Table of Contents

[Question 1 2](#_Toc162299553)

[AddBookController.java 2](#_Toc162299554)

[Book.java 4](#_Toc162299555)

[DeleteBookController.java 5](#_Toc162299556)

[EditBookController.java 7](#_Toc162299557)

[HelloApplication.java 10](#_Toc162299558)

[HelloController.java 11](#_Toc162299559)

[SearchBooksController.java 12](#_Toc162299560)

[ViewAllBooksController.java 14](#_Toc162299561)

[addBook.fxml 16](#_Toc162299562)

[deleteBook.fxml 17](#_Toc162299563)

[editBook.fxml 18](#_Toc162299564)

[hello-view.fxml 19](#_Toc162299565)

[searchBooks.fxml 21](#_Toc162299566)

[viewAllBooks.fxml 22](#_Toc162299567)

[Question 2 22](#_Toc162299568)

[Main.java 22](#_Toc162299569)

[Question 3 24](#_Toc162299570)

[UDPClient.java 24](#_Toc162299571)

[UDPClient2.java 25](#_Toc162299572)

[UDPServer.java 25](#_Toc162299573)

# Question 1

## AddBookController.java

package org.example.i222651;  
  
import javafx.fxml.FXML;  
import javafx.scene.control.Alert;  
import javafx.scene.control.TextField;  
import java.util.regex.Pattern;  
import java.util.ArrayList;  
import java.util.List;  
import java.io.BufferedReader;  
import java.io.BufferedWriter;  
import java.io.FileReader;  
import java.io.FileWriter;  
import java.io.IOException;  
  
public class AddBookController {  
 private List<Book> bookList;  
  
 @FXML  
 private TextField bookTitleField;  
  
 @FXML  
 private TextField authorField;  
  
 @FXML  
 private TextField isbnField;  
  
 @FXML  
 private void initialize() {  
 // Load existing books from file  
 loadBooksFromFile();  
 }  
  
 @FXML  
 private void onAddBookButtonClick() {  
 String title = bookTitleField.getText();  
 String author = authorField.getText();  
 String isbn = isbnField.getText();  
  
 if (title.isEmpty() || author.isEmpty() || !Pattern.*matches*("^(97(8|9))?[0-9]{9}([0-9]|X)$", isbn)) {  
 showAlert(Alert.AlertType.*ERROR*, "Error", "Invalid input or ISBN");  
 } else {  
 // Add new book to the list  
 Book newBook = new Book(title, author, isbn);  
 bookList.add(newBook);  
  
 // Save the updated list to file  
 saveBooksToFile();  
  
 // Show success message  
 showAlert(Alert.AlertType.*INFORMATION*, "Success", "Book added successfully!");  
  
 // Clear input fields  
 clearFields();  
 }  
 }  
  
 private void loadBooksFromFile() {  
 // Initialize bookList if null  
 if (bookList == null) {  
 bookList = new ArrayList<>();  
 } else {  
 bookList.clear(); // Clear existing books before loading  
 }  
  
 try (BufferedReader reader = new BufferedReader(new FileReader("books.txt"))) {  
 String line;  
 while ((line = reader.readLine()) != null) {  
 String[] parts = line.split(",");  
 if (parts.length == 4) {  
 String title = parts[0];  
 String author = parts[1];  
 String isbn = parts[2];  
 boolean available = Boolean.*parseBoolean*(parts[3]);  
 bookList.add(new Book(title, author, isbn));  
 }  
 }  
 } catch (IOException e) {  
 // Handle file read error  
 e.printStackTrace();  
 }  
 }  
  
 private void saveBooksToFile() {  
 try (BufferedWriter writer = new BufferedWriter(new FileWriter("books.txt"))) {  
 for (Book book : bookList) {  
 writer.write(book.getTitle() + "," + book.getAuthor() + "," +  
 book.getIsbn() + "," + book.isAvailable() + "\n");  
 }  
 } catch (IOException e) {  
 // Handle file write error  
 e.printStackTrace();  
 }  
 }  
  
 private void showAlert(Alert.AlertType alertType, String title, String message) {  
 Alert alert = new Alert(alertType);  
 alert.setTitle(title);  
 alert.setHeaderText(null);  
 alert.setContentText(message);  
 alert.showAndWait();  
 }  
  
 private void clearFields() {  
 bookTitleField.clear();  
 authorField.clear();  
 isbnField.clear();  
 }  
  
 public List<Book> getBookList() {  
 return bookList;  
 }  
}

## Book.java

package org.example.i222651;  
  
import java.util.regex.Pattern;  
  
public class Book {  
 private String title;  
 private String author;  
 private String isbn;  
 private boolean availability;  
  
 public Book(String title, String author, String isbn) {  
 this.title = title;  
 this.author = author;  
 this.isbn = isbn;  
 this.availability = true;  
 }  
  
 public Book(String title, String author, String isbn, boolean availability) {  
 this.title = title;  
 this.author = author;  
 this.isbn = isbn;  
 this.availability = availability;  
 }  
  
 public String getTitle() {  
 return title;  
 }  
  
 public void setTitle(String title) {  
 this.title = title;  
 }  
  
 public String getAuthor() {  
 return author;  
 }  
  
 public void setAuthor(String author) {  
 this.author = author;  
 }  
  
 public String getIsbn() {  
 return isbn;  
 }  
  
 public void setIsbn(String isbn) {  
 this.isbn = isbn;  
 }  
  
 public boolean isAvailable() {  
 return availability;  
 }  
  
 public void setAvailable(boolean availability) {  
 this.availability = availability;  
 }  
}

