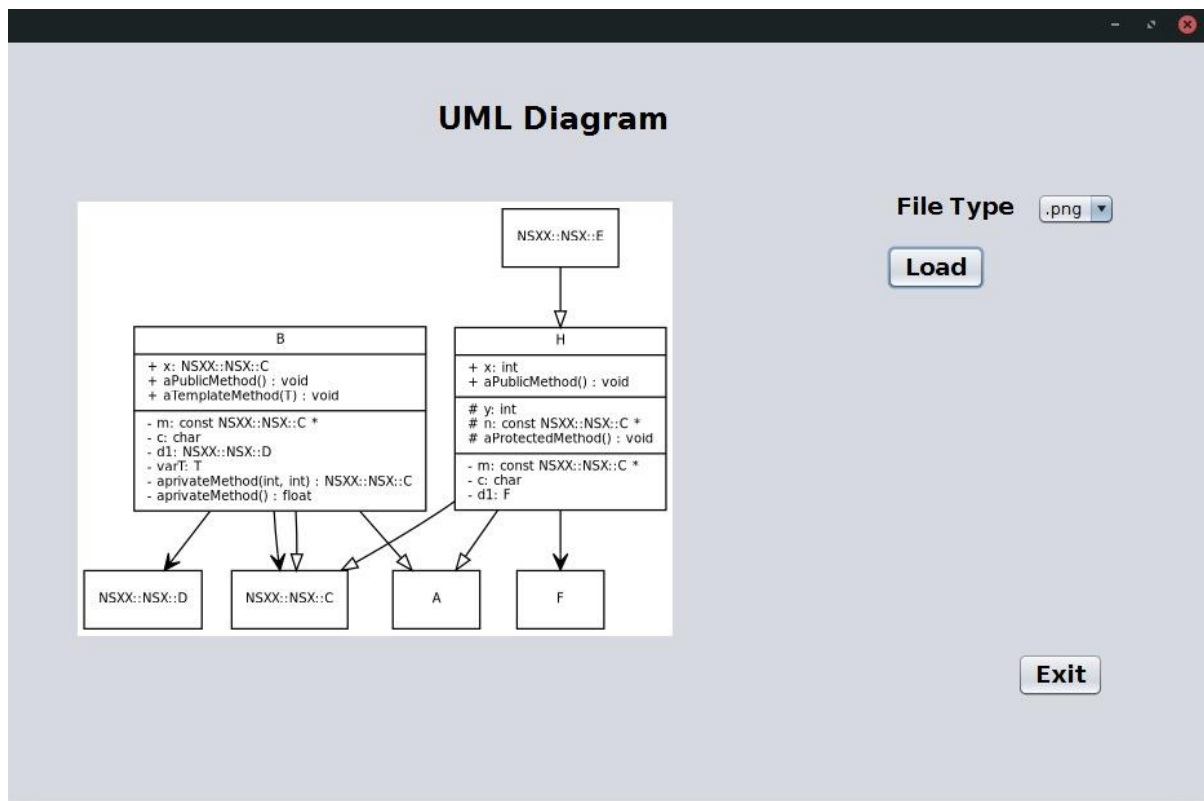
5.1.1 Login Page

It is the login form which lets the user gets authenticated to gain access to the File Chooser Window.

5.1.2   Home Page

This window allows the authenticated user to browse the C++ code file from the local device. The user has to specify the file type [.cpp, .pdf, .txt] and the program to convert the C++ header file to a UML diagram triggers from here.

5.1.3 UML Generator

This form will enable the user to view the generated UML diagram from the code and allow to save it in the desired file type.

5.2   Test Report

| Test Case ID	Test Objective	Test Data	Expected Results	Actual Results	Test Pass/Fail
1.	User Authentication	XYZ	Incorrect Password! Try Again	Incorrect Password! Try Again	Pass
2.	File Format Check	Test.java	Incorrect File Format	Incorrect File Format	Pass
3.	Private Class identification	Private class	Recognised	Recognised	Pass
4	Public Class identification	Public Class	Recognised	Recognised	Pass
5.	Protected Class identification	Protected Class	Recognised	Recognised	Pass
6.	Identify and segregate class names, methods and attributes	class.cpp, class.h	Separated class name, attributes and operations	class name, attributes and operations were separated	Pass
7.	Identify classes with signature	class.cpp,class.h	identified	identified	Pass
8.	Identify Association	class.cpp,class.h	Solid arrow	Solid arrow	Pass
9.	Identify Inheritance	class.cpp, class.h	Hollow arrow	Hollow arrow	Pass
10.	Format conversion	Uml.dot	Uml.svg	Uml.svg	Pass
11.	Identify the cardinality	class.cpp,class.h	Cardinalities detected	It couldn't detect the cardinality	Fail
12.	Identification of Private Methods	Private method()	-(methodname)	-(methodname)	Pass
13.	Identification of Public Methods	Public method()	+(methodname)	+(methodname)	Pass
14.	Identification of Protected Methods	Protected method()	#(methodname)	#(methodname)	Pass
15.	Identification of Composition and aggregation	class.cpp, class.h	Identify and denotes with corresponding notation	Couldn't identify	Fail

