Assignment 2

1DV503 - Database Web Application Programming

Contacts:
Alisa Lincke (alisa.lincke@lnu.se)
For questions, use the forum on Moodle's course page or Slack.

Description

This assignment can be completed in **groups** (max 2 people) or **individually**. In this task, you will develop a web application in JavaScript (NodeJS) or as a Java-based terminal application if you know the Java language using the Visual Studio Code. The application will interact with a database (MySQL Server) using an API (mysql2 connector).

Prerequisites:

- Installed MySQL Server
- Installed mysql2 (https://www.npmjs.com/package/mysql2)

Study Material:

- Lectures 5 and 6

Minimum Score to Pass:

- 60 points

Submission:

Your submission should include solutions to all tasks. Submit your web porject files to Moodle. **Do not submit node_modules folder and any `.sql` files**, as we will use our own database to test the application. Therefore, it is crucial to follow the exact instructions in **Task 1** (configuration of the bookstore database).

Task 1 Use MySQL Workbench to Create and Configure the Bookstore Database (20 points)

1.1 Connect to MySQL Server

- Open **MySQL Workbench** or **DBeaver** and connect to the MySQL Server.

1.2 Create the Database Schema

- Create a schema named **`book_store`** for the bookstore database according to the relational diagram shown in **Figure 1**.

Database Structure

The database consists of five tables:

1. **Books:**
   - This table stores information about the books sold in the bookstore.
   - Each book is classified under a "subject" to allow subject-based searches.
2. **Members:**
   - This table stores information about the application's members.
   - Each member registers with a unique **email address** and **password**.
   - `UserId` is **auto-incremented** and generated by the database.
   - The **email address must be unique** in the members' table.
3. **Orders:**
   - This table stores information about orders placed by members.
   - Each order may contain **one or more books**, and order details are stored in a separate table.
   - The database generates a **unique order number** for each order.
4. **Order Details:**
   - This table stores details for each order, including the **ISBN** and **quantity** of books in the order.
5. **Cart:**
   - This table contains the **ISBN** and **quantity** of each book added to a member's shopping cart.
   - When a member checks out, the **cart is cleared**, and an order is created.

MySQL Workbench symbols will be useful for this task.

1.3 Populate the Books Table

- Use the provided **`books.sql`** script (available on Moodle) to populate the **Books** table.
- The **`books.sql`** file contains data for this database.

