SOFTWARE DESIGN DOCUMENT

PSAJ Archive
LIBTECH
Software Design Document

Cardenas, Maritonee Danielle I.
Chavez, Rustan Angello A.
David, Devonn Carl D.
Delos Santos, Angelo Joe O.

Date: (05/03/2023)

TABLE OF CONTENTS

1.0 INTRODUCTION

1.1 PURPOSE

To manage library operations effectively and automatically while improving the entire library experience, LibTech was created. LibTech seeks to streamline and automate a number of library administration operations. By automating these procedures, LibTech frees up library staff members' time and energy so they can concentrate on offering library users better services. LibTech helps libraries make the best use of their resources. It keeps track of inventory, keeps tabs on book circulation, and offers perceptions into well-liked products or topics. Administrators of libraries can use this data to make well-informed choices about collection growth, resource allocation, and general library planning.

This Software Design Documentation contains a thorough explanation of the system's design and architecture. It gives a high-level overview of the various components, modules, and interactions that will be used to build the system. In order to direct the development process, it provides technical specifications. Data structures, algorithms, interface designs, database schemas, and potential integration with other technologies and systems are all included in this. These specifics guarantee precision and consistency during the application stage. The development team behind LibTech is the main target of the SDD. It acts as a manual, giving programmers a comprehensive grasp of the specifications, technical requirements, and design of the system. They can write code more effectively and with greater consistency thanks to this information.

Stakeholders including library directors, employees, and possible investors should be informed about the SDD. They will be able to learn more about the proposed system's attributes and potential advantages. The document can be reviewed by stakeholders who can then offer insightful input to help guide the system's development and determine whether it satisfies their expectations.

1.2 SCOPE

LibTech is intended to improve the user experience for both library employees and customers by streamlining and automating library operations. It has a variety of features and functionalities that help libraries effectively manage their resources, increase accessibility, and streamline all aspects of library operations.

LibTech offers modules for categorizing and arranging books, keeping track of inventory, managing loans and returns, producing reports, and enhancing user interactions. It has an easy-to-use interface that makes it possible for library personnel to complete duties quickly and for customers to browse and borrow resources with ease.

The primary goals and objectives of LibTech are as follows:

- **Efficiency** : LibTech strives to automate repetitive procedures, expedite library operations, and maximize resource usage. The workload of the personnel is reduced, productivity is increased, and time and resources are used more effectively by doing away with manual operations.
- **Accessibility** : LibTech aims to give users better access to library resources. Users may quickly browse and check out goods thanks to quick catalog searches, checkout systems, and content integration. This improves user experience overall and promotes more frequent library use.
- **Analytics** : Reports and analytics on library operations, book circulation, and usage patterns are produced by LibTech. In order to improve services and resource allocation, administrators can use this information to evaluate performance, spot patterns, and make data-driven decisions.

The key benefits of LibTech includes:

- **Time and Resource Savings** : Personnel workload is reduced by automating manual operations, allowing them to concentrate on higher-value activities.
- **Efficient Resource Management** : LibTech offers information on resource use, popular genres, and book circulation. Administrators can use this information to make informed choices about resource allocation, collection development, and general library planning.
- **Enhanced Reporting** : Administrators can get useful data thanks to LibTech's reporting and analytics features. This encourages the use of data-driven decisions, enabling ongoing development of library operations and services.

1.3 OVERVIEW

In order to provide libraries with cutting-edge solutions, PSAJ Archive developed the LibTech Library Management System. Its goals include better resource management, automation and streamlining of library operations, and user experiences.

PSAJ Archive formed LibTech is committed to using technology to improve library services. They are aware of how libraries' requirements alter over time and how critical it is to effectively manage resources and facilitate their accessibility. Their goal is to provide libraries with the resources they need to meet the demands of

modern users and to provide an intuitive user experience for both patrons and employees.

The SDD for LibTech includes a thorough description of the system's architecture, design, and features. Project managers, testers, stakeholders, and developers involved in the development process can benefit from it as a valuable resource because it offers them thorough insights. The text is divided into a number of clearly arranged sections that explore various facets of the system's development and operation.

➔ ***Introduction*** : This section gives a summary of the document's content, an introduction to LibTech, and its goal. It establishes the scene for the remainder of the document.
➔ ***System Architecture*** : The overall structure of LibTech, including its modules, components, and interactions, is described in this section. It gives a high-level description of the design and insights into how the system's many components interact.
➔ ***Data Design*** : The database design and data structures used by LibTech are the main topics of this section. It provides an overview of the system's data entities, connections, and properties. Database schemas or entity-relationship diagrams may be present.
➔ ***User Interface Design*** : The user interface design for LibTech is described in this section. It addresses the design, usability, and navigation of the user interfaces that library personnel and visitors utilize. It might also have screen flow diagrams, prototypes, or wireframes to show how the interface is to be designed.

1.4 DEFINITIONS AND ACRONYMS

- **Software Design Document (SDD) –** a document defining the components, functionality, and interactions of a software system's design and architecture.
- **LibTech –** The currently under development Library Management System that automates library activities and improves user experience.
- **Cataloging –** The act of assigning unique identities, categorizing, and giving metadata in order to create and organize a catalog of library items, including books, journals, and multimedia materials.
- **Inventory –** The entire stock of books, journals, multimedia, and other products that can be borrowed or used as references from the library.
- **Loans and Returns –** The procedure of handing out library materials for checkout and then getting them back. It includes keeping track of due dates, controlling renewals, and supervising the handling of returned borrowed things.

- **Reports –** Analytical insights into a range of library activities, including circulation, usage patterns, popular genres, and overdue items.
- **User Interface (UI) –** The LibTech visual and interactive elements that users employ to carry out tasks and access system capabilities. Screens, menus, buttons, and navigational elements are all part of it.
- **Module –** An independent component of library technology that does a certain task or offers a group of related capabilities. Modules are intended to be reusable, modular parts of the entire system.
- **Integration –** The process of integrating several software systems, technologies, or modules to function together without any noticeable hiccups. Integrating with other systems, including content management systems, is referred to as integration in the field of library technology.
- **Algorithms –** Formulas or steps used to carry out calculations or address particular problems within the system. The reasoning and processes of processing required to produce desired results are defined by algorithms.
- **Data Structures –** The manner that LibTech organizes and stores data. It encompasses ideas like arrays, lists, databases, or specific structures made to store and access information quickly.
- **Quality Assurance (QA) –** A procedure used to ensure that LibTech complies with predetermined quality requirements. It entails evaluating the system's reliability, performance, and usability through testing, verification, and validation.

