|  |  |
| --- | --- |
| A picture of a winding road and trees  Presentation  CST2550 Coursework 1 | InTRODUCTION  Brief overview  This presentation is intended to demonstrate how the system software functions by examining the design of UML Diagrams, implementation, testing and so on.  Osee Mbiya  Software Engin, Manage & Development |

2. Design

Use Case Diagram:

A screen shot of a diagram

Description automatically generated

Use case diagram provided use case diagram outlines the primary interaction and functionality of the systems.

**-Person:** view info, indicating the steps and decisions involved in the process.

**-Librarian:** add members, issue a Book, and return and display the list of borrowed books by a specific member. And then calculate the fine.

**-Member:** Borrow a book and return a book.

Activity Diagram:

A screenshot of a computer

Description automatically generated

Activity diagram represents the flow of activities or actions within a system:

**-Person:** view info, indicating the steps and decisions involved in the process.

**-Librarian:** add members, issue a Book, and return and display the list of borrowed books by a specific member. And then calculate the fine.

**-Member:** Borrow a book and return a book.

Class Diagram:

**-Person:** This class diagram represents a general human with basic information.

**- Librarian:** Acts as a librarian, managing members, handling book transactions, and calculating fines.

**- Member:** This object represents a library member and can access member information and handle borrowed books.

**- Book:** Represent a book by providing information on the book, the due date, and borrower information.

3. Implementation:

1. How you translate the design into working software:

I took a specific strategy to this coursework, relying on my understanding of methodologies like waterfall to help me finish it. I pulled out my UML diagram for design. I establish the Development Environment and assemble the necessary tools in my programming environment.

I set up version control with a system like Git to handle changes correctly.

Assemble Skeletons in Class:

In your favourite programming language, convert class diagrams into class skeletons or templates.

Based on the data in the UML diagram, define features and procedures.

Put Classrooms into Practice:

Write the actual code for each method and property to start developing the classes.

Observe the UML's design patterns and object-oriented programming (OOP) principles.

1. Explain what the Make file was used for.
2. Explain how and why version control was used:

I used git as my version control system to track changes to my code over time. It has capabilities like branching, merging, and history tracking, allowing numerous developers to work on the same project at the same time while preserving a disciplined and organised development process.

1. Screenshot of the Bitbucket/GitHub repository which clearly shows all commits and commits messages:

A screenshot of a computer

Description automatically generatedSo, first things first, I have created my repository for my coursework.

A screenshot of a computer program

Description automatically generatedAfter that, I cloned my repo to my desktop, so I could commit and push my files and work on it.

A screenshot of a computer program

Description automatically generatedThen I started to commit and push my files to my repository. The screenshot below shows how I used git add . all the files to commit and then push it to the repository. I used the command git status to check my files staged and see the files that are ready to be committed.

A screenshot of a computer program

Description automatically generatedDeleting unnecessary files to keep my repository organized.

Testing approach:

For my test I used

Software demonstration:

Date Class: Shows the day, month, and year of a date.

Person Class: An abstract class containing shared properties such as email, address, and name. To display personal details, it also has a virtual method called display Details().

Book Class: Contains information on a book, including its name, book ID, due date, borrowed status, and borrower ID. It offers procedures for both returning and borrowing books.

Member Class: Represents a member of the library and inherits from Person. It has a vector to store checked books and a member ID. Borrowing, returning, and putting on the show are some of the methods.

Librarian Class: Descended from an individual and symbolises a librarian. It contains information about staff identification, pay, and how to add members, issue and return books, show borrowed books, and compute fines.

Main Function: Provides a menu for engaging with the library system after receiving input regarding librarian details. Adding a member, issuing, returning, and displaying borrowed books, computing penalties, and quitting the program are among the menu options.

The program uses classes, inheritance, and polymorphism to illustrate object-oriented concepts. It mimics routine library tasks like creating new accounts, checking out and returning books, and computing late fees.

In summary,

The project consists of developing a software program to handle member and book information for a small library by utilising a CSV file that is supplied. To permit flexibility in the use of CSV files, adherence to a UML diagram provided by the library is necessary. The submission process involves using a Bitbucket or GitHub repository that contains C++ source code, Catch2 tests, and a Make file; third-party libraries or code are prohibited.

By enabling member addition, book issuance and return, loaned book display, and late book charge computation, the system satisfies the needs of libraries. Notably, the solution places a high priority on user-friendliness and input validation.