## DeleteBookController.java

package org.example.i222651;  
  
import javafx.fxml.FXML;  
import javafx.scene.control.\*;  
import javafx.collections.ObservableList;  
import javafx.collections.FXCollections;  
import javafx.scene.control.SelectionMode;  
import javafx.scene.control.TableColumn;  
import javafx.scene.control.TableView;  
import javafx.scene.control.cell.PropertyValueFactory;  
import java.io.BufferedReader;  
import java.io.FileReader;  
import java.io.IOException;  
import java.io.BufferedWriter;  
import java.io.FileWriter;  
  
public class DeleteBookController {  
 @FXML  
 private TableView<Book> bookTableView; // Corrected variable name  
  
 private ObservableList<Book> bookList;  
  
 public void setBookList(ObservableList<Book> bookList) {  
 this.bookList = bookList;  
 }  
  
 @FXML  
 private void initialize() {  
 // Initialize bookList if it's null  
 if (bookList == null) {  
 bookList = FXCollections.*observableArrayList*();  
 }  
  
 // Load existing books from file  
 loadBooksFromFile();  
  
 // Clear existing columns to avoid duplicates  
 bookTableView.getColumns().clear();  
  
 // Set up columns in the TableView  
 TableColumn<Book, String> titleColumn = new TableColumn<>("Title");  
 titleColumn.setCellValueFactory(new PropertyValueFactory<>("title"));  
 titleColumn.setMinWidth(219);  
  
 TableColumn<Book, String> authorColumn = new TableColumn<>("Author");  
 authorColumn.setCellValueFactory(new PropertyValueFactory<>("author"));  
 authorColumn.setMinWidth(150);  
  
 TableColumn<Book, String> isbnColumn = new TableColumn<>("ISBN");  
 isbnColumn.setCellValueFactory(new PropertyValueFactory<>("isbn"));  
 isbnColumn.setMinWidth(100);  
  
 TableColumn<Book, Boolean> availabilityColumn = new TableColumn<>("Availability");  
 availabilityColumn.setCellValueFactory(new PropertyValueFactory<>("available"));  
 availabilityColumn.setMinWidth(100);  
  
 bookTableView.getColumns().addAll(titleColumn, authorColumn, isbnColumn, availabilityColumn);  
  
 // Enable sorting for columns  
 titleColumn.setSortable(true);  
 authorColumn.setSortable(true);  
 isbnColumn.setSortable(true);  
  
 // Allow single selection mode  
 bookTableView.getSelectionModel().setSelectionMode(SelectionMode.*SINGLE*);  
 }  
  
 @FXML  
 private void deleteSelectedBook() {  
 Book selectedBook = bookTableView.getSelectionModel().getSelectedItem();  
 if (selectedBook != null) {  
 // Remove the book from the list  
 bookList.remove(selectedBook);  
  
 // Update the text file  
 updateBooksFile();  
  
 showAlert(Alert.AlertType.*INFORMATION*, "Delete Book", "Book deleted successfully.");  
 } else {  
 showAlert(Alert.AlertType.*ERROR*, "Error", "Please select a book to delete.");  
 }  
 }  
  
 private void updateBooksFile() {  
 try (BufferedWriter writer = new BufferedWriter(new FileWriter("books.txt"))) {  
 for (Book book : bookList) {  
 String line = book.getTitle() + "," + book.getAuthor() + "," + book.getIsbn() + "," + book.isAvailable();  
 writer.write(line);  
 writer.newLine();  
 }  
 } catch (IOException e) {  
 // Handle file write error  
 e.printStackTrace();  
 }  
 }  
  
 private void loadBooksFromFile() {  
 bookList.clear(); // Clear existing books before loading  
  
 try (BufferedReader reader = new BufferedReader(new FileReader("books.txt"))) {  
 String line;  
 while ((line = reader.readLine()) != null) {  
 String[] parts = line.split(",");  
 if (parts.length == 4) {  
 String title = parts[0];  
 String author = parts[1];  
 String isbn = parts[2];  
 boolean available = Boolean.*parseBoolean*(parts[3]);  
 bookList.add(new Book(title, author, isbn, available));  
 }  
 }  
 } catch (IOException e) {  
 // Handle file read error  
 e.printStackTrace();  
 }  
  
 // Refresh the TableView to display the updated book list  
 bookTableView.setItems(bookList);  
 }  
  
 private void showAlert(Alert.AlertType alertType, String title, String message) {  
 Alert alert = new Alert(alertType);  
 alert.setTitle(title);  
 alert.setHeaderText(null);  
 alert.setContentText(message);  
 alert.showAndWait();  
 }  
}

## EditBookController.java

package org.example.i222651;  
  
import javafx.collections.FXCollections;  
import javafx.collections.ObservableList;  
import javafx.fxml.FXML;  
import javafx.scene.control.\*;  
import javafx.scene.control.cell.PropertyValueFactory;  
import javafx.scene.control.SelectionMode;  
import java.io.BufferedReader;  
import java.io.FileReader;  
import java.io.IOException;  
import java.io.BufferedWriter;  
import java.io.FileWriter;  
  
public class EditBookController {  
 @FXML  
 private TableView<Book> bookTableView;  
  
 private ObservableList<Book> bookList = FXCollections.*observableArrayList*();  
  
 @FXML  
 private TextField titleField;  
  
 @FXML  
 private TextField authorField;  
  
 @FXML  
 private TextField isbnField;  
  
 @FXML  
 private TextField availabilityField;  
  
 @FXML  
 private void initialize() {  
 // Load existing books from file  
 loadBooksFromFile();  
  
 // Set the book list to the TableView  
 bookTableView.setItems(bookList);  
  
 // Set up columns defined in FXML file  
 TableColumn<Book, String> titleColumn = (TableColumn<Book, String>) bookTableView.getColumns().get(0);  
 TableColumn<Book, String> authorColumn = (TableColumn<Book, String>) bookTableView.getColumns().get(1);  
 TableColumn<Book, String> isbnColumn = (TableColumn<Book, String>) bookTableView.getColumns().get(2);  
 TableColumn<Book, Boolean> availabilityColumn = (TableColumn<Book, Boolean>) bookTableView.getColumns().get(3);  
  
