**ICSI499 Capstone Project Report**

# The Red Book


*Project Team*

Kevin Lee (001566787)
Billy Plantin (001460619)
Jean Shin (001481777)
Xavier Lodge (001475223)
.....
College of Engineering and Applied Sciences
University at Albany, SUNY

*Project Sponsor*

Bej'a A. Christmas
Anemoia Studios
anemoia@anemoiastudios.com


05-06-2024

**Acknowledgements**

We would like to acknowledge and thank our sponsor Bej'a A. Christmas and professor Atrey for their patience and giving us the opportunity to work on this project.

**Abstract**

One may find themselves struggling to find accurate or personalized information about the inner working of companies and their prospects, or even overloaded with fragmented information and tumultuous discourse. The Red Book is a mobile application that addresses the need for a comprehensive community-driven platform where users can engage with content specific to their professional interests. The application facilitates user interactions through reviews, posts, and real-time notifications, promoting personal experiences and community engagement. While there are many different types of social networks that are similar to ours, we attempt to consider a variety of features that come together as a unique combination of professional insights and the personalizing of knowledge. By integrating a system of unstifled reviews and interactive micro blogging features, we aim to provide user engagement and a tailored experience to address the lack of personalized and professional knowledge, and the lack of interaction in existing community platforms.

# Contents

# 1 Problem Analysis

In an era characterized by an overwhelming amount of information and disinformation, the Red Book app simplifies the landscape by providing a central platform for social networking and business evaluations. The primary goal of The Red Book is to combine various aspects of social networking and business reviews into a single, user-friendly platform. When observing similar products, it becomes apparent that while certain applications excel in specific functionalities, they often fall short in integrating all the essential features needed for optimal social networking and reviewing services at one place. This unique combination is what The Red Book aims to provide. The integration of this idea poses some challenges, because it is difficult for new products that have similar features to other existing product, to gain traction. The complexity of merging multiple functionalities into a one cohesive application presents technological challenges in terms of the architecture of the system.

Current solutions, such as Glassdoor and LinkedIn, offer platforms for professional networking and business reviews. However, these platforms cater more towards specific demographics or professional groups rather than serving a universal audience. They also either lack aspects of either social networking or personalized reviews. These platforms, while efficient within their niches, fail to provide a service that addresses the needs of all user groups. For example, platforms like LinkedIn are effective in professional settings, but do not support features like business reviews, which can limit this type of user expression and feedback.

The Red Book offers a platform that supports both anonymous and disclosed reviews that can not be commented on by other users, which allows for the sharing of honest feedback without worrying about particular repercussions. Diverse social networking functions also makes The Red Book a key resource for honest business insights and community engagement. The Red Book has a unique combination of features such as a review board, micro blogging features, direct messaging, a live streaming service. The application offers a platform that combines business reviews, social networking, and community engagement, making it a unique application for digital interaction. The Red Book aims to include the general user base to increase professional knowledge-gaining resources and connectivity nationwide.

Key contributions of The Red Book:

- Feedback Mechanism: Offers a structured environment for users to post detailed reviews about their work and educational experiences, promoting transparency and informed decisions.
- Community Building: Facilitates engagement within educational and professional communities by providing a platform for sharing and discussing about personal experiences.

- Interactive Engagement: Enhances user interaction through features like commenting and real-time notifications, keeping the community active and informed.

# 2 Proposed System/Application/Study

## 2.1 Overview

The Red Book is a mobile application of a social network. It contains elements of a review board and micro-blogging features. “Reviews” are in the “home” screen of the application, and are treated separately from “posts” which are displayed in the “campus” screen. Users have an option to create either reviews or posts individually, from the “create” screen. Notifications are also supplied to users in the “notifications” screen to notify them when other users interact with their media. Finally, users directly send and receive messages from other individuals from the “messages” screen.

This section will provide details relating to how each of these screens are organized and navigated, the project requirements, the technical design of our application, and how we applied both computer science and software engineering practices throughout the development of the application.

## 2.2 Project Requirements

We will now detail the user classes and all functional requirements.

### 2.2.1 User Class

Other than the general “user”, there are no additional user classes because all users of the application have access to the same features, possess the same permissions, and their content is treated the same way as all other users.

