Assignment No. 2

INTRODUCTION TO SOFTWARE ENGINEERING (SE1001)

Name: Pir Ahmed Shah
Roll No: 24P-3000
Department: BSE-3B

1. Project Overview

Project Title

Loan Calculator Refactoring, Testing, and Documentation

Overview

The goal of this project was to optimize and refactor a legacy C++ Simple Interest Loan Calculator, addressing common issues found in older codebases: lack of input validation, risks of integer overflow in financial calculations, and reliance on hardcoded values.

The solution involved a complete refactoring into a modular, object-oriented structure using the LoanCalculator class, implementing error checking, fixing numerical precision using long double, etc., Finally, the project established professional software practices by implementing a comprehensive GoogleTest suite and integrating Doxygen for API documentation.

Github Repository:
`https://github.com/AhmedvShah/lagacy-calc-2009`

Tools and Technologies Used

- Language: C++14
- Build System: make
- Compiler: g++
- Testing: GoogleTest (gtest)
- Documentation: Doxygen
- Version Control: Git & GitHub

2. Code Snippets of Bug Fixes and Refactorings

Refactoring 1: Input Validation and Modularization

Issue: The original main.cpp lacked checks for non-positive or irrational inputs, risking runtime errors and meaningless results.

Fix/Refactoring: The validation logic was abstracted into the LoanCalculator::validate_inputs method in loan.cpp, ensuring all critical inputs are positive before calculation.

Snippet (src/loan.cpp - validate_inputs method):

```
bool LoanCalculator::validate_inputs(LoanValue amount, LoanValue rate, LoanValue years) {
  if (amount <= 0.0L) {
      cerr << "Error: Loan amount must be greater than zero." << endl;
      return false;
  }
  if (rate <= 0.0L) {
      cerr << "Error: Interest rate must be greater than zero." << endl;
      return false;
  }
  if (years <= 0.0L) {
      cerr << "Error: Number of years must be greater than zero." << endl;
      return false;
  }
  return true;
}
```

Refactoring 2: Overflow and Precision Fix

Issue: The use of standard double exposed the calculator to potential precision loss or overflow, especially with large loan amounts or tenures, as required by the assignment.

Fix/Refactoring: The project switched to the long double type via a type alias (LoanValue) for all financial calculations and variables to maximize precision and mitigate overflow risk.

Snippet (src/loan.hpp - Type Alias):

```
using LoanValue = long double;
```

Snippet (src/loan.cpp - Calculation Logic):

```
LoanResults LoanCalculator::calculate(LoanValue amount, LoanValue rate, LoanValue years) {

  LoanResults results;



  LoanValue total_interest_calculated = amount * (rate / 100.0L) * years;



  results.total_amount_paid = amount + total_interest_calculated;

  results.total_interest = total_interest_calculated;

  LoanValue total_months = years * 12.0L;



  if (total_months > 0.0L) {

     results.monthly_amount = results.total_amount_paid / total_months;

  } else {

     results.monthly_amount = 0.0L;

  }



  return results;

}
```

Refactoring 3: Removing Hardcoded Values

Issue: The original logic had no mechanism for configuration, requiring changes to source code for adjustments like a default rate.

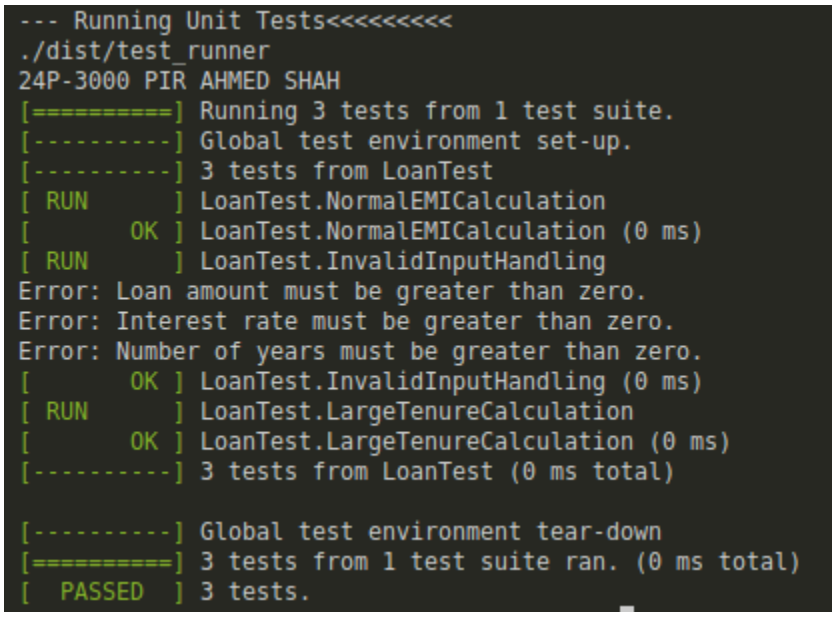
Fix/Refactoring: Implemented a file parsing function in main.cpp to read a default interest rate from a config.txt file, ensuring dynamic configuration.