2.0 SYSTEM OVERVIEW

To enable efficient library management, LibTech provides a wide range of functionalities, including:

➔ ***Catalog Management*** : To catalog and arrange library materials including books, journals, and multimedia objects, LibTech offers tools. It enables librarians to manage resource structure and categorization, assign unique identifiers, and input and update metadata.

➔ ***User Managemen***t : The system gives librarians the ability to manage user accounts, including profile maintenance, registration, and authentication. Updates to user information, borrowing rights, and fine management are supported.

➔ ***Circulation and Borrowing Management*** : By automating the circulation process, LibTech enables librarians to efficiently manage holds, renewals, and checkouts. It monitors loan histories, facilitates smooth transactions, and keeps track of due dates.

➔ ***Report Management*** : The system generates reports and offers insights on library operations, including user activity, inventory analysis, and circulation

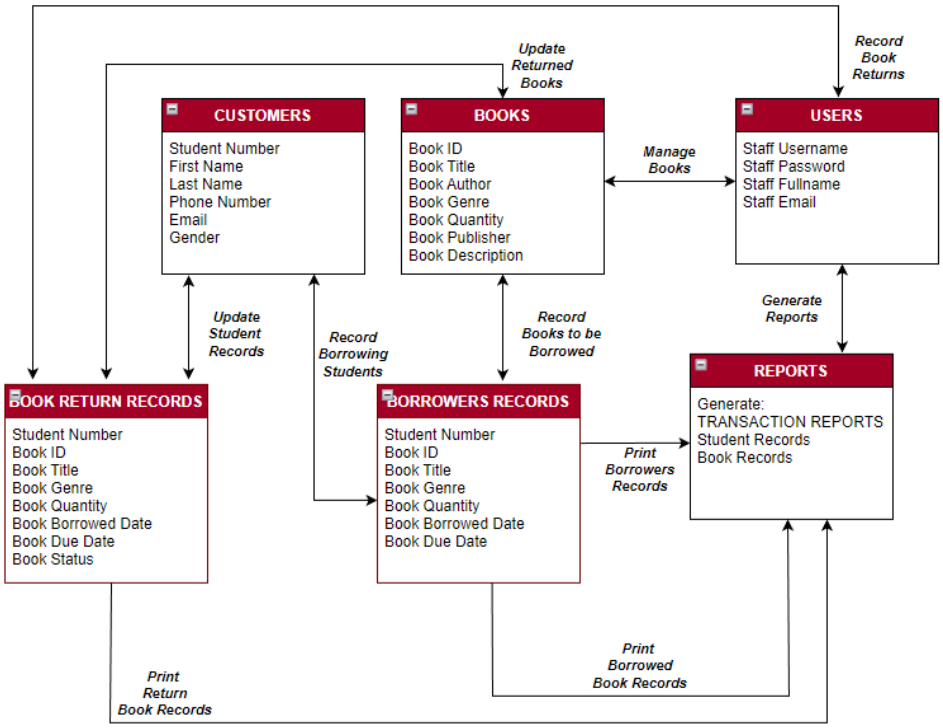
statistics. Making data-driven decisions for resource allocation and service enhancements is made easier for librarians thanks to these insights.

LibTech is created with consideration for the unique setting of libraries. It focuses on the difficulties librarians have managing and organizing enormous resource collections while boosting the user experience. The system is meant to be simple to use, straightforward, and adaptable to the specific needs and workflows of various libraries.

LibTech's design prioritizes scalability, simplicity, and effectiveness. Librarians may easily navigate and interact with it thanks to its clear and simple user interface. The system design is strong, enabling flexibility and scalability to support libraries of different sizes and complexity levels.

3.0 SYSTEM ARCHITECTURE

3.1 ARCHITECTURAL DESIGN



3.2 DECOMPOSITION DESCRIPTION

User Interface Subsystem

The interface via which users communicate with the LibTech system is provided by the User Interface Subsystem. It includes all of the graphical user interface (GUI)

elements, screens, menus, and forms that let users access information and carry out different tasks. To handle user requests, this subsystem controls user input, checks user interactions, shows pertinent data, and interacts with other subsystems.

Catalog Management Subsystem
The management of library resources and catalog data is handled by the catalog management subsystem. It is in charge of duties like expanding the library of resources, updating resource metadata, classifying resources, and controlling their availability. To show search results, make resource-related actions (such borrowing or reserving) possible, and get comprehensive resource information, this subsystem communicates with the User Interface Subsystem.

Circulation Subsystem
The Circulation Subsystem controls how library materials are checked out and returned. It manages tasks including renewing loans, putting holds on products, checking out items, and managing due dates. This subsystem makes sure loans are handled properly, verifies a user's eligibility for borrowing, updates the status of resource availability, and sends alerts when anything is overdue or approaching a deadline. To deliver loan-related information, handle loan requests, and update resource availability, it works with the User Interface Subsystem.

User Management Subsystem
The LibTech system's user accounts, authentication, and access privileges are managed via the user management subsystem. It manages operations including user registration, login, password administration, and profile updates for users. This subsystem handles user-related data, verifies user credentials, and implements access control policies to ensure that users have tailored and safe access. In order to handle user authentication, retrieve user data, and enforce the proper access privileges, it communicates with the User Interface Subsystem.

Reporting and Analytics Subsystem
The Reporting and Analytics Subsystem gathers information from other LibTech subsystems and produces reports and analytical insights. It offers data on user behavior, resource utilization, circulation patterns, and inventory control. To present data in a comprehensible way, this subsystem might use data visualization methods. It engages in data collection interactions with other subsystems and may make use of external reporting tools or databases to produce reports and analytics.

3.3 DESIGN RATIONALE

Data processing needs, system performance, and maintainability were among the criteria used to choose the ***Data-Flow Architecture*** for LibTech in the architectural design section of the document.

➔ *Efficient Data Processing* : The data-flow architecture was selected for LibTech because it can analyze and modify data effectively. In this design, data passes through various components or modules, each of which is in charge of carrying out particular data transformations or operations. Parallel processing is possible with this approach, allowing for quicker and more effective data handling.

