Software Requirements and Design Document

For

Group 26

Version 1.0

Authors:
Kyla Henry
Brian Ho
Beck Boyd
Holly Lapini
Flora Charles

Red = increment 1, Blue = increment 2, green = increment 3

1. Overview (5 points)

Give a general overview of the system in 1-2 paragraphs (similar to the one in the project proposal).

2. Functional Requirements (10 points)

Home must display the current reads that the user may choose to continue reading and it should also display the top reads by other users. Displaying the profile icon for the user in the top right corner that links to the users profile is also another feature that needs to be created and implemented. These are the features of high priority for the home page. The current and top reads should be a horizontal scrolling view where the information continues to display up to 10 current reads that the user is reading and up to 10 top reads by other users.

Search must include a search bar at the top of the page and browse tags below for the user to select the type of story category they wish to read. This is of high priority once the home page is established. The system needs to be able to search stories, profiles or hashtags and have a queue of information display based on their search.

Notifications must include information ranging from story updates to authors posts to inbox messages. This is of low priority as it is an element that shouldn't take long to implement but should instead be required information from the backend to display results on the frontend.

User profile needs to be implemented for this increment. This is of high priority as it needs to connect with the user profile icon on the home page and display user information such as following count, work count, a biography and reading lists they have made public. This is crucial for the continuation of the app as settings then need to be implemented and nested within this page.

The library feature needs to also be implemented in this increment containing a vertical scroll of books the user is currently reading or has read. This needs to be a continuous vertical scroll of all the books. This is of high priority. There should be a pop up that displays when the user holds down the book presenting options such as add to reading list, remove from library, story info and to share the story. These sub features are of medium importance as they are dependent on the main library feature.

Write is the last feature that needs to be implemented in the application. Since this is the last one, it is of high priority. This feature needs to have the options to create a new story or edit/continue a current one. It also needs to be able to upload an image for the cover of the story, a text box for the description of the story and a continue button that leads them to be able to have a huge text box to add the title and a larger text box below to add the story. It should also have an option button menu in the top right corner with options to save, delete or publish. This feature is expected to take the longest, so the sub features are of medium priority.

3. Non-functional Requirements (10 points)

The app must load all pages and user interactions (e.g., opening a book, navigating chapters) within a certain amount of time. This performance of the application is crucial as it ensures a seamless reading experience for users, reducing frustration and maintaining engagement.

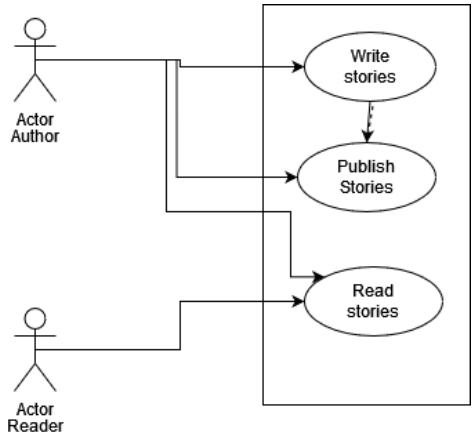
User data including login credentials must be encrypted to ensure authorized access and improve user convenience. It also ensures that the app can support future features that require stronger security features such as payments. We must also implement a security feature to ensure illegal and overly inappropriate content isn't abused on the application in order to provide users with a welcoming environment.

The code base must follow a constant structure to allow for easy updates and reduce technical debt and maintenance cost. Maintainability is important as it also allows for comprehensive documentation to support future development or debugging.

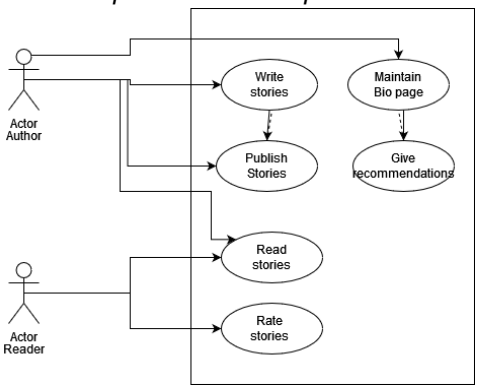
Portability ensures that the content and user data synchronize seamlessly between iOS and other platforms (e.g., Android, web). This enables users to switch devices while retaining their data while also allowing for constant features across all platforms.

Software quality ensures that all updates must undergo automated and manual testing, including regression, performance, and usability tests. This is to ensure that the user experiences a high-quality application, and it also reduces the possibility of bugs and technical debt.

