Database Systems
CS 4347.003

Chocolate Volcanic Library

Jalal Omer

Group 17

Ethan Emmanuel (eje200000)
Pooja Ganapathy (pxg200027)
Nikhil Jahagirdar (nxj200009)
Reuben John (rxj200024)
Victoria Vynnychok (vxv200032)

December 3rd, 2023

Table of Contents

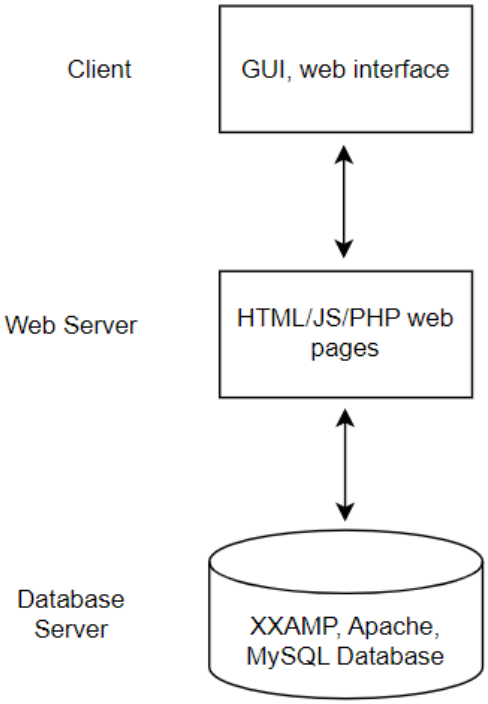
1. Introduction

This project involved creating a database, interface, and webpage for a library to update information about loans, books, librarians, and library card users. We created a website hosted on GitHub Pages (https://vvyn.github.io/volcanic-chocolate-library/). We used HTML, CSS, JavaScript, PHP, and MySQL as our tech stack (https://github.com/vvyn/volcanic-chocolate-library).

In this report, we will go over the system requirements, conceptual design of the database, logical database schema, functional dependencies and database normalization, the database system, user application interface, and our conclusions and future work for this project.

2. System Requirements

2.1 Context Diagram



2.2 Interface Requirements

- The website shall have a page for the Book entity CRUD functions with a section and submit button for each function (create, read, update, delete).
- The website shall have a page for the Librarian entity CRUD functions with a section and submit button for each function (create, read, update, delete).
- The website shall have a page for the Loan entity CRUD functions with a section and submit button for each function (create, read, update, delete).
- The website shall have a page for the User entity CRUD functions with a section and submit button for each function (create, read, update, delete).

2.3 Functional Requirements

- The system shall let a user create a Book entity.
- The system shall let a user read a Book entity.
- The system shall let a user update a Book entity.
- The system shall let a user delete a Book entity.
- The system shall let a user create a Librarian entity.
- The system shall let a user read a Librarian entity.
- The system shall let a user update a Librarian entity.