➔ *Modularity and Reusability* : Because each component or module in the data-flow architecture is in charge of a particular data processing task, it encourages modularity. The system is simpler to comprehend, create, and maintain because of its modular design. Additionally, because modules can be utilized in various system components or even in separate projects, the modular structure of the architecture enables component reuse. This lowers redundancy and improves development efficiency.

➔ *Maintainability* : The data-flow design provides a distinct separation of concerns, which improves maintainability. It is simpler to find problems and correct them or to make changes without affecting the entire system because each component or module concentrates on a particular data processing task. Because data flows through the system in a structured way, making it simpler to track and debug data-related issues, this architecture also encourages code readability.

There are trade-offs and factors to be considered even if the Data-Flow Architecture has several benefits:

➔ *Increased Complexity* : The implementation of a data-flow architecture might add complexity since several components must be built to manage data dependencies and guarantee proper data flow. This intricacy may affect the work and duration of development.

➔ *Overhead of Data Flow Management* : Designing and coordinating the data flow between components carefully is necessary. This overhead may have an effect on system performance and call for more resources for data synchronization and routing

➔ *Learning Curve* : Data flow patterns and communication methods may need a certain amount of expertise and awareness on the part of developers working with the data-flow architecture. The development team can have a learning curve as a result of this.

4.0 DATA DESIGN

4.1 DATA DESCRIPTION

Using the SQLite database, LibTech converts the library management system's information domain into data structures. The SQLite database structures, controls and organizes the main data or system elements.

The data is kept in a single file via the serverless, lightweight database engine SQLite. For embedded devices or applications that need a local database, it is frequently utilized. To store and manage the main data or system entities, LibTech can make use of SQLite.

The associations between entities in SQLite are frequently defined by primary key and foreign key restrictions. If the "Users" and "Resources" entities are related, the "Users" table, for example, might have a foreign key column that points to the primary key of the "Resources" database. This may lead to associations being established and the preservation of referential integrity.

To manipulate and arrange the contents of the database, SQLite offers a number of SQL (Structured Query Language) procedures. To access, add to, update, and delete data from the tables, utilize SQL queries. In order to do calculations or create derived data, SQLite also supports a variety of data aggregation and manipulation features.

The SQLite database file itself serves as the main data storage item when LibTech uses SQLite. All of the tables, columns, and structured data are contained in the database file. Any related files or resources, such as cover photos or papers, can also be stored as binary data in the SQLite database using the proper data types.

4.2 DATA DICTIONARY

Add Book Function		
Variable	**Data Type**	**Description**
id	QString	Stores the ID of the book
bookName	QString	Stores the name of the book
author	QString	Stores the name of the author
genre	QString	Stores the genre of the book

quantity	int	Stores the quantity of the book
publisher	QString	Stores the name of the publisher
publishedDate	QDate	Stores the published date of the book
description	QString	Stores the description of the book

Add Member Function		
Variable	**Data Type**	**Description**
id	QString	Stores the ID of the member
firstName	QString	Stores the first name of the member
lastName	QString	Stores the last name of the member
contactNumber	QString	Stores the contact number of the member
email	QString	Stores the email address of the member
gender	QString	Stores the gender of the member

Book List Function		
Variable	**Data Type**	**Description**
booksTable	QTableView*	Points to the table view widget displaying the books
searchLineEdit	QLineEdit*	Points to the line edit widget for searching books
sortComboBox	QComboBox*	Points to the combo box widget for sorting books
idLabel	QLabel*	Points to the label widget displaying book ID
bookNameLabel	QLabel*	Points to the label widget displaying book name

authorLabel	QLabel*	Points to the label widget displaying book author
genreLabel	QLabel*	Points to the label widget displaying book genre
quantityLabel	QLabel*	Points to the label widget displaying book genre
publisherLabel	QLabel*	Points to the label widget displaying book publisher
publishedDateLabel	QLabel*	Points to the label widget displaying published date
descriptionLabel	QLabel*	Points to the label widget displaying book description

Borrow Book Function		
Variable	**Data Type**	**Description**
tableView	QTableView*	Pointer to the QTableView widget used to display the table
tableModel	QStandardItemModel*	Pointer to the QStandardItemModel used for the table view
headers	QStringList	List of column headers for the table view
bookId	QString	Stores the book ID entered by the user
memberId	QString	Stores the member ID entered by the user
query	QSqlQuery	SQL query object used for executing queries
bookName	QString	Stores the name of the book retrieved from the database
author	QString	Stores the author of the book retrieved from the database
genre	QString	Stores the genre of the book retrieved from the database

description	QString	Stores the description of the book retrieved from the database
quantity	int	Stores the quantity of the book retrieved from the database
fullName	QString	Stores the full name of the member retrieved from the database
issuanceDate	QDate	Stores the issuance date of the book
returnDate	QDate	Stores the return date of the book
notes	QString	Stores the notes for the issuance
bookId	QString	Stores the book ID for the issuance
memberId	QString	Stores the member ID for the issuance
rowData	QStringList	List of data for a single row in the table view
items	QList<QStandardItem*>	List of QStandardItem objects representing the data for a single row
confirmDelete	QMessageBox::StandardButton	Stores the user's response to the delete confirmation dialog
selectedIndexes	QModelIndexList	List of selected indexes in the table view
row	int	Stores the row index of the selected row in the table view
deleteQuery	QSqlQuery	SQL query object used for deleting an issuance
updateQuantityQuery	QSqlQuery	SQL query object used for updating the book quantity
updateStatusQuery	QSqlQuery	SQL query object used for updating the book status

Edit Book Function		
Variable	**Data Type**	**Description**
m_id	QString	Stores the ID of the book
m_bookName	QString	Stores the name of the book
m_author	QString	Stores the author of the book
m_genre	QString	Stores the genre of the book
m_quantity	int	Stores the quantity of the book
m_publisher	QString	Stores the publisher of the book
m_publishedDate	QDate	Stores the published date of the book
m_description	QString	Stores the description of the book
ui->idLineEdit	QLineEdit	Displays and allows editing of the book ID
ui->bookNameLineEdit	QLineEdit	Displays and allows editing of the book name
ui->authorLineEdit	QLineEdit	Displays and allows editing of the author
ui->genreLineEdit	QLineEdit	Displays and allows editing of the genre
ui->quantitySpinBox	QSpinBox	Displays and allows editing of the quantity
ui->publisherLineEdit	QLineEdit	Displays and allows editing of the publisher
ui->dateEdit	QDateEdit	Displays and allows editing of the published date
ui->descriptionTextEdit	QTextEdit	Displays and allows editing of the description

