Please fill in your name and student ID in the table below.			
Student Name	*AMOGH DATH KALASAPURA ARUNKUMAR*	*NITISH CHOWDARY YARLAGADDA*	*JASPREET KAUR*
Student Number	24168333	24148241	24170277
Course and Year	2025-2026		
Module Code	CMP5332		
Module Title	Object Oriented Programming		
Module Leader	Dr. Quanbin Sun		
Assessment item:	Interactive Library System		

- You **must use the provided skeleton code and Javadoc** to implement the required features (see checklist below). You may add new fields, methods, or classes, but you must not remove any existing methods or alter the class or project structure.
- Your team may collaborate while drafting the code. However, for submission, **each feature must be implemented individually by one person and reviewed by another team member**.
- You must complete the **first six features in the 40%** grade band before progressing to higher grades. For each subsequent band, you must complete **at least two features** from the previous band.

How will you get marked?

- No viva → no mark.
- All features working + demonstrated in the viva → eligible for minimum mark for the band.
- Good OO design in the code → eligible for the maximum mark.
- Strong explanation of OO design choices in the viva → awarded the maximum mark.
- Missing features → reduced marks and possible grade band drop.

Note: Each student receives an individual mark based on their viva performance and the features they implemented/reviewed.

Feature and Responsibility Checklist

Prerequisites

- ➢ 1. Download the skeleton code and import to Eclipse.
- ➢ 2. Try the commands marked with "*" and study the code.
- ➢ 3. Rename the package "cmp5332" to your group number (e.g. a2) and make a start!

Hint: Do read and refer to the **Javadoc** for your implementation.

Achieving a mark of 40% to maximum of 49%		
	Implemented by	**Reviewed by**

➢ 4.1. Add new Patrons (members) to the system. System should store at least the following information for each member: ID, Name, Phone Number and List of Books Borrowed.	*JASPREET KAUR*	*AMOGH DATH*
➢ 4.2. Show a book stored in the system - must implement getDetailsLong().	*JASPREET KAUR*	*AMOGH DATH*
➢ 4.3. Show a patron's details in the system – should also display the borrowed books if implemented the borrow feature.	*NITISH CHOWDARY*	*AMOGH DATH*
➢ 4.4. List all Patrons stored in the system.	*JASPREET KAUR*	*AMOGH DATH*
➢ 4.5. Borrow: Issue Books to Patrons. - When a book is issued to a Patron a Loan object must be created holding a reference to the Book being issued, to the Patron borrowing the Book and the Due Date of the Loan. This object should be added to the Book, and the book should be added to the Patron's list of borrowed Books.	*AMOGH DATH*	*NITISH CHOWDARY*
➢ 4.6. Return issued books. - When a member returns a book the status of the returned book should be updated to reflect its availability. Also, book must be removed from the Patron's list of borrowed Books.	*AMOGH DATH*	*NITISH CHOWDARY*
➢ 4.7. Renew: Renew an existing Loan. - If loan exists, the due date should be renewed.	*NITISH CHOWDARY*	*AMOGH DATH*
➢ 4.8. Save the status of the system to the backend storage (i.e. text file storage). - When the system is closed (via "exit" command), the library data should be stored in three different files (books.txt, patrons.txt and loans.txt). Save/Load books has been implemented as an example; When the system starts it should load the status of the library from the text files.	*AMOGH DATH*	*NITISH CHOWDARY*
Achieving a mark of 50% to maximum of 59%		
	Implemented by	**Reviewed by**
➢ 5.1. Add a publisher property to the Book object and make the appropriate changes to the program to ensure that this information can be captured when a new Book is created. Also ensure that this information will be stored to and correctly loaded from the file storage.	*AMOGH DATH*	*NITISH CHOWDARY*
➢ 5.2. Add an email property to the Patron object and make the appropriate changes to the program to ensure that this information can be captured when a new Patron is created. Also ensure that this information will be stored to and correctly loaded from the file storage.	*AMOGH DATH*	*NITISH CHOWDARY*

➢ 5.3. Implement Unit Tests to the above **changes made to the Book and Patron classes**.	*JASPREET KAUR*	*AMOGH DATH*
Achieving a mark of 60% to maximum of 69%		
Note. For all GUI implementations, they **MUST use corresponding Commands** for library operations.		
	Implemented by	**Reviewed by**
➢ 6.1. Add GUI: If a Book is on loan, display a popup window to show the Patron details when the book is selected/clicked.	*AMOGH DATH*	*NITISH CHOWDARY*
➢ 6.2. Add GUI: When *Menu->Patrons->View* is selected, in the main window, show all Patrons and their details including the number of books they have on loan.	*AMOGH DATH*	*NITISH CHOWDARY*
➢ 6.3. Add GUI: If a Patron has Books on loan, display a popup window to show the details of the books.	*AMOGH DATH*	*NITISH CHOWDARY*
➢ 6.4. Add GUI: When *Menu->Patrons->Add* is selected, display a popup window. The popup should display a form that allows the addition of a new Patron to the system.	*AMOGH DATH*	*NITISH CHOWDARY*
➢ 6.5. Extend the functionality of the library system to allow for storing data to the file storage after the execution of commands that change the state of the system (e.g. "addbook", "return"). If the system fails to store the data on the file storage due to an error (e.g. file is already in used or corrupted), the program must inform the user and rollback any changes made to the system prior to the error. You can change the file permission to "read-only" to test this functionality.	*AMOGH DATH*	*NITISH CHOWDARY*
Achieving a mark of 70% to maximum of 79%		
	Implemented by	**Reviewed by**
➢ 7.1. Soft delete books. A book must not be removed from the text file; instead set a Boolean flag in Book class to mark it as deleted. Deleted books must not appear in the Books view. Only books not on loan can be deleted. *Hint: add a new command in the terminal.*	*AMOGH DATH*	*NITISH CHOWDARY*
➢ 7.2. Soft delete patrons. A patron must not be removed from the text file; instead set a Boolean flag in Patron class to mark them as deleted. Deleted patrons must not appear in the Patrons view. Only patrons with no books on loan can be deleted. *Hint: add a new command in the terminal.*	*AMOGH DATH*	*NITISH CHOWDARY*
➢ 7.3. Impose a limit on the maximum number (default: 2) of books that a patron can borrow and use that during issuing books for a patron.	*AMOGH DATH*	*NITISH CHOWDARY*

➢ 7.4. Extend the implementation for the GUI application to add the Delete functionality for both Books and Patrons using the GUI.	*AMOGH DATH*	*NITISH CHOWDARY*
➢ 7.5. Add Javadoc documentation for the **newly created** methods (i.e., delete book/patron in 7.1 and 7.2).	*NITISH CHOWDARY*	*AMOGH DATH*
Achieving a mark of 80% and over		
Note 1. You must complete all features above before attempting this grade, or they won't be marked. **Note 2.** All GUI operations must be mouse-only. No typing or text entry is allowed; the interface must provide selectable options for all actions.		
	Implemented by	**Reviewed by**
➢ 8.1. Show the loan history for a given patron (either in command line or via GUI).	*AMOGH DATH*	*NITISH CHOWDARY*
➢ 8.2. Implement GUI: *Menu->Books->Issue*. Issue a book to a patron.	*AMOGH DATH*	*NITISH CHOWDARY*
➢ 8.3. Implement GUI: *Menu->Books->Return*. Return a book.	*NITISH CHOWDARY*	*AMOGH DATH*
➢ 8.4. Add GUI: *Menu->Books->Renew*. Renew a book.	*NITISH CHOWDARY*	*AMOGH DATH*