### 2.2.2 Functional Requirements

- Functional Requirement 1.1
  *Home Screen*
  - Reviews are listed on the home screen in accordance to the geographical distance of the content of the review to the user viewing the review.
  - Users can "like" a review by tapping on a like icon associated with the review, incrementing the like counter.

  - Users can filter the reviews by a specified geographical distance of the reviews shown, from the user.
  - The display content of the review consists of the review title, body, author, and optional images.
  - Users can archive a review, and archived reviews can be accessed on the home screen.
  - Tapping on a review should take you to the focused view of the review.
  - Tapping on the profile picture of the author of the review should take you to the profile screen of the author.

- Functional Requirement 1.2
  *Campus Screen*
  - Posts are listed on the campus screen in accordance to the other users that the user follow, in chronological order.
  - Users can like a post by tapping on a like icon associated with the review, incrementing the like counter.
  - Users can comment on the posts of other users, and comments should be displayed in chronological order.
  - The display content of the post consists of the post title, body, author, and optional images.
  - Users can archive a post, and archived posts can be accessed on the campus screen.
  - Tapping on a post should take you to the focused view of the post.
  - Tapping on the profile picture of the author of the post should take you to the profile screen of the author.

- Functional Requirement 1.3
  *Create Screen*
  - Both reviews and posts can be distinguished and chosen to be created from the create screen.
  - Reviews to be created must consist of a title, name of the organization being reviewed, zip code of the organization being reviewed, the review content, and optional images, to be uploaded.
  - Posts to be created must consist of a title, the post content, and optional images, to be uploaded.

- Functional Requirement 1.4
  *Notifications Screen*

  - A notification must be created when another user interacts with content owned by the user, such as liking and commenting on posts, reviews, and comments.
  - Tapping on a notification should take you to the focused content of the notification, whether it be the originating post, review, or comment.
  - Notifications that have not been opened should have an indication that they have not been checked.
  - A number should be displayed on the "badge" over the notification icon, representing the number of notifications that have an unchecked status.
  - Notifications can be searched using a search bar or filtered by the attributes of the notification.

- Functional Requirement 1.5
  *Messages Screen*
  - Every message that has been initiated by the user or their recipient should be tabularly displayed in order of last modification.
  - Messages on the messages screen should have a truncated view of the last sent message.
  - A time should be displayed on each message indicating the time of the last message that has been sent.
  - A number should be displayed on a message object, representing the number of messages (from the recipient) that have an unchecked status.
  - A number should be displayed on the "badge" over the messages icon, representing the number of messages that have an unchecked status.
  - The messaging screen and interface should be displayed when tapping on a message from the messages screen.

- Functional Requirement 2.1
  *Archive*
  - All "bookmarked" content can be viewed in the archive tab, separated by posts and reviews.
  - Tapping on a bookmarked object should take you to the focused view of the content of the bookmark.
  - Posts and reviews can be unarchived from the archive page.
  - Posts and reviews can be unarchived from the focused view of the content, coming from the archive page.

- Functional Requirement 2.2
  *Profile*
  - Profile should contain public information about the user such as, their profile picture, banner picture, username, job title, location, and follower and following count.
  - Profile should contain the list of posts, reviews, and media that they own and published.
  - Profile should contain a list of all media types that the user of the profile marked as liked.
  - Tapping on a post, review, or media object should take you to the focused view of the corresponding content.
  - Users' public information can be edited from their profile, by tapping on a button to take the user to a screen where they can edit their public profile details.
- Functional Requirement 2.3
  *Messaging Screen*
  - A chronological account of all messages with the recipient should be displayed, with the recipient's messages aligned to the left, and the user's messages aligned to the right side of the screen.
  - Users are able to tap an input field to type a message using their mobile keyboard.
  - Users are able to tap a button to upload media from their mobile storage.
  - Messages should be sent and received in real-time.
  - The time of incoming or outgoing messages should be displayed if a particular amount of time has passed since the last message has been sent.
- Functional Requirement 3.1
  *Signing, Authorization, and Validation*
  - Users are able to login from the sign-in screen using an existing username and password, and after successful validation.
  - Users are able to reset their password using a password reset form and email confirmation process.
  - Users are able to sign-up using an email validation and confirmation process.
  - Users are able to register and sign-in using a Google or Apple account.