Edit Members Function		
Variable	**Data Type**	**Description**
m_id	int	Stores the ID of the member
m_firstName	QString	Stores the first name of the member
m_lastName	QString	Stores the last name of the member
m_contactNumber	int	Stores the contact number of the member
m_email	QString	Stores the email address of the member
m_gender	QString	Stores the gender of the member
ui->memberIdLineEdit	QLineEdit	Displays and allows editing of the member ID
ui->firstNameLineEdit	QLineEdit	Displays and allows editing of the member's first name
ui->lastNameLineEdit	QLineEdit	Displays and allows editing of the member's last name
ui->contactNumberLineEdit	QLineEdit	Displays and allows editing of the member's contact number
ui->emailLineEdit	QLineEdit	Displays and allows editing of the member's email address
ui->genderComboBox	QComboBox	Displays and allows selection of the member's gender

Return Book Function		
Variable	**Data Type**	**Description**
tableView	QTableView*	Pointer to a QTableView object used to display the table view
tableModel	QStandardItemModel*	Pointer to a QStandardItemModel object used as the model for the table view

headers	QStringList	List of column headers for the table view
layout	QVBoxLayout*	Pointer to a QVBoxLayout object used as the layout for the table view container
sortingOptions	QStringList	List of sorting options for the combo box
db	QSqlDatabase	QSqlDatabase object used to connect to the database
createTableQuery	QSqlQuery	QSqlQuery object used to create the "issuance" table
query	QSqlQuery	QSqlQuery object used to retrieve data from the "issuance" table
bookId	QString	String variable to store the book ID
memberId	QString	String variable to store the member ID
issuanceDate	QString	String variable to store the issuance date
returnDate	QString	String variable to store the return date
notes	QString	String variable to store the notes
status	QString	String variable to store the status
items	QList<QStandardItem*>	List of QStandardItem objects used to store table view data
selectedIndex	QModelIndex	QModelIndex object representing the selected index in the table view
row	int	Integer variable representing the row of the selected index in the table view
bookName	QString	String variable to store the book name retrieved from the books table

genre	QString	String variable to store the genre retrieved from the books table
coverImageData	QByteArray	QByteArray variable to store the cover image data retrieved from the books table
firstName	QString	String variable to store the first name of the member
lastName	QString	String variable to store the last name of the member
fullName	QString	String variable to store the full name of the member
coverImage	QPixmap	QPixmap object used to display the cover image
updateQuantityQuery	QSqlQuery	QSqlQuery object used to update the book quantity in the "books" table
updateBookStatusQuery	QSqlQuery	QSqlQuery object used to update the book status in the "books" table
updateIssuanceQuery	QSqlQuery	QSqlQuery object used to update the issuance status in the "issuance" table
confirmation	QMessageBox::StandardButton	StandardButton variable to store the user's confirmation choice

Edit Members Function		
Variable	**Data Type**	**Description**
parent	QWidget*	Pointer to the parent widget
ui	Ui::viewmembers*	Pointer to the user interface for the viewmembers dialog
query	QSqlQuery	SQL query object for retrieving member information from the database
model	QStandardItemModel*	Model for the table view displaying member information

searchText	const QString&	Text entered in the search line edit for filtering members
selectionModel	QItemSelectionModel*	Selection model for the members table view
selectedRows	QModelIndexList	List of selected rows in the members table view
confirmation	QMessageBox::StandardButton	User's confirmation for member deletion
selectedRow	QModelIndex	Index of the selected row in the members table view
firstName	QString	First name of the selected member
lastName	QString	Last name of the selected member
parent	QWidget*	Pointer to the parent widget

Login Function		
Variable	**Data Type**	**Description**
ui->username	QLineEdit	Allows input and retrieval of the username
ui->password	QLineEdit	Allows input and retrieval of the password

Splash Screen Function		
Variable	**Data Type**	**Description**
parent	QWidget*	Pointer to the parent widget
ui	Ui::SplashScreen*	Pointer to the user interface for the splash screen window
progressbar	QProgressBar*	Pointer to the progress bar widget

label_value	QLabel*	Pointer to the label widget displaying the current progress value
loginScreen	loginscreen*	Pointer to the login screen window

Start Screen Function		
Variable	**Data Type**	**Description**
parent	QWidget*	Pointer to the parent widget
ui	Ui::StartScreen*	Pointer to the user interface for the start screen window
LoadingScreen	SplashScreen*	Pointer to the splash screen window

History Function		
Variable	**Data Type**	**Description**
model	QSqlQueryModel*	Pointer to the QSqlQueryModel object used to store and manage the data retrieved from the database for the history table.
database	QSqlDatabase	QSqlDatabase object used to establish a connection to the database.
ui	Ui::history*	Pointer to the user interface object for the history dialog.
tableViewHistory	QTableView*	Pointer to the QTableView widget used to display the history data.
history	QWidget*	Pointer to the QWidget object representing the history dialog window.
printer	QPrinter	QPrinter object used for printing the history table.

dialog	QPrintDialog	QPrintDialog object used to configure the printer settings before printing.
rowcount	int	The number of rows in the history table.
columnCount	int	The number of columns in the history table.
cellWidth	const int	The width of each cell in the printed table.
cellHeight	const int	The height of each cell in the printed table.
tableWidth	int	The total width of the printed table.
tableHeight	int	The total height of the printed table.
tableX	int	The X-coordinate of the top-left corner of the printed table.
tableY	int	The Y-coordinate of the top-left corner of the printed table.
painter	QPainter	QPainter object used for painting the table and its contents.
headerText	QString	The text of the table header for each column.
text	QString	The text of each cell in the table.
query	QSqlQuery	QSqlQuery object used for executing SQL queries on the database.
loadData	bool	A flag indicating whether to load data into the model.