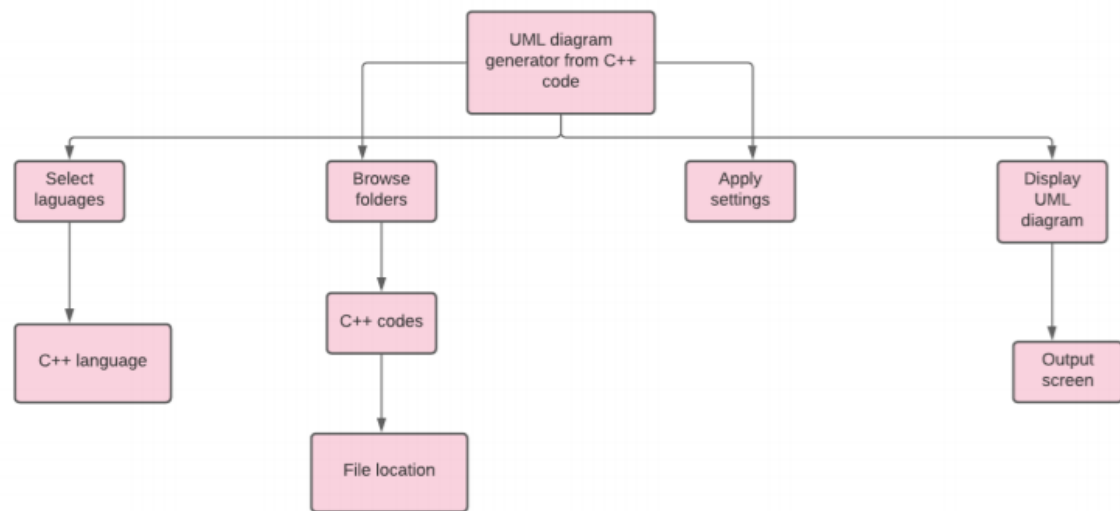# 6. SCHEDULE, TASKS AND MILESTONES
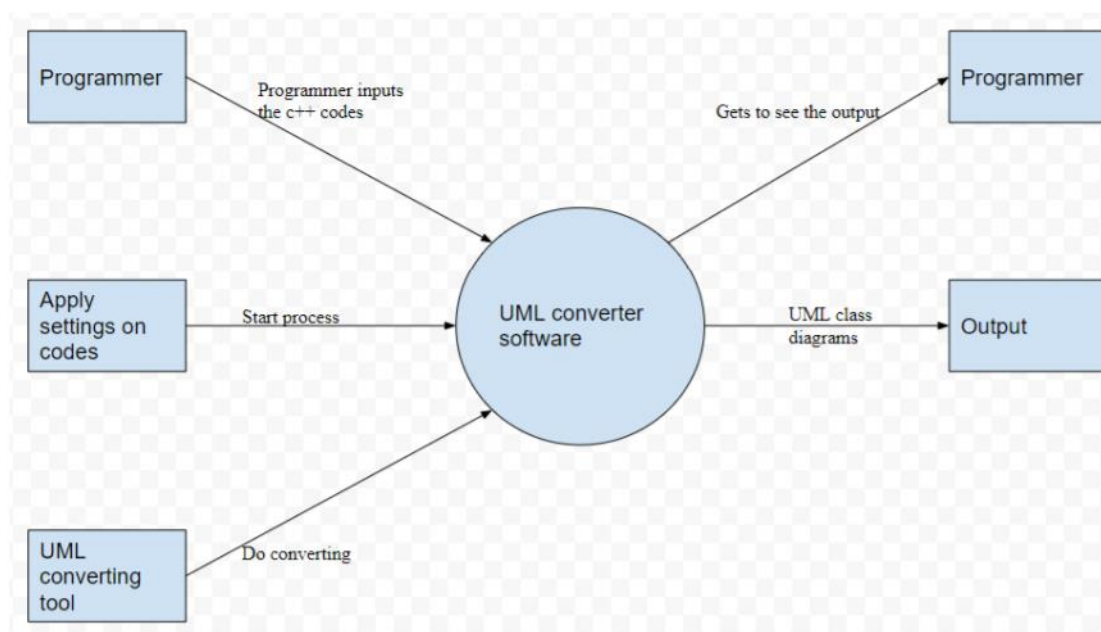
Gantt chart:
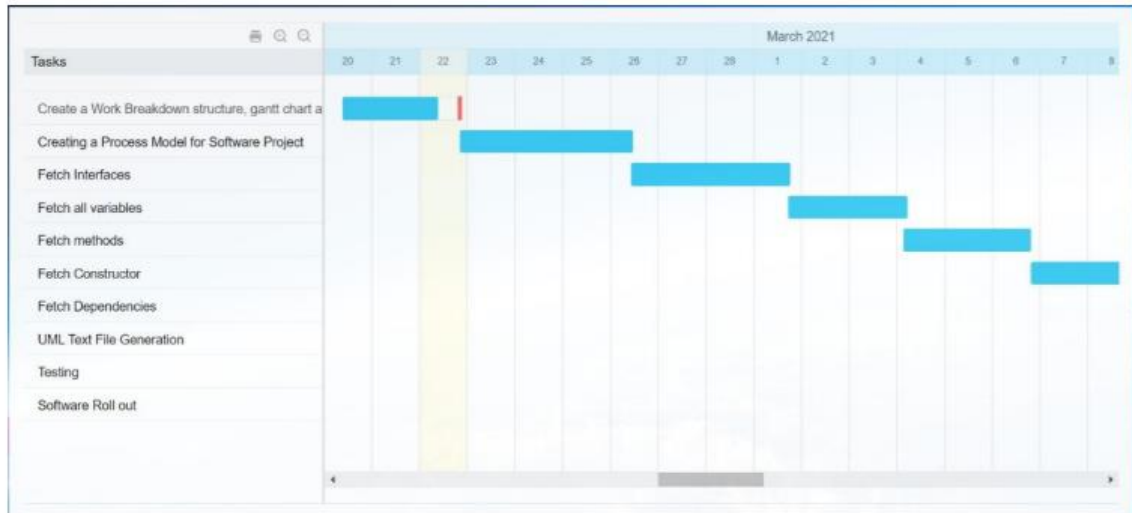
