**INDY 1 RED — Book Selector App**

**CS 4850 - Section 03 – Fall 2024**

**September 1, 2024**

| Francesca Del Aguila  (Team Leader)  Development, Website | Thomas Roberts  Documentation, Test | Tristan Sanford  Development, Website |
| --- | --- | --- |

**Team Members:**

| Name | Role | Cell Phone / Alt Email |
| --- | --- | --- |
| Francesca Del Aguila (Team Leader) | Development, Website | 678.920.0165  francescydel23[@gmail.com](mailto:student2@gmail.com) |
| Thomas Roberts | Documentation, Test | 404.834.1861  thomas@bwroberts.com |
| Tristan Sanford | Development, Website | 706.461.3929  tristanrsanford@gmail.com |
| Sharon Perry | Project Owner, Advisor | 770.329.3895  [Sperry46@kennesaw.edu](mailto:Sperry46@kennesaw.edu) |

**Table of Contents**

[**1. Introduction and Overview 3**](#)

[**2. Design Considerations 3**](#)

[2.1. Assumptions and Dependencies 3](#)

[2.2. General Constraints 3](#)

[2.3. Development Methods 4](#)

[**3. Architectural Strategies 4**](#)

[**4. System Architecture 5**](#)

[**5. Detailed System Design 5**](#)

[5.1. Classification 5](#)

[5.2. Definition 5](#_zha097rxrdop)

[5.3. Internal Structure 6](#)

[5.4. Constraints 6](#)

[5.5. Resources 6](#)

[5.6. Interface/Exports 7](#)

[5.7. Design Mockups 8](#_rblcmxazej4m)

[**6. Software Deployment 9**](#_cwu4472sa1um)

[**7. Glossary 9**](#_m8whmbbsdc7u)

[**8. Bibliography 10**](#)

[**9. Appendix 10**](#_3n515aced918)

# **Introduction and Overview**

The Book Selector app is a mobile application designed to enhance book discovery through a swipe-based interface similar to dating apps like Tinder and Hinge. The app helps users efficiently find new books based on their interests while also providing a platform for authors and publishers to reach readers.

Users can swipe left to skip a book or swipe right to add it to their To-Be-Read (TBR) list. If they are unsure about a book, they can scroll down for more details, like key tropes, genre, and quotes.

This document outlines the app’s architecture and detailed design specifications, modifying the requirements from the SRS into a technical guide for developers. The intended audience includes the project manager, the design team, the development team, and the testing team. The overall design includes a frontend user interface, a backend server for managing data, and connections to third-party APIs for book info. The detailed design specifications are below.

# **Design Considerations**

For the Book Selector app to be finalized, there are key issues that need to be addressed, which include making sure there is compatibility across mobile devices and operating systems(OS), make sure the external book data APIs are manageable for us to implement, and for us to optimize performance for a medium to large datasets.

## ***Assumptions and Dependencies***

While developing the Book Selector app, it is assumed that the users will access the app through mobile devices running Apple iOS and Android platforms. The app will be optimized for these operating systems, and it is expected that users will be familiar with the swipe-based interface.

It is assumed the external API that will be used will be reliable in retrieving book data and having a consistency of information presented to the users. This will mean our algorithm will provide the best way to sort our users to their preferences with our external API integrated in the app.

## ***General Constraints***

* API Limitations:
  + Google Books API has a daily cell cap, which restricts the number of book matches users can receive daily.
  + A cap on the number of swipes will be based on the available API quota.
* Mobile Device Limitations:
  + Efficient coding practices required due to limited processing power and memory capacity
  + Having smooth performance and quick load times are important.
* Security Considerations:
  + User data will require us to be compliant with data protection regulations.
  + Secure authentication methods will be established to secure sensitive data as necessary.
* Network Communication:
  + Dependent on external data sources for book information may limit the communication due to the API restrictions.
  + The app will need to remain responsive and able to work when there are large amounts of book data in the system.
* User Experience(UX) design:
  + The app must provide a user-friendly interface to make sure that there is a positive experience while using the app.
  + The app should be designed to accommodate different screen sizes on phones to make the appearance not awkward.
* Verification and validation requirements:
  + The app will undergo testing to ensure it runs smoothly and meets performance and quality standards
  + A group of testers will use the app on various platforms and devices to ensure it works correctly for a wide range of users under different conditions.
* Data management:
  + Data must be managed to ensure the app can handle large datasets without slowing the app down.

