Library Management Program Tool

Project Design

Version 11

**Group 1 [Ursula Richardson, John Kucera, Jason Martin]**

University of Maryland Global Campus

CMSC 495 6380 Current Trends and Projects in Computer Science

Prof. Hung Dao

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Revision Number | Date | Description | Author |
| 1 | 1 Feb 2022 | Creation and formatting of this document | John Kucera |
| 2 | 2 Feb 2022 | Added Pseudocode for: GUI, Login Menu, Change Password Menu, Browse All Books Menu, View Checked-Out Books Menu | John Kucera |
| 3 | 5 Feb 2022 | Added Pseudocode for: Admin Settings Menu, Admin Book List Info Page, Admin User List Page, Books Table, and Users Table | Jason Martin |
| 4 | 6 Feb 2022 | Added Sequence and Class Diagrams | Ursula Richardson |
| 5 | 7 Feb 2022 | Restructured Database Tables to include new Checked Out Table and moved Admin Settings page to Main Menu drop down options. | Jason Martin |
| 6 | 7 Feb 2022 | Cleaned up pseudocode in first 5 classes. Added Unresolved Risk section | John Kucera |
| 7 | 7 Feb 2022 | Revised Sequence Diagrams | Ursula Richardson |
| 8 | 15 Feb 2022 | Further Revised Sequence Diagrams | Ursula Richardson |
| 9 | 1 March 2022 | Revised all pseudocode to match the code we have in our current working program. Updated all Java classes added, updated all wording to fit the previous subsystem diagram. | John Kucera |
| 10 | 2 March 2022 | Revised all Sequence Diagrams to match the code we have in our current working program. Updated all wording to fit the previous subsystem diagram. | John Kucera |
| 11 | 5 March 2022 | Revise pseudocode to match finalized code. Added updated class diagram. | John Kucera |

**PROJECT DESIGN**

**Class Diagram**:

(Created using <https://app.diagrams.net/>)

A picture containing text, receipt, screenshot

Description automatically generated

**Sequence Diagrams**:

(Created diagrams using: <http://www.plantuml.com/plantuml/uml/>. Copy-paste the text commands under each diagram into this website in order to re-create that diagram.)

**Scenario 1: Start-Up Scenario**

* Description: The user types correct MySQL credentials in the Output Console. The Login Dialog is displayed. The user then types correct credentials into the Login Dialog and clicks the “Log In” button.
* Precondition: The user has started running the LMP Project and is prompted with MySQL credentials in the Output Console.
* Post-condition: The user has successfully logged in, and the Login Dialog has been disposed of. The GUI (Input Subsystem) Frame is displayed. If the user is an admin, both the “Main Menu” and “Admin Settings” menus are present in the menu bar. Otherwise, only the “Main Menu” menu is present in the menu bar.

A picture containing graphical user interface

Description automatically generated

* PlantUML Text Commands to re-create diagram:

@startuml

!theme plain

IDEConsole -[#0000FF]-> User: Prompt for user's MySQL credentials

User -> IDEConsole: Enter MySQL credentials.

LoginDialog -[#0000FF]-> User: Prompt for user's LMP credentials.

User -> LoginDialog: Type Username + Password,\nClick "Login" Button.

LoginDialog -> UsersTable: Compare input username and password to\nUsers Table to find a match.

UsersTable -[#0000FF]-> LoginDialog: Return Authentication Confirmation,\nUserID, and User Admin Status.

LoginDialog -[#0000FF]-> User: Dispose of Login Dialog. Display main JFrame,\nwith "Admin Settings" menu if user is an Admin.

@enduml

**Scenario 2:** **Shut-Down Scenario**

* Description: User clicks the “Log Out” menu item at top of GUI Frame.
* Precondition: User is logged in and the GUI Frame is displayed.
* Post-condition: User has logged out. The GUI Frame has been disposed of and the Login Dialog is displayed.

Graphical user interface, text, application

Description automatically generated

* PlantUML Text Commands to re-create diagram:

@startuml

!theme plain

User -> GuiFrame: Click "Log Out" Menu Item.

GuiFrame -[#0000FF]-> User: Frame is disposed of.\nLogin Dialog is displayed.

@enduml

**Scenario 3:** **Normal Operation 1 - User browses, views book info, and checks out book.**

* Description: GUI Frame is displayed and user clicks “Main Menu” menu, then clicks “Browse Books” menu item. When Browse Books Panel is displayed in Frame, the user types their desired phrases in title and author text fields, then clicks “Search” button. When search results are displayed in “Results” table, user selects a book and clicks “Show Book Info” button. After viewing the Book Info Dialog, user closes the dialog and clicks “Check Out Book” button.
* Precondition: User is logged in and the GUI Frame is displayed.
* Post-condition: “Browse Books” Panel is displayed with user’s search results in “Results” table. Books Table and Checked-Books Table have been modified from the user’s book check-out.

