Learning Report – Applied System Development Life Cycle and Software Testing

- R Sriniketan
99002487(PS no)

L&T Technology Services

GLOBAL ENGINEERING ACADEMY

Genesis

Ver. Rel. No.	Release Date	Prepared. By	Reviewed By	To be approved By	Remarks/Revision Details
1	07/10/2020	R Sriniketan	Nisarga Hs & Safura zohareeen	Dr Prithvi Sekhar	Comments
2	08/10/2020	R Sriniketan	Nisarga Hs & Safura zohareeen	Dr Prithvi Sekhar	
3	09/10/2020	R Sriniketan	Nisarga Hs & Safura zohareeen	Dr Prithvi Sekhar	

Document History

Table of Contents

Table of Figures

ACTIVITY 1: SYSTEM DEVELOPMENT

INTRODUCTION

Formal Definition: An ATM, which stands for automated teller machine, is a specialized computer that makes it convenient to manage a bank account holder's funds. It allows a person to check account balances, withdraw or deposit money, print a statement of account activities or transactions.

My product "ATM Machine": An automated teller machine (ATM) is an electronic banking outlet that allows customers to complete basic transactions without the aid of a branch representative or teller. Anyone with a credit card or debit card can access money at most ATMs.

ATMs are convenient, allowing consumers to perform quick self-service transactions such as deposits, cash withdrawals, bill payments, and transfers between accounts. Fees are commonly charged for cash withdrawals by the bank where the account is located, by the operator of the ATM, or by both. Some or all of these fees can be avoided by using an ATM operated directly by the bank that holds the account. The functionalities include withdrawing money, depositing money and printing the balance of the account and other details.

SWOT Analysis of the product:

Strength	Weakness	Opportunities	Threats
Easy to use	Cashless payments on the rise	More advanced features yet to be added	ATM frauds
Access to cash in a lesser time	ATM may be out of cash	Easy to build the machines	ATM Security
Secure transaction	Privacy might be compromised	Any time cash can be obtained	Lesser transactions using ATM

Figure 1 aaaa

Table 2 SWOT Analysis of the product

Requirements and Research:

AGEING OF THE PRODUCT	COST OF THE PRODUCT
ATM installation took off towards the end of the 1970 to withdraw money	$ 300,000 in the 1970s in US initially when the machine was invented
Further enhancements by adding more features like depositing money	Rs 400000 in the early 2000s
Chip added in the ATM card for security purposes	Rs 850000-1000000 is the current price of the ATM machine

Table 3 Ageing v/s costing of the product

High level requirements:

ID	DESCRIPTION
HL_01	Check balance function to check the balance of the account
HL_02	Withdraw function to withdraw money
HL_03	Print a statement of account

Table 4 High level requirements of ATM

Low level requirements:

ID	DESCRIPTION
LL_01_HL01	Enter the required details to check the balance
LL_02_HL02	Enter the correct ATM pin
LL_03_HL03	Account details should be updated after a transaction