Tool used: bitrix
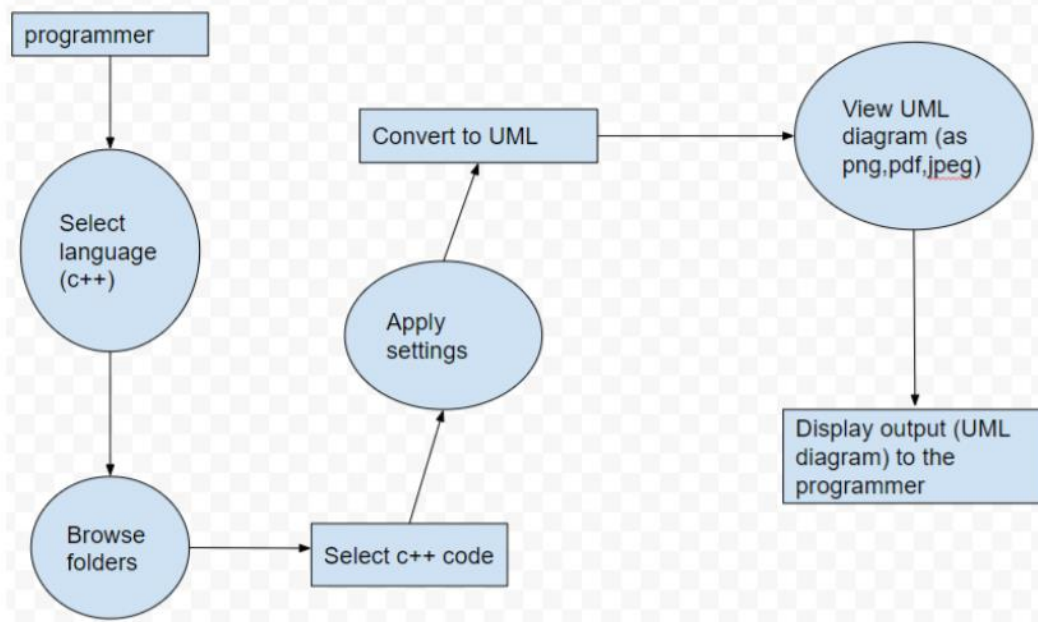
Work breakdown structure (process based):
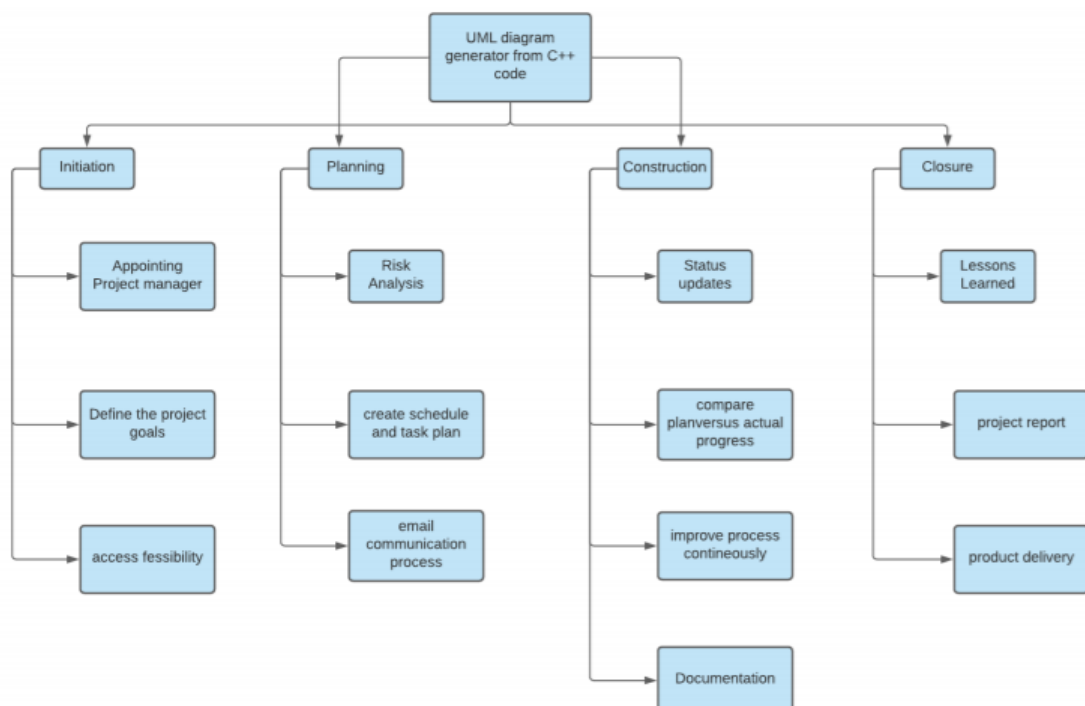
Tool used: Lucid