 // Set cell factories to display data in columns  
 titleColumn.setCellValueFactory(new PropertyValueFactory<>("title"));  
 authorColumn.setCellValueFactory(new PropertyValueFactory<>("author"));  
 isbnColumn.setCellValueFactory(new PropertyValueFactory<>("isbn"));  
 availabilityColumn.setCellValueFactory(new PropertyValueFactory<>("available"));  
  
 // Bind selected book's details to text fields for editing  
 bookTableView.getSelectionModel().selectedItemProperty().addListener((observable, oldValue, newValue) -> {  
 if (newValue != null) {  
 titleField.setText(newValue.getTitle());  
 authorField.setText(newValue.getAuthor());  
 isbnField.setText(newValue.getIsbn());  
 availabilityField.setText(newValue.isAvailable() ? "True" : "False");  
 }  
 });  
  
 // Allow single selection mode  
 bookTableView.getSelectionModel().setSelectionMode(SelectionMode.*SINGLE*);  
 }  
  
 @FXML  
 private void saveChanges() {  
 Book selectedBook = bookTableView.getSelectionModel().getSelectedItem();  
 if (selectedBook != null) {  
 // Update selected book's details with values from text fields  
 selectedBook.setTitle(titleField.getText());  
 selectedBook.setAuthor(authorField.getText());  
 selectedBook.setIsbn(isbnField.getText());  
 selectedBook.setAvailable(Boolean.*parseBoolean*(availabilityField.getText()));  
  
 // Save the updated list to file  
 saveBooksToFile();  
  
 // Show success message  
 showAlert(Alert.AlertType.*INFORMATION*, "Success", "Book details updated successfully!");  
  
 // Refresh the TableView after saving changes  
 bookTableView.refresh();  
 } else {  
 showAlert(Alert.AlertType.*ERROR*, "Error", "Please select a book to edit.");  
 }  
 }  
  
 private void loadBooksFromFile() {  
 if (!bookList.isEmpty()) {  
 bookList.clear(); // Clear existing books before loading  
 }  
  
 try (BufferedReader reader = new BufferedReader(new FileReader("books.txt"))) {  
 String line;  
 while ((line = reader.readLine()) != null) {  
 String[] parts = line.split(",");  
 if (parts.length == 4) {  
 String title = parts[0];  
 String author = parts[1];  
 String isbn = parts[2];  
 boolean available = Boolean.*parseBoolean*(parts[3]);  
 bookList.add(new Book(title, author, isbn, available));  
 }  
 }  
 } catch (IOException e) {  
 // Handle file read error  
 e.printStackTrace();  
 }  
  
 // Refresh the TableView to display the updated book list  
 bookTableView.setItems(bookList);  
 }  
  
 private void saveBooksToFile() {  
 try (BufferedWriter writer = new BufferedWriter(new FileWriter("books.txt"))) {  
 for (Book book : bookList) {  
 writer.write(book.getTitle() + "," + book.getAuthor() + "," +  
 book.getIsbn() + "," + book.isAvailable() + "\n");  
 }  
 } catch (IOException e) {  
 // Handle file write error  
 e.printStackTrace();  
 }  
 }  
  
 private void showAlert(Alert.AlertType alertType, String title, String message) {  
 Alert alert = new Alert(alertType);  
 alert.setTitle(title);  
 alert.setHeaderText(null);  
 alert.setContentText(message);  
 alert.showAndWait();  
 }  
}

## HelloApplication.java

package org.example.i222651;  
  
import javafx.application.Application;  
import javafx.fxml.FXMLLoader;  
import javafx.scene.Scene;  
import javafx.stage.Stage;  
  
import java.io.IOException;  
  
public class HelloApplication extends Application {  
 @Override  
 public void start(Stage stage) throws IOException {  
 FXMLLoader fxmlLoader = new FXMLLoader(getClass().getResource("hello-view.fxml"));  
 Scene scene = new Scene(fxmlLoader.load(), 800, 600);  
 stage.setTitle("Library Management System");  
 stage.setScene(scene);  
 stage.show();  
 }  
  
 public static void main(String[] args) {  
 *launch*(args);  
 }  
}

## HelloController.java

package org.example.i222651;  
  
import javafx.fxml.FXML;  
import javafx.event.ActionEvent;  
import javafx.fxml.FXMLLoader;  
import javafx.scene.Scene;  
import javafx.stage.Stage;  
import java.io.IOException;  
  
public class HelloController {  
 @FXML  
 private void openAddBookDialog(ActionEvent event) {  
 try {  
 FXMLLoader loader = new FXMLLoader(getClass().getResource("addBook.fxml"));  
 Scene scene = new Scene(loader.load(), 800, 600);  
 Stage stage = new Stage();  
 stage.setScene(scene);  
 stage.setTitle("Add Book");  
 stage.show();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
 @FXML  
 private void openEditBookDialog(ActionEvent event) {  
 try {  
 FXMLLoader loader = new FXMLLoader(getClass().getResource("editBook.fxml"));  
 Scene scene = new Scene(loader.load(), 800, 600);  
 Stage stage = new Stage();  
 stage.setScene(scene);  
 stage.setTitle("Edit Book");  
 stage.show();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
 @FXML  
 private void openDeleteBookDialog(ActionEvent event) {  
 try {  
 FXMLLoader loader = new FXMLLoader(getClass().getResource("deleteBook.fxml"));  
 Scene scene = new Scene(loader.load(), 800, 600);  
 Stage stage = new Stage();  
 stage.setScene(scene);  
 stage.setTitle("Delete Book");  
 stage.show();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
 @FXML  
 private void openSearchBooksDialog(ActionEvent event) {  
 try {  
 FXMLLoader loader = new FXMLLoader(getClass().getResource("searchBooks.fxml"));  
 Scene scene = new Scene(loader.load(), 800, 600);  
 Stage stage = new Stage();  
 stage.setScene(scene);  
 stage.setTitle("Search Books");  
 stage.show();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
 @FXML  
 private void openViewAllBooksDialog(ActionEvent event) {  
 try {  
 FXMLLoader loader = new FXMLLoader(getClass().getResource("viewAllBooks.fxml"));  
 Scene scene = new Scene(loader.load(), 800, 600);  
 Stage stage = new Stage();  
 stage.setScene(scene);  
 stage.setTitle("View All Books");  
 stage.show();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
}