Table 5 Low level requirements of ATM

DESIGN OF THE SYSTEM

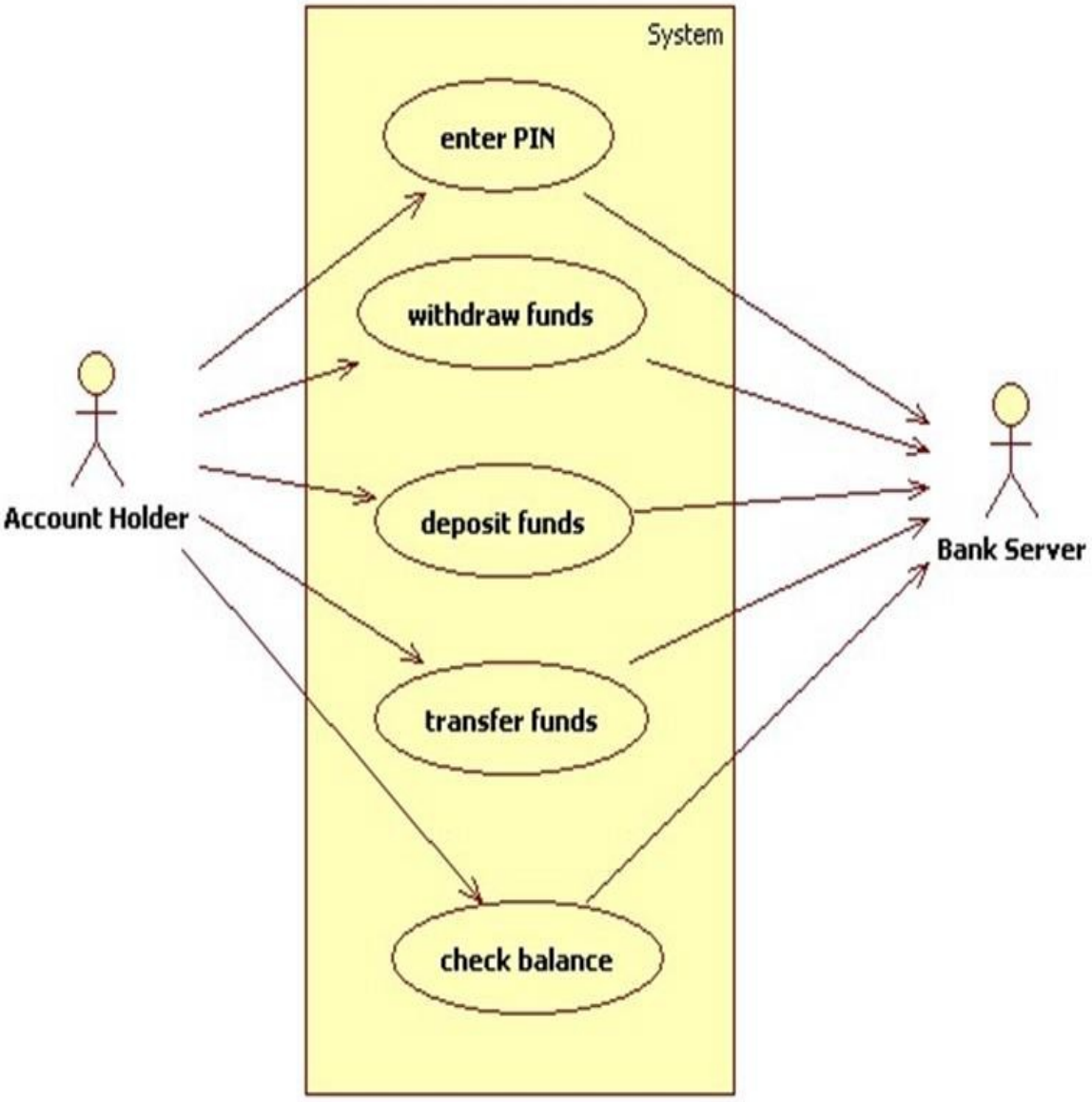

Figure 6 Use case diagram of an ATM machine

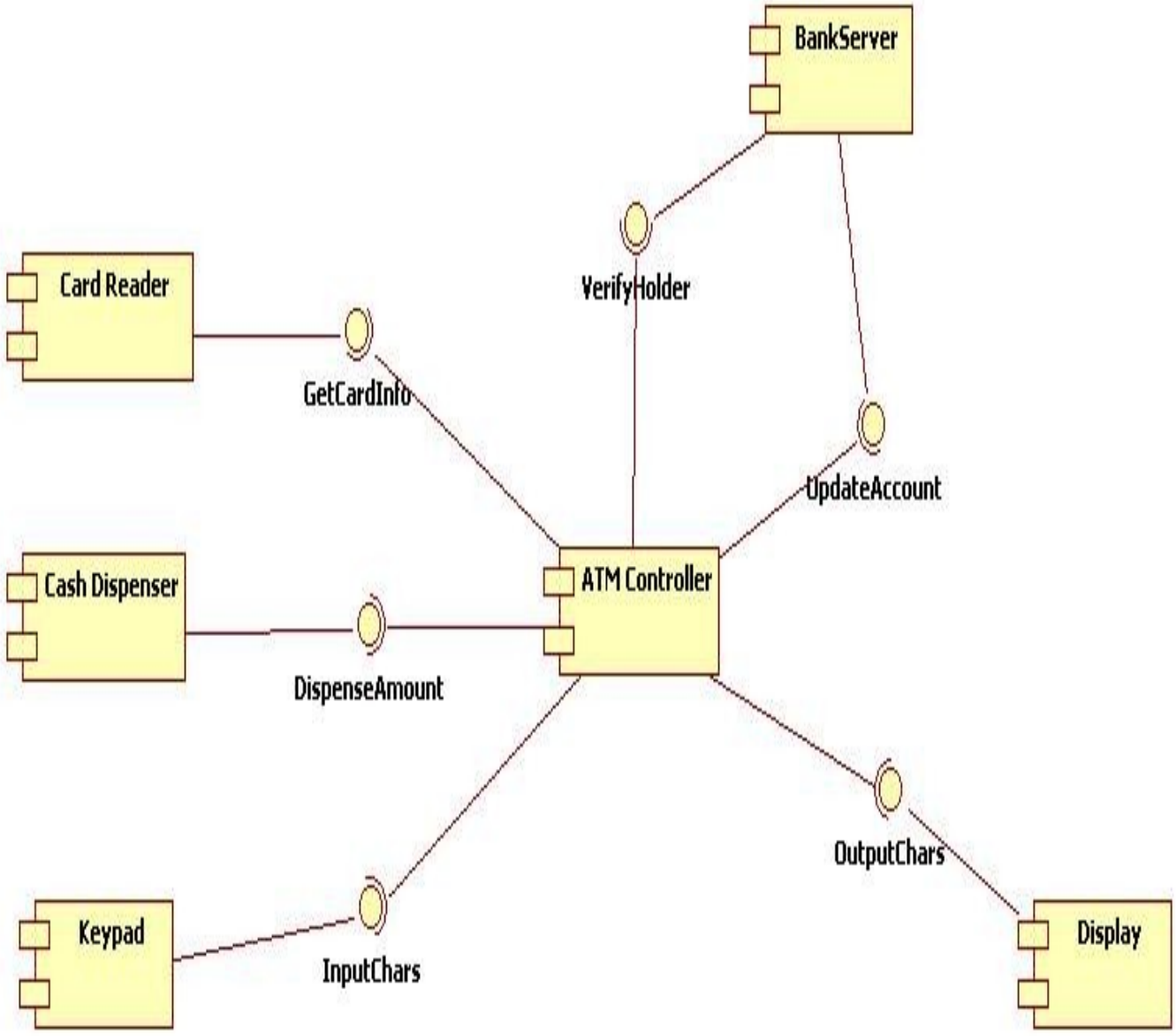

Figure 7 Component diagram describing withdrawal of cash from the ATM machine

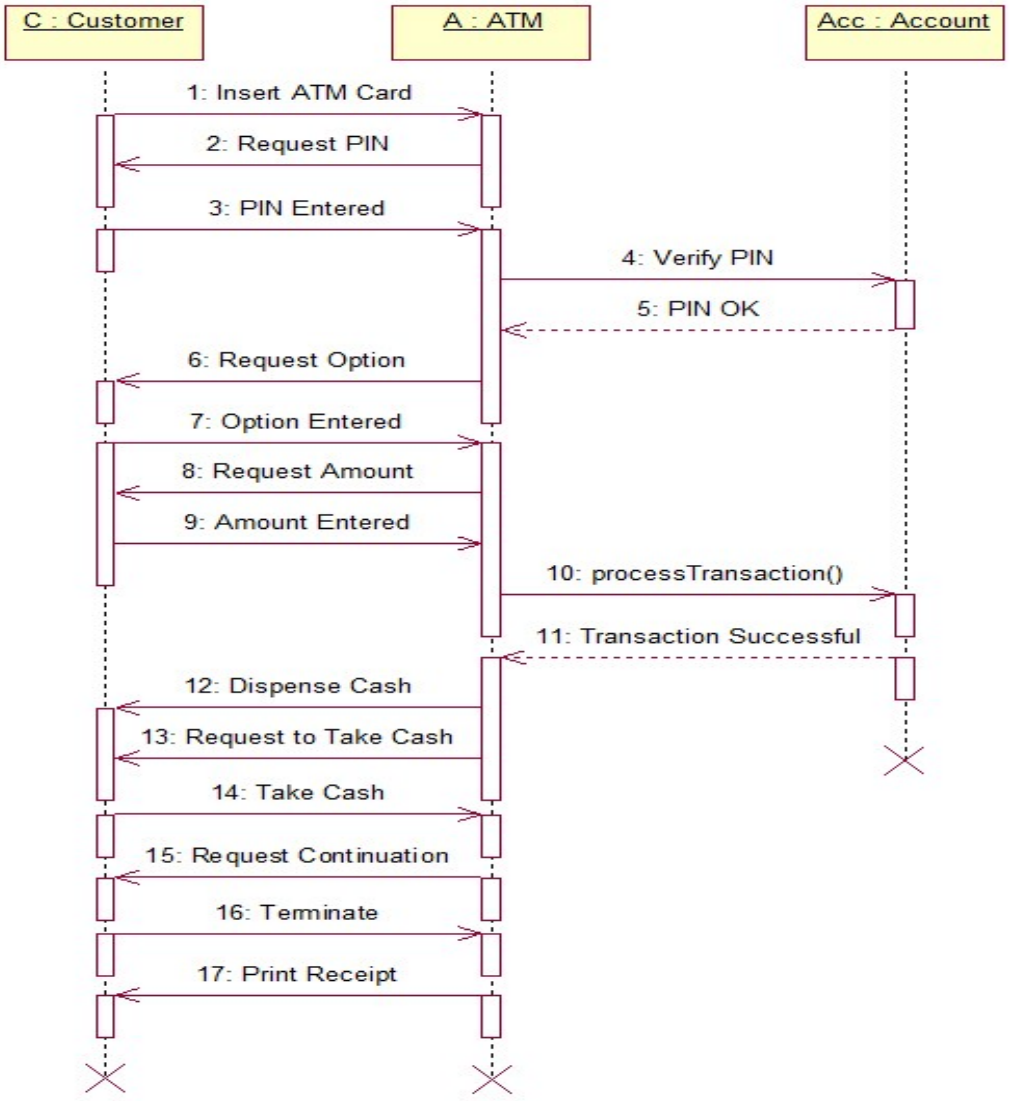

Figure 8 Sequence diagram of ATM machine describing withdrawal of cash

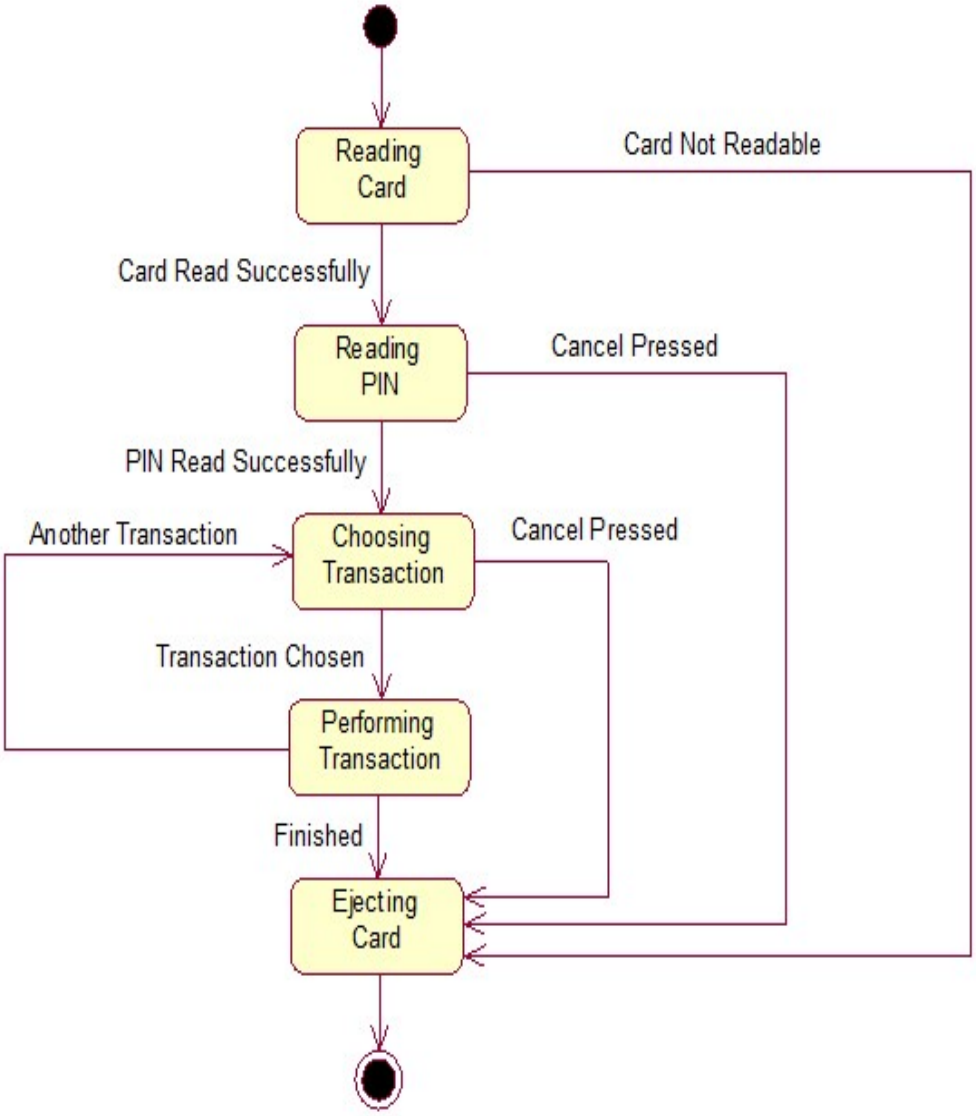

Figure 9 State diagram of an ATM machine

TEST PLAN

Requirement based test plan:

ID	DESCRIPTION	PRE-CONDITION	EXPECTED INPUT	EXPECTED OUTPUT	ACTUAL OUTPUT
HL_01	Check balance	Account should exist	ATM Pin	Print balance	Balance is checked
HL_02	Withdraw money	Account should be present	Amount to be withdrawn	Amount dispensed to the user	Money is withdrawn
HL_03	Printing balance	Account should be present	ATM Pin	Balance to be printed	Balance is printed
LL_01_HL_01	Enter the required details to check the balance	Account should be present	ATM Pin	Transaction should be processed and details should be updated	Transaction should be processed and details should be updated
LL_02_HL_02	Enter the correct ATM pin	Balance should be greater than zero	ATM Pin	Requested transaction should be processed	Requested transaction should be processed
LL_03_HL_03	Account details should be updated after a transaction	A transaction should be made	Transaction type and it's details	Account details are updated	Account details are updated

Table 5 Test plan of the ATM

Scenario based test plan:

1) When a user enters a wrong ATM pin.

2) When a user enters a wrong denomination to be withdrawn.

3) When a user enters more money to be withdrawn than the balance in the account

Boundary based test plan:

1) When the user enters zero as money to be withdrawn

2) When a user wants to withdraw more money than it is present in the ATM machine

