Product Requirements

Team **<LETR LIB>**

Team Members:

Layan Alateeq [1]

Tea Disho [2]

Era Fejza [3]

Rosilda Bajrami [4]

❖ Brief problem statement

The library management system is a software created to satisfy the needs of both users and librarians . The library management system offers a wide range of features, making it smooth and user-friendly. By using their components, such as book availability, quantity, and users ID or phone number, etc., users may easily search for books and other users.This approach makes sure users can find their requested books quickly, promoting effective use of the library's resources. Users may handle their borrowing demands without needless delays or problems of any kind by having access to these types of operations.

A vital tool for modern libraries, the library management system serves as a link between librarians and users. It's efficient features, makes library resources easily accessible, and improves the entire library experience thanks to its user-friendly design and useful features. This system provides smooth and seamless interactions, resulting in an orderly and delightful library environment/community by bringing together librarians and users.

❖ System requirements

The library management system should perform efficiently, handling a large number of users and reservations without slowing down. It should be scalable to accommodate the library's growth in books and users. Reliability is crucial, ensuring uninterrupted operation and data protection. Usability is important for both managers and users, with an intuitive interface and easy navigation. The system should also be easily maintainable, with well-documented code and support for future enhancements.

//Libraries used in this program//

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "structures.h"
#include "get_data.h" //tea
#include "search_by_user.h" //layan
#include "search_by_title.h" //layan
#include "search_by_author.h" //tea
#include "upcoming_due_date.h" // layan
#include "overdue_books.h" //tea
#include "edit_book.h" //rosi
#include "edit_by_user.h" //layan
#include "delete_book.h" //era
#include "delete_user.h" //tea
#include "add_book.h" //era
#include "add_user.h" //rosi
er fbds//global Definition//
#define MAX_PEOPLE 150
#define MAX_BOOKS 300
```

❖ Users profile

The library management system will be used by two main groups: the Library Administrator/Manager and the Library Users. The Manager can handle user accounts, while Users can borrow and add books as needed. The system includes separate systems for borrowing, reservations, and reporting.

- ❖ Librarians/managers have to be highly proficient in using computers and software and will extensively use the system for various important task that require handling with important data.
- ❖ Library users, on the other hand, have diverse computer understanding levels, raging from advanced to limited experience. The program offers users a straightforward comprehension of its functions, requiring only basic computer knowledge.
- ❖ It is crucial for the system to be user-friendly, providing clear instructions and intuitive interfaces to accommodate users with different levels of proficiency. It focuses on being user-friendly, with an easy-to-use interface that combines Manager and User functions.

❖ Feature requirements (user stories)

No.	User Story Name	Description	Release
1.	void readDataFromFile(FILE* file, struct person* p, int* numPeople)	The program stores the data in separate files .The data gets saved in different arrays so there are two separate arrays ;one for books and one for borrowers	02/06/2023
2.		The data gets displayed for the manager to check that every field is consistent.An efficient way to make sure the manager is in coordination with everything.Ensure effective coordination and communication between the manager and all relevant aspects.	02/06/2023
3.	void delete_user(const char* filename, const char* key, int search_type)	Based on users request the program gives an option to delete their account.A necessary implementation to assure the library puts borrowers and their experiences first.	02/06/2023
4.	void print_due_date(const char* filename)	Displaying and giving the necessary notifications for the user's upcoming due dates.Ensures that the data is properly managed and users will not misuse the library's functionalities.	02/06/2023
5.	void search_books_by_author(const char* filename, const char* Author)	Both the user and manager can search by books author.Enables program users to utilize various search methods to create a customized user interface according to their preferences.	02/06/2023