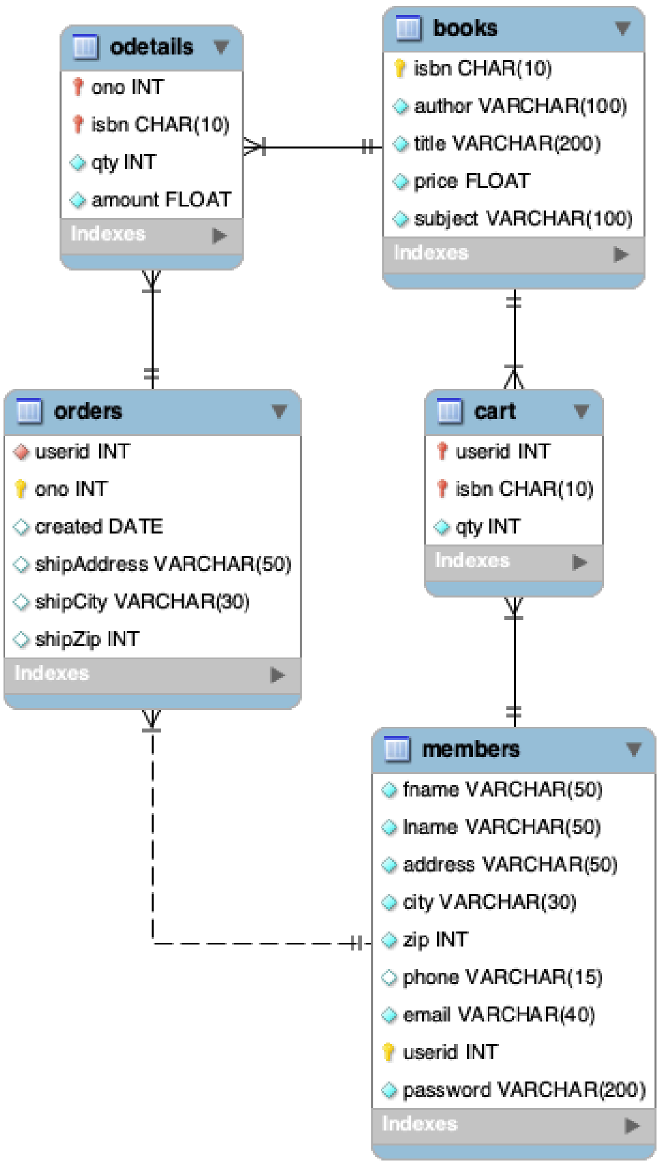


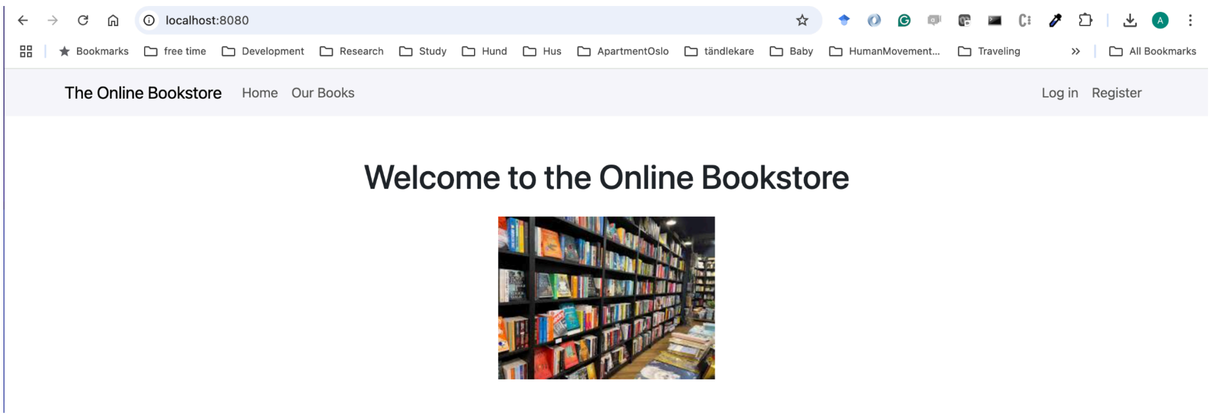
Figure 1. Relational Model Diagram for the Bookstore Database

⚠ **Note! Use the same names for all tables and attributes, as we will use our database to test your application.**

Task 2 - The BookStore Application (80 points)

2.1 Main Page or Home Page

Create a simple main page (see the example below) with a members menu that allows Login and Register.