**(Sequence Diagram on next page)**

Graphical user interface, text, application

Description automatically generated

* PlantUML Text Commands to re-create diagram:

@startuml

!theme plain

User -> GUIFrame: User Clicks "Main Menu" menu, then clicks\n"Browse Books" menu item.

GUIFrame -[#0000FF]-> User: Display Browse Books Panel in Frame.

BrowseBooksPanel -[#0000FF]-> User: Display text fields “title” and “author”\n+ “Search”, "Show Book Info", and "Check Out Book" buttons.

User -> BrowseBooksPanel: User types desired phrases in title and author fields,\nthen clicks "Search" button.

BrowseBooksPanel -> BooksTable: Takes user input and request\nmatching books + information.

BooksTable -[#0000FF]-> BrowseBooksPanel: Return matching books + information.

BrowseBooksPanel -[#0000FF]-> User: In "Results" table, display list of matching\nbooks retrieved from Books Table.

User -> BrowseBooksPanel: User selects a book and clicks "Show Book Info" button.

BrowseBooksPanel -> BooksTable: Request information for selected book.

BooksTable -[#0000FF]-> BrowseBooksPanel: Return information for selected book.

BrowseBooksPanel -[#0000FF]-> User: Display Book Info Dialog containing information\nretrieved from Books Table.

User -> BrowseBooksPanel: User closes Book Info Dialog, then clicks\n"Check Out Book" button.

BrowseBooksPanel -> UsersTable: Request current user's User ID.

UsersTable -[#0000FF]-> BrowseBooksPanel: Return current user's User ID.

BrowseBooksPanel -> CheckedBooksTable: Request addition of selected book to\nchecked-out books of user with stored User ID.

BrowseBooksPanel -> BooksTable: Request decrement of "Quantity\nAvailable" for selected book.

BrowseBooksPanel -[#0000FF]-> User: Check-out confirmation message is displayed.

@enduml

**Scenario 4**: **Normal Operation 2 – User views checked-out books and returns a book.**

* Description: GUI Frame is displayed and user clicks “Main Menu” menu, then clicks “View Checked-Out Books” menu item. When View Checked-Out Books Panel is displayed in Frame, the user selects a book and clicks “Return Book” button. After viewing return confirmation dialog, user closes the dialog.
* Precondition: User is logged in and the GUI Frame is displayed.
* Post-condition: “View Checked-Out Books” Panel is displayed with user’s checked-out books in table, the returned book no longer being present. Books Table and Checked-Books Table have been modified from the user’s book return.

Graphical user interface, application

Description automatically generated

* PlantUML Text Commands to re-create diagram:

@startuml

!theme plain

User -> GUIFrame: User Clicks "Main Menu" menu, then clicks\n"View Checked-Out Books" menu item.

GUIFrame -[#0000FF]-> User: Display View Checked-Out Books Panel in Frame.

ViewCheckedOutBooksPanel -> UsersTable: Request current user's User ID.

UsersTable -[#0000FF]-> ViewCheckedOutBooksPanel: Return current user's User ID.

ViewCheckedOutBooksPanel -> CheckedBooksTable: Request checked-out\nbooks for currently stored user ID.

CheckedBooksTable -[#0000FF]-> ViewCheckedOutBooksPanel: Return user's checked-out\nbooks.

ViewCheckedOutBooksPanel -> BooksTable: Request checked-out books title/author.

BooksTable -[#0000FF]-> ViewCheckedOutBooksPanel: Return checked-out books title/author.

ViewCheckedOutBooksPanel -[#0000FF]-> User: In a table, display user's checked-out books retrieved from\nChecked-Books Table, along with "Return Book" button.

User -> ViewCheckedOutBooksPanel: User selects a book and clicks "Return" button.

ViewCheckedOutBooksPanel -> CheckedBooksTable: Request removal for selected book.

ViewCheckedOutBooksPanel -> BooksTable: Request increment of "Quantity Available" for selected book.

ViewCheckedOutBooksPanel -[#0000FF]-> User: Display return confirmation message.\nRe-display checked-out books with returned book no longer present.