## SearchBooksController.java

package org.example.i222651;  
  
import javafx.collections.FXCollections;  
import javafx.collections.ObservableList;  
import javafx.fxml.FXML;  
import javafx.scene.control.\*;  
import javafx.scene.control.cell.PropertyValueFactory;  
  
import java.io.BufferedReader;  
import java.io.FileReader;  
import java.io.IOException;  
  
public class SearchBooksController {  
 @FXML  
 private TextField searchField;  
  
 @FXML  
 private TableView<Book> searchTableView;  
  
 private ObservableList<Book> searchResults = FXCollections.*observableArrayList*();  
  
 private ObservableList<Book> bookList;  
  
 @FXML  
 private void initialize() {  
 // Initialize searchResults if it's null  
 if (searchResults == null) {  
 searchResults = FXCollections.*observableArrayList*();  
 }  
  
 // Load existing books from file  
 loadBooksFromFile();  
  
 // Clear existing columns to avoid duplicates  
 searchTableView.getColumns().clear();  
  
 // Set up columns in the TableView  
 TableColumn<Book, String> titleColumn = new TableColumn<>("Title");  
 titleColumn.setCellValueFactory(new PropertyValueFactory<>("title"));  
 titleColumn.setMinWidth(219);  
  
 TableColumn<Book, String> authorColumn = new TableColumn<>("Author");  
 authorColumn.setCellValueFactory(new PropertyValueFactory<>("author"));  
 authorColumn.setMinWidth(150);  
  
 TableColumn<Book, String> isbnColumn = new TableColumn<>("ISBN");  
 isbnColumn.setCellValueFactory(new PropertyValueFactory<>("isbn"));  
 isbnColumn.setMinWidth(100);  
  
 TableColumn<Book, Boolean> availabilityColumn = new TableColumn<>("Availability");  
 availabilityColumn.setCellValueFactory(new PropertyValueFactory<>("available"));  
 availabilityColumn.setMinWidth(100);  
  
 searchTableView.getColumns().addAll(titleColumn, authorColumn, isbnColumn, availabilityColumn);  
  
 // Enable sorting for columns  
 titleColumn.setSortable(true);  
 authorColumn.setSortable(true);  
 isbnColumn.setSortable(true);  
  
 // Allow single selection mode  
 searchTableView.getSelectionModel().setSelectionMode(SelectionMode.*SINGLE*);  
 }  
  
 @FXML  
 private void searchBooks() {  
 searchResults.clear(); // Clear previous search results  
  
 String searchText = searchField.getText().trim().toLowerCase();  
  
 if (searchText.isEmpty()) {  
 showAlert(Alert.AlertType.*ERROR*, "Error", "Please enter a search query.");  
 return;  
 }  
  
 for (Book book : bookList) {  
 if (book.getTitle().toLowerCase().contains(searchText) || book.getAuthor().toLowerCase().contains(searchText)) {  
 searchResults.add(book);  
 }  
 }  
  
 // Display search results in the TableView  
 searchTableView.setItems(searchResults);  
 }  
  
 private void loadBooksFromFile() {  
 bookList = FXCollections.*observableArrayList*(); // Initialize bookList  
  
 try (BufferedReader reader = new BufferedReader(new FileReader("books.txt"))) {  
 String line;  
 while ((line = reader.readLine()) != null) {  
 String[] parts = line.split(",");  
 if (parts.length == 4) {  
 String title = parts[0];  
 String author = parts[1];  
 String isbn = parts[2];  
 boolean available = Boolean.*parseBoolean*(parts[3]);  
 bookList.add(new Book(title, author, isbn, available));  
 }  
 }  
 } catch (IOException e) {  
 // Handle file read error  
 e.printStackTrace();  
 }  
  
 // Refresh the TableView to display the updated book list  
 searchTableView.setItems(bookList);  
 }  
  
 private void showAlert(Alert.AlertType alertType, String title, String message) {  
 Alert alert = new Alert(alertType);  
 alert.setTitle(title);  
 alert.setHeaderText(null);  
 alert.setContentText(message);  
 alert.showAndWait();  
 }  
}

## ViewAllBooksController.java

package org.example.i222651;  
  
import javafx.fxml.FXML;  
import javafx.scene.control.TableColumn;  
import javafx.scene.control.TableView;  
import javafx.collections.FXCollections;  
import javafx.collections.ObservableList;  
import javafx.scene.control.cell.PropertyValueFactory;  
  
import java.io.BufferedReader;  
import java.io.FileReader;  
import java.io.IOException;  
  
public class ViewAllBooksController {  
 @FXML  
 private TableView<Book> searchTableView;  
  
 private ObservableList<Book> bookList;  
  
 @FXML  
 private TableColumn<Book, String> titleColumn;  
  
 @FXML  
 private TableColumn<Book, String> authorColumn;  
  
 @FXML  
 private TableColumn<Book, String> isbnColumn;  
  
 @FXML  
 private TableColumn<Book, Boolean> availabilityColumn;  
  
 @FXML  
 private void initialize() {  
 // Initialize bookList if it's null  
 if (bookList == null) {  
 bookList = FXCollections.*observableArrayList*();  
 }  
  