## ***Development Methods***

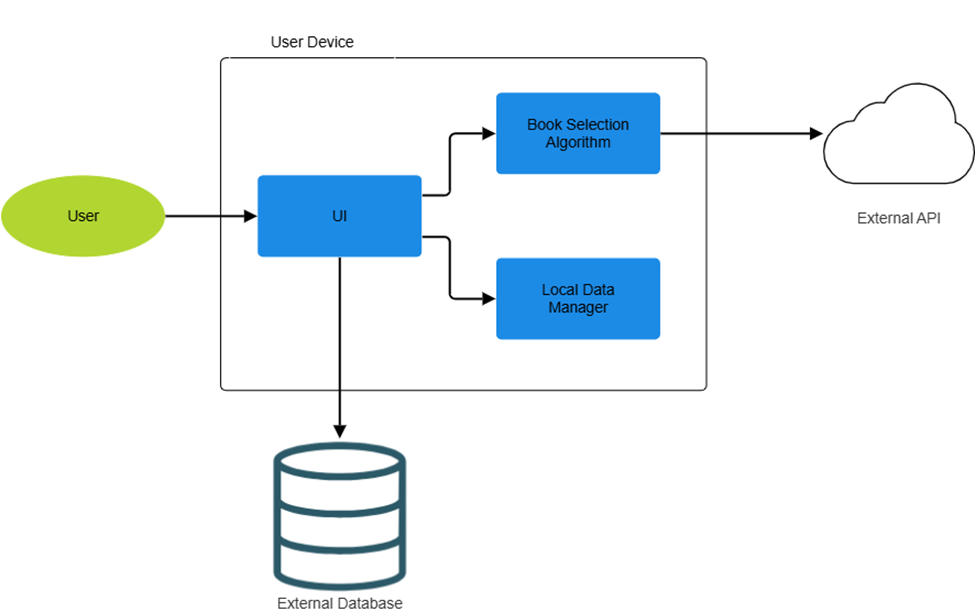
We will be using a hybrid approach that combines primarily waterfall and some agile methodology. Our methodology is influenced by the nature of the class which promotes the waterfall method through assignments (linear progression that matches the class progression). This gives our project more structure and forces us to work sequentially which is ideal for ensuring diligent planning and completion. However, the waterfall method does not allow for additional features to be added during a single iteration in the development process. This project may iteratively add more features or fix bugs as needed towards the end of the project which is representative of agile development.

# **Architectural Strategies**

We are using React Native to develop our application. This impacts our architectural strategies as we are only required to write the program once in JavaScript instead of having to write multiple programs in iOS or Android’s native language. The downside is that our application may not perform as efficiently, but performance is not a quality that needs to be optimized at this stage.

Our program will utilize object-oriented programming styles. The data and functions regarding the data are closely related, and OOP will help manage the data. For instance, a book object would be optimal for storing data such as ID, title, author, synopsis, genre, and tropes. Since we are focused on data management, we are disinclined to use procedural and functional programming styles.

# **System Architecture**

**

*Diagram 1. Application Architecture*

The user interface (UI) is designed to handle user input and manage information exchanges with the Book Selection, Local Data, and Database components. The Local Data component manages details about the TBR (To Be Read) and DNF (Did Not Finish) lists, which should be directly accessible to the user. The Book Selection component requires access to Local Data to generate recommendations and connect to an external API. This API, such as Google Books, will provide the actual book information. Lastly, user login details and profile settings will be stored in the Database component.