Snippet (src/main.cpp - read_default_interest_rate function):

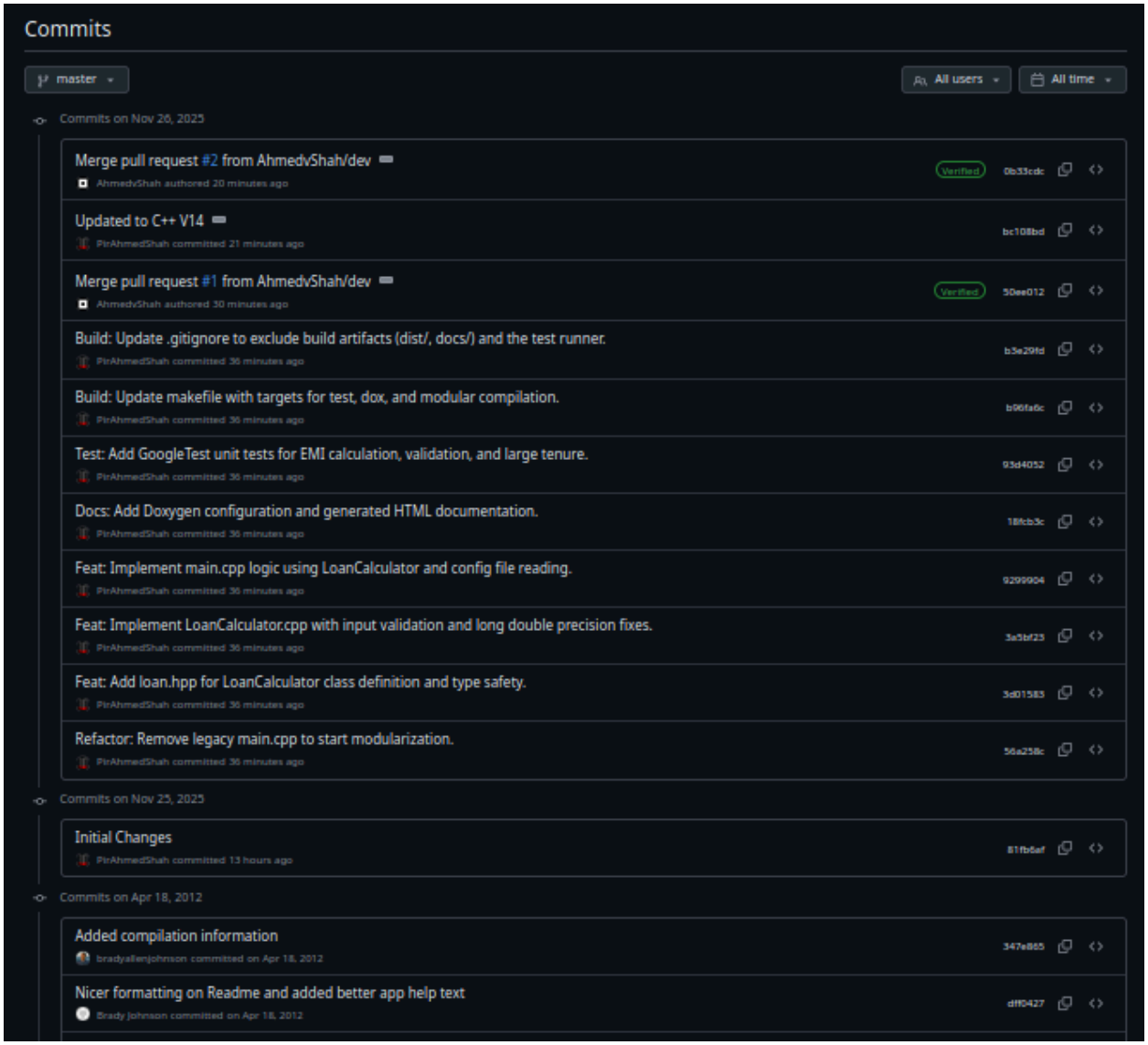
```
/**
* @brief Attempts to read a default interest rate from a configuration file.
* @param filename The name of the configuration file (e.g., "config.txt").
* @return The default interest rate as LoanValue, or 0.0L if reading fails.
*/
LoanValue read_default_interest_rate(const string &filename)
{
   ifstream cfg_file(filename);
   LoanValue default_rate = 0.0L;

   if (cfg_file.is_open())
   {
       string line;
       if (getline(cfg_file, line))
       {
           stringstream ss(line);
```