6.	Add books	Any member of the system can add new books. The program asks for the quantity, and opens space for the new books to be added. They might be existing books also and their quantity is changed as soon as inserted. Undoubtedly, managers can perform this action too.	02/06/2023
7.	Check existing books	As a member or the manager wants to insert new books, it might be the case that the book is existent in the library.After inserting book details, the user gets notified about book's info. This operation is done for checking purposes too, in order to know if a book is part of the library system or not.	02/06/2023

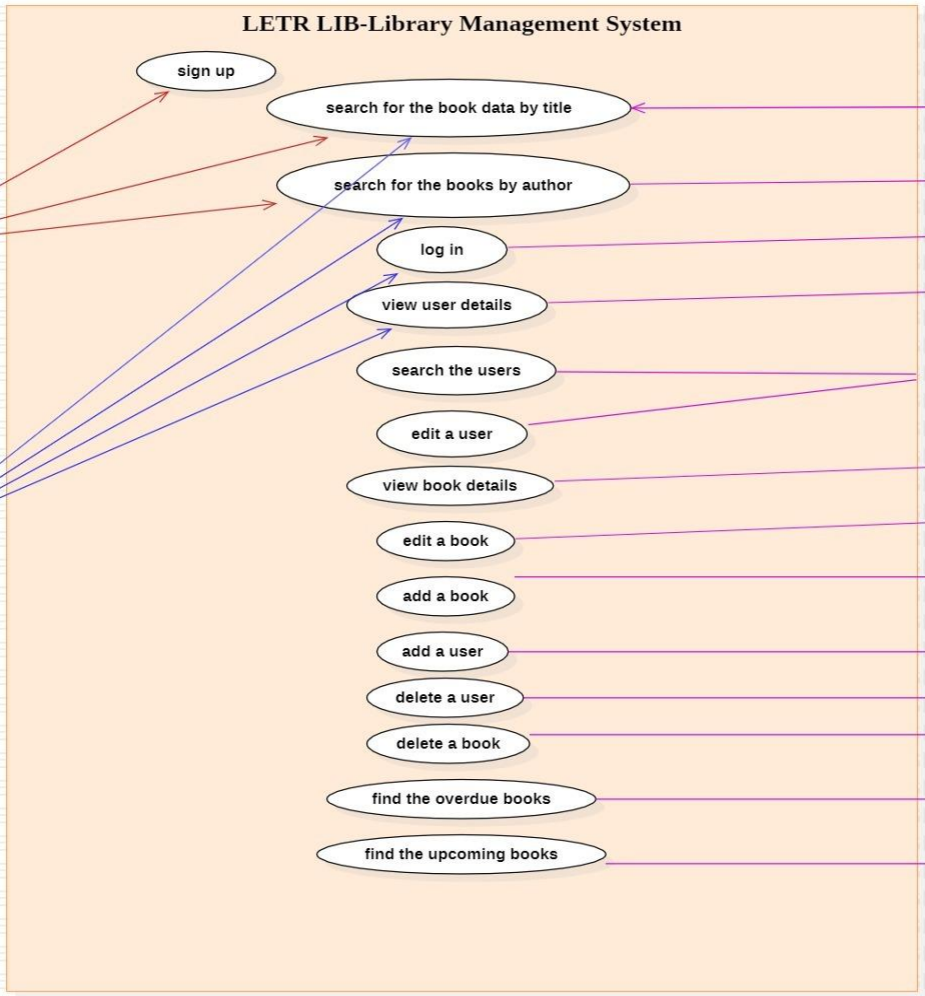
8.	Delete by author	Only the manager has the right to delete a book.Depending on their choice or circumstance, they can choose to delete the book by its author. This operation is mostly used when the manager wants to delete books from a specific author.	02/06/2023
9.	Delete by title	The manager can decide to delete/remove a book from the library by using its title.When a member borrows a book or when the book gets removed from the library, they can use this operation.	02/06/2023
10.	Editing user details	only the manager can edit the user details, they have to first search the user by name, surname, ID, or phone number, then they have access to user details, the manager can then select which part of the user they want to edit	02/06/2023
11.	Upcoming due date	a member can borrow their books for upto 9 days, as a manager i can check the members that are coming close to the deadline to return their books, which displays if the deadline is today, tomorrow or in 3 days.	02/06/2023
12.	Search by user	the manager can search the data{name, surname, ID, phone number, books they have borrowed, how many days until user has to return their books if they borrowed any} of any user by either phone number, name, surname & ID.	02/06/2023
13.	Search by title	the user, manager, and member use this function to go through the books in the database{the file} and check if the book they are look for exists or not	02/06/2023
14.	void editBook(Book b[], int numBooks)	This function allows the manager to search and edit book details such as author name,amount and quantity.It uses a loop to ask the manager for input,searches for the book by name in an array of books,performs the requested edit and then updates the book details in the file.	02/06/2023
15.	void addUser(struct person p[], int *numUsers)	This function allows the manager to input information for a new user and adds it to an array of person structures. It increments the number of users, writes the updated user list to a file named "user.txt", and displays a success message.	02/06/2023

❖ Use case diagram

A use case diagram consists of use cases (represented by oval shapes) that describe the interactions between actors (in this case are Manager,User and Member) and the system. Actors are represented by stick figures and are connected to the use cases with lines that indicate the type of interaction.

>Use cases describe the various ways in which actors interact with the system.For instance , in this case they interact by logging in,signing up ,searching for the book by title or author , viewing user details,searching the users ,editing the users,viewing book details,editing books,adding books,adding users,deleting users,deleting books,finding the overdue and upcoming books.

> Use case diagrams are useful for understanding the requirements of a system and identifying the features and functionality that need to be developed. They can help developers and stakeholders to communicate and understand the interactions between the system and its users.



Use Case Number:	***UC-01***
Use Case Name:	***Add books in the library***
Overview:	Any member of the system can add new books in the library. The program asks for the quantity, and opens space for the new books to be added. They might be existing books also and their quantity is changed as soon as inserted. Undoubtedly, managers can perform this action too.
Actor(s):	***Library Managers and Members***
Pre condition(s):	*-User has to be a verified manager* *-User has to be a valid member with a personal account*
Scenario Flow:	*1- User selects the "Add Books" option* *2- User adds the number of books they want to insert.* *3- The program receives the book details.* *4- If the books are already part of the library, their quantity is changed.* *5- Non existing books are now added to the library.*
	Alternate Flows: Include the post condition for each alternate flow if different from the main flow.
Post Condition:	*-The system adds the newly added book in the library.* *-New added book is displayed in the list of books.* *-Existing books quantity has changed.*