# **Detailed System Design**

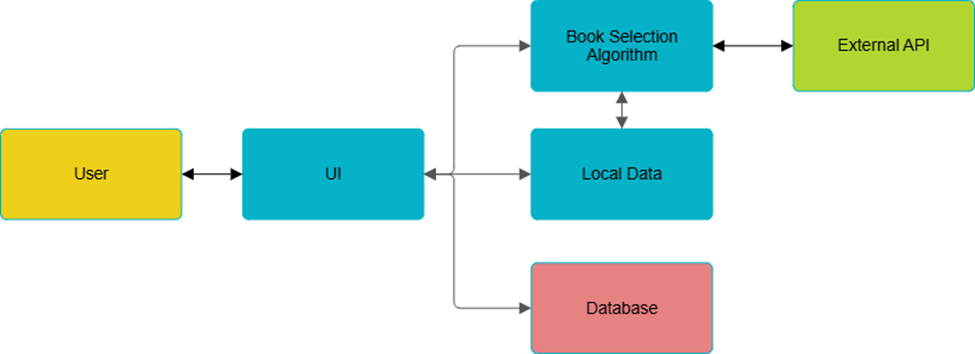
## ***Classification***

Our project is a mobile application that provides data discovery services. It is composed of higher-level classes (e.g. UI) and lower-level functions. Since we do not know all of the details for each system and how they will interact with each other, we grouped all of the systems for this section and gave general details. After our early, initial prototype phase, we will be able to update this section and provide the granular details, a classification, definition, constraints, resources, and interface/exports, for each system.

## ***Definition***

Despite being classified as a data discovery service, our app is not intended to provide experiences such as business-driven performance enhancement. Instead, it engages with users similar to social networking services such as Hinge. The application does provide insight in the form of book recommendations. For example, the app may recommend books by genre or author because the user interacts with similar genres or authors.

## ***Internal Structure***



*Diagram 2. Information Flow*

The internal structure is illustrated by Diagram 2. Each class is an individual component that has its own functions. This structure is listed below:

* User Interface Class
* Login Page
* User Page
* Discovery Page
* Library Page
* Archive Page
* Book Selecting Algorithm Class
* Local Data Manager Class
* Database Class
* External API Class

## ***Constraints***

The software may be limited by the user’s bandwidth. Since book information is downloaded from the external API, the app may slow down while downloading data. Additionally, there may be a limit to the number of queries the external API will accept. This means that if the user reaches this limit, the app may cease functionality until the API limit resets.

The backend and database handle login requests, account management, and other user data. This could pose a limitation if too many requests are sent within a given time.

## ***Resources***

This application requires local memory, processing power, and a strong internet connection. There are no race/deadlock conditions since all data is tied to an individual user and individual user actions are sequential. Another resource will be the external API which is responsible for returning book information. Lastly, the mobile app will utilize a backend server/database for user login, account management (reset password/change email), and other user data.

## ***Interface/Exports***

* 1. User Interface Class

Login Page

* function displayLogin().
* function displayCreateAccount().
* function displayResetPassword().

User Page

* function displayUserProfile().
* function displayUserPreferences().
* function displayNotificationPreference().

Discovery Page

* function displayCurrentBook(). This function pops a book from the global object “queue” and displays the book.
* function displayBookInfo(object currentBook). This function displays the extra information stored in the object currentBook.

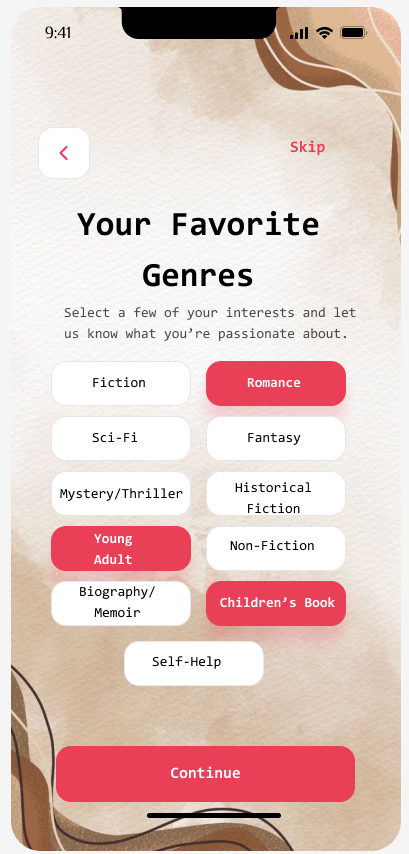
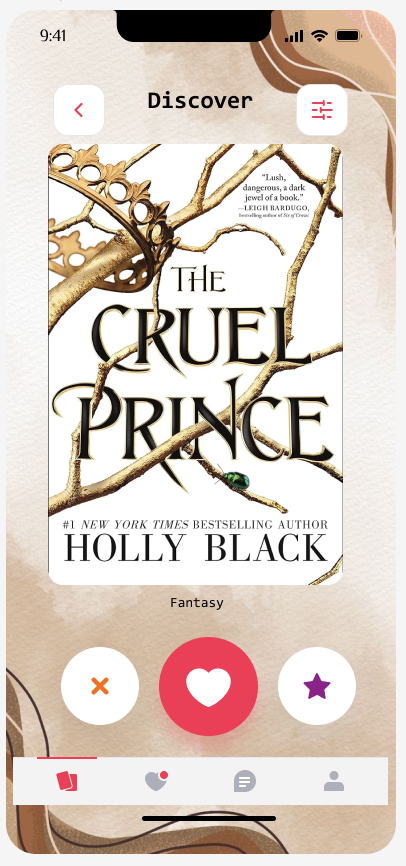
Library Page