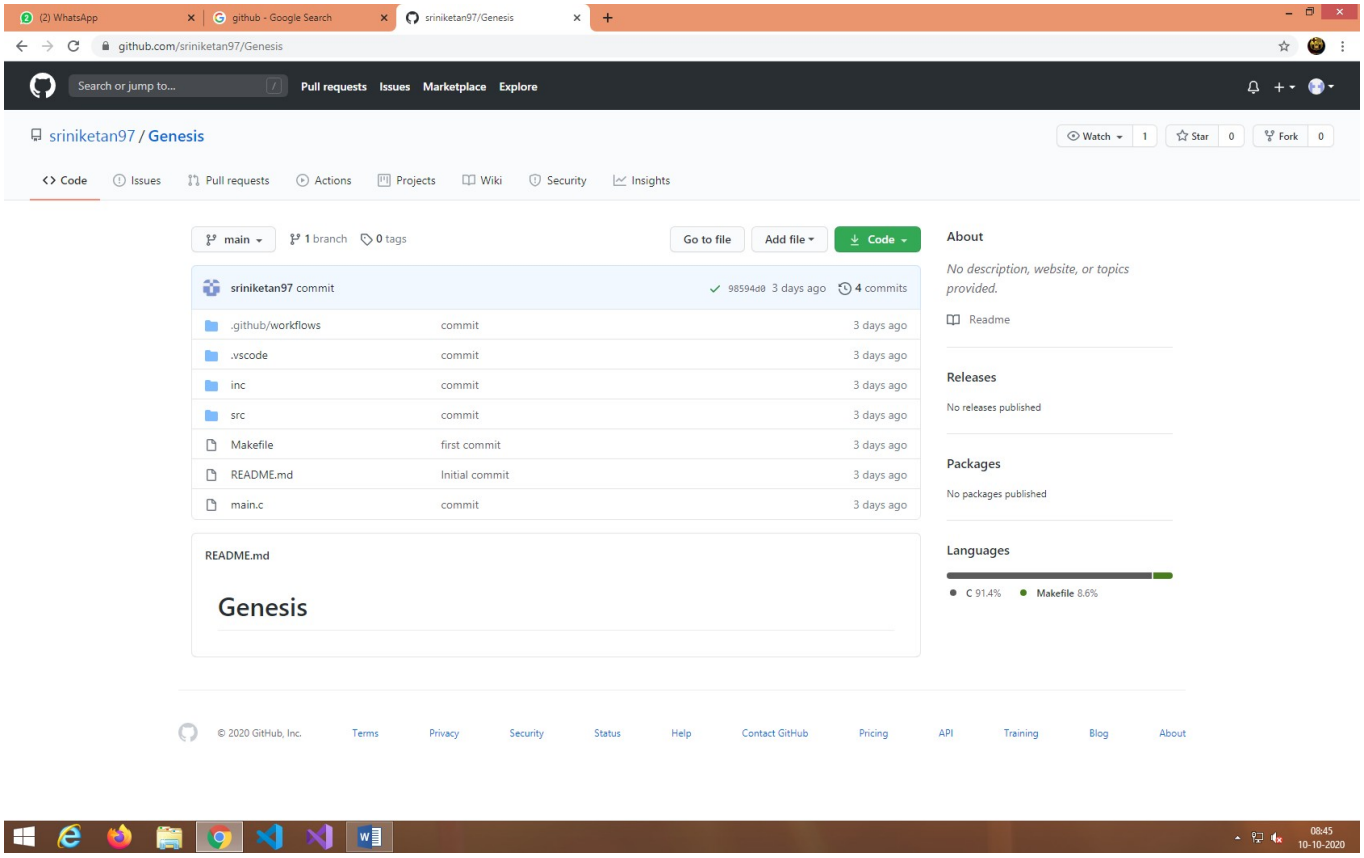

Figure 10 Github file system containing the CI/CD of the ATM machine

REFERENCES - https://github.com/sriniketan97/Genesis

Things I liked doing the activity – SWOT analysis of the product
Things in which I faced difficulty – Makefile in github

ACTIVITY 2: CALCULATOR

Introduction:

A calculator app is one of the most basic yet important apps on your phone. You need to deal with calculations every day and a calculator app allows you to use your smartphone for all the calculations on the go. Even though the default calculator app in most Android smartphones today is fairly feature packed and capable enough to handle a few complex equations. If you need a scientific calculator that can help you with all types of calculations and math problems, you will have to go for a third-party calculator app.

The calculator we have designed will have,

- Simple Calculations like addition, subtraction, multiplication, division and modulo division.
- Scientific Operations like nth power of a number, square root of a given number, factorial of a number and multiplicative inverse of a number.
- Conversion operations like currency conversion (US Dollars to Indian Rupees), Length conversion
- (Feet to Inches) and time conversion (Hours to Minutes).

Requirements:

High Level Requirements:

ID	Description
HL_01	A mobile calculator app that should perform simple calculations, scientific calculations and conversions.
HL_02	The calculator is developed using standard C language and should run on all machines supporting gcc compiler.
HL_03	Should display following menu bar to users like-1. Add, 2. Subtract, 3. Multiply, 4. Divide, 5. Modulus, 6. Power, 7. Square root, 8. Factorial, 9. Inverse, 10. Currency, 11. Length, 12.Time, 13. Exit

Table 6 High Level Requirements for Calculator

Low Level Requirements:

ID	Description
LL_01	Should exit when entered 13
LL_02	Prevent users from divide by zero error.
LL_03	Can use either one or two operands
LL_04	Should display “Invalid Selection” when user chooses menu option less than zero and greater than 13

Table 7 Low Level Requirements for Calculator

System design:

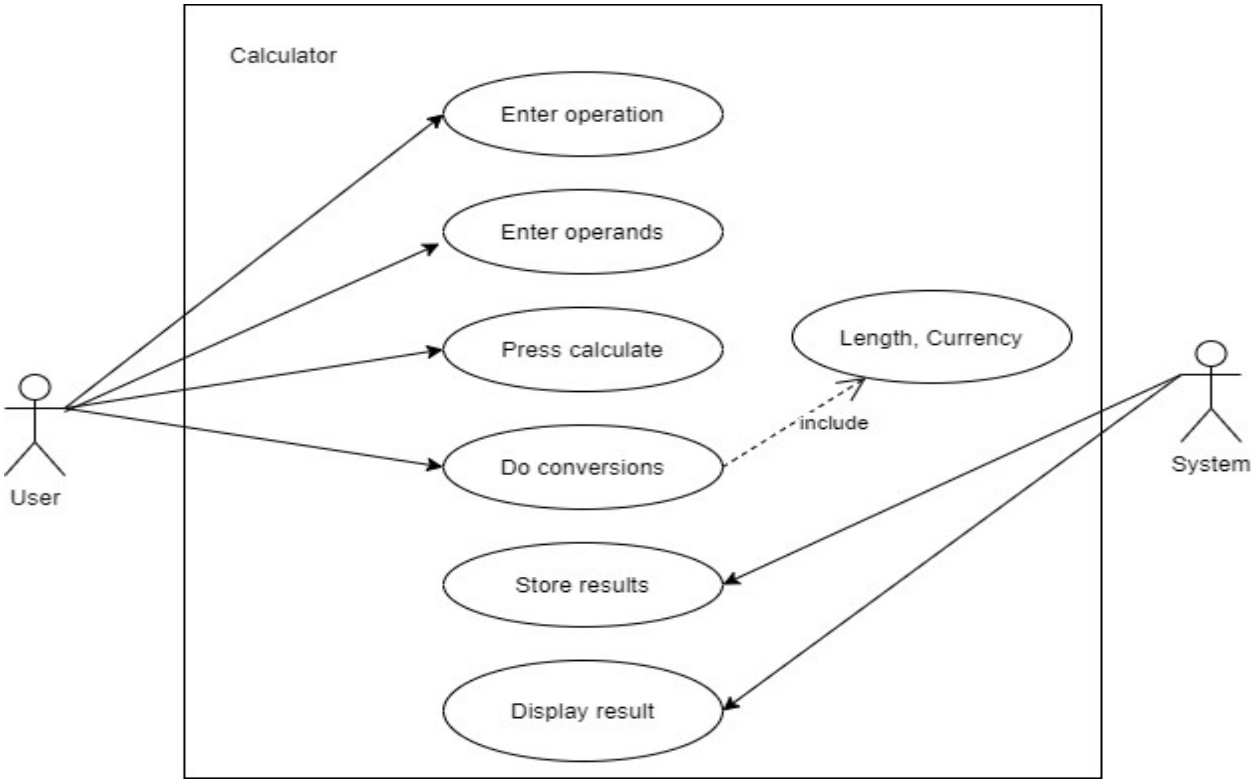

Figure 11 Use case diagram of calculator

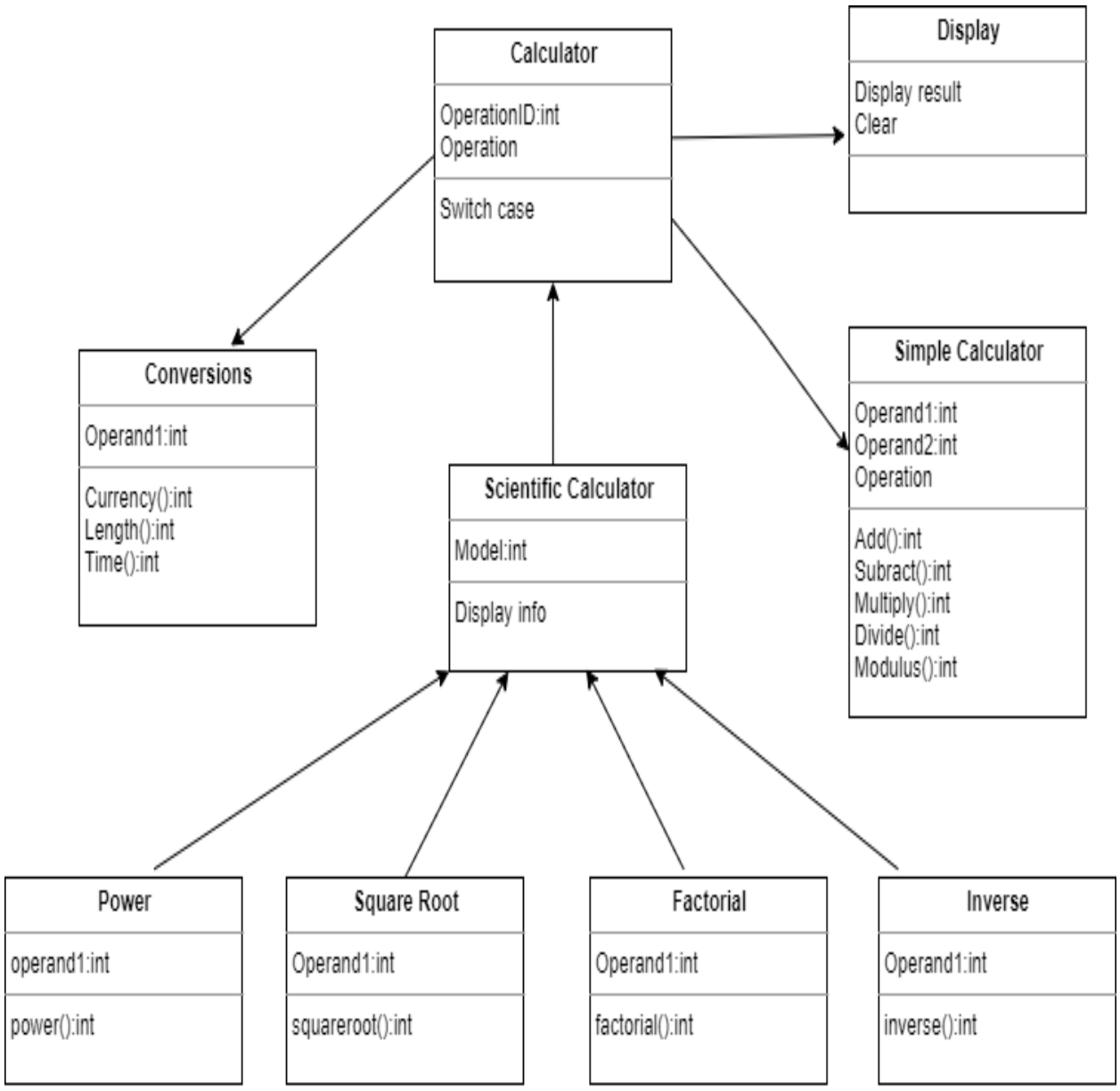

Figure 12 Class diagram of calculator

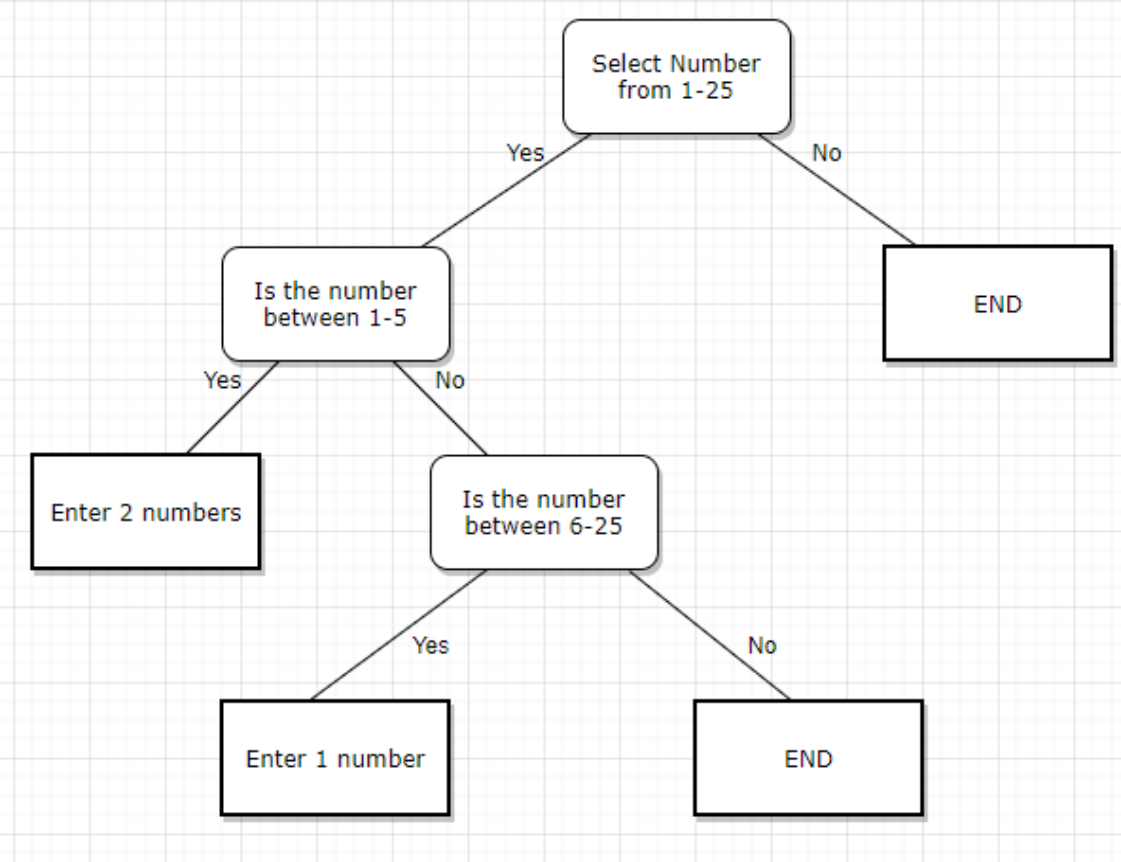

Figure 13 Selection of Operation Logic for a calculator

Test Plan:

Requirement based testing:

ID	Description	Pre-Condition	Expected input	Expected output	Actual output
T_01	Work For function add	Operands must be integers	7+26	33	33
T_02	Work For function add	Operands must be integers	234578+345698	580276	Error: Output value is in double
T_0	Work for	Operand	2*4	8	8

3	function multiply	s must be integers			
T_04	Work for function multiply	Operands must be integers	2456*3456	8487936	Error: Output value is in double
T_05	Work for function divide	Operands must be integers	4/0	Divide by zero error	Divide by zero error
T_06	Work for function square root	Operand must be integer	25	5	5

Table 8 Test plan for calculator

Boundary condition testing:
1) When the number is too large to perform an operation
2) When a negative number is given as an input to find the square root

Scenario based testing:
1) When the user enters a character value instead of a number
2) When the user enters an operand which is undefined

REFERENCES (CI/CD) - https://github.com/99002487/genesis_sdlc

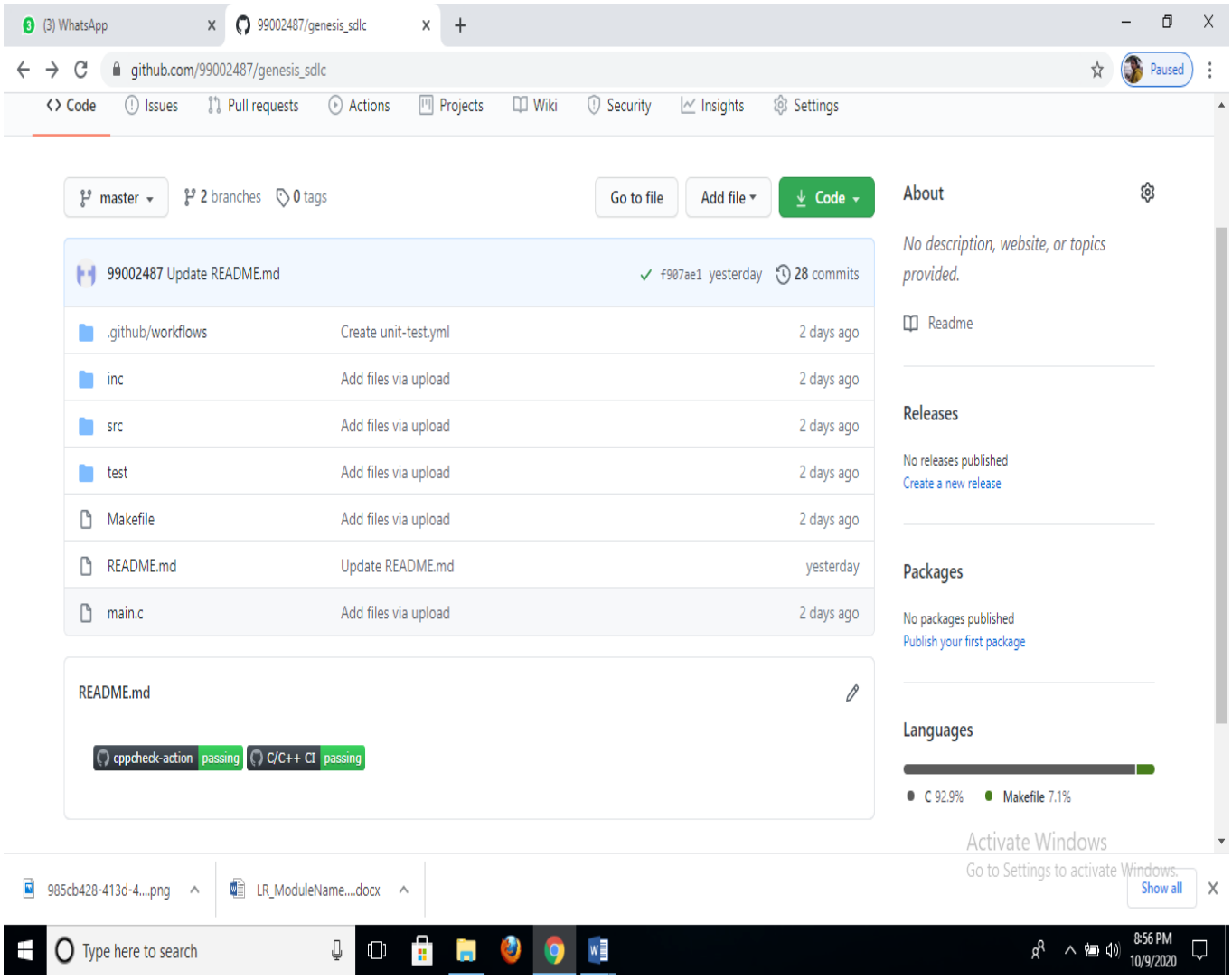

Figure 14 Github file system containing the CI/CD for the calculator

Things I liked doing the activity – Writing the test plan.
Things I faced difficulty in – Getting the cpp check done.

ACTIVITY 3: Agile methodology and concepts

Theme: To build software for an ATM machine that is efficient and easy to use for a user and the user can have hassle free experience in performing various actions that an ATM machine can perform to get easy access to money at any time of the day.

Epic:
1) To build a function which checks the balance of the user's account and displaying the same on the ATM screen.
2) To build a function which withdraws money for the user according to the amount entered by the user
3) To build a function which prints the statement of account after a transaction is processed.

User stories:
1) As a Customer I want to Login to my account using card and PIN code so that I can perform the transactions.
Acceptance Criteria –
• System must validate the card and pin code
• In case Customer enters wrong Pin code three times then the system locks the card.
2) As a Customer I want to to check the balance of my bank account So that I can perform transactions.
Acceptance Criteria –
• Customer needs to be logged in before checking balance.
• Balances is displayed.
3) As a Customer I want to withdraw cash from my bank account through ATM So that I may save my time.
Acceptance Criteria –
• Customer needs to be logged in before withdrawing cash.
• System checks to see if the request amount exceeds the balance
4) As a Customer I want to logout from my bank account through ATM So that I may end up my ATM session.
Acceptance Criteria –
• System asks user if the user wants session report and receipt for the entire session.
• If yes then the receipt is dispensed
• User is logged off from the account

Scrum: The requirements gathering process is planned for the 1st sprint wherein all the user requirements are gathered for further processing.
The next sprint is planned to create a check balance functionality wherein all the requirements are considered and the appropriate functionality is built and the release is planned at the end of the sprint.
In the further sprints the other functionalities are planned where each functionality is released for each of the sprint and correspondingly integrating the functionality with the main project.