5.0 COMPONENT DESIGN

User Interface Subsystem

Algorithm for handling user requests

1. Receive user input.
2. Validate and sanitize user input.
3. Determine the user's requested action.
4. Process the action based on the user's input.
5. Retrieve and display relevant data from other subsystems.
6. Update the user interface with the appropriate response.
7. Repeat steps 1-6 until the user exits the interface.

Catalog Management Subsystem

Algorithm for managing library resources

1. Receive requests to add, update, or delete resources.
2. Validate and sanitize resource data.
3. Perform the requested operation on the catalog.
4. Update resource metadata and availability status.
5. Communicate with the User Interface Subsystem to display search results and resource information.
6. Repeat steps 1-5 for any subsequent resource management requests.

Circulation Subsystem

Algorithm for handling library material circulation

1. Receive user requests for checking out, returning, renewing, or placing holds on library materials.
2. Verify user eligibility for borrowing and handle authentication.
3. Update resource availability status and due dates based on the requested action.
4. Generate notifications for overdue or approaching deadline items.
5. Communicate with the User Interface Subsystem to provide loan-related information and handle loan requests.
6. Repeat steps 1-5 for any subsequent circulation requests.

User Management Subsystem

Algorithm for managing user accounts and authentication

1. Receive requests for user registration, login, password administration, and profile updates.
2. Validate and sanitize user data.
3. Perform the requested operation on the user management system.
4. Verify user credentials for authentication.
5. Enforce access control policies based on user roles and privileges.

6. Communicate with the User Interface Subsystem to handle user authentication, retrieve user data, and enforce access privileges.
7. Repeat steps 1-6 for any subsequent user management requests.

Reporting and Analytics Subsystem

Algorithm for generating reports and analytics

1. Gather data from other subsystems or external sources.
2. Process and analyze the collected data.
3. Generate statistical reports, insights, or visualizations.
4. Present the reports and analytics to the user or stakeholders.
5. Communicate with other subsystems to collect necessary data and utilize external reporting tools or databases if required.
6. Repeat steps 1-5 for any subsequent reporting and analytics requests.

6.0 HUMAN INTERFACE DESIGN

6.1 OVERVIEW OF USER INTERFACE

LibTech provides a range of features and functionalities that enable them to efficiently interact with the system and perform various tasks.

Catalog Search and Resource Access

- The User Interface Subsystem's search feature allows users to look up library contents like books, journals, or multimedia files.
- To locate the needed materials, they might enter keywords, titles, authors, or other pertinent details.
- The search results will be displayed together with pertinent information like title, author, availability status, and location.

User Registration and Account Management

- The User Interface Subsystem is where new users can create a library account.
- They will give the relevant facts, such as name, phone number, and perhaps a special code.
- The software will check the data, create a new user account, and assign a special code or ID.
- The User Interface Subsystem allows users to manage their account details, contact information, passwords, and borrowing history.