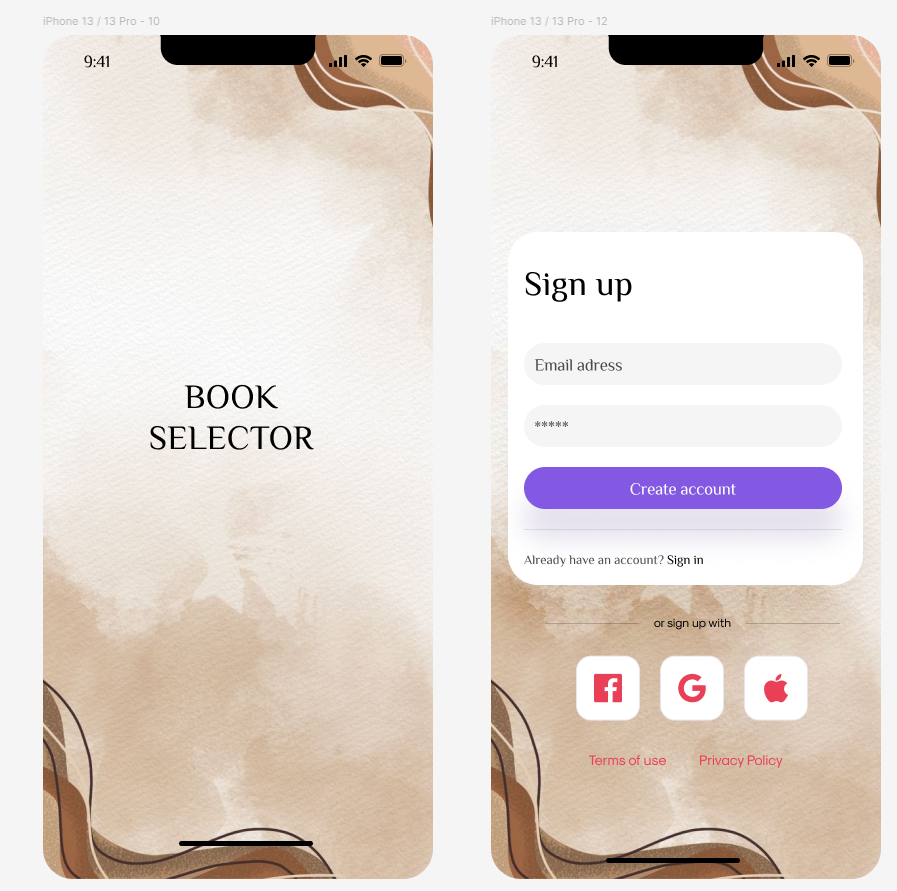
* function displayTBRList().
* function displayBook(var bookID).

Archive Page

* function displayDNFList().
* function displayBook(var bookID).
  1. Book Selection Class
* function generateBookRating(object book).
* function sortRecommendations(object queue).
  1. Local Data Manager Class
* function addBook(var listID, object book).
* function removeBook(var listID, var bookID).
* function getTBRList(). Returns TBR list.
* function getDNFList(). Returns DNF list.
* function storePreferences(var preferences).
  1. External Database Class
* function login(var username, var password).
* function createAccount(var username, var password, var email).
* function resetPassword(var email).
* function changeName(var username, var email).
* function changeEmail(var username, var email).
* function changeNotification(object settings).
  1. External API Class
* function getBooks().

## ***5.7. Design Mockups***





# **Software Deployment**

Our application will be developed using the React Native framework which generates native code for Android, iOS, and web-based platforms. We will use the Google Books API to extract book data for our recommendation algorithm.