### 2.2.3 Non-functional Requirements

- Non-Functional Requirement 3.1
  *Security*
  - Users' data should be protected against security threats and vulnerabilities. Signing processes should be encrypted, and users should be authenticated before being able to modify the database.
- Non-Functional Requirement 3.2
  *Ease of use*
  - The application should use an intuitive and user-friendly interface, where functionalities are easy to understand and use.
- Non-Functional Requirement 3.3
  *Performance*
  - The application must use efficient queries to retrieve and upload data quickly and displays must be updated responsively.
- Non-Functional Requirement 3.4
  *Availability*
  - The application must will be usable on both iOS and android devices, and services must be continuously functional.

### 2.2.4 Operating Requirements

- Operating System: iOS and android on mobile devices
- Technology Stack:
  - Client Side: JavaScript code is executed within the React Native framework's environment, which then interacts with the native layers of the device, as well was device APIs.
  - Server Side: Our particular cohort will only focus on the frontend of the application, while another cohort develops the backend.

## 2.3 Technical Design

The data flow diagram as seen in Figure 1, represents the flow of data within the application, illustrating the API calls and data structure of various screens and components. As depicted, the user's id is immediately stored in a context, because it is used for most interactions to the server. The data is managed and accessed across different components without needing to explicitly pass it as a prop through every level of the component tree. The arrows represent a visual transfer of screens, while black arrows represent a transfer of data. The application is database-driven, where all modifications happen in the security of the backend server, and visual representations as the front end are derived from data retrieved from the backend.