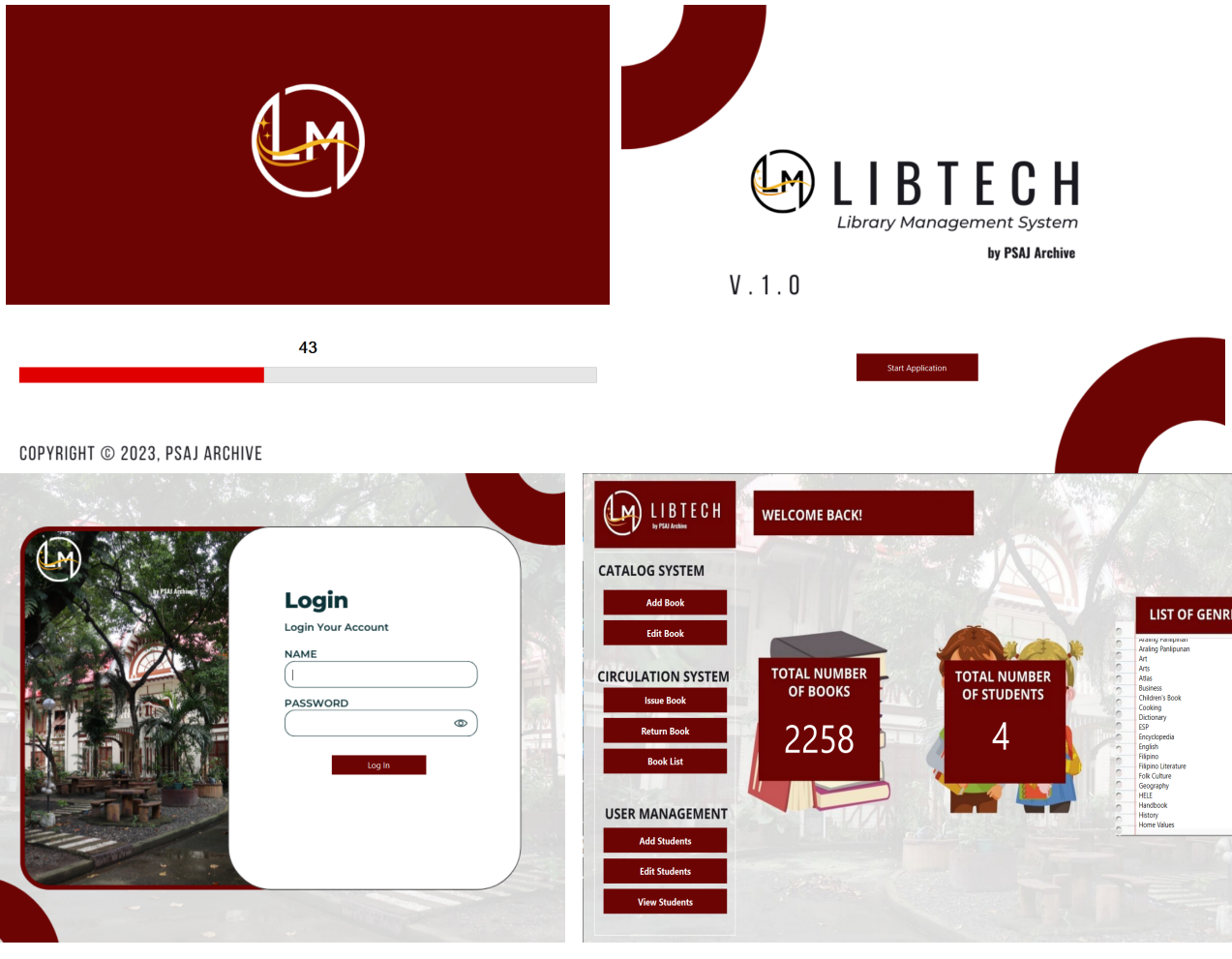
Borrowing and Returning Materials

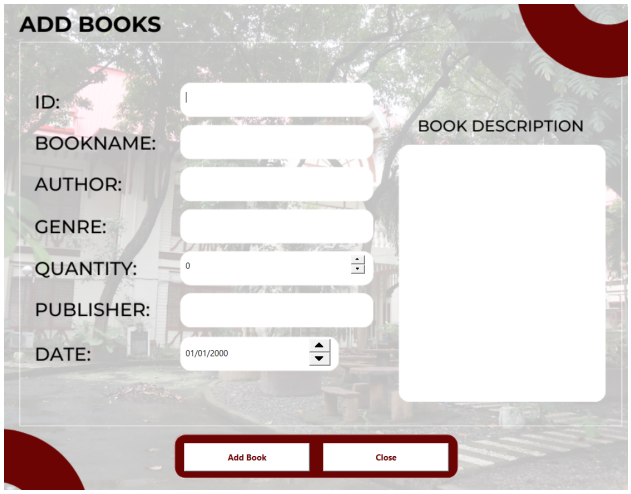
- By choosing the desired items and starting the checkout procedure through the User Interface Subsystem, users can borrow library materials.
- The system will confirm the user's eligibility and revise the status of the borrowed items' availability.
- Users will be informed of the due date and any other loan-specific instructions.
- Users will use the User Interface Subsystem to check in goods while returning materials, enabling the system to update availability and clear the related loan records.

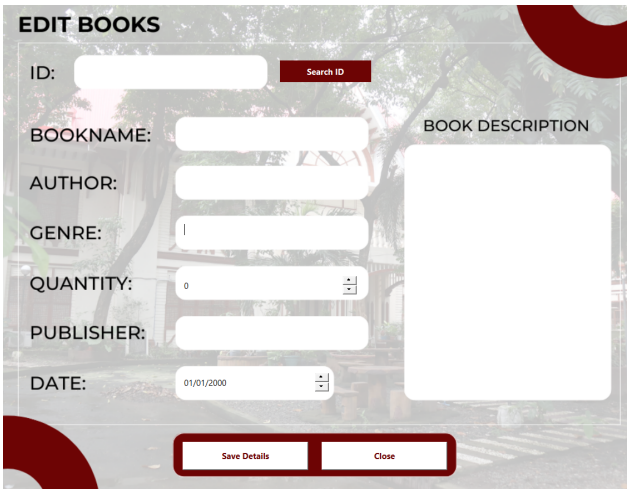
Reporting and Analytics

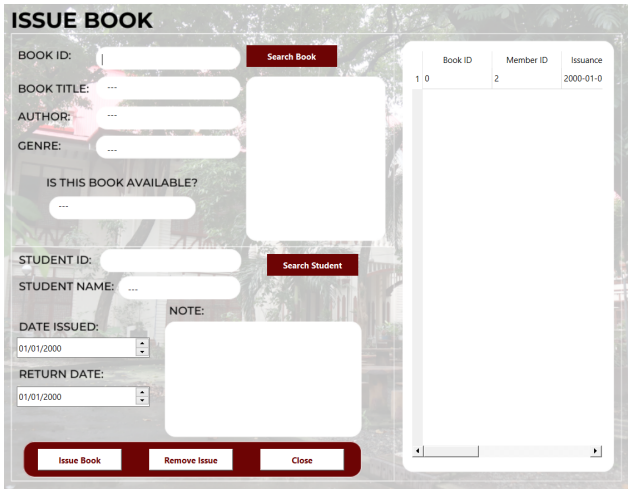
- The user behavior, resource usage, circulation patterns, and inventory management will all be taken into account when the reporting and analytics subsystem generates reports and analytics.
- These reports and insights are accessible through the User Interface Subsystem to those with the necessary access rights, such as administrators or library staff.
- To aid users in understanding library operations and making wise decisions, the system will offer feedback in the form of graphs, statistics, or summaries.