 // Load books from the "books.txt" file  
 loadBooksFromFile("books.txt");  
  
 // Set the book list to the TableView  
 searchTableView.setItems(bookList);  
  
 // Set up columns in the TableView  
 titleColumn.setCellValueFactory(new PropertyValueFactory<>("title"));  
 authorColumn.setCellValueFactory(new PropertyValueFactory<>("author"));  
 isbnColumn.setCellValueFactory(new PropertyValueFactory<>("isbn"));  
 availabilityColumn.setCellValueFactory(new PropertyValueFactory<>("available"));  
 }  
  
 // Load books from the specified file  
 private void loadBooksFromFile(String filename) {  
 try (BufferedReader reader = new BufferedReader(new FileReader(filename))) {  
 String line;  
 while ((line = reader.readLine()) != null) {  
 String[] parts = line.split(",");  
 if (parts.length == 4) {  
 String title = parts[0];  
 String author = parts[1];  
 String isbn = parts[2];  
 boolean available = Boolean.*parseBoolean*(parts[3]);  
 bookList.add(new Book(title, author, isbn, available));  
 }  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
}

## addBook.fxml

<?xml version="1.0" encoding="UTF-8"?>  
  
<?import javafx.scene.control.Button?>  
<?import javafx.scene.control.Label?>  
<?import javafx.scene.control.TextField?>  
<?import javafx.scene.layout.AnchorPane?>  
<?import javafx.scene.layout.HBox?>  
<?import javafx.scene.layout.Pane?>  
<?import javafx.scene.layout.VBox?>  
<?import javafx.scene.text.Font?>  
  
<HBox maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="600.0" prefWidth="800.0" xmlns="http://javafx.com/javafx/21" xmlns:fx="http://javafx.com/fxml/1" fx:controller="org.example.i222651.AddBookController">  
 <children>  
 <VBox prefHeight="600.0" prefWidth="800.0">  
 <children>  
 <Pane prefHeight="579.0" prefWidth="800.0">  
 <children>  
 <Label layoutX="324.0" layoutY="51.0" text="Add Book">  
 <font>  
 <Font size="35.0" />  
 </font>  
 </Label>  
 <AnchorPane layoutX="175.0" layoutY="161.0" prefHeight="275.0" prefWidth="450.0">  
 <children>  
 <Label graphicTextGap="5.0" layoutX="35.0" layoutY="36.0" prefHeight="18.0" prefWidth="70.0" text="Title:">  
 <font>  
 <Font size="16.0" />  
 </font>  
 </Label>  
 <TextField fx:id="bookTitleField" layoutX="197.0" layoutY="32.0" prefHeight="26.0" prefWidth="205.0" />  
 <TextField fx:id="authorField" layoutX="197.0" layoutY="85.0" prefHeight="26.0" prefWidth="205.0" />  
 <TextField fx:id="isbnField" layoutX="197.0" layoutY="139.0" prefHeight="26.0" prefWidth="205.0" promptText="Example: 1000345678" />  
 <Label graphicTextGap="5.0" layoutX="35.0" layoutY="89.0" prefHeight="18.0" prefWidth="70.0" text="Author:">  
 <font>  
 <Font size="16.0" />  
 </font>  
 </Label>  
 <Label graphicTextGap="5.0" layoutX="35.0" layoutY="143.0" prefHeight="18.0" prefWidth="70.0" text="ISBN:">  
 <font>  
 <Font size="16.0" />  
 </font>  
 </Label>  
 <Button layoutX="332.0" layoutY="210.0" mnemonicParsing="false" onAction="#onAddBookButtonClick" prefHeight="30.0" prefWidth="70.0" text="Add">  
 <font>  
 <Font size="16.0" />  
 </font>  
 </Button>  
 </children>  
 </AnchorPane>  
 </children>  
 </Pane>  
 <Pane prefHeight="84.0" prefWidth="800.0">  
 <children>  
 <Label layoutX="10.0" layoutY="33.0" prefHeight="17.0" prefWidth="781.0" text="Enter Details and then add Book" />  
 </children></Pane>  
 </children>  
 </VBox>  
 </children>  
</HBox>

## deleteBook.fxml

<?xml version="1.0" encoding="UTF-8"?>  
  
<?import javafx.scene.control.Button?>  
<?import javafx.scene.control.Label?>  
<?import javafx.scene.layout.AnchorPane?>  
<?import javafx.scene.layout.VBox?>  
<?import javafx.scene.text.Font?>  
<?import javafx.scene.control.TableView?>  
<?import javafx.scene.control.TableColumn?>  
  
<AnchorPane prefHeight="600.0" prefWidth="800.0" xmlns="http://javafx.com/javafx/21" xmlns:fx="http://javafx.com/fxml/1" fx:controller="org.example.i222651.DeleteBookController">  
 <VBox layoutX="115.0" layoutY="119.0" prefWidth="571.0" spacing="10">  
 <Label fx:id="selectedBookLabel" prefHeight="17.0" prefWidth="0.0" />  
 <Label text="Select a book to delete:">  
 <font>  
 <Font size="16.0" />  
 </font>  
 </Label>  
 <TableView fx:id="bookTableView" prefHeight="300.0" prefWidth="500.0">  
 <columns>  
 <TableColumn text="Title" fx:id="titleColumn" />  
 <TableColumn text="Author" fx:id="authorColumn" />  
 <TableColumn text="ISBN" fx:id="isbnColumn" />  
 <TableColumn text="Availability" fx:id="availabilityColumn" />  
 </columns>  
 </TableView>  
 </VBox>  
 <Label layoutX="306.0" layoutY="52.0" text="Delete Book">  
 <font>  
 <Font size="35.0" />  
 </font>  
 </Label>  
 <Button layoutX="618.0" layoutY="512.0" onAction="#deleteSelectedBook" text="Delete">  
 <font>  
 <Font size="16.0" />  
 </font>  
 </Button>  
 <Label layoutX="10.0" layoutY="578.0" prefHeight="17.0" prefWidth="781.0" text="Select a book and then delete it" />  
</AnchorPane>

editBook.fxml

<?xml version="1.0" encoding="UTF-8"?>  
  
<?import javafx.scene.control.Button?>  
<?import javafx.scene.control.Label?>  
<?import javafx.scene.control.TextField?>  
<?import javafx.scene.layout.AnchorPane?>  
<?import javafx.scene.layout.VBox?>  
<?import javafx.scene.text.Font?>  
<?import javafx.scene.control.TableView?>  
<?import javafx.scene.control.TableColumn?>  
  
