[Document title]

[Document subtitle]

[School]

[Course title]

Table of Contents

[Project Requirements 2](#_Toc149819097)

[Introduction 2](#_Toc149819098)

[Objectives 2](#_Toc149819099)

[Scope 2](#_Toc149819100)

[System Design 3](#_Toc149819101)

[Flowcharts 3](#_Toc149819102)

[Algorithms and Pseudocode 3](#_Toc149819103)

[Justification of Design Tools 4](#_Toc149819104)

[Development of the System 4](#_Toc149819105)

[Class and Structure Definitions 4](#_Toc149819106)

[Functions and Methods 5](#_Toc149819107)

[Data Structures 6](#_Toc149819108)

[File Handling 7](#_Toc149819109)

[Testing and Debugging 7](#_Toc149819110)

[Testing Methodology 7](#_Toc149819111)

[Issue Tracking 7](#_Toc149819112)

[Test Cases and Results 8](#_Toc149819113)

[Evaluation 9](#_Toc149819114)

[User Feedback 9](#_Toc149819115)

[Usability Problems Solved: 9](#_Toc149819116)

[System Strengths 10](#_Toc149819117)

[Areas for Improvement 10](#_Toc149819118)

[User Interfaces 11](#_Toc149819119)

[Librarian Interfaces 11](#_Toc149819120)

[Member Interfaces 11](#_Toc149819121)

# Project Requirements

## Introduction

* The library management system project began in response to a request from the Ha Tsolo Ward Councilor to solve the existing issues of manual record-keeping within the Ha Tsolo Community Centre library. The existing manual system has various flaws, including the maintenance of book records, member records, book issuing, and the calculation of overdue fees. A computerized library administration system is thought necessary to modernize and improve the library's services.

## Objectives

* The Ward Councilor highlighted the following as the project's major goals:
* To automate the management of book records, containing information such as book title, author, category, publication date, and availability.
* To automate member record administration by capturing information such as member ID, name, gender, address, and membership status.
* To allow the library to offer books to members while automatically generating return dates (books can be loaned for up to 7 days, including weekends).
* To give members and non-members access to the library's book catalog, where they can search for books by category, author, release date, and other pertinent criteria.
* Members may borrow no more than three books under the following conditions:
* Condition 1: No member may borrow the same book (with the same title, author, and publication date) more than once within the same borrowing time.
* Condition 2: Members who owe the library overdue fees are not permitted to borrow further books.
* To allow the librarian to execute necessary tasks such as adding new members or books, amending current member or book records, issuing books, listing members who owe fees, and listing issued books per week, categorized by category, title, and author.
* To design a user-friendly system with clear, understandable user interfaces and a smooth navigation experience.

## Scope

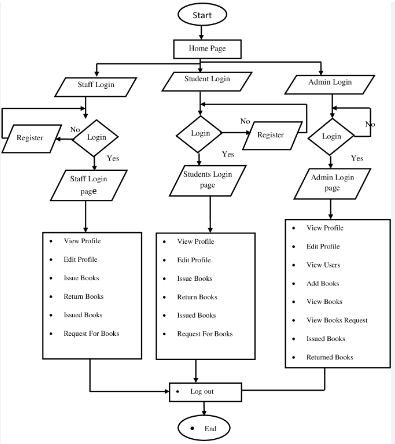
The library management system's capabilities include the following:

* Book record creation and administration, including title, author, category, date of publication, and availability status.
* Member record creation and management, including member ID, name, gender, and address.
* Book distribution to members, including automatic development of return dates (up to 7 days, including weekends).
* Implementation of book borrowing requirements, guaranteeing that no books are duplicated by title, author, or publication date.
* Overdue fees for members with outstanding books are calculated.
* Members and non-members alike can search and browse the library's book catalog using various criteria.
* Librarian operations are used to add, amend, and delete member and book records.
* Members who owe fees are listed in descending order of the amount owed.
* Weekly book releases, organized by genre, title, and author.
* Future additions could include the introduction of online services for member registration and book reservations, user authentication for increased security, and expanded reporting and analytics capabilities for the library management staff.