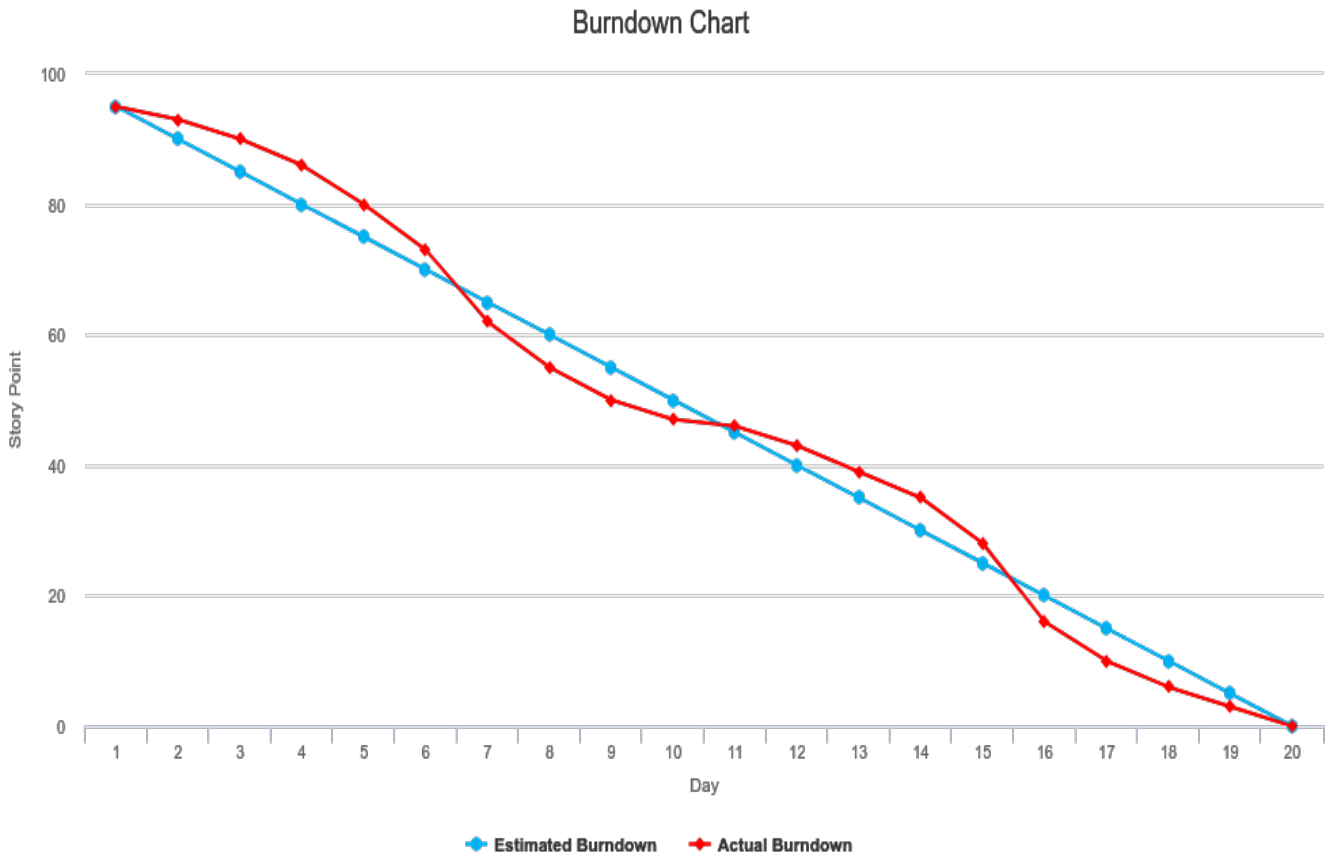

Figure 15 A burndown chart in a sprint in an agile model of software development

Things I liked doing the activity – Writing the user stories

Things I faced difficulty in – Getting the difference between epic and user stories.

ACTIVITY 4: Make file in github

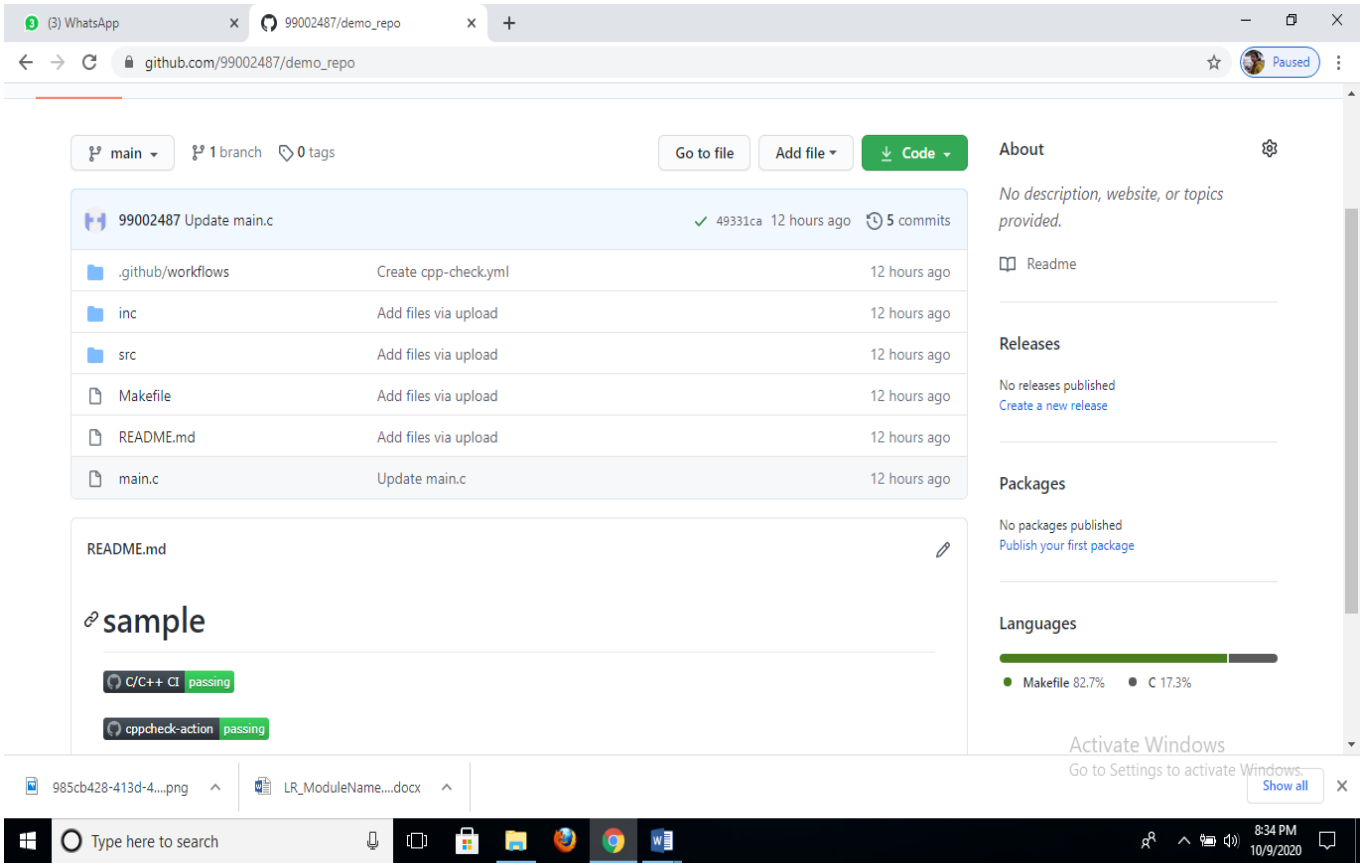

Figure 16 A screenshot of the files on the github which makes use of a makefile

Summary of the activity 4: A simple code was written to understand how the makefile on github works in which a simple program is taken to understand how the makefile works.