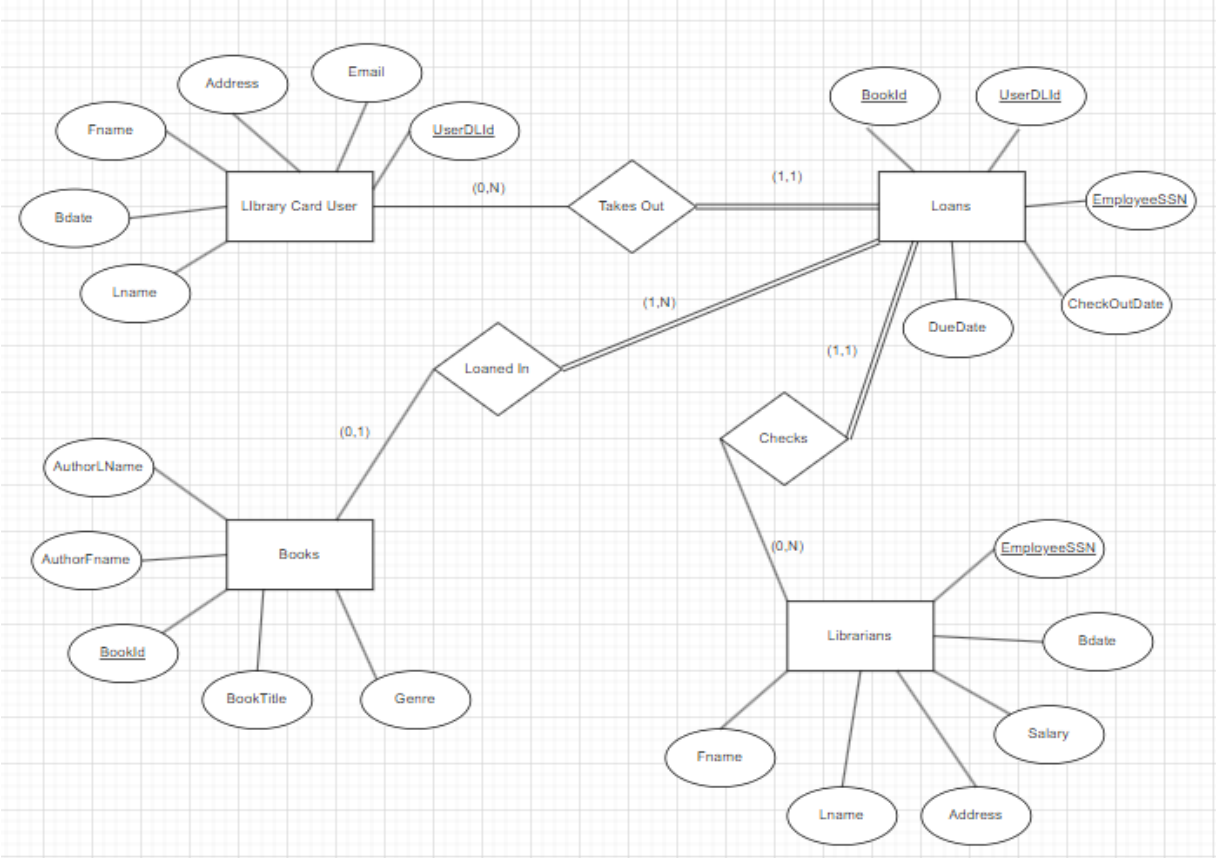
- The system shall let a user delete a Librarian entity.
- The system shall let a user create a Loan entity.
- The system shall let a user read a Loan entity.
- The system shall let a user update a Loan entity.
- The system shall let a user delete a Loan entity.
- The system shall let a user create a User entity.
- The system shall let a user read a User entity.
- The system shall let a user update a User entity.
- The system shall let a user delete a User entity.

2.4 Non-Functional Requirements

- The system shall respond to input errors within 5 seconds.
- The system shall use HTML, CSS, and JavaScript.
- The system shall use MySQL, XXAMP, and Apache.
- The system shall use PHP, XXAMP, and Apache to connect the HTML web interface to the database.

3. Conceptual Design of the Database

3.1 ER Diagram



3.2 Data Dictionary

i. Library card user

Fname (VARCHAR 30)	Lname (VARCHAR 30)	UserDLId (CHAR 8)	Bdate (DATE)	Address (VARCHAR 30)	Email (VARCHAR 100)

ii. Books

AuthorFname (VARCHAR 30)	AuthorLname (VARCHAR 30)	BookTitle (VARCHAR 100)	Genre (VARCHAR 50)	BookId (INT)

iii. Loans

BookId (INT)	UserDLId (CHAR 8)	EmployeeSSN (CHAR 9)	CheckOutDate (DATE)	DueDate (DATE)