@enduml

**Scenario 5:** **Normal Operation 3 – User changes password.**

* Description: GUI Frame is displayed and user clicks “Main Menu” menu, then clicks “Change Password” menu item. When Change Password Panel is displayed in Frame, the user types correct credentials, a new password, and the new password again for confirmation. The user then clicks the “Confirm Password Change” button.
* Precondition: User is logged in and the GUI Frame is displayed.
* Post-condition: Change Password Panel is displayed in Frame, and user’s password has been changed in Users Table. Confirmation message is displayed.

A picture containing diagram

Description automatically generated

* PlantUML Text Commands to re-create diagram:

@startuml

!theme plain

User -> GUIFrame: User Clicks "Main Menu" menu, then clicks\n"Change Password" menu item.

GUIFrame -[#0000FF]-> User: Display Change Password Panel in Frame.

ChangePasswordPanel -[#0000FF]-> User: Display "Current Password", "New Password",\nand "Confirm New Password" password fields\nwith "Confirm Change" button.

User -> ChangePasswordPanel: User types correct credentials and new password\ninto fields, then clicks "Confirm Change" button.

ChangePasswordPanel -> UsersTable: Compare input current password to\nUsers Table to confirm authentication.

ChangePasswordPanel -> UsersTable: Request input password change for user.

ChangePasswordPanel -[#0000FF]-> User: Display password change confirmation message.

@enduml

**Scenario 6:** **Normal Operation 4 – Admin lists books, edits book info, and removes a book.**

* Description: GUI Frame is displayed and user clicks “Admin Settings” menu, then clicks “List Books” menu item. When List Books Panel is displayed in Frame, the user selects a book and clicks the “Edit Book Info” button. After editing the book information in the “Edit Info” dialog, the user clicks “Confirm edits” button. After closing the dialog, the user selects another book and then clicks “Remove Book” button. The removal confirmation dialog pops up, and the user closes it.
* Precondition: User is logged in and the GUI Frame is displayed. User is an Admin and therefore has access to “Admin Settings” menu.
* Post-condition: List Books Panel is displayed in Frame, one book in the Books Table has been edited, and another book has been removed from Books Table.

Table

Description automatically generated with medium confidence

* PlantUML Text Commands to re-create diagram:

@startuml

!theme plain

User -> GUIFrame: User Clicks "Admin Settings" menu, then clicks\n"List Books" menu item.

GUIFrame -[#0000FF]-> User: Display List Books Panel in Frame.

ListBooksPanel -> BooksTable: Request list of books including information.

BooksTable -[#0000FF]-> ListBooksPanel: Return list of books including information.

ListBooksPanel -[#0000FF]-> User: Display books in a table, along with\n"Edit Book Info" and "Remove Book" buttons.

User -> ListBooksPanel: User selects a book then clicks "Edit Book Info" button.

ListBooksPanel -> BooksTable: Request information for selected book.

BooksTable -[#0000FF]-> ListBooksPanel: Return information for selected book.

ListBooksPanel -[#0000FF]-> User: Display a dialog containing editable\ntext fields with current book information\nand a "Confirm edits" button.

User -> ListBooksPanel: User types edits into dialog, then clicks "Confirm edits" button.

ListBooksPanel -> BooksTable: Request input edits for selected book.

ListBooksPanel -[#0000FF]-> User: Display edit confirmation message in dialog.

User -> ListBooksPanel: User closes dialog, then selects another book.\nUser clicks "Remove Book" button.

ListBooksPanel -> BooksTable: Request removal of selected book.

ListBooksPanel -[#0000FF]-> User: Display removal confirmation dialog.

User -> ListBooksPanel: User closes dialog.

@enduml

**Scenario 7: Normal Operation 5 – Admin adds book**

* Description: GUI Frame is displayed and user clicks “Admin Settings” menu, then clicks “Add Book” menu item. When Add Book Panel is displayed in Frame, the user types in information for new book. User then clicks “Add Book” button. The addition confirmation dialog pops up, and the user closes it.
* Precondition: User is logged in and the GUI Frame is displayed. User is an Admin and therefore has access to “Admin Settings” menu.
* Post-condition: Add Book Panel is displayed in Frame, and one book has been added to Books Table.

Diagram

Description automatically generated

* PlantUML Text Commands to re-create diagram:

@startuml

!theme plain

User -> GUIFrame: User Clicks "Admin Settings" menu, then clicks\n"Add Book" menu item.

GUIFrame -[#0000FF]-> User: Display Add Book Panel in Frame.

AddBookPanel -[#0000FF]-> User: Display editable text fields to hold new\nbook information and a "Add Book" button.

User -> AddBookPanel: User types in information for new\nbook and clicks "Add Book" button.

AddBookPanel -> BooksTable: Request addition of new\nbook with input info.

