./

GENESIS - Learning Outcome & Mini-project Summary Report



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ver. Rel. No.** | **Release Date** | **Prepared. By** | **Reviewed By** | **To be Approved** | **Remarks/Revision Details** |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Details**

Contents

[Contents 3](#_Toc55470819)

[Miniproject -1 [Individual] 4](#_Toc55470820)

[Module/s 4](#_Toc55470821)

Intermediate C++ and Advanced C++

→ STL

→ Default and parameterized constructor

→ Operator Overloading

→ File Operations [4](#_Toc55470822)

[Objectives & Requirements 4](#_Toc55470823)

[Design 4](#_Toc55470824)

[Test Plan 4](#_Toc55470825)

[Implementation Summary 4](#_Toc55470826)

[Video Summary 4](#_Toc55470827)

[Git Link 4](#_Toc55470828)

[Git Dashboard 4](#_Toc55470829)

[Summary 4](#_Toc55470830)

[Individual Contribution & Highlights 5](#_Toc55470831)

[Summary 5](#_Toc55470832)

[Challenges faced and how were they overcome 5](#_Toc55470833)

[Future Scope (If applicable) 5](#_Toc55470834)

[Miniproject -2 [Team/Individual] 6](#_Toc55470835)

[Module/s 6](#_Toc55470836)

Linux and OS

[6](#_Toc55470837)

[Objectives & Requirements 6](#_Toc55470838)

[Design 6](#_Toc55470839)

[Test Plan 6](#_Toc55470840)

[Implementation Summary 6](#_Toc55470841)

[Git Link 6](#_Toc55470842)

[Git Dashboard 6](#_Toc55470843)

[Summary 6](#_Toc55470844)

[Individual Contribution & Highlights 6](#_Toc55470845)

[Summary 6](#_Toc55470846)

[Challenges faced and how were they overcome 6](#_Toc55470847)

# Miniproject -1 [Individual]

## Module/s

“Module linked to the miniproject- Intermediate/ Advanced C++

### Topic and Subtopics

→ STL(Standard Template Library)

→ Default and parameterized constructor

→ Operator Overloading

→ File Operations

STL was implemented using map container. When we read data from .DATA file, the data is stored in map container, which has key value pair. Default and parameterized constructors were used. Operator overloading was used on insertion and extraction operator because here we are passing user defined objects which are not defined, so we will overload insertion and extraction operator. Regarding file operations, reading file, writing file. Ostream, ofstream and istream is used. In ostream, contents are written to the file and in ofstream file is created and contents are written to it and in in-stream file is read.

## Objectives & Requirements

High Level Requirements

|  |  |
| --- | --- |
| ID | Description |
| HL\_01 | Open Account function to open account |
| HL\_02 | Deposit function to deposit money |
| HL\_03 | Withdraw function to withdraw money |
| HL\_04 | Balance Enquiry to check for balance |
| HL\_05 | Close Account function to close account |
| HL\_06 | Display details of all accounts |