iv. Librarians

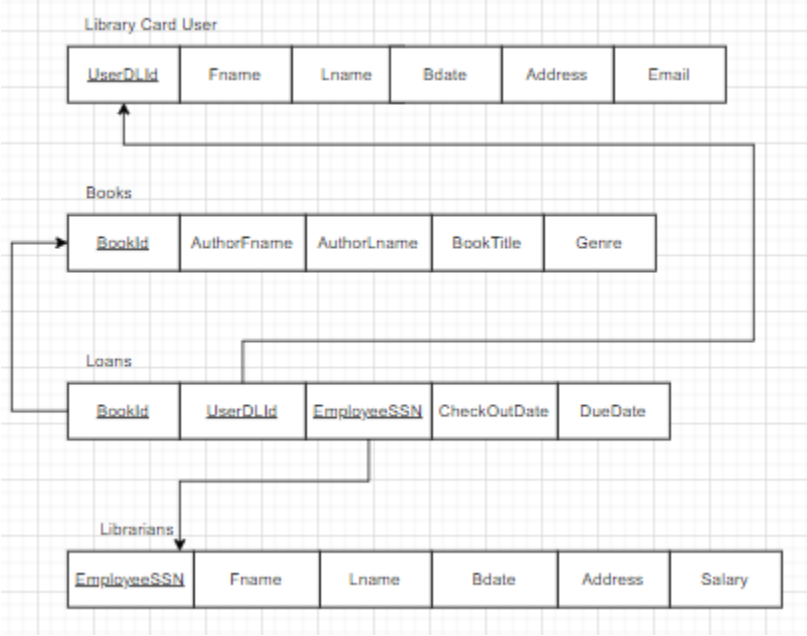
Fname (VARCHAR 30)	Lname (VARCHAR 30)	EmployeeSSN (CHAR 9)	Bdate (DATE)	Address (VARCHAR 30)	Salary (DECIMAL 10,2)

3.3 Business Rules

- The userDLId has to be unique for each user entity. (Uniqueness constraint)
- The BookId has to be unique for each book entity. (Uniqueness constraint)
- The BookId, UserDLId, and EmployeeSSN have to be unique for each loan entity. (Uniqueness constraint)
- The EmployeeSSN has to be unique for each librarian entity. (Uniqueness constraint)
- Each entity had to have every field filled out for each database entry.
- A loan can only be taken out by a library card user.
- A library can be a library card user.

4. Logical Database Schema

4.1 Schema Diagram



4.2 Schema SQL Statements

Creating the database

```sql
CREATE DATABASE library
USE library;

-- Create the Library Card User table
CREATE TABLE LibraryCardUser (
    UserDLId CHAR(8) NOT NULL,
    Fname VARCHAR(30),
    Lname VARCHAR(30),
    BDate DATE NOT NULL,
    Address VARCHAR(30),
    Email VARCHAR(100),
    PRIMARY KEY (UserDLId)
);

-- Create the Books table
CREATE TABLE Books (
    BookID INT NOT NULL,
    AuthorFname VARCHAR(30),
    AuthorLname VARCHAR(30),
    BookTitle VARCHAR(100) NOT NULL,
    Genre VARCHAR(50),
    PRIMARY KEY (BookID)
);

-- Create the Loans table
CREATE TABLE Loans (
    BookID INT,
    UserDLId CHAR(8),
    EmployeeSSN CHAR(9),
    CheckOutDate DATE NOT NULL,
    DueDate DATE NOT NULL,
    FOREIGN KEY (BookID) REFERENCES Books (BookID),
    FOREIGN KEY (UserDLId) REFERENCES LibraryCardUser (UserDLId),
    FOREIGN KEY (EmployeeSSN) REFERENCES Librarians (EmployeeSSN)
);

-- Create the Librarians table
CREATE TABLE Librarians (
    EmployeeSSN CHAR(9) NOT NULL,
    Fname VARCHAR(30),
    Lname VARCHAR(30),
    Bdate DATE NOT NULL,
    Address VARCHAR(30),
    Salary DECIMAL(10, 2),
    PRIMARY KEY (EmployeeSSN)
);
```