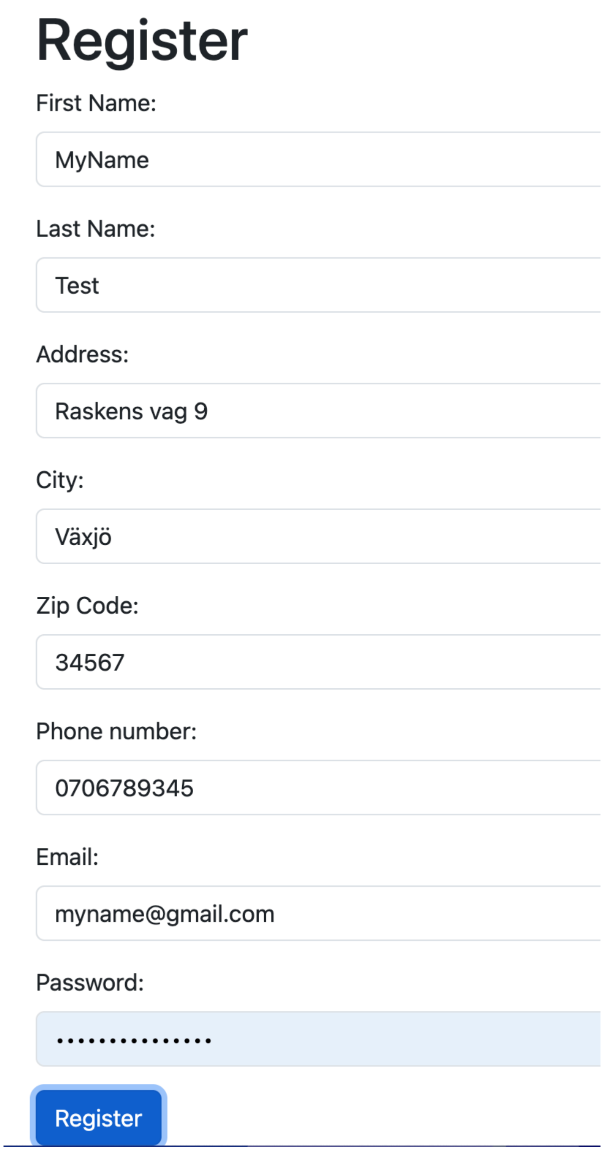


2.2 Register page

Create a registration form that allows new users to sign up and become members of the bookstore.
The form should collect the following details:

- First Name
- Last Name
- Address
- City
- Zip code
- Phone number
- Email (must be unique)
- Password (encrypted)

Validate the user input (e.g., email format, zip code) before inserting them into the database.
Store the user data in the ***members*** table in the MySQL database.
Ensure that duplicate email addresses are not allowed.
Example of the Registration form


Register

First Name:

MyName

Last Name:

Test

Address:

Raskens vag 9

City:

Växjö

Zip Code:

34567

Phone number:

0706789345

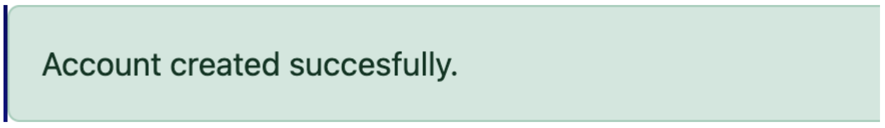
Email:

myname@gmail.com

Password:

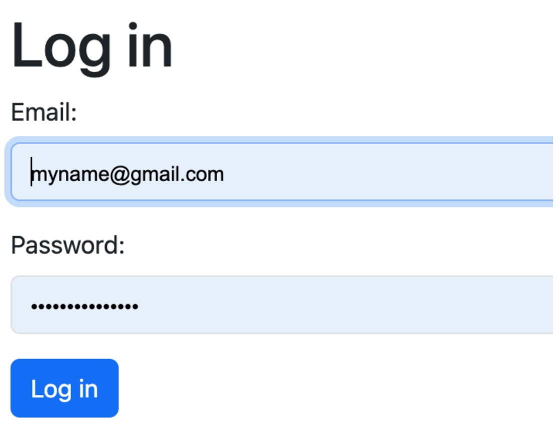
•••••••••••••••

After the user clicks Register buttons, he/she sees a notification about successfully creating account.


Account created succesfully.

2.2 Login page

Create the login form where the user enters email and password, and implement user input validation. An example of the Login form is shown below.


Log in

Email:

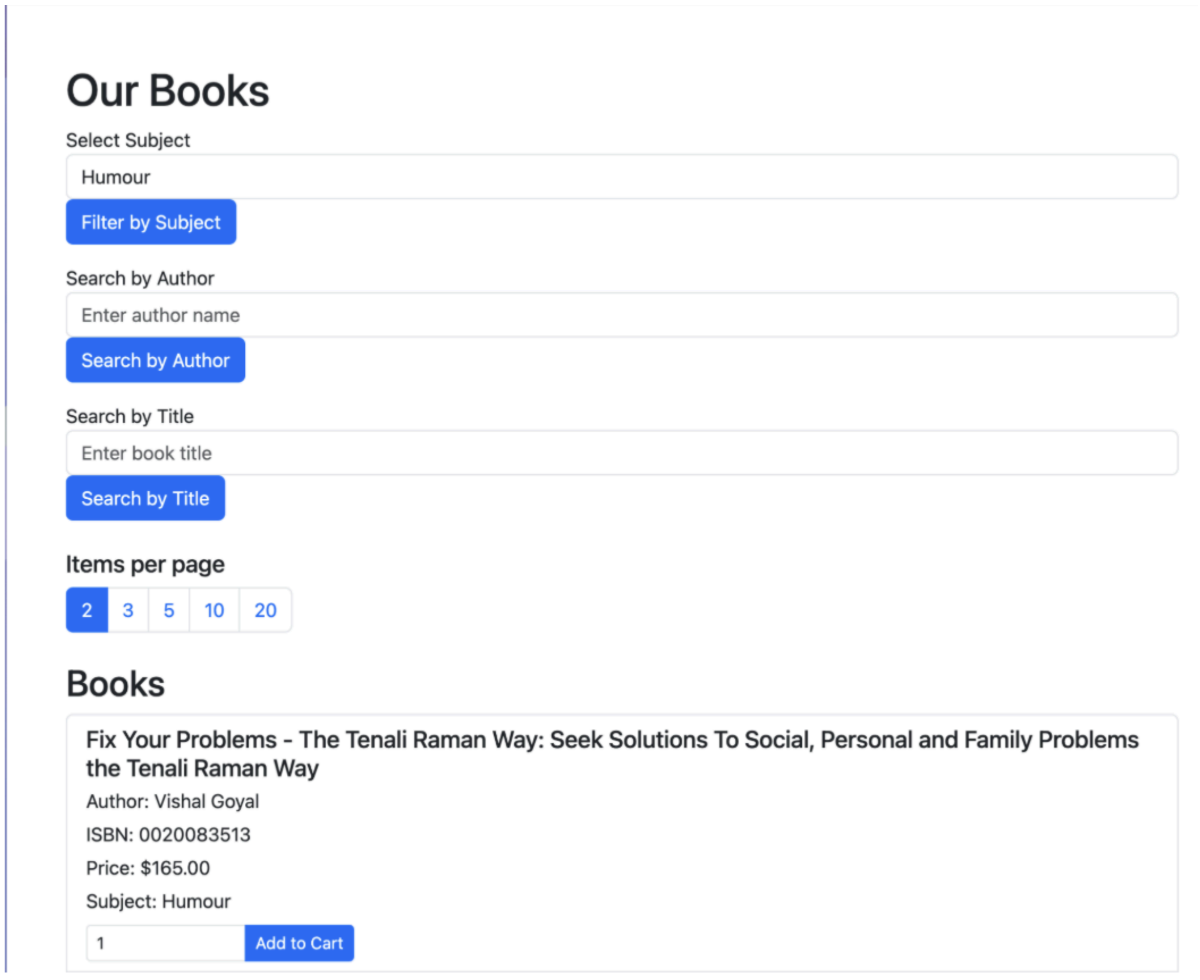
myname@gmail.com

Password:

•••••••••••••••

2.3 Search for a Book

After the member login, display his/her name, Log out button should be visible, and the content of the main page should allow users to select a subject search by author and/or title. After the user selects the subject, he/she can see five books at the screen (either as a table, list, or other UI component). It should be able to view the next five books using pagination. See the example below.


Our Books

Select Subject: Humour

Search by Author: Enter author name

Search by Title: Enter book title

Items per page: 2 3 5 10 20

Books

Fix Your Problems - The Tenali Raman Way: Seek Solutions To Social, Personal and Family Problems the Tenali Raman Way