# System Design

## Flowcharts

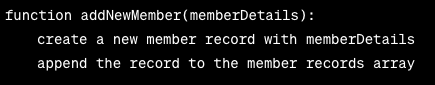
Below are flowcharts that illustrate the workflow of the library management system. These visual representations provide an overview of how different parts of the system interact:



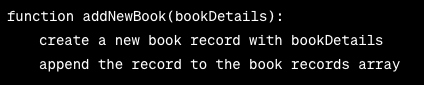
## Algorithms and Pseudocode

Below are pseudocode examples for critical functions within the library management system:

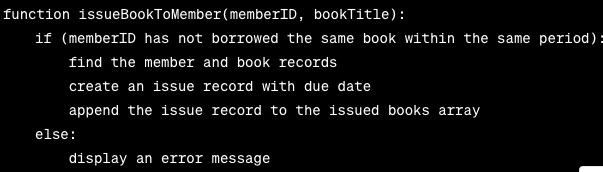
* Pseudocode: Adding a New Member



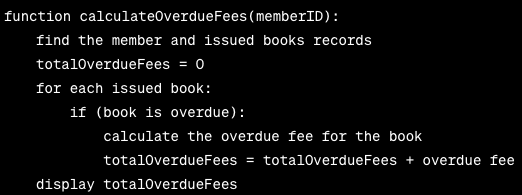
* Pseudocode: Adding a New Book



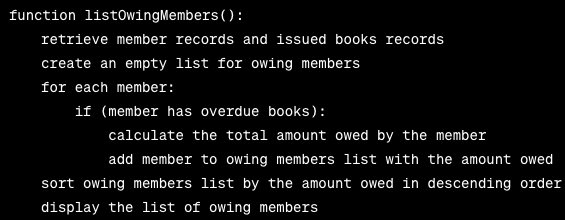
* Pseudocode: Issuing a Book to a Member



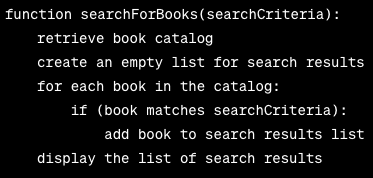
* Pseudocode: Calculating Overdue Fees



* Pseudocode: Listing Owing Members



* Pseudocode: Searching for Books



## Justification of Design Tools

The programming language C++ was chosen for this project due to its suitability for system development, specifically its strong features and ability to manage large data structures. C++ provides:

* Object-Oriented Programming (OOP): C++ provides OOP, allowing the creation of classes and structures, which is perfect for modeling library management system entities such as books and members.
* Efficient Memory Management: C++ provides precise memory management control, which is essential for dealing with dynamic data structures and minimizing memory leaks.
* Standard Template Library (STL): The STL contains pre-built data structures such as vectors, which are critical for effectively managing arrays of records.
* File Handling: C++ has comprehensive file handling capabilities, allowing for the structured storage and retrieval of member and book information.
* Portability: Because C++ is a widely supported language, the system is compatible with a wide range of platforms and operating systems.
* C++ is well-known for its performance, which is critical in maintaining the system's responsiveness, especially when dealing with potentially enormous numbers of records and transactions.

The usage of classes, structures, functions, arrays, and files is justified by their inherent strengths in data and operation structuring, code modularity, and efficient and structured data administration. These design tools were used to assure the system's maintainability, extensibility, and ability to meet the project's objectives.

# Development of the System

## Class and Structure Definitions

In the library management system, we have defined two essential classes: Book and Member. These classes help structure and manage book and member records effectively.

**Class: Book**

Attributes:

title (string): The title of the book.

author (string): The author of the book.

category (string): The category to which the book belongs (e.g., fiction, science, language).

dateOfPublication (string or date): The date of publication of the book.

availability (boolean): Indicates whether the book is available for borrowing.

Methods:

displayDetails(): A method to display the book's details.

isAvailable(): A method to check if the book is available for borrowing.

**Class: Member**

Attributes:

* memberID (string): The unique identifier for the member.
* name (string): The name of the member.
* gender (string): The gender of the member.
* address (string): The address of the member.
* membershipStatus (boolean): Indicates whether the member is an active member.

Methods:

* displayDetails(): A method to display the member's details.
* isMemberActive(): A method to check the membership status.