  
<AnchorPane prefHeight="600.0" prefWidth="800.0" xmlns="http://javafx.com/javafx/21" xmlns:fx="http://javafx.com/fxml/1" fx:controller="org.example.i222651.EditBookController">  
 <VBox layoutX="47.0" layoutY="150.0" prefHeight="300.0" prefWidth="397.0" spacing="10">  
 <children>  
 <TableView fx:id="bookTableView" prefHeight="300.0" prefWidth="500.0">  
 <columns>  
 <TableColumn text="Title" prefWidth="150" />  
 <TableColumn text="Author" prefWidth="150" />  
 <TableColumn text="ISBN" prefWidth="150" />  
 <TableColumn text="Availability" prefWidth="150" />  
 </columns>  
 </TableView>  
 </children>  
 </VBox>  
 <Label layoutX="10.0" layoutY="578.0" prefHeight="17.0" prefWidth="781.0" text="Select Book to edit and then save changes" />  
 <VBox layoutX="491.0" layoutY="150.0" prefHeight="144.0" prefWidth="266.0" spacing="25.0">  
 <children>  
 <TextField fx:id="titleField" promptText="Title">  
 <font>  
 <Font size="16.0" />  
 </font>  
 </TextField>  
 <TextField fx:id="authorField" promptText="Author">  
 <font>  
 <Font size="16.0" />  
 </font>  
 </TextField>  
 <TextField fx:id="isbnField" promptText="ISBN">  
 <font>  
 <Font size="16.0" />  
 </font>  
 </TextField>  
 <TextField fx:id="availabilityField" promptText="Availability (true/false)">  
 <font>  
 <Font size="16.0" />  
 </font>  
 </TextField>  
 </children>  
 </VBox>  
 <Button layoutX="637.0" layoutY="403.0" onAction="#saveChanges" text="Save Changes">  
 <font>  
 <Font size="16.0" />  
 </font>  
 </Button>  
 <Label layoutX="327.0" layoutY="62.0" text="Edit Book">  
 <font>  
 <Font size="35.0" />  
 </font>  
 </Label>  
</AnchorPane>

hello-view.fxml  
<?xml version="1.0" encoding="UTF-8"?>  
  
<?import javafx.scene.control.Label?>  
<?import javafx.scene.control.Menu?>  
<?import javafx.scene.control.MenuBar?>  
<?import javafx.scene.control.MenuItem?>  
<?import javafx.scene.layout.HBox?>  
<?import javafx.scene.layout.Pane?>  
<?import javafx.scene.layout.VBox?>  
<?import javafx.scene.text.Font?>  
  
<HBox maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="600.0" prefWidth="800.0" xmlns="http://javafx.com/javafx/21" xmlns:fx="http://javafx.com/fxml/1" fx:controller="org.example.i222651.HelloController">  
 <children>  
 <VBox prefHeight="504.0" prefWidth="802.0">  
 <children>  
 <Pane prefHeight="114.0" prefWidth="800.0">  
 <children>  
 <MenuBar prefHeight="30.0" prefWidth="798.0">  
 <menus>  
 <Menu mnemonicParsing="false" text="File">  
 <items>  
 <MenuItem mnemonicParsing="false" text="New" />  
 <MenuItem mnemonicParsing="false" text="Open" />  
 <MenuItem mnemonicParsing="false" text="Save" />  
 <MenuItem mnemonicParsing="false" text="Save As" />  
 <MenuItem mnemonicParsing="false" text="Exit" />  
 </items>  
 </Menu>  
 <Menu mnemonicParsing="false" text="Edit">  
 <items>  
 <MenuItem fx:id="addBookMenuItem" onAction="#openAddBookDialog" text="Add Book" />  
 <MenuItem fx:id="editBookMenuItem" onAction="#openEditBookDialog" text="Edit Book" />  
 <MenuItem fx:id="deleteBookMenuItem" onAction="#openDeleteBookDialog" text="Delete Book" />  
 </items>  
 </Menu>  
 <Menu mnemonicParsing="false" text="View">  
 <items>  
 <MenuItem fx:id="viewAllBooksMenuItem" onAction="#openViewAllBooksDialog" text="View All Books" />  
 <MenuItem fx:id="searchBooksMenuItem" onAction="#openSearchBooksDialog" text="Search Books" />  
 </items>  
 </Menu>  
 </menus>  
 </MenuBar>  
 </children>  
 </Pane>  
 <Pane prefHeight="592.0" prefWidth="800.0">  
 <children>  
 <Label layoutX="148.0" layoutY="190.0" text="Library Management System">  
 <font>  
 <Font size="40.0" />  
 </font>  
 </Label>  
 <Label layoutX="299.0" layoutY="248.0" text="Sumeed Jawad Kanwar">  
 <font>  
 <Font size="20.0" />  
 </font>  
 </Label>  
 <Label layoutX="364.0" layoutY="278.0" text="22I-2651">  
 <font>  
 <Font size="18.0" />  
 </font>  
 </Label>  
 </children>  
 </Pane>  
 <Pane prefHeight="93.0" prefWidth="800.0">  
 <children>  
 <Label layoutX="9.0" layoutY="5.0" prefHeight="17.0" prefWidth="781.0" text="Welcome to the Library Management System" />  
 </children>  
 </Pane>  
 </children>  
 </VBox>  
 </children>  
</HBox>

searchBooks.fxml  
<?xml version="1.0" encoding="UTF-8"?>  
  
<?import javafx.scene.control.Button?>  
<?import javafx.scene.control.Label?>  
<?import javafx.scene.control.TableColumn?>  
<?import javafx.scene.control.TableView?>  
<?import javafx.scene.control.TextField?>  
<?import javafx.scene.layout.AnchorPane?>  
<?import javafx.scene.layout.VBox?>  
<?import javafx.scene.text.Font?>  
  