# **Glossary**

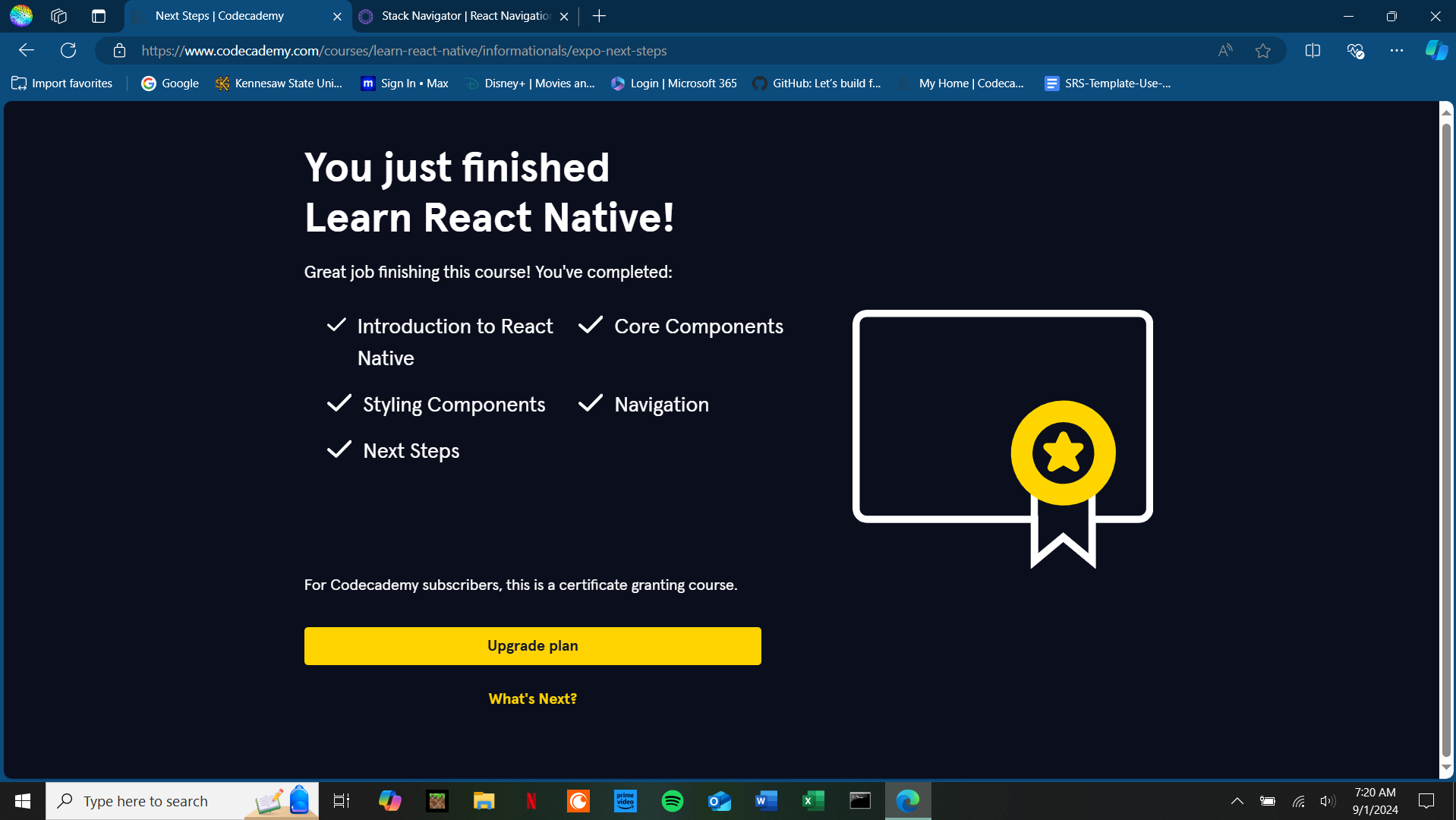
1. **TBR:** “To be read;” the list of books a user wants to read
2. **DNF:** “Did not finish;” label for when a user does not finish a book
3. **UX**: User Experience
4. **OS**: Operating Systems
5. **UI**: User Interface