## Functions and Methods

The following functions and methods are implemented for the library management system:

**Adding a New Member:**

* Function: addNewMember(memberDetails)
* Purpose: Adds a new member to the system.
* Inputs: memberDetails (attributes of the member).
* Outputs: None.

**Adding a New Book:**

* Function: addNewBook(bookDetails)
* Purpose: Adds a new book to the system.
* Inputs: bookDetails (attributes of the book).
* Outputs: None.

**Issuing a Book to a Member:**

* Function: issueBookToMember(memberID, bookTitle)
* Purpose: Issues a book to a member, considering borrowing conditions.
* Inputs: memberID (member's unique identifier) and bookTitle (title of the book).
* Outputs: None.

**Modifying Member or Book Records:**

* Functions: modifyMemberRecord(memberID, newMemberDetails) and modifyBookRecord(bookTitle, newBookDetails)
* Purpose: Modifies the details of a member or book.
* Inputs: memberID or bookTitle for identification and newMemberDetails or newBookDetails for updated information.
* Outputs: None.

**Deleting Member or Book Records:**

* Functions: deleteMemberRecord(memberID) and deleteBookRecord(bookTitle)
* Purpose: Deletes a member or book record from the system.
* Inputs: memberID or bookTitle for identification.
* Outputs: None.

**Listing Owing Members:**

* Function: listOwingMembers()
* Purpose: Lists members who owe fees in descending order of the amount owed.
* Inputs: None.
* Outputs: List of owing members.

**Listing Issued Books per Week:**

* Function: listIssuedBooksPerWeek()
* Purpose: Lists issued books per week, categorized by category, title, and author.
* Inputs: None.
* Outputs: List of issued books per week.

**Searching for Books:**

* Function: searchForBooks(searchCriteria)
* Purpose: Searches the book catalog based on specified criteria.
* Inputs: searchCriteria (search parameters).
* Outputs: List of books matching the search criteria.

## Data Structures

Arrays and other data structures are used to handle records in the system. Arrays are used to organize and retrieve data by storing book records, member records, and issued books records. As an example:

* Book records are stored as an array of book objects (Book[]).
* Member records are stored in an array of member objects (Member[]).
* An array of issued book objects keeps records of books that members are currently borrowing.

## File Handling

Files are used to store member records indefinitely. Member information is saved in a file, allowing data to be retained even when the system is restarted. The file structure ensures that member records can be loaded and changed as needed into the system.

Methods for Handling Files:

* During system startup, reading member records from a file.
* When adding or changing member records, write them to the file.
* When members overdue payments or their membership status changes, the file is updated.

# Testing and Debugging

## Testing Methodology

To assure the system's correctness and robustness, we use both unit testing and integration testing for the library management system.

* Unit testing: was performed on each individual function and method, such as adding a new member, issuing a book, and calculating overdue costs. Unit tests were created to ensure that each component of the system functions as expected.
* Integration Testing: Following successful unit testing, we conducted integration testing to explore how different system components interact with one another. When we issued a book to a member, for example, we double-checked that the member and book data were correctly linked.
* A combination of automated and manual testing methods was used. Automated testing was utilized for recurring and complex scenarios, while manual tests were performed to evaluate the system's usability and user-friendliness.

## Issue Tracking

Several concerns arose during the development process, including logic flaws, input validation challenges, and unexpected behavior. The following procedure was used to track and address these issues:

* Unit testing, integration testing, and user testing were used by developers to identify issues.
* Issue Tracking: Each issue was tracked in a centralized issue tracking system. The documentation contained a description of the problem, its location in the source code, and its impact on the system.
* Prioritization of Issues: To prioritize debugging efforts, issues were classified by severity (e.g., critical, major, minor). The most critical issues that had a major impact on system functionality were fixed first.
* Debugging and Code Changes: Developers reviewed the code to determine the source of problems. Print statements and code inspection were utilized as debugging tools and methodologies. To address the issues, code changes were done.
* Testing Following Fixes: Following code changes to address the issues, the impacted components were retested to ensure that the problems were handled and that no new issues arose.
* Regression testing was undertaken to ensure that issue remedies did not generate new problems elsewhere in the system.
* User Feedback: User feedback was critical in identifying and resolving issues. To solve usability and user interface difficulties, librarians and members provided essential feedback.
* Documentation Updates: Any updates or remedies to issues were documented to keep a complete record of the development process.