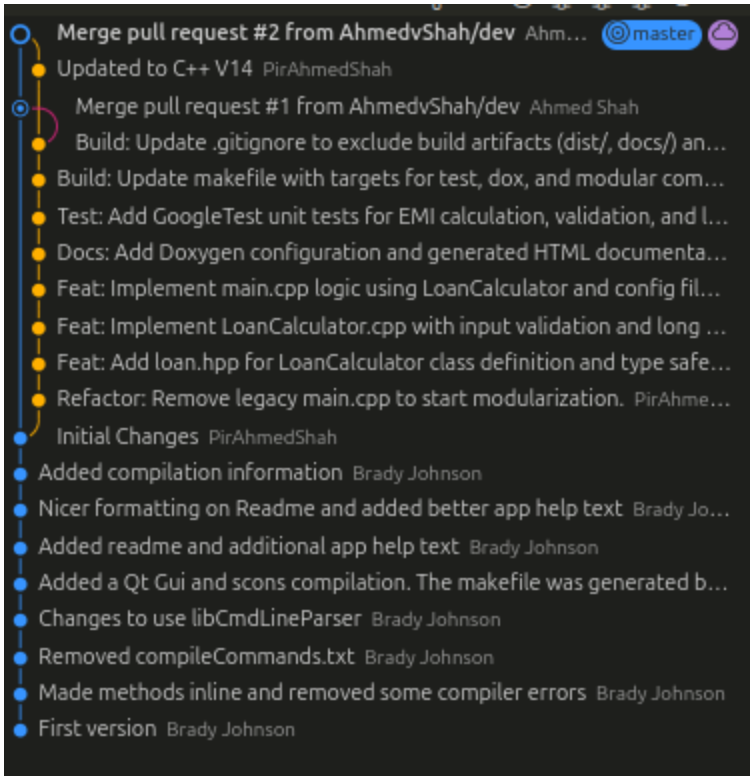
```
            string token;
            if (ss >> default_rate)
            {
            }
        }
        cfg_file.close();
        return default_rate;
    }
    return 0.0L;
}
```

3. Test Output Screenshot



4. Test Output Screenshot




Merge pull request #2 from AhmedvShah/dev Ahm…
master
Updated to C++ V14 PirAhmedShah
Merge pull request #1 from AhmedvShah/dev Ahmed Shah
Build: Update .gitignore to exclude build artifacts (dist/, docs/) an…
Build: Update makefile with targets for test, dox, and modular com…
Test: Add GoogleTest unit tests for EMI calculation, validation, and l…
Docs: Add Doxygen configuration and generated HTML documenta…
Feat: Implement main.cpp logic using LoanCalculator and config fil…
Feat: Implement LoanCalculator.cpp with input validation and long …
Feat: Add loan.hpp for LoanCalculator class definition and type safe…
Refactor: Remove legacy main.cpp to start modularization. PirAhme…
Initial Changes PirAhmedShah
Added compilation information Brady Johnson
Nicer formatting on Readme and added better app help text Brady Jo…
Added readme and additional app help text Brady Johnson
Added a Qt Gui and scons compilation. The makefile was generated b…
Changes to use libCmdLineParser Brady Johnson
Removed compileCommands.txt Brady Johnson
Made methods inline and removed some compiler errors Brady Johnson
First version Brady Johnson