4.3 Expected Database Operations and Estimated Data Volumes

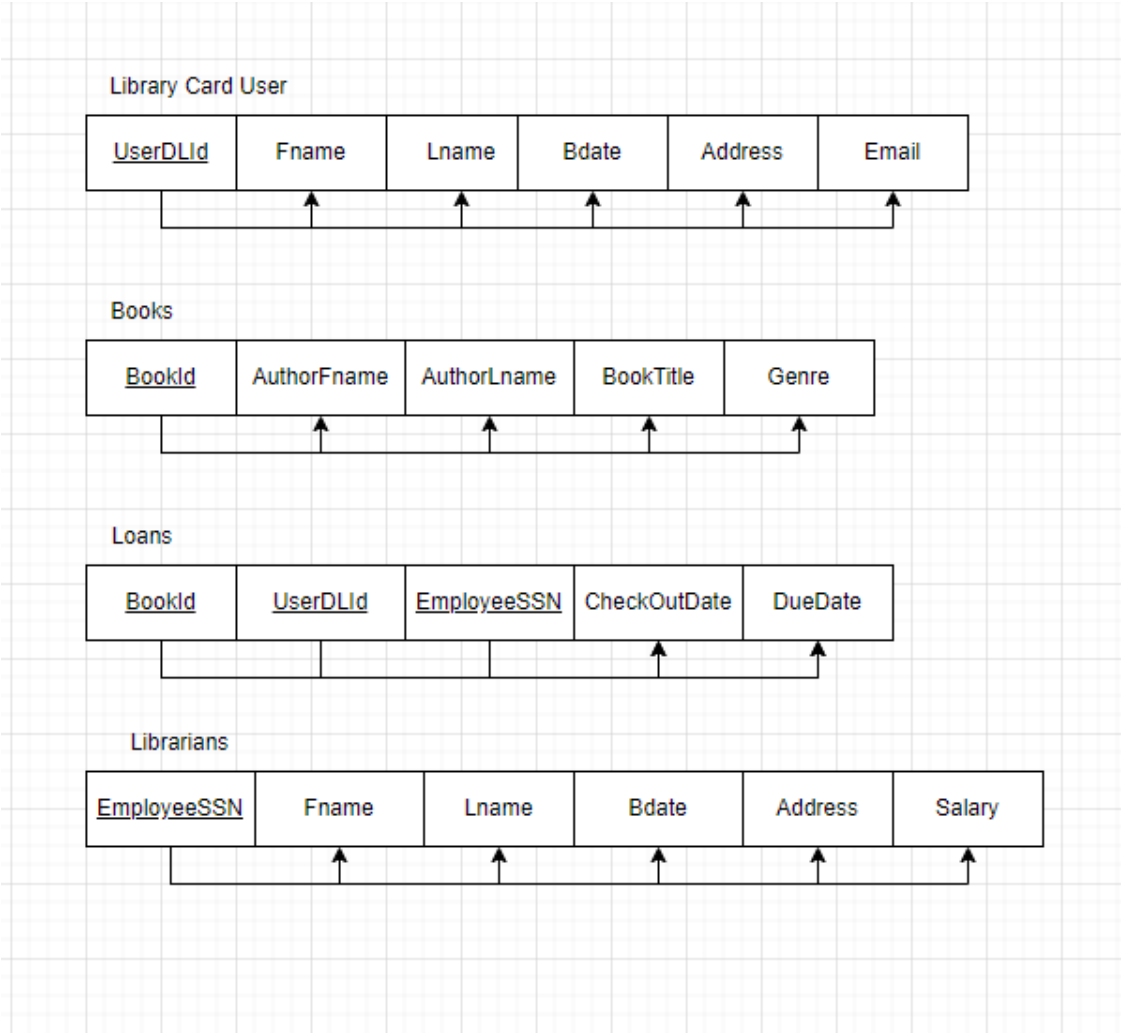
Expected Database Operations

- Insertion
- Deletion
- Update
- Retrieval

Estimated Data Volumes

The data that is inputted must be valid and pertain to the rules of the database in order to maintain the integrity of the database system. In terms of volumes, our application can withstand large amounts of inputs but has limitations past an arbitrary large number.