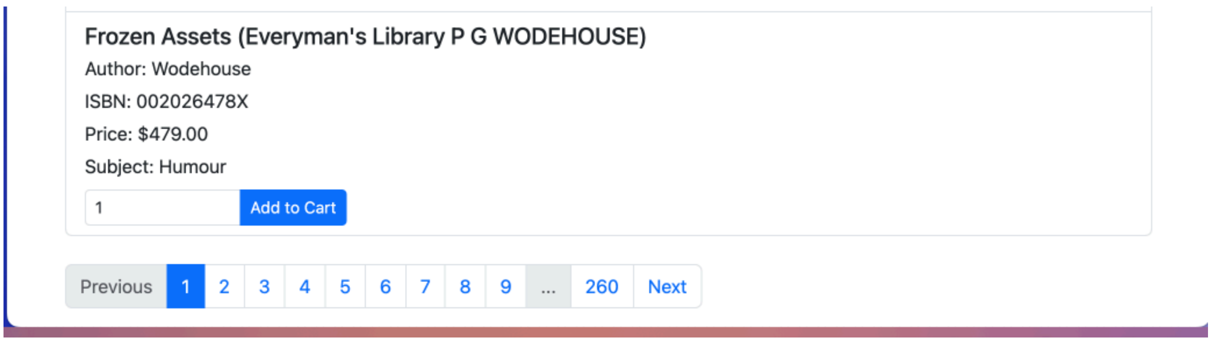
Author: Vishal Goyal

ISBN: 0020083513

Price: $165.00

Subject: Humour

1


Frozen Assets (Everyman's Library P G WODEHOUSE)

Author: Wodehouse

ISBN: 002026478X

Price: $479.00

Subject: Humour

1 Add to Cart

Previous 1 2 3 4 5 6 7 8 9 ... 260 Next

You should use **OFFSET** and **LIMIT** in your SQL query and not fetch all books at once, even if there are not many books in this database. There are subjects that have no books, so display a message to the user in such cases.

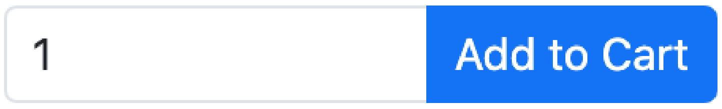
Search by Author

- The user enters a **part of an author's first name**.
- The program retrieves all authors whose **first name starts with** the entered string.
- The search should be **case-insensitive** (e.g., "Ali" or "ali" should return the same results).
- Example: If the user enters **"Ali"** or **"ali"**, the program should display authors such as:
  - Alison
  - Ali
  - Alim
  - Alik

Search by Title

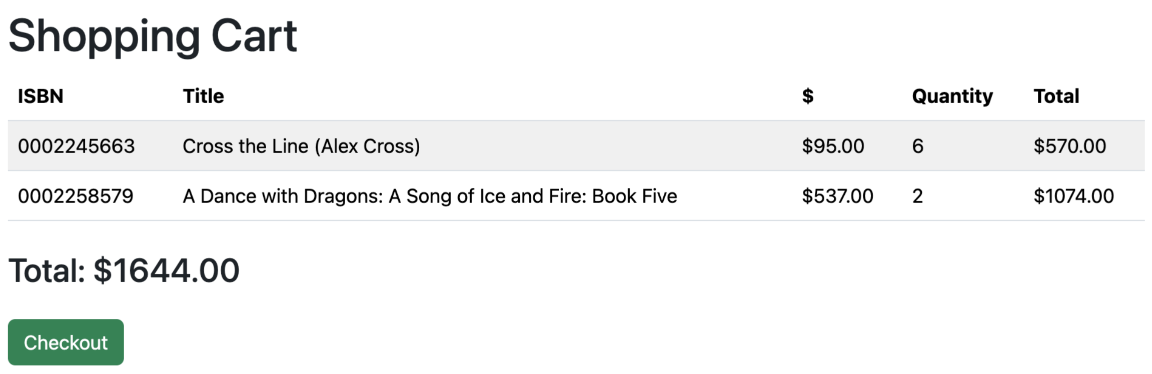
- The user enters a **word**.
- The program retrieves all **book titles that contain** the entered word.
- The search should be **case-insensitive** (e.g., "Sport" and "sport" should return the same results).
- Example: If the user searches for **"Sport"** or **"sport"**, the program should display book titles such as:
  - "The World of Sport"
  - "Sport and Fitness"
  - "History of Sport"

Users can specify a quantity (positive number) and click button “Add to cart”. The user can add multiple books to the shopping cart. Information about the books added to the cart should be stored in the cart table. If a user adds a book that is already in the cart, the program should update the quantity of that book instead of adding a duplicate entry. Ensure the total quantity reflects the updated amount.



2.4 View Cart

The user should be able to view his/her cart, and the following information should be displayed to the user as shown in the figure bellow.