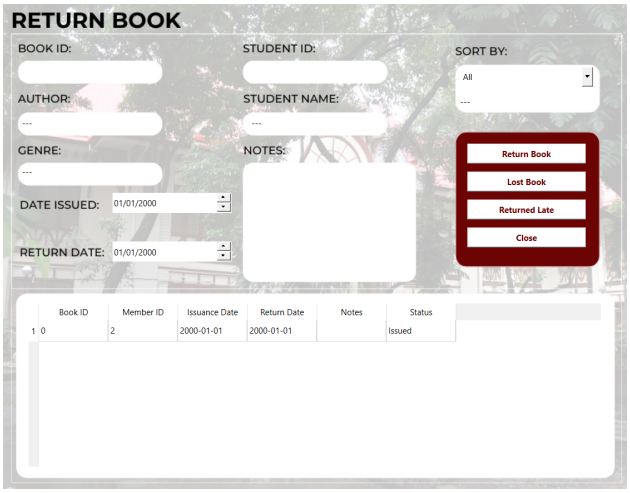
6.2 SCREEN IMAGES

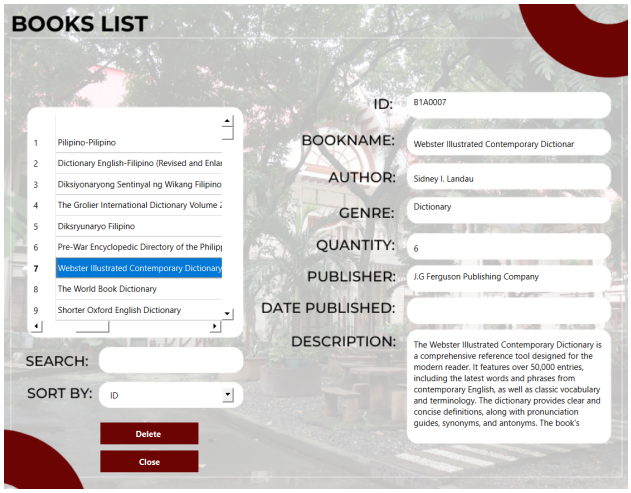


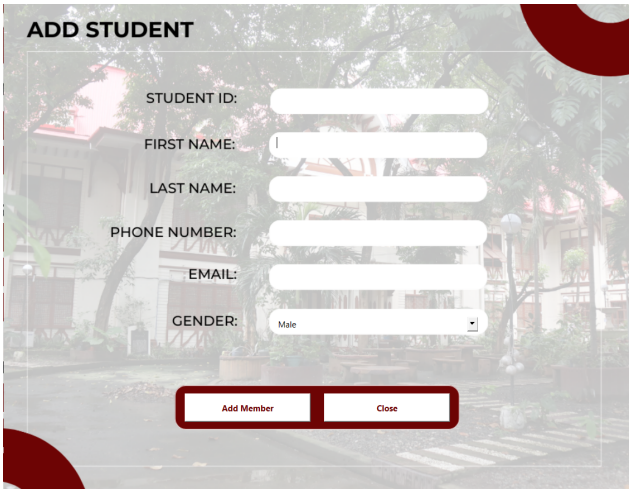

ADD BOOKS
ID:
BOOKNAME:
AUTHOR:
GENRE:
QUANTITY:
0
PUBLISHER:
DATE:
01/01/2000
BOOK DESCRIPTION
Add Book
Close

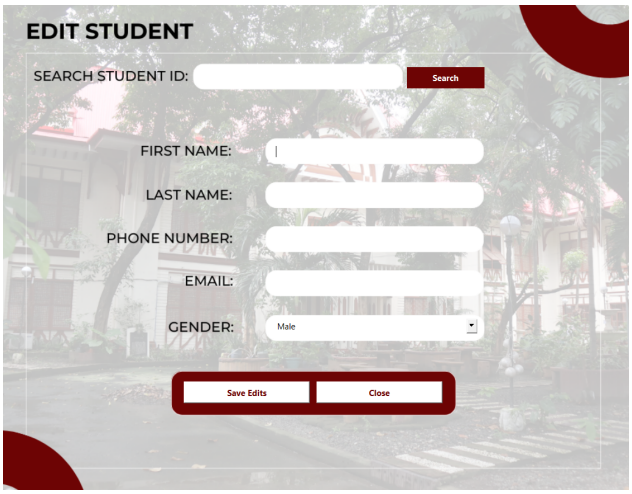

EDIT BOOKS
ID:
Search ID
BOOKNAME:
AUTHOR:
GENRE:
QUANTITY:
0
PUBLISHER:
DATE:
01/01/2000
BOOK DESCRIPTION
Save Details
Close

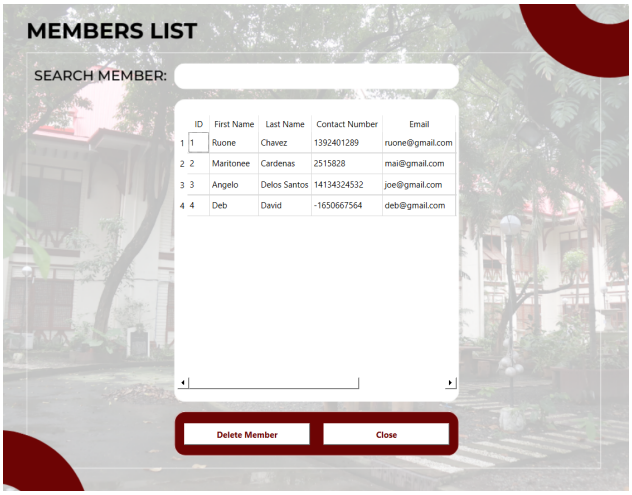

ISSUE BOOK
BOOK ID:
Search Book
BOOK TITLE:
AUTHOR:
GENRE:
IS THIS BOOK AVAILABLE?
STUDENT ID:
Search Student
STUDENT NAME:
NOTE:
DATE ISSUED:
01/01/2000
RETURN DATE:
01/01/2000
Book ID
Member ID
Issuance
1 0 2 2000-01-0
Issue Book
Remove Issue
Close

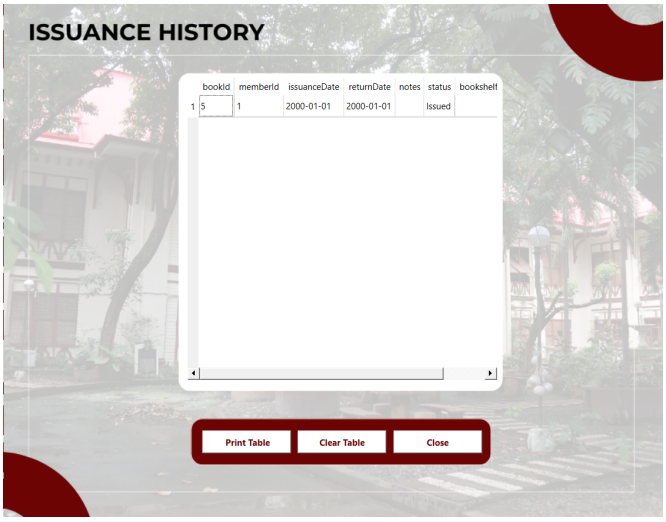

RETURN BOOK
BOOK ID:
AUTHOR:
GENRE:
DATE ISSUED:
01/01/2000
RETURN DATE:
01/01/2000
STUDENT ID:
STUDENT NAME:
NOTES:
SORT BY:
All
Return Book
Lost Book
Returned Late
Close
Book ID
Member ID
Issuance Date
Return Date
Notes
Status
1 0 2 2000-01-01 2000-01-01 Issued


BOOKS LIST
1 Pilipino-Pilipino
2 Dictionary English-Filipino (Revised and Enla
3 Diksiyonaryong Sentinyal ng Wikang Filipino
4 The Grolier International Dictionary Volume
5 Diksryunaryo Filipino
6 Pre-War Encyclopedic Directory of the Philip
7 Webster Illustrated Contemporary Dictionary
8 The World Book Dictionary
9 Shorter Oxford English Dictionary
SEARCH:
SORT BY: ID
Delete
Close
ID: B1A0007
BOOKNAME: Webster Illustrated Contemporary Dictionar
AUTHOR: Sidney I. Landau
GENRE: Dictionary
QUANTITY: 6
PUBLISHER: J.G Ferguson Publishing Company
DATE PUBLISHED:
DESCRIPTION:
The Webster Illustrated Contemporary Dictionary is a comprehensive reference tool designed for the modern reader. It features over 50,000 entries, including the latest words and phrases from contemporary English, as well as classic vocabulary and terminology. The dictionary provides clear and concise definitions, along with pronunciation guides, synonyms, and antonyms. The book's


ADD STUDENT
STUDENT ID:
FIRST NAME:
LAST NAME:
PHONE NUMBER:
EMAIL:
GENDER:
Male
Add Member
Close







6.3 SCREEN OBJECTS AND ACTIONS

Splash Screen – A welcome screen that shows up when the application is started. It gives a visual depiction of the name or logo of the application, and it might also show loading information or other pertinent introduction material.