AddBookPanel -[#0000FF]-> User: Display addition confirmation message in dialog.

User -> AddBookPanel: User closes dialog.

@enduml

**Scenario 8:** **Normal Operation 6 – Admin lists users and removes a user.**

* Description: GUI Frame is displayed and user clicks “Admin Settings” menu, then clicks “List Users” menu item. When List Users Panel is displayed in Frame, the user selects a user and clicks the “Remove User” button. The removal confirmation dialog pops up, and the user closes it.
* Precondition: User is logged in and the GUI Frame is displayed. User is an Admin and therefore has access to “Admin Settings” menu.
* Post-condition: List Users Panel is displayed in Frame and one user has been removed from Users Table.

Diagram

Description automatically generated

* PlantUML Text Commands to re-create diagram:

@startuml

!theme plain

User -> GUIFrame: User Clicks "Admin Settings" menu,\nthen clicks "List Users" menu item.

GUIFrame -[#0000FF]-> User: Display List Users Panel in Frame.

ListUsersPanel -> UsersTable: Request list of users\nincluding information.

UsersTable -[#0000FF]-> ListUsersPanel: Return list of users\nincluding information.

ListUsersPanel -[#0000FF]-> User: Display users in a table, along with\n"Remove User" button.

User -> ListUsersPanel: User selects a user and clicks "Remove User" button.

ListUsersPanel -> UsersTable: Request removal\nof selected user.

ListUsersPanel -[#0000FF]-> User: Display removal confirmation dialog.

User -> ListUsersPanel: User closes dialog.

@enduml

**Scenario 9: Normal Operation 7 – Admin adds user**

* Description: GUI Frame is displayed and user clicks “Admin Settings” menu, then clicks “Add User” menu item. When Add User Panel is displayed in Frame, the user types in information for new user. User then clicks “Add User” button. The addition confirmation dialog pops up, and the user closes it.
* Precondition: User is logged in and the GUI Frame is displayed. User is an Admin and therefore has access to “Admin Settings” menu.
* Post-condition: Add User Panel is displayed in Frame, and one user has been added to Users Table.

Diagram

Description automatically generated

* PlantUML Text Commands to re-create diagram:

@startuml

!theme plain

User -> GUIFrame: User Clicks "Admin Settings" menu, then clicks\n"Add User" menu item.

GUIFrame -[#0000FF]-> User: Display Add User Panel in Frame.

AddUserPanel -[#0000FF]-> User: Display editable text fields to hold new\nuser information and a "Add User" button.

User -> AddUserPanel: User types in information for new\nuser and clicks "Add User" button.

AddUserPanel -> UsersTable: Request addition of new\nuser with input info.

AddUserPanel -[#0000FF]-> User: Display addition confirmation message in dialog.

User -> AddUserPanel: User closes dialog.

@enduml

**Scenario 10: Error-handling 1 – User inputs invalid username and/or password.**

* Description: Two times, user types incorrect credentials (first username and then password) into the Login Dialog and clicks the “Log In” button.
* Precondition: The user has opened the LMP project in their IDE and has successfully built and started running the project. The Login Dialog is displayed on the screen, awaiting user input.
* Post-condition: The user has not logged in, and the Login Dialog is still displayed with a denial message.

Graphical user interface, text, application

Description automatically generated

* PlantUML Text Commands to re-create diagram:

@startuml

!theme plain

User -> LoginDialog: Type Incorrect Username + Correct Password,\nClick "Login" Button.

LoginDialog -> UsersTable: Compare input username and password to\nUsers Table, finds no username match.

UsersTable -[#0000FF]-> LoginDialog: Return Authentication Denial.

LoginDialog -[#0000FF]-> User: Display message notifying user of\nincorrect username and/or password.

User -> LoginDialog: Type Correct Username + Incorrect Password,\nClick "Login" Button.

LoginDialog -> UsersTable: Compare input username and password to\nUsers Table, finds incorrect password.

UsersTable -[#0000FF]-> LoginDialog: Return Authentication Denial.

LoginDialog -[#0000FF]-> User: Display message notifying user of\nincorrect username and/or password.

@enduml

**Scenario 11: Error-handling 2 – User attempts to check out unavailable book.**

* Description: GUI Frame is displayed containing the Browse Books Panel. Search terms have been searched for, displaying matching books in the “Results” table. User selects a book and clicks “Check Out Book”. Since the “Quantity Available” = 0 for that book, a message is displayed that notifies user that book is unavailable and cannot be checked out.
* Precondition: User is logged in and the GUI Frame is displayed containing the Browse Books Panel. Search terms have been searched for, displaying matching books in the “Results” table.
* Post-condition: “Browse Books” Panel is displayed with user’s search results in “Results” table. No check-out has been done.