Shopping Cart

ISBN	Title	$	Quantity	Total
0002245663	Cross the Line (Alex Cross)	$95.00	6	$570.00
0002258579	A Dance with Dragons: A Song of Ice and Fire: Book Five	$537.00	2	$1074.00

Total: $1644.00

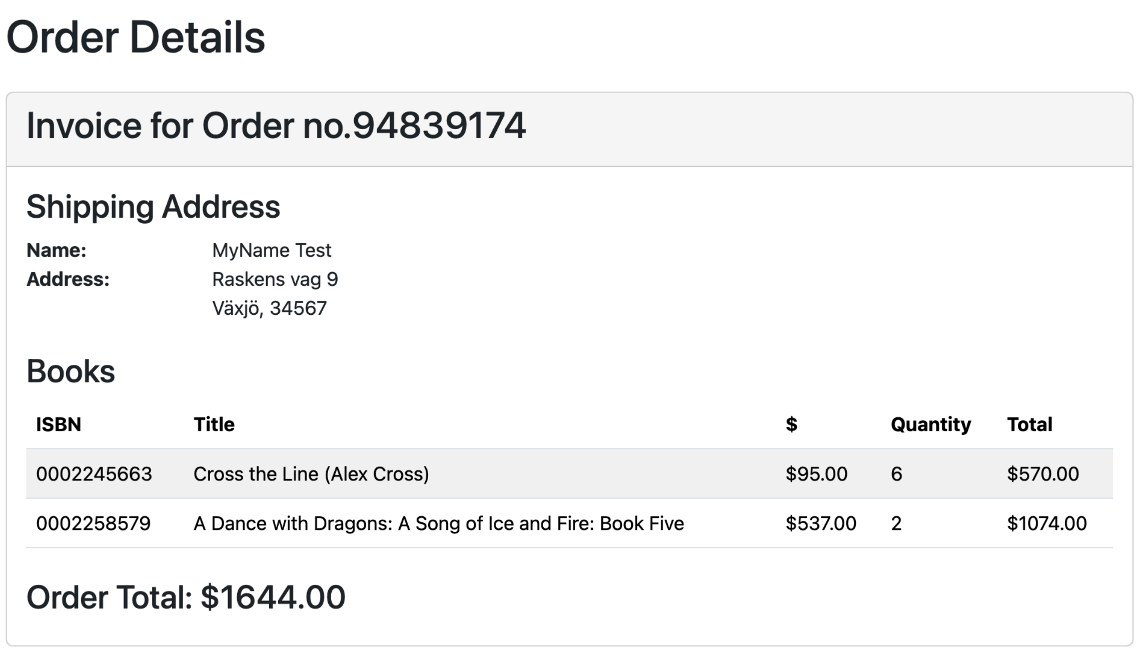
2.4 Checkout

When the user presses the button Checkout, the order invoice should be generated and displayed.

Invoice Details:

1. **Use the user's registered address** for delivery.
2. **Save the order** in the `"order"` table with the following details:
   - **Order date** (`created` date) – the date the order was placed.
   - **Delivery date** – automatically set to **one week from the order date** (not stored in the database, only displayed on the screen).
   - **Delivery address (shippedAddres,shippedCity,shippedZipCode)** – taken from the member's registered address.
3. **Save order details** in the `"order_details"` table:
   - **Book ISBN** – the identifier of each book in the order.
   - **Quantity (qty)** – the number of copies of each book ordered.
   - **Amount** – calculated as `(quantity * book price)`.

This ensures that every purchase is correctly recorded, and users receive a clear **invoice with delivery details** on the screen as shown in the figure below.


Order Details

Invoice for Order no.94839174

Shipping Address

Name:	MyName Test
Address:	Raskens vag 9
	Växjö, 34567

Books

ISBN	Title	$	Quantity	Total
0002245663	Cross the Line (Alex Cross)	$95.00	6	$570.00
0002258579	A Dance with Dragons: A Song of Ice and Fire: Book Five	$537.00	2	$1074.00

Order Total: $1644.00

2.5 Log out

When the user clicks Log out, he/she see the home welcoming page.

The user interface may have a different design, but the **information** displayed and the **user interactions** with the UI must remain consistent with the assignment's specifications.