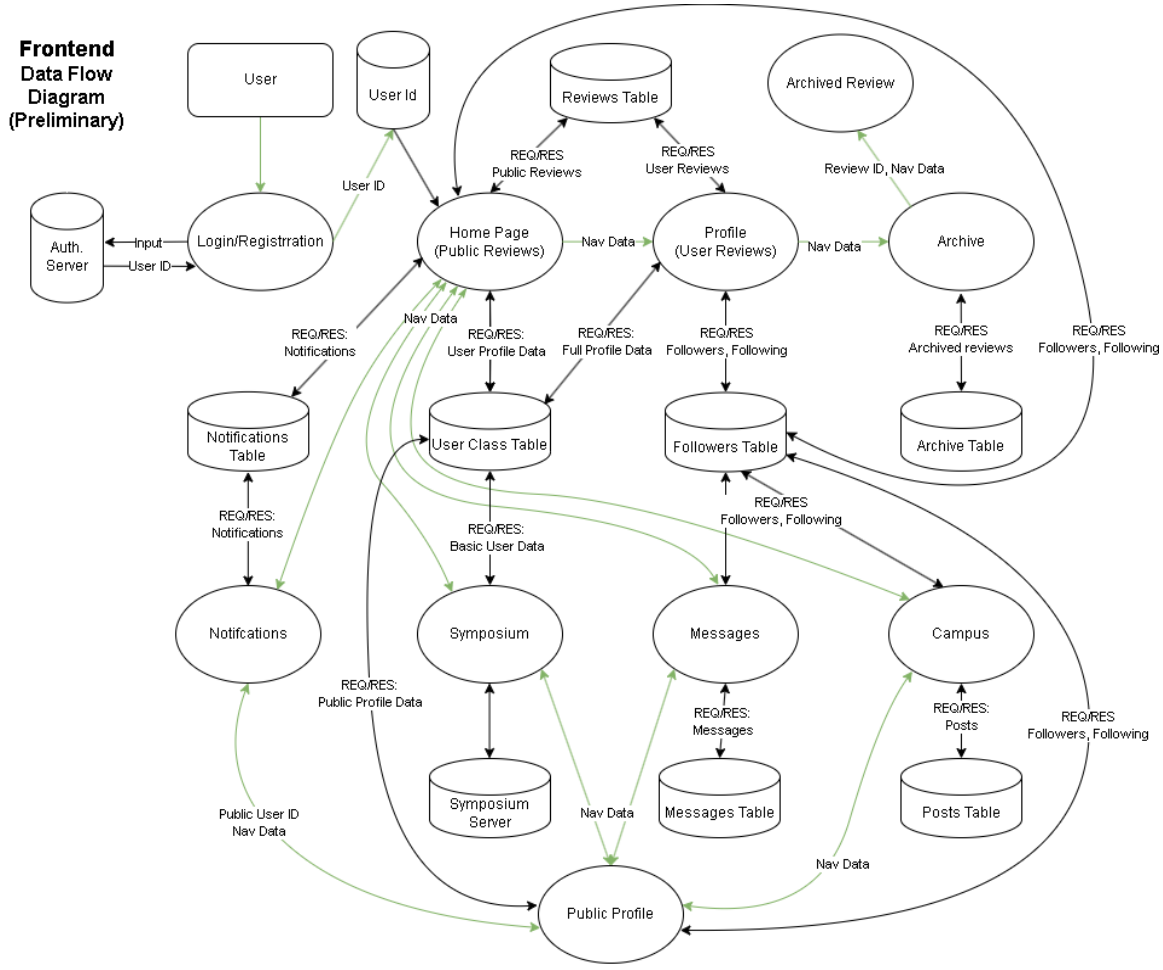


Figure 1: Data Flow Diagram

The "signing" process, as we call it, is the process that encapsulates signing in, signing out, signing up, password resetting, and the validation and authorization that come with these

processes. The whole process must follow a particular flow of interactions to ensure fluidity and security. Each of these steps is designed to confirm the identity of the user and secure access to the application's features. Additionally, robust encryption and secure storage techniques should be implemented to protect sensitive data and credentials throughout the authentication process. By implementing multi-factor authentication and continuous authorization sequences, the system enhances overall security and gains user trust.