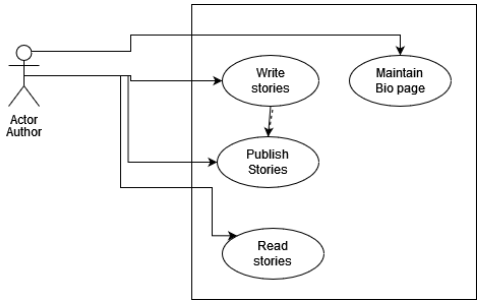
4. Use Case Diagram (10 points)



- *Authors will need to be able to access a "write" tab that allows them to create a "story" using a text box or image upload. The "story" will need to save the title, the chapters, and the written or uploaded information.*
- *Authors will need to have the ability to "publish" their stories. This toggles the "reader's" ability to see the available stories.*
- *Both actors will be able to "read" stories in scrollable chapters format. Stories will have a main page with all of its information (such as title, author name, description) and separated into chapters*



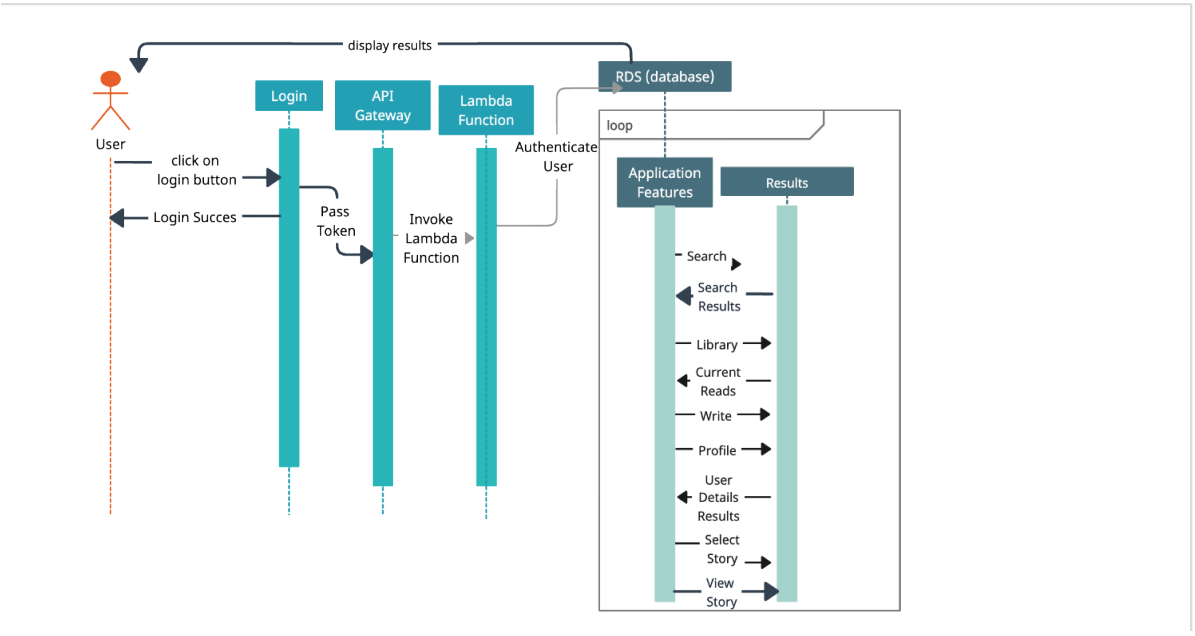
- *Authors will need a new tab that allows them to view and edit a simple biography about themselves*
- *A "recommendations" box will be under the biography, which will be editable by authors and contain recommended titles of stories on the app*
- *Readers will be able to post public "ratings" to stories which will be viewable on the story's main page*



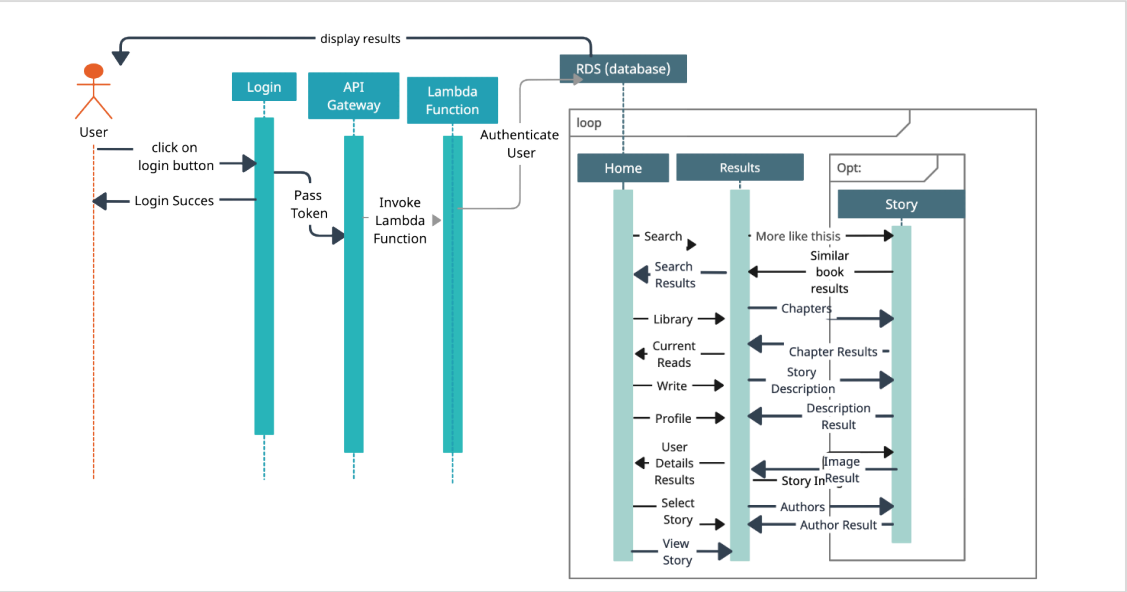
- *A switch to focusing on the "authors" has us drop "readers" entirely*