Use Case Number:	***UC-02***
Use Case Name:	***Delete books from the library***
Overview:	The manager can decide to delete/remove a book from the library by using **its title or its author.** They can use this operation when a member borrows a book or when the book gets removed from the library,
Actor(s):	**Library Managers Only**
Pre condition(s):	-User has to be a verified manager -The book needs to be existent on the library
Scenario Flow:	1- User selects the "Delete Books" option 2- User adds the number of books they want to delete. 2- User adds the option by which they want to delete books (by title/ by author) 3- The program receives the book details and checks for book availability. 4- The program deletes the book if the option is "by title". 5- The program deletes all books of an author if the option is "by author".
	Alternate Flows: Include the post condition for each alternate flow if different from the main flow.
Post Condition:	-The system deletes books from the library. -Deleted book is not displayed in the list of books.

Use Case Number:	***UC-03***
Use Case Name:	***Get data***
Overview:	The program stores the data in separate files .The data gets saved in different arrays so there are two separate arrays ;one for books and one for borrowers . In this function you also have the option to display the data to check if we read the info correctly .User don't have access
Actor(s):	**Library Managers**
Pre condition(s):	-User has to be a verified manager
Scenario Flow:	1- Manager selects the "Get Data" option 2- Then all the data of the files get printed instantly ,all of them divided with dash lines
Post Condition:	-The system prints the files in their respective order based on the order the users imputed them

Use Case Number:	**UC-04**

Use Case Name:	**Search by author**
Overview:	Any member of the system can search by author in the library. The user inputs the authors name that gets saved in a char variable .Enables program users to utilize various search methods to create a customized user interface according to their preferences. Undoubtedly, managers can perform this action too.
Actor(s):	**Library Managers and Members**
Pre condition(s):	-User has to be a verified manager -User has to be a valid member with a personal account
Scenario Flow:	1- User selects the "Search Books by Author" option 2- User adds the name of the author they're trying to find 3- The program saves the author name in a char and compares it with the column consisting all the other author names. 4- If the author is found it prints all the books under their name. 5- Else a message "The author you were looking for was not found" gets printed in the screen.
Post Condition:	

Use Case Number:	***UC-05***
Use Case Name:	***Overdue books***
Overview:	Displaying and giving the necessary notifications for the user's upcoming due dates.Ensures that the data is properly managed and users will not misuse the library's functionalities.
Actor(s):	**Library Managers and Members**
Pre condition(s):	-User has to be a verified manager
Scenario Flow:	1- Manager selects the "Overdue Books" option 2- If the overdue date is equal to 0 then the information of each user (name ,surname,ID,address) so each member of the structure gets printed automatically.
Post Condition:	-The program prints to the screen of the manager the users information so they can notify them.

Use Case Number:	***UC-06***
Use Case Name:	***Delete user***
Overview:	Based on users request the program gives an option to delete their account.A necessary implementation to assure the library puts borrowers and their experiences first.
Actor(s):	**Library Managers and Members**
Pre condition(s):	-User has to be a verified manager
Scenario Flow:	1- User selects the "Delete Users" option 2- User adds the searching type so pressing 1 for ID, 2 for name, 3 for surname, 4 for phone. 3-The user also adds the searching key (so f.e. the specific ID) 4- The program receives the users details. 5- If found the program deletes the row consisting of the users information. 5- The user doesn't exist anymore in the file
Post Condition:	-The system deletes the row of that found user in the file .

Use Case Number:	***UC-07***
Use Case Name:	***edit user details***
Overview:	only the manager can edit the user details, they have to first search the user by name, surname, ID, or phone number, then they have access to user details, the manager can then select which part of the user they want to edit
Actor(s):	***Library Managers***
Pre condition(s):	***-User has to be a verified manager***