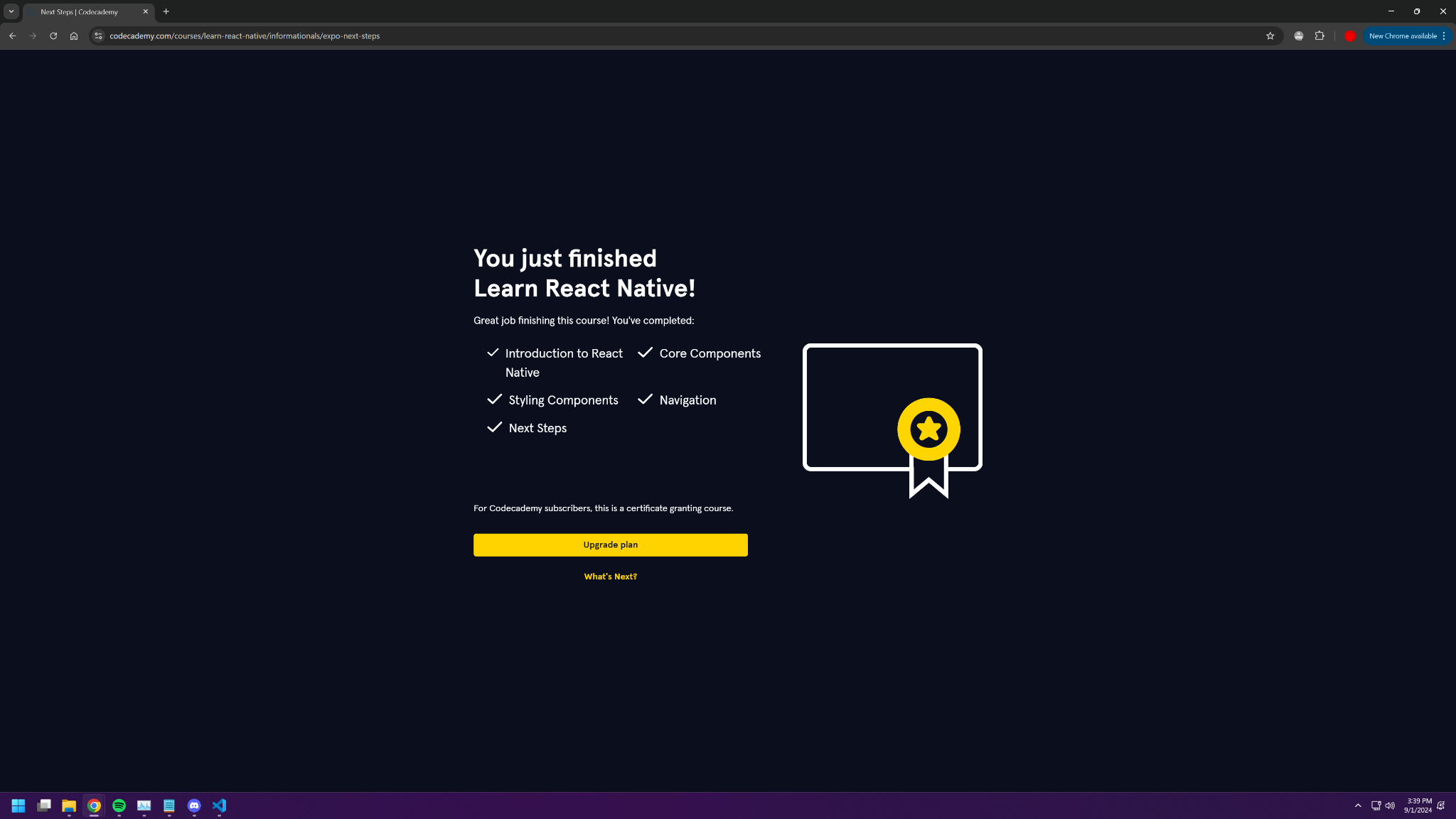
# **Bibliography**

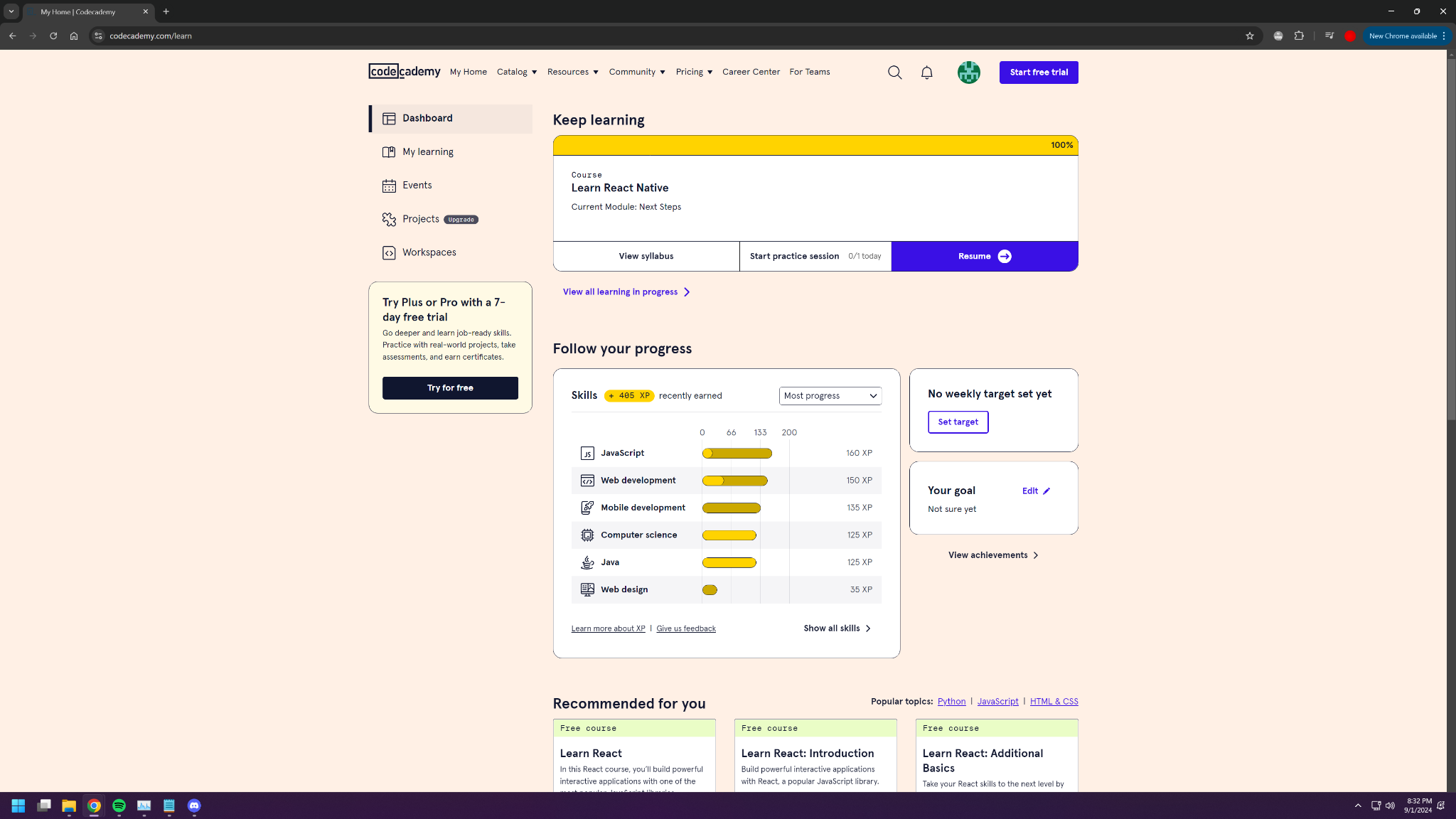
McGuire, M. (2024, March 24). *Top 4 software development methodologies*. Synopsys. Retrieved August 30, 2024, from <https://www.synopsys.com/blogs/software-security/top-4-software-development-methodologies.html>