Diagram

Description automatically generated

* PlantUML Text Commands to re-create diagram:

@startuml

!theme plain

User -> BrowseBooksPanel: User selects book and\nclicks "Check Out Book" button.

BrowseBooksPanel -> BooksTable: Request decrement of "Quantity\nAvailable" for selected book.

BooksTable -[#0000FF]-> BrowseBooksPanel: Rejects request because "Quantity\nAvailable" for book is 0.

BrowseBooksPanel -[#0000FF]-> User: Check-out denial\nmessage is displayed.

@enduml

**Scenario 12: Error-handling 3 – User enters incorrect input when changing password.**

* Description: GUI Frame is displayed and contains Change Password Panel. The user makes two attempts to change password. First, user enters incorrect current password and clicks “Confirm Change” button. Denial message is displayed. Second, user enters mismatched “new password” and “confirm new password” input and clicks “Confirm Change” button. Denial message is displayed.
* Precondition: User is logged in and the GUI Frame is displayed containing Change Password Panel.
* Post-condition: Change Password Panel is displayed in Frame, containing a denial message. User’s password has not been changed in Users Table.

Text, letter

Description automatically generated

* PlantUML Text Commands to re-create diagram:

@startuml

!theme plain

User -> ChangePasswordPanel: User types incorrect "current password",\nbut matching "new password"\nand "confirm new password" input into fields,\nthen clicks "Confirm Change" button.

ChangePasswordPanel -> UsersTable: Request password change for user,\ncompare input "current password" to\nUsers Table to confirm authentication.

UsersTable -[#0000FF]-> ChangePasswordPanel: Find that input is incorrect,\nreturns Authentication denial.

ChangePasswordPanel -[#0000FF]-> User: Display password change denial message\ndue to incorrect "current password".

User -> ChangePasswordPanel: User types correct "current password",\nbut mismatched "new password"\nand "confirm new password" input into fields,\nthen clicks "Confirm Change" button.

ChangePasswordPanel -[#0000FF]-> User: Display password change denial message\ndue to mismatched "new password"\nand "confirm new password".

@enduml

**Scenario 13: Error-handling 4 – User attempts to remove themself from User List.**

* Description: GUI Frame is displayed and contains List Users Panel. The user selects their own username and clicks “Remove User” button. As a result, a denial message is displayed in a dialog.
* Precondition: User is logged in and the GUI Frame is displayed containing List Users Panel.
* Post-condition: List Users Panel is displayed in Frame, with a dialog containing a denial message. **User has not been removed from Users Table.**

Graphical user interface, text, application, email

Description automatically generated

* PlantUML Text Commands to re-create diagram:

@startuml

!theme plain

User -> ListUsersPanel: User selects their own username and clicks "Remove User" button.

ListUsersPanel -[#0000FF]-> User: Display dialog containing denial message.

@enduml

**Pseudocode**: (Note: Class DatabaseInitialization has details on how data is retrieved from MySQL Database)

* **GUI (Input Subsystem)**:

Class Gui extends JFrame {

Create “mainPanel” JPanel;

Create Strings “DB\_URL”, “USER”, “PASS”

Constructor Gui() {

Create JFrame components and characteristics;

* + - * Create “menuBar” JMenuBar, containing “mainMenu” and “adminMenu” JMenus.

On Open:

int CurrentUserID

Bool AdminStatus

SELECT AdminStatus

FROM LMP\_Users

WHERE UserID = CurrentUserID;

If AdminStatus != 1:

Create JFrameDialog “Insufficient Rights to Access”

Else:

Load Respective Admin Panel Constructor

* + - * + When “mainMenu” is Pressed:

Display drop-down options, including:

“Browse Books” When Pressed:

repaintMain(1);

“View Checked-Out Book” When Pressed:  
repaintMain(2);

“Change Password” When Pressed:  
repaintMain(3);

“Log Out” When Pressed:

clearGUI();

new Login();

* + - * + When “adminMenu” is Pressed:

Display drop-down options, including:

“List Books” When Pressed:  
repaintMain(4);

“Add Book” When Pressed:  
repaintMain(5);

“List Users” When Pressed:  
repaintMain(6);

“Add User” When Pressed:  
repaintMain(7);

* + - * setJMenuBar(menuBar);
      * add(mainPanel);
      * mainPanel.add(new JLabel(“Welcome Message”));
      * Set Exit on Close button, Title, Non-resizable, Size, Location, Visible