Start Application Button – A button displayed on the splash screen. Allows the user to initiate the application by clicking the button, which takes them to the login screen.

Log-in Screen – A screen that requests login information from users. To log into the system, users enter their username and password.

Dashboard Screen – After a successful login, you can visit the application's home page. This gives a summary of the options and characteristics of the system. On the dashboard, users can access various functionalities using buttons or menu items.

Add Book Button – A button visible on the dashboard. In order to add a new book to the library catalog, users can enter information about it in a form or modal window by clicking this button.

Edit Book Button – A button visible on the dashboard. In the book management screen or catalog view, each book entry has an edit icon or button next to it.

Issue/Remove Issue Button – A button under circulation management on the dashboard. Used to add a book to a user's account or remove a book that has already been provided. By pressing the corresponding button, the corresponding operation is started, updating both the user's and the book's borrowing records.

Return Book Button – A button under circulation management on the dashboard. It enables customers to give back a rented book. This button's click updates the user's borrowing history as well as the book's availability status.

Lost Book Button – A button displayed on the return book screen. When clicked, a book is marked as lost. This process generates relevant reports, maintains the book's status, and could entail intricate calculations or other related procedures.

Returned Late Button – A button displayed on the return book screen. Shows that a book has been returned after its due date. This button's click may result in the assessment of fines or penalties or the creation of pertinent reports.

Book List Button – A button displayed on the dashboard. Clicking this button takes the user to the book list screen, where they can view all the books in the library catalog.

Search Book Button – A button or search icon displayed on the book list screen. Allows users to search for specific books by entering keywords, book titles, authors, or other relevant search criteria. Initiates the search functionality and displays matching results.

Sort By Function – A drop-down menu or sorting options displayed on the book list screen. Enables users to sort the book list based on different criteria such as title, author, publication date, or availability. Changes the display order of the book list accordingly.

Add Member Button – A button displayed on the dashboard under the user management section. Clicking this button opens a form or modal window where users can enter details about a new member to add them to the system's member database.

Search Student Button – A button or search icon displayed on the member list screen. Allows users to search for specific members by entering keywords, member names, IDs, or other relevant search criteria. Initiates the search functionality and displays matching results.

Edit Member Button – A button displayed on the dashboard under the user

management section. It enables users to edit a particular member's details. The user can edit or amend the existing member information by clicking this button, which opens a form or modal window prefilled with it.

Save Student Details Button – A button displayed on the edit member screen. Allows users to save the updated or newly entered details of a member. Clicking this button updates the member's information in the system's member database.

Member List Button – A button displayed on the dashboard under the user management section. Clicking this button takes the user to the member list screen, where they can view all the registered members in the system.

Delete Member Button – A button displayed under the member list screen. Allows users to delete a specific member from the system's member database. Clicking this button prompts for confirmation and, upon confirmation, removes the member's information from the system.

Print Button – A button or print icon displayed on the dashboard. Clicking this button initiates the printing functionality, allowing users to generate physical copies of the displayed information. The content to be printed can vary based on the context, such as book details, member details, or generated reports.

7.0 REQUIREMENTS MATRIX

Requirement	Component	Input	Output	Relationship	Details
REQ1	User Interface Subsystem	User actions	Displayed information	Interacts with other subsystems	The user interface subsystem receives user actions, such as button clicks or form inputs, and displays the relevant information based on the user's interactions.
REQ2	Catalog Management	Resource details	Updated catalog data	Interacts with User	The catalog management

	Subsystem			Interface Subsystem	subsystem receives resource details from the user interface subsystem and updates the catalog data accordingly.
REQ3	Circulation Subsystem	Loan requests	Loan status and updates	Interacts with User Interface Subsystem	The circulation subsystem processes loan requests received from the user interface subsystem and provides loan status updates based on the user's actions.
REQ4	User Management Subsystem	User registration and login details	User authentication and access control	Interacts with User Interface Subsystem	The user management subsystem handles user registration and login details, verifies user credentials, and controls access to the system's functionalities.
REQ5	Reporting and Analytics Subsystem	System data	Generated reports and analytics	Interacts with other subsystems	The reporting and analytics subsystem gathers data from various subsystems, analyzes it, and generates reports and analytics for decision-making purposes.
REQ6	Database	Data input	Stored data	Supports	The database

		and retrieval		data storage and retrieval for all subsystems	component stores and retrieves data for various subsystems, ensuring data integrity and availability.
REQ7	Printing Functionality	Print button action	Printed documents	Utilizes system data	The printing functionality allows users to print various documents, such as reports, loan receipts, or member details, based on the system data.
REQ8	Search Functionality	Search query	Search results	Interacts with Catalog Management Subsystem	The search functionality receives search queries from users and retrieves relevant search results from the catalog management subsystem based on the query.
REQ9	Sorting Functionality	Sort option selection	Sorted data	Interacts with Catalog Management Subsystem	The sorting functionality allows users to select different sort options for catalog data, and the subsystem retrieves and displays the sorted data accordingly.
REQ10	Member Management	Member details	Updated member	Interacts with User	The member management

			records	Interface Subsystem and Database	component receives member details from the user interface subsystem, updates the member records in the database, and reflects the changes in the user interface.
REQ11	Book Management	Book details	Updated book records	Interacts with User Interface Subsystem and Database	The book management component receives book details from the user interface subsystem, updates the book records in the database, and reflects the changes in the user interface.