# **Appendix**

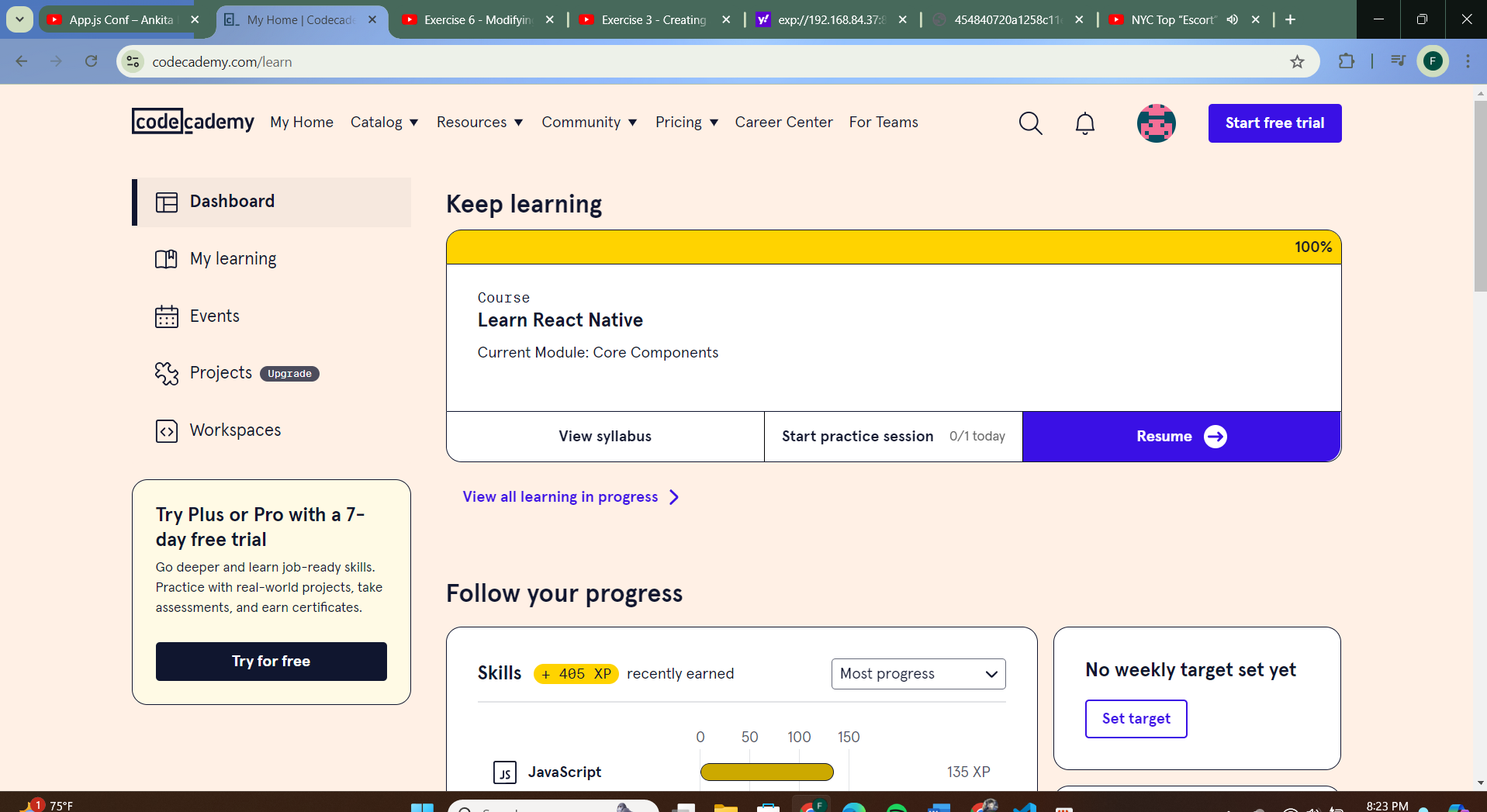
Proof of completion - Thomas Roberts:



Proof of completion - Tristan Sanford:  




Proof of completion - Francesca Del Aguila:



# 3.0 Functional Requirements

## 3.1 Login Page

3.1.1: Login with username (email) and password

3.1.2: Create account with email and password (must verify email)

3.1.3: Reset password

## 3.2 User Page

3.2.1: Manage user profile: changing name and email

3.2.2: Set user preferences for book genres, authors, and tropes

3.2.3: Disable app notifications

## 3.3 Main Swipe-Based Discovery Page

3.3.1: After being shown a book from the recommendation algorithm, swipe right to add the book to your TBR list or swipe left to advance to the next book

3.3.2: Tap on the recommended book to see extra information such as the book author, genre, quotes, and related reviews.

## 3.4 TBR Library Page

3.4.1: View a library of TBR books added from the app

3.4.2: Tap on a book in the library to view extra information such as the book author, genre, quotes, and related reviews

3.4.3: Tap on a book in the library to rate it (including DNF) and move it to the archive

## 3.5 Archive Page

3.5.1: View a library of read/dismissed books from the TBR Library

3.5.2: Tap on a book in the library to view extra information such as the rating given, book author, genre, quotes, and related reviews and remove from the archive

## 3.6 Recommendation Algorithm

3.6.1: A custom algorithm will recommend books for the main page based on user preferences, previous swipes, and book ratings

## 3.7 Notifications

3.7.1: Notifications will be sent to remind users to find a new book

## 3.8 Backend

3.8.1: User data will be stored and retrieved via a backend service

3.8.2: Book data will be extracted from an external API