}

Create 8 Private Listener Classes corresponding to the 8 menu items;

Method void repaintMain(int item) {

clearGUI();

switch (item) {

case 1: mainPanel.add(new BrowseBooks());

case 2: mainPanel.add(new ViewChecked());

case 3: mainPanel.add(new ChangePassword());

case 4: mainPanel.add(new AdminBookList());

case 5: mainPanel.add(new AdminAddBook());

case 6: mainPanel.add(new AdminUserList());

case 7: mainPanel.add(new AdminAddUser());

}

mainPanel.repaint();

}

Method void clearGUI() {

repaint mainPanel;

}

Getter methods for MySQL credentials: DB\_URL, USER, PASS.

Method Boolean isInteger(String string) {

if (Integer.parseInt(string) is an Integer) {

return true;

}

else {

return false;

}

}

Void main() {

Create Scanner mysqlIn;

Print MySQL credentials prompt in Output Console;

Scan for user input, store into USER and PASS;

Database Initialization;

Test Data Population;

new Login();

}

}

* **Login Dialog**:

Class Login extends JDialog {

Create “Username” JTextField;

Create “Password” JPasswordField;

Create “denial” JLabel;

Create “CURRENT\_USERNAME” String;

Constructor Login() {

Create Login Dialog components;

* + - * Create Header, Username, Password JLabels
      * Create “Log in” JButton When Pressed:

Authenticate input username and password by comparing to Users Table;

IF user is valid:

Store user in CURRENT\_USERNAME;

dispose dialog;

IF user is invalid:

display “denial” JLabel

* + - * Organize Swing components with GridBagLayout and BorderLayout
      * Dialog Window listener: Exit program upon closing Login Dialog
      * Set modality, title, non-resizable, size, location, visible

}

}

* **Change Password Panel**:

Class ChangePassword extends JPanel {

Create “current”, “new”, “confirm new” JPasswordFields;

Create “denial” JLabel;

Constructor ChangePassword() {

Create Change Password JPanel components;

* + - * Create Header, “current”, “new”, “confirm new” JLabels;
      * Create “Confirm Password Change” JButton When Pressed:

Connect to MySQL database

authenticate input by comparing to Users Table;  
if (input is valid) {

add new information to Users Table;

display confirm message in “denial” JLabel;

}

else {

display denial message in “denial” JLabel;

}

clearFields();

* + - * Organize Swing components with GridBagLayout and BorderLayout

}

}

* **Browse All Books Panel**:

Class BrowseBooks extends JPanel {

Database Initialization;

Create “SearchQuery” JTextField;

Create “denial” JLabel;

Create “Check Out Book” JButton

* + - * + Enabled ONLY IF a table row is selected
        + When Pressed:

If (selectedBook.getBookQuantity() > 0) {

SELECT \* from Books Table where BookName = selectedBook

UPDATE Books Table SET AmountInStock+=1

SELECT UserID from Users Table where UserName = CURRENT\_USERNAME

INSERT INTO Checked-Books Table VALUES (CheckOutNumber, Current Date, Book Name, User ID)

Set “denial” to confirmation message;

}

else {

Set “denial” to unavailable message;

}

Create “Show Book Info” JButton;

* + - * + Enabled ONLY IF a table row is selected
        + When Pressed:

new showBookInfo();

Constructor BrowseBooks() {

Create Browse All Books JPanel components;

* + - * Create Header, “title/author”, “results” JLabels;
      * Create “Search” JButton When Pressed:

Compare input from “Title/Author” JTextfield to Books Table;

Store matching results in ResultSet;  
“Results”.addRow(ResultSet(Title, Author))

* + - * Create “Results” JTable
        + Columns: Title, Author
        + setSelectionModel(new ForcedListSelectionModel());
      * Organize Swing components with GridBagLayout and BorderLayout

Class RowSelectionListener: if row is selected, enable “Show Book Info” and “Check Out Book” JButtons

}

Class ShowBookInfo extends JDialog{

Database Initialization;

Create “title”, “author”, “isbn”, “quantity total”, “quantity available” JTextFields (set un-editable);

Constructor ShowBookInfo() {

Create Dialog components:

* Create “title”, “author”, “isbn”, “quantity total”, “quantity available” JLabels;
* Retrieve data from Books Table to display the book information
* Organize Swing components with GridBagLayout
* Set modality, title, non-resizable, size, location, visible.

