**User Documentation:** The first screen the user sees is the welcoming page, the user has the option to login if they already have an account. If they do not have an account, they will also have the option to create an account using the signup button. This will bring them to a page where they will need to enter their first and last name, email, and a password. From there, the user will need to go back to the login page and enter their new details. If the user tries to login with an account details that are incorrect, they will be prompted to try to login again. If the user logs in with valid account details, they will be brought to a page where they can search for books based on their the title, author, or genre. From there a list will be brought up under the search bar that relates to their search. The user will then be able to save a book to their profile with the save book button. The user can then view all of their saved books by going to their saved page. On their saved page, they will be able to unsave a book which will remove it from their saved book list.

**Technical Documentation:**

On the client side, the main three interconnected HTML files are Librapp.html, signup.html, and search.html. They serve respectively three main purposes: login, registration, and book searching. The Librapp.html will refer to signup.html if registration is needed, and then refer to search.html after the login is done. Moreover, we have our own custom style.css and Librapp.css files for the aesthetic of the website. However, these html and css files only determine the structure and design but the functionality of the webpage. Making the website to work requires the javascript parts.

For Librapp.html and signup.html, the javascript codes are inserted internally into the html files. Those codes take the user inputs (name, email, password) and send them to the server side. For the search.html, the javascript file api.js is referred externally. Based on the user choices on searching metrics (by author, by title, by genre, trending), it sends the according call to the server side using JQuery and Ajax calls. Now, it comes to the job of the server side to handle received data and respond back to these calls.

On the server side, there is a server listening to HTTP requests and sending back HTTP response messages. The request messages usually contain user information and the server needs to verify and update them. In order to do so, the server needs to connect to a database that helps with update, verification, and book queries.

In summary, the main work flow of the application is as follows. The first program that starts off the application is the Librapp.html, this opens the html website and shows the login. This program also handles the redirect to the create sign up page which is controlled by the signup.html. From there once the user creates a sign up and goes back to the login screen and they will log in. When creating an account the UserProfile.java saves the account to the database. When they enter their credentials, the database will be called to check if that account has been created. If it has, they will be transferred to the search screen, if not they will be told to try again. Once on the search screen, search.html will control the search function and call to our database and api to search for the books entered by the user.

**Installation requirements:**

1. Download the Librapp folder (from Github).
2. You will need some version of Java (Java 18 works, other versions may work but 18 definitely will). This can be found at this link on [oracle.com](https://www.oracle.com/java/technologies/downloads/#jdk18-windows).
3. Load the driver “mysql-connector-java-8.0.29.jar” from Librapp folder to run “Server.java” file
   1. **On Windows:** 
      1. Type “**for %i in (java.exe) do @echo. %~$PATH:i**” in your command prompt to find the directory of your jdk version.
      2. From that directory, go to the folder \jre\lib\ext.
      3. Copy the file “mysql-connector-java-8.0.29.jar” into that folder.
      4. Open the command prompt and change current folder to Librapp
      5. Enter “**javac Server.java**” for compiling and then enter “**java Server**” into the command prompt to run “Server.java”
   2. **On Mac:**
      1. Open the terminal and go to the Librapp folder
      2. Enter “**javac -classpath “.:mysql-connector-java-8.0.29.jar” Server.java**” for compiling.
      3. Enter “**java -classpath “.:mysql-connector-java-8.0.29.jar” Server**” to run “Server.java”.
4. Open Librapp.html in your browser

**Team's technical skills and goals:** Our team consisted of 5 people, Nic Sarkis was the project manager, Henry Pierce was the technical manager, Nam Hoang was the back-end developer, Annie Dima was the database developer/administrator, and Jake Hoffman was the front-end developer. Each person contributed different things to our group to make our project come to fruition. Everyone participated in creating different models/diagrams. Nic managed all of the customer interaction and assisted in the creation of the book class (this class is no longer used). Henry created the search/api functions using the google books api, we later switched this to our own database so that we could implement our own search functions/filters. However the api that we used was great so that we could test other parts of our project without having to create a database first. Nam worked out the create profile system, he managed to make the profile saving feature work on his computer then worked with us to make it work on all of ours. This was a very big step for us as the implementation/linking of all of our code worked though his breakthrough. Annie worked on the SQL databases, that being both the user profile and the book database. She created the data that is currently used in our final submission. Like we mentioned earlier, the only way we were able to make our sort functions work was to create our own book database. Jake created the html code that the user sees and made is aesthetically pleasing. There was a lot that we needed to learn, none of us had taken any html classes, only Annie had taken a sql class, and we had to learn what a search api was and how to use it. Our overall goal was to create a webpage that had the ability to store user profile details and save books to their profile. We were able to achieve this goal.

**External Libraries (Citations):** The first library we used was Bootstrap. We got it from getbootstrap.com, and used it in search.html to make the search page look better. It was used throughout search.html to help with formatting.

Another library we used was jQuery. We got it from jquery.com. It was used in api.js and Librapp.html to create an ajax connection to Server.java so we could query and get results from our database.

The third libraries we used were Java HTTP Server imported from “com.sun.net.httpserver” and “java.net.InetSocketAddress”. They were used to handle a HTTP request from the client and return a HTTP response back to the client side.

The final library we used was the JDBC API imported from “java.sql”. It was used to execute SQL requests to a database server and receive a response back.

**Bonuses:** Our group has attempted a couple of the extra credit opportunities. We were able to make 2 sql databases and implement them into our code. We created a sql database for our books and our user profiles. Our program is also a web application that can run on any browser since it is html based.