Work breakdown structure (product based):

Tool used: Lucid

State Transition Diagram:

[Lucid]

[Lucid]

7 a.   Cost Analysis

Since this software is an app-based process and the only costing for the software would be app maintenance which could be recovered from advertising cost from advertisement on download link or app. The app would be provided free of cost to the users with unlimited usage.

7 b.   Result & Analysis

We were able to build the tool we aimed and the tool is able to convert the code file into UML diagram with great ease and without error. The tool is able to accept or deny the file and other source material. The tool is ready to go for service with good user interface. The tool has a login page, home page and result page.

All the analysis such as cost analysis, testing and maintenance were performed successfully with positive results mentioned above in the document. The tool converts only C++ file but the error is nil.

7 c Code explanation Video links

Part 1 (75% implementation + Parser Module and UML Generator Module in detail) :
https://drive.google.com/file/d/1IZ2HcCV_NTLESxI1yLTcv5x-ki0J-fFN/view?usp=sharing

Part 2 (100% demo of the project and explanation) :
https://drive.google.com/file/d/1Erji_5nkp2xM_z4IekQquAuTardw3cub/view?usp=sharing

GitHub Repository

https://github.com/abhinav-dholi/Software_J_Comp/tree/master/UML_Generator

8. **References**

We suffered around various sites and gathered information about the things related to the
software building and learnt many things using the following sites:

- https://clang.llvm.org/
- https://graphviz.org/documentation/
- https://www.tutorialspoint.com/java/java_documentation.htm
- https://github.com/llvm-mirror/clang/tree/master/bindings/python
- https://www.guru99.com/java-swing-gui.html#:~:text=GUI%20(Graphical%20User%20Interface)%20in,easy%20interfaces%20for%20Java%20applications
- https://www.guru99.com/java-swing-gui.html#:~:text=GUI%20(Graphical%20User%20Interface)%20in,easy%20interfaces%20for%20Java%20applications
- https://www.tutorialspoint.com/uml/uml_class_diagram.htm
- https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-class-diagram/
- https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-class-diagram/
- https://link.springer.com/chapter/10.1007/978-3-642-04117-4_31