Things I liked doing the activity – Writing the simple code.
Things I faced difficulty in – None.

ACTIVITY 5: Tools used in Agile and V model

S L No	Tools	Description	Link
1	**Mind maps**	• tool to capture ideas, requirements and help organize a conversation with many tangents	Mind mapping
2	Reqtest	• Prioritizing Requirements • Highly customizable • instant reviewing tracking of changes	https://reqtest.com/features/requirement-management/?utm_source=softwaretestinghelp
3	Visure	• End-to-end req. • Traceability, • Req. management, • Req. Gathering, reusability	https://visuresolutions.com/
4	Jama	• Align tests & requirements, • Real-time collaboration, reuse requirements, etc.	https://www.jamasoftware.com/?utm_medium=media%20partner&utm_source=software%20testing%20help&utm_campaign=top%20rm%20tools

Tools used in design phase (V-model):

SL No	Tools	Description	Link
1	Star UML	A popular UML modeling tool	Star UML
2	Open text provision	An extensive business process architecture tool	Open Text Provision
3	Visual Paradigm	A design and management tool for business IT development	Visual Paradigm

Tools used in testing phase (V-model):

	Tools	Description	Link
Unit Testing	Junit	• Used for Java programming language. • This tool test data first and then inserted in the piece of code.	https://www.guru99.com/junit-tutorial.html
	Nunit	• Use for all .net languages. • It supports data-driven tests which can run in parallel.	https://nunit.org/
	JMockit	• Code coverage tool with line and path metrics. • Allows mocking API with recording and verification syntax.	http://jmockit.github.io/index.html
	EMMA	• Support coverage types like method, line, basic block.	http://emma.sourceforge.net/
Integration testing	Squish (Froglogic)	• Commercial cross-platform GUI and regression testing tool that can test applications based on a variety of GUI technologies	https://en.wikipedia.org/wiki/Squish_(Froglogic)
	Rational Integration tester	• It gives the scripting free environment for developing business process integration projects and tests for SOA messaging.	https://www.ibm.com/support/knowledgecenter/SSBLQQ_9.2.0/com.ibm.rational.rit.gs.doc/topics/c_ritov_test_methodology.html
System testing	Squish	• prevails as the single GUI testing solution for all your applications under test (AUT) •	https://www.froglogic.com/squish/

Acceptance testing	Usersnap	• It's an easy to use UAT solution that helps QA teams verify if a certain solution works for the user.	https://usersnap.com/blog/types-user-acceptance-tests-frameworks/

Tools used in maintenance phase (V-model):

Tools	Description	Link
KANBAN	• Visualize the flow of your work • Manage and improve the flow • Implement feedback loops	https://www.ntaskmanager.com/blog/best-kanban-tools/
SCRUM	• It manages the complex work • The scrum framework is based on continuous learning.	https://en.wikipedia.org/wiki/Scrum_(software_development)

Tools used in coding phase (V-model):

Sl No	Tool	Description	Link
1	Visure	• End-to-end req. • Traceability, • Req. management, • Req. Gathering, reusability	https://visuresolutions.com/
2	Reqtest	• Prioritizing Requirements • Highly customizable • instant reviewing tracking of changes	https://reqtest.com/features/requirement-management/?utm_source=softwaretestinghelp
3	**Mind maps**	• tool to capture ideas, requirements and help organize a conversation with many tangents	Mind mapping
4	Jama	• Align tests & requirements, Real-time collabration	https://www.jamasoftware.com/?utm_medium=media%20partner&utm_source=software%20testing%20help&utm_campaign=to

TOOLS IN AGILE MODEL:

Tools used in requirement phase (Agile model):

SI. No	Tools	Description	Link
1	**Source control tools**	• Git is widely supported, and many teams now use its hosting services to keep their code organized.	https://techbeacon.com/app-dev-testing/top-agile-tools-keep-software-engineers-productive
2	**ClickUp**	• Only project management tool where the goal at the forefront of feature-driven development was "to move quickly and easily" • ensures you can prevent bottlenecks.	https://clickup.com/blog/agile-tools/
3	**Jira**	• Jira is an agile based solution.	https://clickup.com/blog/agile-tools/
4	LeanKit	• Simple modeling columns for work flow and note cards for work items – move by drag and drop • Metrics Tools to analyze and extract information. Ability to monitor on micro or macro level.	http://agiletools.info/leankit/

Tools used in development phase (Agile model):

Tools	Description	References
Source control tools	To track, test progress and organizing the code structure of the development.	• Git- https://git-scm.com/ • Mercurial-https:// mercurial.selenic.com/
Continuous Integration	To run unit tests that ensure the software is performing correctly after	• Github - https://github.com/ • Travis CI- https://travis-

Tools	all the new code is added to the stack.	ci.org/
Team Management Tools	To track active collab and daily progress of the work done by individuals and also for keeping track of manual changes done	• Github- https://github.com/ • Active Collab - • https://activecollab.com/ • Agile Bench - • http://www1.agilebench.com/

Tool	Description	Link
Worksoft	• Enable Agile adoption by building automation closer to development sprint when documentation is top of mind • Quickly identify and document existing business processes and variations • Achieve end-to-end business process testing across enterprise applications • Efficiently keep pace with complex application landscapes and frequent application updates	https://www.worksoft.com/solutions/agile-devops-testing
PractiTest	• It can be integrated with tools like JIRA, Jenkins, Selenium, TestComplete etc. • It has a hierarchical tree structure to manage and find information. • The powerful and customized dashboard provides relevant and accurate information. • Easily imports existing data. • Complex database queries can be easily generated.	https://www.practitest.com/?utm_medium=listings&utm_source=sth&utm_campaign=agile+testing+tools
JIRA	• JIRA supports an agile methodology like Scrum, Kanban, etc. • It has a strong reporting feature which provides access to dozens of reports with real-time team performance. • Plans and forecasts roadmap and are able to make informed decisions. • It can be integrated with the developer tools for end to end	https://www.atlassian.com/software/jira/agile

	traceability	
JMeter	• JMeter is an open source tool. • Graphical analysis is possible for the performance measurement of the application under different types of load. • JMeter can be used for static and dynamic resources like Servlets, Java Objects, and FTP servers to measure their performance.	http://jmeter.apache.org/index.html

Tools used in maintenance phase (Agile model):

Tools	Description	Link
KANBAN	• Visualize the flow of your work • Manage and improve the flow • Implement feedback loops	https://www.ntaskmanager.com/blog/best-kanban-tools/
SCRUM	• It manages the complex work • The scrum framework is based on continuous learning.	https://en.wikipedia.org/wiki/Scrum_(software_development)

Things I liked doing the activity – Learning about different tools which helps in the different phases of SDLC.

Things I faced difficulty in – Finding the different sources for the tools