5. Class Diagram and/or Sequence Diagrams (15 points)

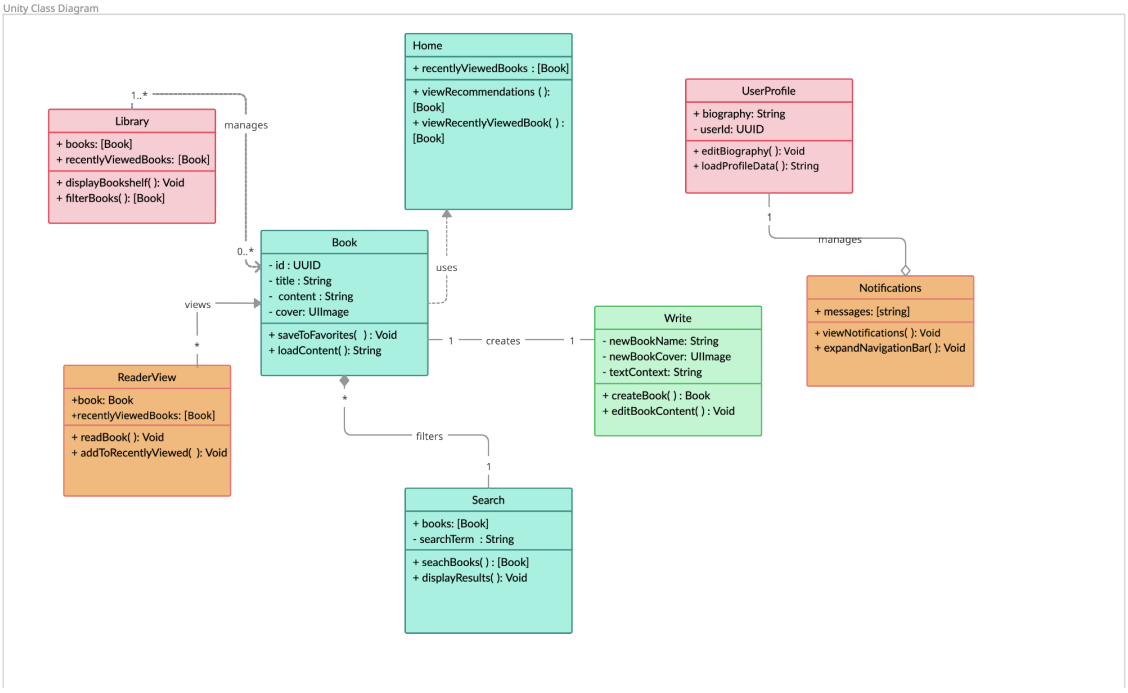
Increment 1:



Increment 2:



Increment 3:



6. Operating Environment (5 points)

The software will operate on the EXPO platform, which is made to create, build and run applications of both IOS, Android and web. Currently, there are no other software components other than the built in dependencies within the package.json file.

After the switch, the software will operate on the XCode platform, being built and created in Swift. There are currently no dependencies as everything is running dynamically currently, and will soon run on IOS devices.

7. Assumptions and Dependencies (5 points)

Third party authentication for sign-in such as (apple, google, or manual sign in), should remain available and functionable. Any disruption or policy change from these providers could affect user login to the application.

Device compatibility is also an important dependency when building the application. We are assuming that the majority of users will be on devices running iOS 14.0 or later. Supporting older devices or future iOS updates may require additional development effort.

Third party tools such as firebase for push notifications and backend services should continue to be maintained and updated. If any third-party SDKs are deprecated or experience downtime, the app's functionality could be disrupted.

The app will rely on a cloud-based backend for data story, content delivery and user authentication. If there are API changes to Firebase or service outages on the server, the applications performance and reliability can be impacted.