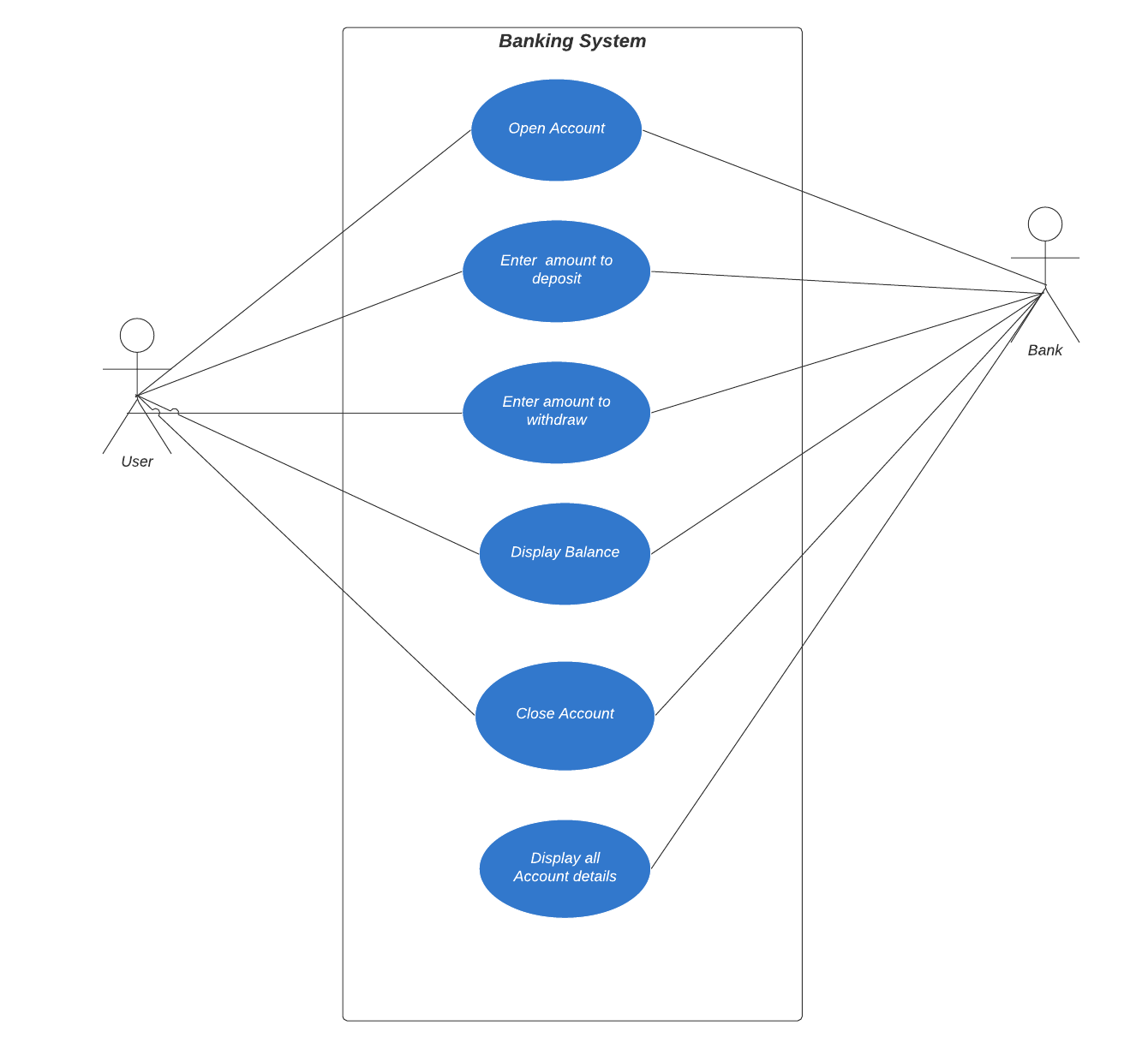
Low Level Requirements

|  |  |
| --- | --- |
| ID | Description |
| LL\_01\_HL\_01 | Enter valid first and last name to create account |
| LL\_02\_HL\_02 | Enter valid account number first and then enter appropriate amount to deposit |
| LL\_03\_HL\_03 | Enter valid account number and enter amount less than bank balance to withdraw |
| LL\_04\_HL\_04 | Enter valid account number for balance enquiry |
| LL\_05\_HL\_05 | Enter valid account number to close Account |
| LL\_06\_HL\_06 | Account details of accounts should be displayed |

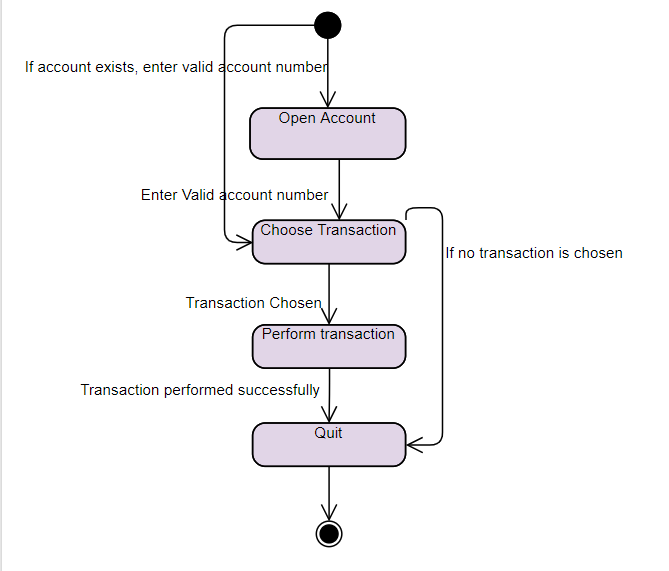
## Design

1.Behavioral Diagram:

1.a. Use Case Diagram

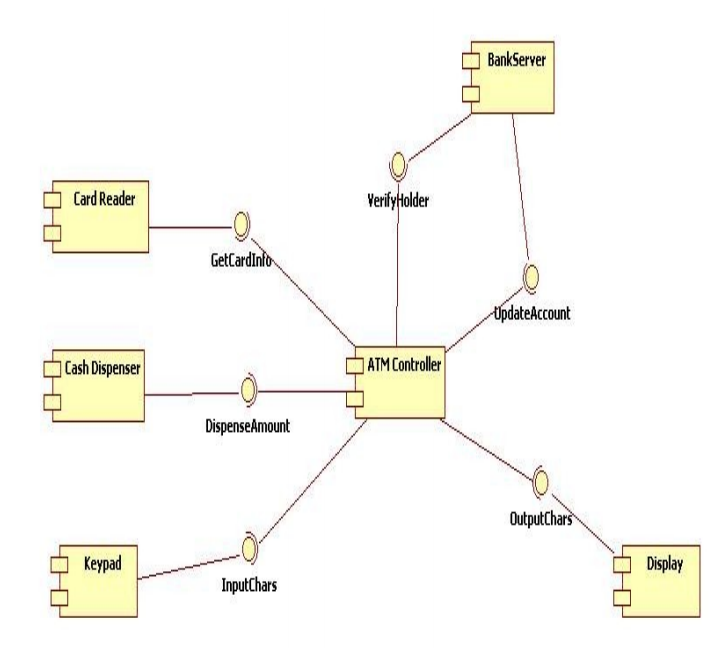


1.b. State Diagram



2. Structural Diagram

2. a Component Diagram



## Test Plan

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Description | Pre-Condition | Expected Input | Expected Output | Actual Output |
| HL\_01 | Open Account | None | First and Last Name | Account created successfully |  |
| HL\_02 | Deposit money | Account should exist | Enter valid account number | Amount deposited successfully |  |
| HL\_03 | Withdraw money | There should be amount in bank | Enter valid account number and amount to be withdrawn | Amount withdrawn successfully |  |
| HL\_04 | Balance Enquiry | Account should exist | Enter valid account number | Balance displayed |  |
| HL\_05 | Close Account | Account should exist | Enter valid account number | Account closed successfully |  |
| HL\_06 | Display details of Accounts | Accounts should exist | None | Display account details of all users |  |
| LL\_01\_HL\_01 | Enter valid first and last name to create account | None | First and last name | Account is created successfully and message is displayed. |  |
| LL\_02\_HL\_02 | Enter valid account number and then enter appropriate amount to deposit | Account should exist | Account number and amount to deposit should be entered | Amount is deposited successfully |  |
| LL\_03\_HL\_03 | Enter valid account number and enter amount less than balance to withdraw | Account should exist | Enter account number and amount to be withdrawn should be entered | Amount is withdrawn and message is displayed |  |
| LL\_04\_HL\_04 | Enter valid account number for balance enquiry | Account should be present | Enter valid account number | Balance is displayed |  |
| LL\_05\_HL\_05 | Enter valid account number to close account | Account should be present | Enter valid account number | Account successfully closed and message is displayed |  |
| LL\_06\_HL\_06 | Display all account details | Accounts should be present | None | Account details of all accounts displayed. |  |

## Implementation Summary

“Section focused toward’ s implementation aspects. Here it is only core summary while all the details are in the Git Repo

Note: The GitHub private repo should be documented (Readme.md files at each folder level)

Ensure code quality and clean code and description practices

Mandatory: To add the GitHub user - **stepin654321** as a contributor to the repo”

The project is implemented in C++. Here I have used concepts like constructors(default and parameterized), file operations, Standard Template Library. Some functions which are implemented are Deposit, withdraw, display particular user details, display all accounts details, Open account and close Account. When user enters the details, it gets stored in the file and when details are retrieved from file, it is stored in Map STL container which has key as account number and value field as account details. Some file operations are used are: istream,ostream and ofstream. Here ofstream is used to open the existing file and istream is used to read contents from the file and ostream is used to display the output to the user. All these operations are overloaded because we pass user defined objects to these functions and it is not originally designed to accept user defined objects, so we will overload these operators to make sure it accepts user defined objects. There is empty class to check if amount entered to be withdrawn is greater than balance. If yes, it throws error.