<AnchorPane prefHeight="600.0" prefWidth="800.0" xmlns="http://javafx.com/javafx/21" xmlns:fx="http://javafx.com/fxml/1" fx:controller="org.example.i222651.SearchBooksController">  
 <VBox layoutX="50.0" layoutY="194.0" prefHeight="298.0" prefWidth="700.0" spacing="10">  
 <TableView fx:id="searchTableView" prefHeight="300.0" prefWidth="500.0">  
 <columns>  
 <TableColumn fx:id="titleColumn" prefWidth="262.0" text="Title" />  
 <TableColumn fx:id="authorColumn" prefWidth="167.0" text="Author" />  
 <TableColumn fx:id="isbnColumn" minWidth="4.0" prefWidth="137.0" text="ISBN" />  
 <TableColumn fx:id="availabilityColumn" minWidth="0.0" prefWidth="133.0" text="Availability" />  
 </columns>  
 </TableView>  
 </VBox>  
 <Label layoutX="297.0" layoutY="58.0" text="Search Books">  
 <font>  
 <Font size="35.0" />  
 </font>  
 </Label>  
 <Button layoutX="680.0" layoutY="143.0" onAction="#searchBooks" text="Search">  
 <font>  
 <Font size="16.0" />  
 </font>  
 </Button>  
 <TextField fx:id="searchField" layoutX="52.0" layoutY="143.0" prefHeight="35.0" prefWidth="621.0" promptText="Enter Title or Author">  
 <font>  
 <Font size="16.0" />  
 </font>  
 </TextField>  
 <Label layoutX="10.0" layoutY="583.0" prefHeight="17.0" prefWidth="781.0" text="Search book by title or author name" />  
</AnchorPane>

viewAllBooks.fxml

<?xml version="1.0" encoding="UTF-8"?>  
  
<?import javafx.scene.control.Label?>  
<?import javafx.scene.control.TableColumn?>  
<?import javafx.scene.control.TableView?>  
<?import javafx.scene.layout.AnchorPane?>  
<?import javafx.scene.layout.VBox?>  
<?import javafx.scene.text.Font?>  
  
<AnchorPane prefHeight="600.0" prefWidth="800.0" xmlns="http://javafx.com/javafx/21" xmlns:fx="http://javafx.com/fxml/1" fx:controller="org.example.i222651.ViewAllBooksController">  
 <VBox layoutX="50.0" layoutY="159.0" prefHeight="362.0" prefWidth="700.0" spacing="10">  
 <TableView fx:id="searchTableView" prefHeight="368.0" prefWidth="700.0">  
 <columns>  
 <TableColumn fx:id="titleColumn" prefWidth="262.0" text="Title" />  
 <TableColumn fx:id="authorColumn" prefWidth="167.0" text="Author" />  
 <TableColumn fx:id="isbnColumn" minWidth="4.0" prefWidth="137.0" text="ISBN" />  
 <TableColumn fx:id="availabilityColumn" minWidth="0.0" prefWidth="133.0" text="Availability" />  
 </columns>  
 </TableView>  
 </VBox>  
 <Label layoutX="287.0" layoutY="57.0" text="View All Books">  
 <font>  
 <Font size="35.0" />  
 </font>  
 </Label>  
 <Label layoutX="10.0" layoutY="583.0" prefHeight="17.0" prefWidth="781.0" text="View All Books in the Library" />  
</AnchorPane>

# Question 2

## Main.java

package jdbcExample;  
  
import java.sql.\*;  
  
/\* Entities and Attributes:  
Patients Table:  
Attributes: patientId (primary key), name, age, contactInformation  
  
Doctors Table:  
Attributes: doctorId (primary key), name, specialization, contactInformation  
  
Appointments Table:  
Attributes: appointmentId (primary key), patientId (foreign key referencing Patients table), doctorId (foreign key referencing Doctors table), appointmentDate, appointmentTime, reasonForVisit  
  
DoctorSchedule Table:  
Attributes: scheduleID (primary key), doctorID (foreign key referencing Doctors table), day, schedule  
\*/  
  
public class Main {  
 public static void main(String[] args) {  
 Connection connection = null;  
 Statement statement = null;  
 String user = "root";  
 String password = "root";  
  
 try {  
 Class.*forName*("com.mysql.cj.jdbc.Driver");  
 connection = DriverManager.*getConnection*("jdbc:mysql://localhost:3306/q2", user, password);  
 System.*out*.println("Connected Successfully");  
  
 statement = connection.createStatement();  
  