Scenario Flow:	1- User selects the “edit user” option 2- manager inputs the method they want to find the user 3- the member is found 4- the manager is asked to input what they want to edit 5- the manager inputs what they want to edit
	Alternate Flows: Include the post condition for each alternate flow if different from the main flow.
Post Condition:	-The system adds the edited user details into the list of users -New edited details is added to the list of users -Existing user changes

Use Case Number:	***UC-08***
Use Case Name:	***Upcoming due date***
Overview:	A member can borrow their books for up to 9 days. As a manager I can check the members that are coming close to the deadline to return their books, which displays if the deadline is today, tomorrow or in 3 days.
Actor(s):	***Library Managers***
Pre condition(s):	-User has to be a verified manager
Scenario Flow:	1- User selects the “upcoming due date” option 2- the users with less than 4 days left are displayed
	Alternate Flows: Include the post condition for each alternate flow if different from the main flow.
Post Condition:	- the system displays the users with 3 or less days left to returning their books

Use Case Number:	***UC-09***
Use Case Name:	***Searching users***
Overview:	The manager can search the data{name, surname, ID, phone number, books they have borrowed, how many days until the user has to return their books if they borrowed any} of any user by either phone number, name, surname & ID.
Actor(s):	***Library Managers***
Pre condition(s):	-User has to be a verified manager
Scenario Flow:	1- User selects the “search user” option 2- the manager is asked to choose the methods they want to search for the user 3- the manager enters their option 4- the data is read through the system 5- the system displays all the users details
	Alternate Flows: Include the post condition for each alternate flow if different from the main flow.
Post Condition:	- the system requires some information about the user - the system displays the users information

Use Case Number:	***UC-10***
Use Case Name:	***searching by title***
Overview:	the user, manager, and member use this function to go through the books in the database{the file} and check if the book they are look for exists or not
Actor(s):	***Library Managers, users, library members***
Pre condition(s):	-User has to be a verified manager, members have to be verified by their ID,
Scenario Flow:	1- User selects the “search by title” option 2- the user is asked to input the title of the books 3- the user inputs the title of the book 4- the data is read through the system 5- the system displays all the books details
	Alternate Flows: Include the post condition for each alternate flow if different from the main flow.
Post Condition:	- the system requires some information about the user - the system displays the users information

Use Case Number:	**UC-11**
Use Case Name:	**Add user in the library**
Overview:	Only the manager can add new users in the library. The program asks for user information and stores it in the next index of the array.
Actor(s):	**Library Managers**
Pre condition(s):	-User has to be a verified manager
Scenario Flow:	1- UseR selects the "add user" option and then the program asks the manager to enter information for a new user. 2- The numUser increments. 3- The program opens the user details . 4- It then iterates through the p array of person structures up to the current number of users and writes the new user details to the file. 5- The program closes the file and a new user is added successfully.
	Alternate Flows: Include the post condition for each alternate flow if different from the main flow.
Post Condition:	-The system adds information for a new user. -The numUsers parameter increments. -New added user is displayed in the list of users.

Use Case Number:	**UC-12**
Use Case Name:	**Edit book details**
Overview:	Only the manager can edit the book details, they have to first search the book by name, then they can select which part of the book details they want to edit.
Actor(s):	Library Managers
Pre condition(s):	-User has to be a verified manager
Scenario Flow:	1- User selects the “edit book” option 2-The manager is asked to input the name of the book that wants to edit. 3- The manager searches for the book that wants to edit by name. 4-The manager selects which part of the book details wants to edit. 5-Book details are edited successfully.
	Alternate Flows: Include the post condition for each alternate flow if different from the main flow.
Post Condition:	-The system adds the edited book details into the list of books. -New edited book is added to the list of books. -Existing book changes .