### Video Summary

“Please upload a short video on the repo for the walkthrough of the project (Team/Individual) less than 7min and less than 30MB File Size. Start is the Standard opening slide with title of miniproject + Team members followed by the walkthrough ”

### Git Link

“Link to the repo”

### Git Dashboard

“Screenshot of the GitHub Repo page with all the badges and summary”

### Summary

Banking project is implemented using C++. This application supports opening account, deposit, withdraw, display account details, close account. Display all account details is specifically for bank officials to see details of all bank accounts. Details entered by user gets stored in the file and then it is retrieved when user requests for it.

STL Map container is used to store the data retrieved from file and then it is displayed to the user. Account number is generated randomly for every user and if user wants to do any transaction, they have to enter valid account number. Overall, this application supports basic banking operations.

#### Git inspector summary

“In linux install gitinspector and Run the command –

gitinspector -H -l -m -T -w -r --grading --format=html > gitinsp.html

and upload the same to your repo and paste the snapshot in the report”

#### Build

The application is developed using Microsoft Vs Code. The code is compiled using g++ compiler. I have used MingW.MinGW is a compiler system based on the GNU GCC and Binutils projects that compiles and links code to be run on Win32 (Windows) systems. It provides C, C++ and Fortran compilers plus other related tools. 'MinGW' refers to the "Minimalist GNU for Windows" project.

#### Code quality and Issues or Bug Tracking

“Brief on code quality, errors and warnings flagged (issues created) and fixed ”

#### Unit Testing

Google Test was used to do unit Testing. Each function was thoroughly tested. Credit function was tested to check if amount is deposited, whether it is actually deposited. Debit function was tested to check if amount was debited successfully. Testing was also done to check if amount entered to withdraw is greater than bank balance. If it was greater, then test fails or else test passes. Test was also done for display function which displays the details of the particular user.

## Individual Contribution & Highlights

### Challenges faced and how were they overcome

→ Challenge was to figure out proper file operations for the applications. Figured out by searching on internet

→ Choosing the appropriate STL for the application. Checked all STL containers and felt Map is better container because it is faster and since account number is unique for each user, so went ahead with map.

### 

### 

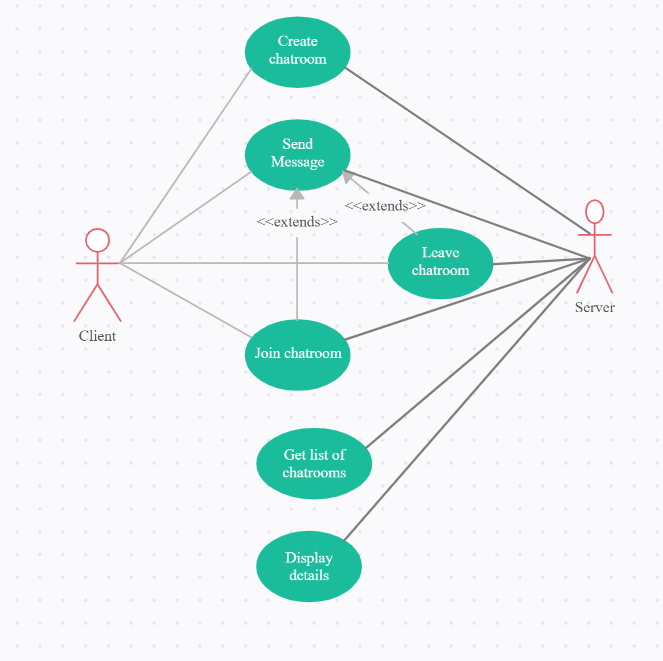
# Miniproject -2 [Team/Individual]