## Test Cases and Results

Here are some sample test cases for major functions and processes, along with their inputs, expected results, and actual results:

**Case 1: Introducing a New Member**

* Input: Member information (name, gender, and address).
* The expected outcome is that a new member record is created and saved in the system.
* Actual Result: A new member record is created and saved successfully.

**Case 2: Giving a Book to a Member**

* Enter the following information: member ID, book title.
* Expected Result: A book with a due date is issued to the member.
* Actual Result: The member receives the book, and the due date is accurately created.

**Case 3: Calculating Past-Due Fees**

* Enter: Member ID
* Overdue costs for the member are calculated as expected.
* Actual Result: Overdue costs for the member's overdue books are calculated correctly.

**Test Case 4: Book Searching**

* Input: Search criteria (for example, category or author).
* The expected result is a list of books that fit the search criteria.
* Actual Result: The system presents books that fit the search criteria correctly.

# Evaluation

## User Feedback

User feedback was critical during testing in identifying usability issues and enhancing the system. Feedback was gathered from both librarians and library users. Among the comments received were:

**Problems with Usability:**

* Because of the quantity of features and settings, some users found the user interface to be slightly overwhelming.
* There were some misunderstandings about the restrictions on borrowing books, particularly about the rules for borrowing the same book or the limit on the number of books a member may borrow.

**Responsiveness and performance:**

* When looking for books or processing membership data, some users noticed minor delays, which impaired the overall user experience.

**Accessibility:**

* A few people suggested that improved accessibility features, such as larger fonts for those with visual impairments, be implemented.

## Usability Problems Solved:

The following efforts were done to resolve these usability issues:

* User Interface streamlined: We streamlined the user interface by grouping related operations and making navigating more intuitive. To help users comprehend the system, we have incorporated tooltips and explanatory information.
* User Training: We held training workshops for librarians to ensure they completely understood the borrowing conditions, which assisted them in explaining these restrictions to members.
* Performance Optimization: We optimized the code to increase system performance and responsiveness, making searches and record processing more efficient.
* Enhancements to Accessibility: We incorporated accessibility features such as text size modifications and keyboard navigation shortcuts to improve the user experience for people with special needs.

## System Strengths

Among the significant strengths and advantages of the library management system are:

* Manual Task Automation: The system successfully automates tasks related to book and member record administration, book issuance, and fee computation, decreasing library staff workload and minimizing errors.
* User-Friendly Interfaces: The user interfaces are designed to be user-friendly, with clear and understandable screens, allowing librarians and members to engage with the system easily.
* Efficient Data Storage: Using data structures and file handling ensures efficient data storage and retrieval, improving system stability and data integrity.
* Conditions Enforcement: The system effectively enforces borrowing conditions, preventing issues such as duplicate book borrowing and guaranteeing compliance with overdue charge rules.
* Customizable Searches: The system allows members to search for books based on a variety of parameters, resulting in a more personalized and efficient book discovery experience.

## Areas for Improvement

While the library management system has many merits, there are certain areas where it can be improved or expanded in the future:

* Consider implementing online capabilities such as member registration and book reservations to provide library users with greater convenience and accessibility.
* User Authentication: Use user authentication for librarian actions to improve system security and accountability.
* Reporting and Analytics: Improve reporting and analytics capabilities to provide insights into book circulation trends and member involvement to the library management staff.
* Mobile Compatibility: Make the system mobile-friendly by allowing members to access the catalog and perform actions on their smartphones or tablets.
* Consider localization options to allow users who choose to use the system in languages other than English, thereby catering to a diverse community.
* Integration with Library Catalogs: Consider integrating the system with external library catalogs and databases to grow the book collection and improve the library's resources.
* User Documentation: Create user manuals or documentation to assist librarians and members in properly using the system.

These areas for development represent potential to improve the system's functionality, user experience, and overall service quality to the community. To fulfill changing demands and expectations, future development might build on the current system's basis.

# User Interfaces

## Librarian Interfaces

## Member Interfaces