# Software Implementation and Testing Document

# For

# Group 17

Version 1.0

**Authors**:
Bryce B
Phuong N
Mason H

## 1. Programming Languages (5 points)

Python - This language constitutes the majority of our project and is used in app.py to define the bulk of our application's functionality. This was chosen for its relative ease of use and our team's comfortability in programming in it. Python also works well with HTML files through Flask and with SQL through SQLite3.

CSS - This was used to style our web pages and its compatibility with HTML.

SQL - Used in our database components to our application, this manages the user logins, books, reading lists, and user reviews. We are also using SQL to write triggers which automatically change average_rating to reflect new reviews created on our website.

HTML - Chosen for familiarity and its ability to work with Python. In particular, we use this for the web page templates for our .html files.

JavaScript - Used very briefly in review.html to insert a function which enables the "Leave a review" menu to appear and disappear dynamically. Chosen for its compatibility with HTML.

## 2. Platforms, APIs, Databases, and other technologies used (5 points)

GoogleBooksAPI - We are using this API to search for books and return all of the information about a book (page count, author, cover image, etc.) and split the data into an object of class Book which is usable with the rest of our website's functions. This API is core to our project's ability to search for books in order to review or log them.

InitializeDb.py - We are using this file to initialize our SQL databases which stores user logins and associated user data, user reviews, book data of recently searched books, and reading lists.

## 3. Execution-based Functional Testing (10 points)

Our functional testing was conducted by actively interacting with our website and performing various tests to ensure that all functional requirements listed in our RD were met. We tested the system from both user and non-user perspectives, in order to test the different permission levels. For example, we verified that logged-in users who chose to be "Readers" could successfully add a book to their "My Books" section. Additionally, we tested that non-users attempting the same action were correctly redirected to the login page, and to give users a personalized page depending on their roles. Through frequent testing and our hands-on approach, we were able to identify and address any issues in user access control and functionality as they arose.

## 4. Execution-based Non-Functional Testing (10 points)

While we did not conduct extensive non-functional testing, certain aspects were naturally evaluated as we used the website throughout development. For example, we frequently interacted with the site and ensured that page load times weren't excessive, such as queries taking more than three seconds. Similarly, usability and stability were informally assessed as we navigated the site, tested features, and fixed issues as they arose. Although these aspects were not rigorously tested, our continuous usage provided a general assurance of performance and reliability.

## 5. Non-Execution-based Testing (10 points)

Our non-execution-based testing is primarily done through code reviews and inspections in which we review the commits and documentation left by one another on GitHub. We also communicate

what we’re doing and the code that we’re submitting with one another on a shared discord. Outside of these methods, we don’t employ any other non-execution-based testing styles.