 statement.execute("CREATE TABLE IF NOT EXISTS Patients (patientId INT PRIMARY KEY, name VARCHAR(50), age INT, contactInformation VARCHAR(50))");  
 statement.execute("CREATE TABLE IF NOT EXISTS Doctors (doctorId INT PRIMARY KEY, name VARCHAR(50), specialization VARCHAR(50), contactInformation VARCHAR(50))");  
 statement.execute("CREATE TABLE IF NOT EXISTS Appointments (appointmentId INT PRIMARY KEY, patientId INT, doctorId INT, appointmentDate DATE, appointmentTime TIME, reasonForVisit VARCHAR(255), FOREIGN KEY (patientId) REFERENCES Patients(patientId), FOREIGN KEY (doctorId) REFERENCES Doctors(doctorId))");  
 statement.execute("CREATE TABLE IF NOT EXISTS DoctorSchedule (scheduleId INT PRIMARY KEY, doctorId INT, day VARCHAR(20), schedule VARCHAR(255), FOREIGN KEY (doctorId) REFERENCES Doctors(doctorId))");  
  
  
 statement.execute("INSERT INTO Patients (name, age, contactInformation) VALUES ('Will', 35, '+923001234567')");  
 statement.execute("INSERT INTO Doctors (name, specialization, contactInformation) VALUES ('Dr. Lecter', 'Neurology', '+923007654321'), ('Dr. Bloom', 'Internal Medicine', '+923009876543')");  
 statement.execute("INSERT INTO DoctorSchedule (DoctorID, Day, Schedule) VALUES (1, 'Monday', '8:00 AM - 4:00 PM')");  
 statement.execute("INSERT INTO DoctorSchedule (DoctorID, Day, Schedule) VALUES (2, 'Wednesday', '10:00 AM - 6:00 PM')");  
  
 statement.execute("INSERT INTO Appointments (patientId, doctorId, appointmentDate, appointmentTime, reasonForVisit) VALUES (1, 1, '2024-03-25', '08:00:00', 'Persistent headaches')");  
 statement.executeUpdate("DELETE FROM Appointments WHERE patientId = 1");  
  
 statement.executeUpdate("UPDATE Doctors SET contactInformation = '+923009513567' WHERE name = 'Dr. Bloom'");  
 statement.executeUpdate("UPDATE DoctorSchedule SET Schedule = '1:00 PM - 9:00 PM' WHERE DoctorID = 2 AND Day = 'Wednesday'");  
  
 statement.close();  
 connection.close();  
 } catch (Exception exc) {  
 exc.printStackTrace();  
 }  
 }  
}

# Question 3

## UDPClient.java

import java.net.\*;  
import java.util.Scanner;  
  
public class UDPClient {  
 public static void main(String[] args) throws Exception {  
 DatagramSocket ds = new DatagramSocket();  
 InetAddress ip = InetAddress.*getByName*("localhost");  
 Scanner scanner = new Scanner(System.*in*);  
  
 while (true) {  
 System.*out*.print("Enter your message (type 'GoodBye' to exit): ");  
 String message = scanner.nextLine();  
 byte[] sendData = message.getBytes();  
 DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length, ip, 3000);  
 ds.send(sendPacket);  
  
 if (message.equalsIgnoreCase("GoodBye")) {  
 System.*out*.println("Client disconnected.");  
 break;  
 }  
  
 byte[] receiveData = new byte[1024];  
 DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);  
 ds.receive(receivePacket);  
  
 String serverMessage = new String(receivePacket.getData(), 0, receivePacket.getLength());  
 System.*out*.println("Server says:" + serverMessage);  
 }  
  
 ds.close();  
 scanner.close();  
 }  
}

## UDPClient2.java

import java.net.\*;  
import java.util.Scanner;  
  
public class UDPClient2 {  
 public static void main(String[] args) throws Exception {  
 DatagramSocket ds = new DatagramSocket();  
 InetAddress ip = InetAddress.*getByName*("localhost");  
 Scanner scanner = new Scanner(System.*in*);  
  
 while (true) {  
 System.*out*.print("Enter your message (type 'GoodBye' to exit): ");  
 String message = scanner.nextLine();  
 byte[] sendData = message.getBytes();  
 DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length, ip, 3000);  
 ds.send(sendPacket);  
  
 if (message.equalsIgnoreCase("GoodBye")) {  
 System.*out*.println("Client disconnected.");  
 break;  
 }  
  
 byte[] receiveData = new byte[1024];  
 DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);  
 ds.receive(receivePacket);  
  
 String serverMessage = new String(receivePacket.getData(), 0, receivePacket.getLength());  
 System.*out*.println("Server says:" + serverMessage);  
 }  
  
 ds.close();  
 scanner.close();  
 }  
}

## UDPServer.java

import java.net.\*;  
import java.util.\*;  
  
public class UDPServer {  
 public static void main(String[] args) throws Exception {  
 DatagramSocket ds = new DatagramSocket(3000);  
 byte[] receiveData = new byte[1024];  
 List<InetAddress> clientAddresses = new ArrayList<>();  
 List<Integer> clientPorts = new ArrayList<>();  
  
 while (true) {  
 DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);  
 ds.receive(receivePacket);  
  
 String clientMessage = new String(receivePacket.getData(), 0, receivePacket.getLength());  
 InetAddress clientAddress = receivePacket.getAddress();  
 int clientPort = receivePacket.getPort();  
  
 if (!clientAddresses.contains(clientAddress)) {  
 clientAddresses.add(clientAddress);  
 clientPorts.add(clientPort);  
 }  
  
 System.*out*.println("Client " + clientAddress.getHostAddress() + ":" + clientPort + " says: " + clientMessage);  
  
 if (clientMessage.equalsIgnoreCase("GoodBye")) {  
 int clientIndex = clientAddresses.indexOf(clientAddress);  
 clientAddresses.remove(clientIndex);  
 clientPorts.remove(clientIndex);  
 System.*out*.println("Client " + clientAddress.getHostAddress() + ":" + clientPort + " disconnected.");  
 continue;  
 }  
  
 String response = " " + clientMessage;  
 DatagramPacket sendPacket = new DatagramPacket(response.getBytes(), response.length(), clientAddress, clientPort);  
 ds.send(sendPacket);  
 }  
 }  
}