}

* **View Checked-Out Books Panel**:

Class ViewChecked extends JPanel {

Database Initialization;

Create “Return Book” JButton

* + - Enabled ONLY IF a table row is selected
    - When Pressed:

Remove book from CheckedBooks Table;  
Increment book quantity available in Books Table;  
Create JDialog containing “Return confirmation” JLabel;  
On Dialog close: re-paint “Checked-out books” JTable;

Constructor ViewChecked() {

Create View Checked-Out Books JPanel components:

* + - * Create Header JLabel;
      * Create “Checked-out books” JTable;
        + Columns: Title, Author, Check-Out date, Check-Out Period
        + setSelectionModel(new ForcedListSelectionModel());
      * Retrieve UserID of CURRENT\_USERNAME from Users Table
      * In JTable, display books retrieved from CheckedBooks Table that have currently stored User ID
      * Organize Swing components with GridBagLayout and BorderLayout

}

Class RowSelectionListener: if row is selected, enable “Return Book” JButton

}

* **(Admin Settings) List Books Panel:**

Class AdminBookList extends JPanel {

Create “Edit Book Info” JButton

* + - Enabled ONLY IF a table row is selected
    - When Pressed:  
      new EditBookInfo(selectedBook);

Create “Remove Book” JButton

* + - Enabled ONLY IF a table row is selected
    - When Pressed:

Remove book from Books Table;  
Create JDialog containing “Removal Confirmation” JLabel;

Constructor AdminBookList() {

Create AdminBookList JPanel components:

* + - * Create Header JLabel;
      * Create “All Books” JTable;
        + Columns: Title, Author
        + setSelectionModel(new ForcedListSelectionModel());
      * In JTable, display data retrieved from Books Table
      * Organize Swing components with GridBagLayout and BorderLayout

}

Class RowSelectionListener: if row is selected, enable “Remove Book” JButton

Class EditBookInfo extends JDialog{

Create “title”, “author”, “isbn”, “quantity total”, “quantity available” JTextFields (set editable);

Create “denial” JLabel;

Constructor EditBookInfo(String bookNameIn) {

SELECT “title”, “author”, “isbn”, “quantity total”, “quantity available” from Books Table where BookName = selectedBook

Create Dialog components:

* + Create “title”, “author”, “isbn”, “quantity total”, “quantity available” JLabels;
  + Display Book info retrieved from Books Table in text fields
  + Create “Confirm Edits” button.
    - When Pressed:  
      if (!isInteger(quantity total || quantity available)) {

Set “Denial” JLabel to notify that quantities must be integers

} else {

Update Books Table with user input in text fields;

Set “Denial” JLabel to show confirmation; }

* + Organize Swing components with GridBagLayout
  + Set modality, title, non-resizable, size, location, visible.

}

}

}

* **(Admin Settings) Add Book Panel:**

Class AdminAddBook extends JPanel {

Database Initialization;

Create “title”, “author”, “isbn”, “quantity total”, “quantity available” JTextFields;

Constructor ShowBookInfo() {

Create Dialog components:

* Create “title”, “author”, “isbn”, “quantity total”, “quantity available” JLabels;
* Create “Add Book” JButton
  + - When Pressed:  
      if (!isInteger(quantity total || quantity available)) {

Create JDialog containing “Denial” JLabel, notifying that quantities must be integers

} else {

Add book with input info to Books Table;  
Create JDialog containing “Addition Confirmation” JLabel; }

* Organize Swing components with GridBagLayout

}

* **(Admin Settings) List Users Panel:**

Class AdminUsersList extends JPanel {

Create “Remove User” JButton

* + - Enabled ONLY IF a table row is selected
    - When Pressed:

if (CURRENT\_USERNAME = selectedUser) {

Create Dialog notifying that user cannot remove themself

} else {

Remove book from Users Table;  
Create JDialog containing “Removal Confirmation” JLabel; }

Constructor AdminUserList() {

Create AdminUserList JPanel components:

* + - * Create Header JLabel;
      * Create “All Users” JTable;
        + Columns: Username, Name
        + setSelectionModel(new ForcedListSelectionModel());
      * In JTable, display data retrieved from Users Table
      * Organize Swing components with GridBagLayout and BorderLayout

}

Class RowSelectionListener: if row is selected, enable “Remove User” JButton

}

* **(Admin Settings) Add User Panel:**

Class AdminAddUserextends JPanel {

Database Initialization;

Create “first name”, “last name”, “username”, “password” JTextFields;

Create “admin” JCheckBox;

Constructor ShowBookInfo() {

Create Dialog components:

* Create “first name”, “last name”, “username”, “password”, “admin?” JLabels;
* Create “Add User” JButton
  + - When Pressed:

Add user with input info to Users Table;  
Create JDialog containing “Addition Confirmation” JLabel;

* Organize Swing components with GridBagLayout

}

* **Additional Java Classes**:

Class ForcedListSelectionModel extends DefaultListSelectionModel {

(Initially has no table rows selected. Once a table row is selected, only a single row may be selected at once, and it cannot be deselected unless a different row is selected.)