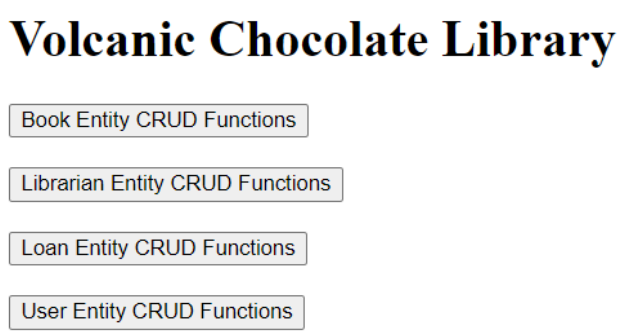
5. Functional Dependencies and Database Normalization



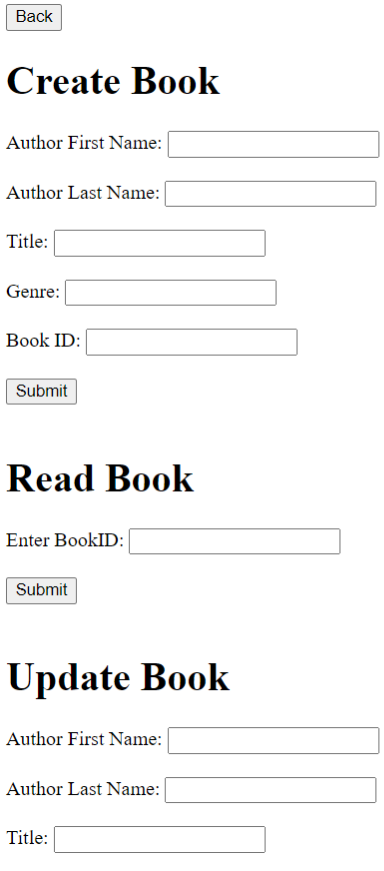
6. The Database System

To download the code and run the website locally, go to this link: https://github.com/vvyn/volcanic-chocolate-library.

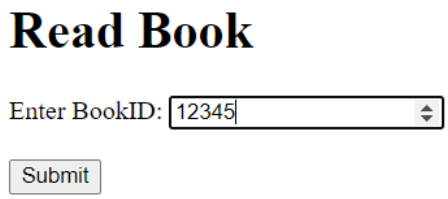
Open the website to this page:



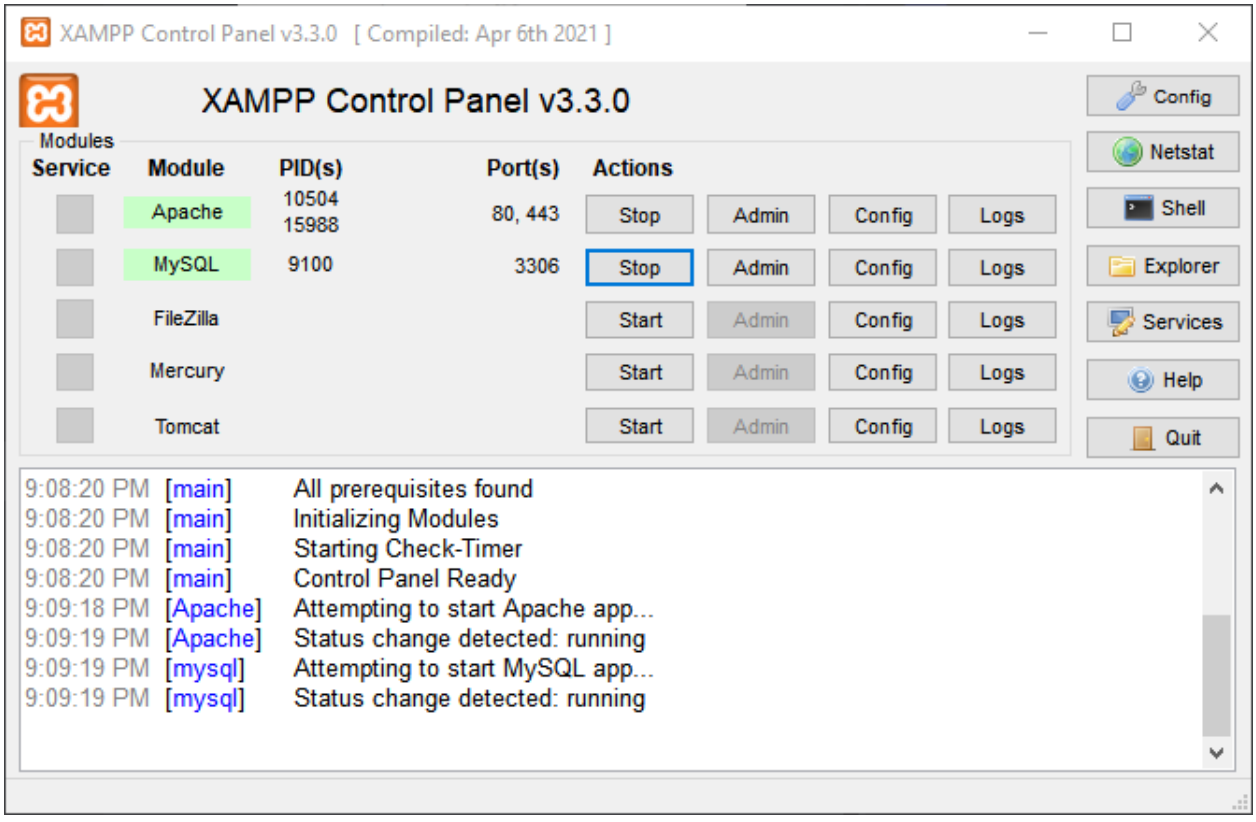
Then in order to execute a CRUD function on an entity, click on one of the buttons such as the Book one to see this web page:



You can press the back button to select a different entity to execute a CRUD function on, or scroll down to the function you wish to execute. For example, to execute the read book function enter the BookID and click submit:

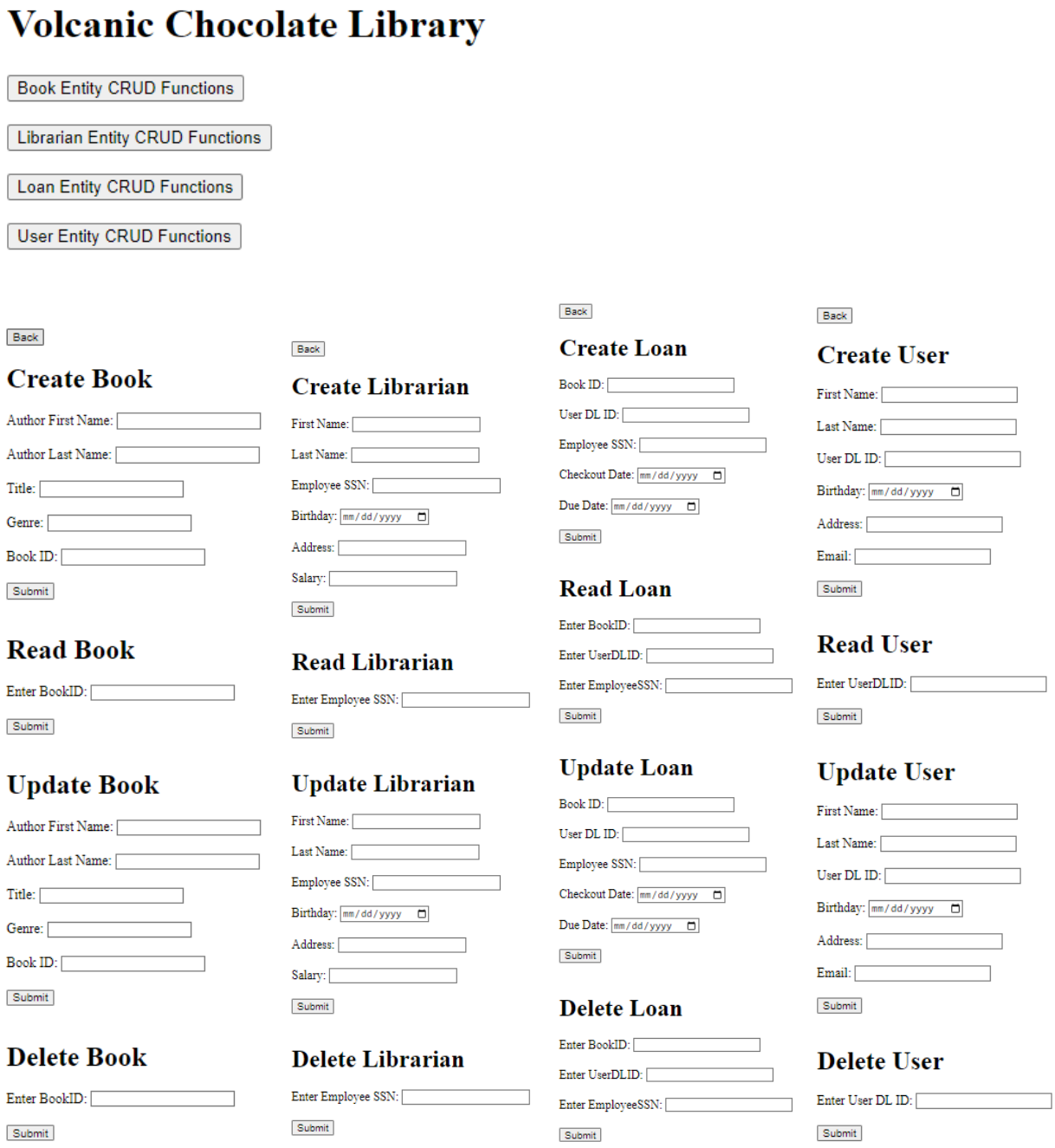


In order to get our system to work, you have to install XXAMP and Apache onto your computer. You can do so by going to this website: https://www.apachefriends.org/.



7. User Application Interface

The user application interface was built using HTML and JavaScript code. This was uploaded to GitHub and hosted on GitHub Pages. Users can access the website at this link: https://vvyn.github.io/volcanic-chocolate-library/. Users get presented with the choice of 4 buttons related to the 4 entities (Book, Librarian, Loan, User) to do a CRUD function on. Once a user chooses an entity, they get taken to a page to execute the CRUD (create, read, update, delete) functions on the entity.



8. Conclusions and Future Work

This project was a learning experience for our team because we haven't used SQL or PHP before. We've also never created a database before. We've all worked with HTML, CSS, and JavaScript before, but learning how to integrate those code files for a website with PHP and XXAMP was a long experience with many mishaps.

Possible future work may include updating the UI of the website and adding more CSS styling. Currently, it just uses HTML tags and elements with very minimal styling. Additionally, the organization of the website. Currently we have the CRUD functions separated by entity. We could create a separate web page for each CRUD function for each entity.

The database can also be expanded to include more entities and fields for each one. For example, adding a Genre entity with specific tags and ids as fields.

9. References

Ojha, Raghwendra. “How to Connect HTML to Database.” Raghwendra, 29 Dec. 2018, www.raghwendra.com/blog/how-to-connect-html-to-database-with-mysql-using-php-example/

10. Appendix

/doc
    Group17-Phase5
    Slides
/project
    Code
README