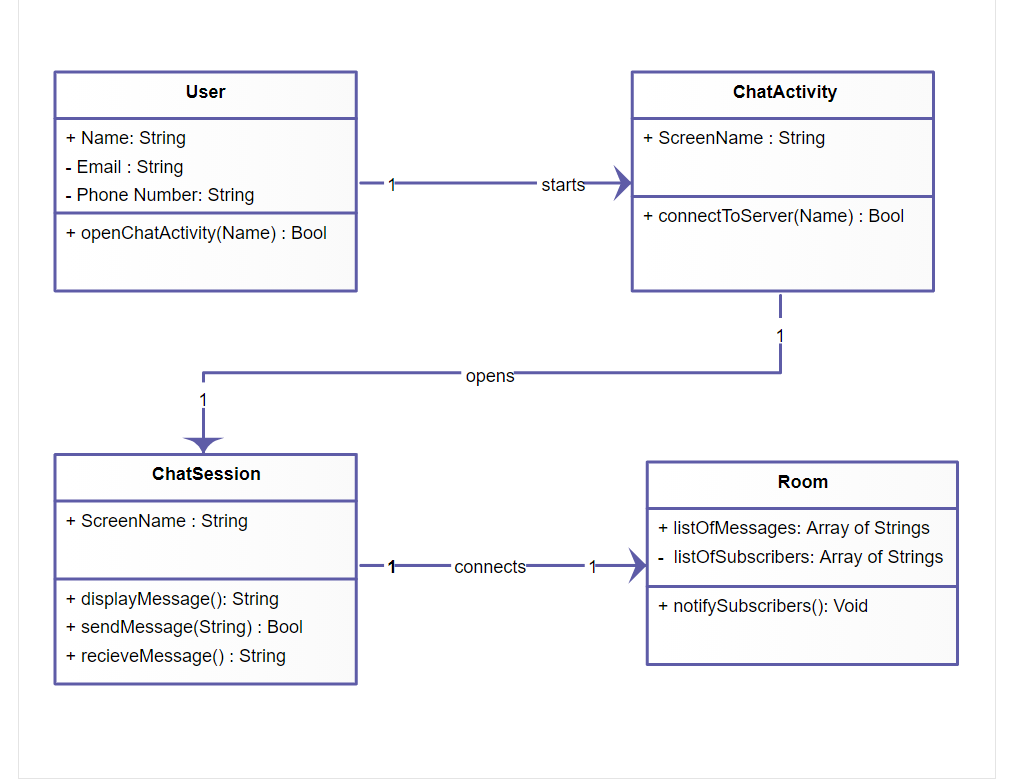
## Linux and OS

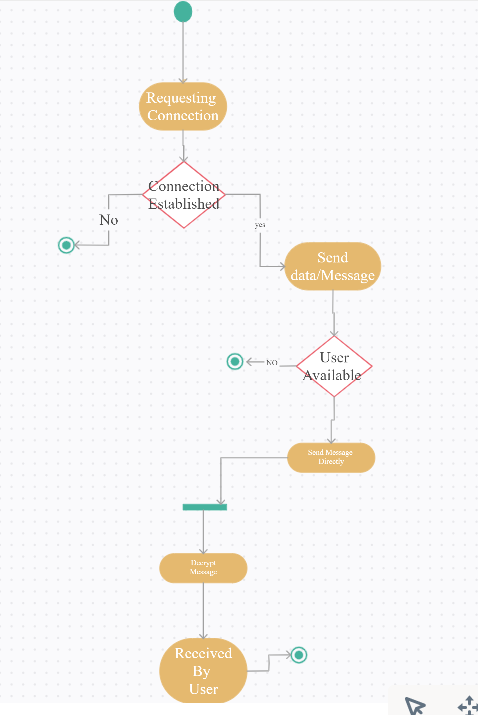
### Topic and Subtopics

## Objectives & Requirements

## Design







## Test Plan

## Implementation Summary

### Git Link

### Git Dashboard

### Summary

#### Git inspector summary

#### Build

#### Code quality

#### Unit Testing

#### Issues

## Individual Contribution & Highlights

### Summary

### Challenges faced and how were they overcome