Overrides all necessary methods: clearSelection() and removeSelectionInterval();

}

Class DatabaseInitialization {

Method void initializeDatabase() {

Connection =DriverManager.getConnection(DB\_URL,USER,PASS)

Statement = connection.createStatement();

statement.executeQuery(Check if Books Table is created)

if (Books Table does not exist) {

CREATE TABLE LMP\_Books(ISBN VARCHAR(17), BookName VARCHAR(255), AmountInStock int, AmountOwned int, BookAuthor varchar(255))

}

statement.executeQuery(Check if Users Table is created)

if (Users Table does not exist) {

CREATE TABLE LMP\_Users(UserName varchar(255), UserPassword varchar(255), FirstName varchar(255),

LastName varchar(255), AdminStatus bool)

}

statement.executeQuery(Check if CheckedBooks Table is created)

if (CheckedBooks Table does not exist) {

CREATE TABLE LMP\_CheckedBooks(CheckOutDate DATE, PRIMARY KEY (CheckOutNumber), ISBN varchar(17), UserID int, FOREIGN KEY(ISBN) REFERENCES LMP\_Books(ISBN), FOREIGN KEY(UserID) REFERENCES LMP\_Users(UserID))

}

}

Method void populateTestData() {

Connection =DriverManager.getConnection(DB\_URL,USER,PASS)

Statement = connection.createStatement();

statement.executeQuery(SELECT Users Table);

if (Users Table has no data) {

INSERT INTO Users Table VALUES (Example Users)

}

statement.executeQuery(SELECT Books Table);

if (Books Table has no data) {

INSERT INTO Books Table VALUES (Example Books)

}

statement.executeQuery(SELECT CheckedBooks Table);

if (CheckedBooks Table has no data) {

INSERT INTO CheckedBooks VALUES (Example Checked-Out Books)

}

}

Method void insertBOOK(String BookName, String BookAuthor, String ISBN, intAmountInStock, intAmountOwned) {

Connection =DriverManager.getConnection(DB\_URL,USER,PASS)

Statement = connection.createStatement();

statement.executeQuery(INSERT INTO Books Table (parameters));

}

Method void ADD\_user(String first, String last, String user, String password, Boolean adminStatus) {

Connection =DriverManager.getConnection(DB\_URL,USER,PASS)

Statement = connection.createStatement();

statement.executeQuery(INSERT INTO Users Table (parameters));

}

Method Boolean CurrentUserAdminStatus(String username) {

Connection =DriverManager.getConnection(DB\_URL,USER,PASS)

Statement = connection.createStatement();

statement.executeQuery(SELECT AdminStatus from Users Table WHERE Username = CURRENT\_USERNAME);

return AdminStatus;

}

}

**Unresolved Risks and Risk Mitigation**:

We do not identify any fixes to our risks in the design, and so the previously identified risks remain the same. They are:

* The program might not function efficiently on a system with older and weaker components – for example, a slow CPU, a RAM with little memory, a hard disk that full of other programs that slow the system down. **Possible Risk Mitigation**: Acquire and install components that better fit the intended operating platform, indicated in Project Plan. Alternatively, find a different system entirely that is closer to the intended operating platform.
* Another possible risk is the leaking of authentication information for users. **Possible Risk Mitigation**: Protect the username/password database with its own password to enter. Encrypt all usernames and passwords so they are unreadable by the human eye.
* The system doesn’t currently have any built-in way to back up its data and if the system were to become corrupted all its data would be lost. **Possible Risk Mitigation**: Add an automatic back-up of each of the database tables that could be used to recover data.
* The ability to create or remove people as admins is shared by all admin accounts. While convenient, this means that any singular admin account can completely lock out every other account. **Possible Risk Mitigation**: Add a management account to handle the creation of Admin accounts to prevent malicious users from locking out other admins.
* Since we are using a SQL database, SQL injection is a possible intrusion method into our data particularly the database’s Users Table. **Possible Risk Mitigation**: A mitigation for this would be ensuring all data be used in queries is parameterized and has been input validated before being acted upon.
* Currently the system only uses a password for authenticating the user. **Possible Risk Mitigation**: Adding a multifactor authentication system that could use a user’s library card or email address to confirm the user’s authenticity and increase account security.