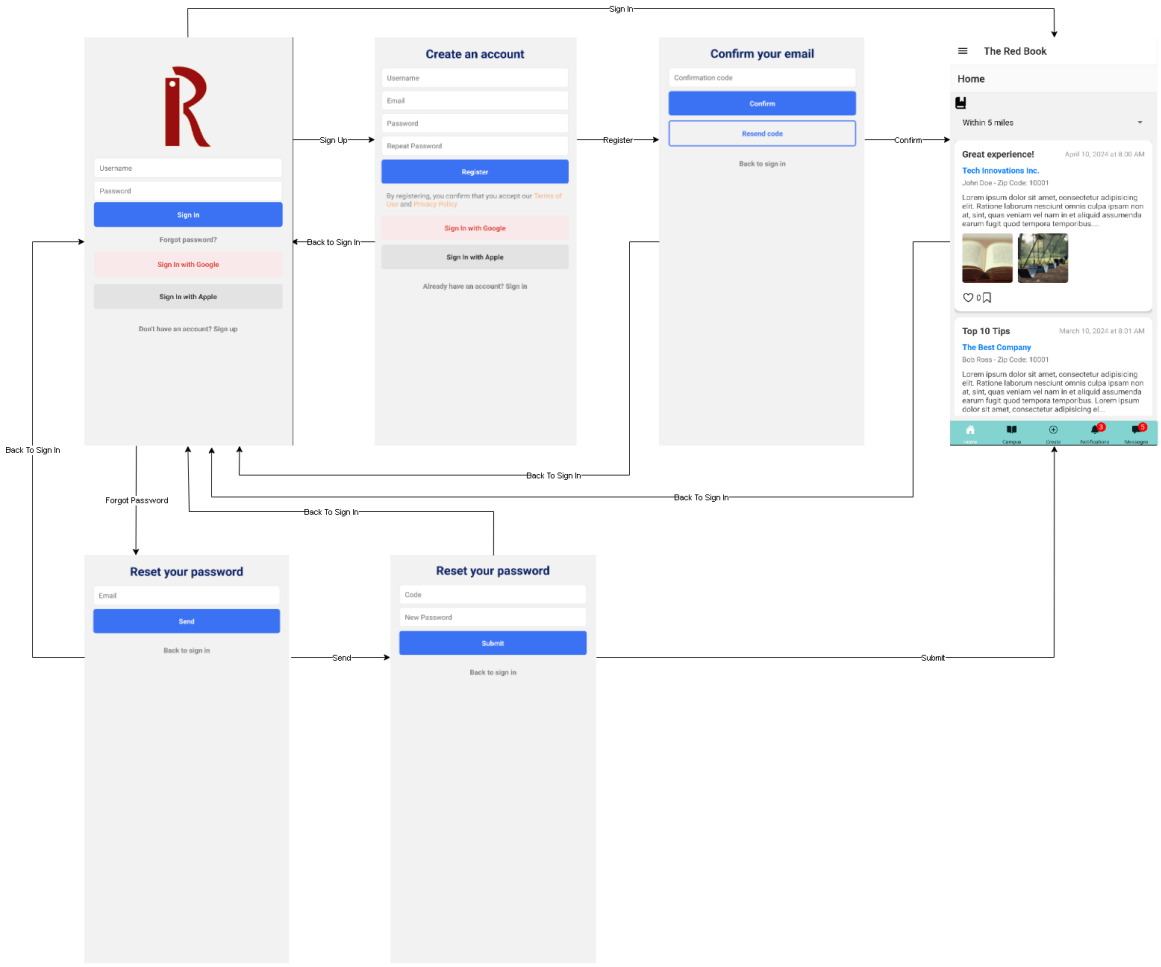


Figure 2: "Signing" Flow Diagram

## 2.4 System Implementation

- Programming Tools and Development Environment
  - React Native Framework: Utilized for building native apps using JavaScript and React.

  - Expo CLI: Used command-line environment that helps in starting, building, and deploying React Native applications.

  - Visual Studio Code: Utilized as development environment for its extensive support for JavaScript and React Native.

  - Node.js: Used as the runtime environment for managing the project's dependencies.

  - npm: Used as package manager to install and manage third-party libraries and dependencies

  - Android Studio: Used for virtual device testing on android devices.

- Significant Libraries, Dependencies, and Built-in Technologies

  - @react-navigation/native: Used navigation components for handling the routing and switching between different screens in the application. Many different types of navigation methods are used throughout the app on top of this library.

  - React Context API (useContext): Utilized for global state management across the application, particularly for maintaining user authentication state. Simplifies prop-drilling and enhances performance by preventing unnecessary re-renders.

  - axios: Handles HTTP requests to external APIs, used for fetching or posting data to a server. This will be used significantly more in the future when backend endpoints are established.

  - useState, useEffect, useCallback: React hooks for managing component state and lifecycle, which are essential for handling states and creating interactive user interfaces.

  - Expo Image Picker: Enables features to access the device's gallery or camera for image uploading functionalities.

## 2.5 Use of Computer Science Theory and Software Development Fundamentals

### Use of Computer Science Theories

Here we will detail the computer science and software engineering theories used in designing and developing The Red Book.


Figure 3: Expo framework CLI and compiler to launch and test the application, used in Visual Studio Code's terminal, which can access Android Studio's virtual devices.

**Object-Oriented Programming** Concepts in object-oriented programming are used to manage the relationships between the screens and components of the application. React components such as Post.js and Comment.js, are designed as classes or functional components with hooks that define their own state and behavior. This encapsulation allow for the re-usability of code and easier maintenance. Components inherit these properties from parent components, which simplifies the modification and scaling of features such as the "like" functionality and the commenting system.

**Space-Time Complexity Considerations** The application implements many features that require space-time complexity considerations. For example, there is a feature to display comments in reverse chronological order. It was decided that comments for posts would

be retrieved from the server only when present and necessary, and posts themselves would not contain their comments. Instead, comments are retroactively appended to posts, to secure load times for features with higher priority while saving object space in the database and maintaining modularity. The chronological sorting algorithm itself has a time complexity of O(n log n), which ensures efficiency as the number of comments increases. The performance remains acceptable even with a large dataset, enhancing user experience by reducing load times.

**Data Structures** Knowledge of data structures is necessary to know the necessary type and structure of data to request and receive from the server, and payloads would have to be appropriately parsed and restructured to display the frontend details correctly.

### Use of Software Development Fundamentals

Here we will detail the Software Development Fundamentals used in designing and developing The Red Book

**React Native Programming** The application uses React Native's component-based architecture in order to create reusable code. We use about 15 components multiple times throughout the application, which reduces duplication and simplifies the development and testing process while also creating a modular environment.

**Code and Design Documentation** We created and used a Figma rendering to reference requirements and visual design aspects. We also maintained documentation of our progression for a backend team to review and maintain clarity on existing parts of the code and expected implementations.

**Responsive Programming** We carefully designed our user interface styles such that it would look and function appropriately across different scales of device dimensions, by using flexible layouts and referring to the operating system of the device to define the values of certain styling options.

# 3 Experimental Design and Testing

As for the frontend of the application, which is the system our cohort is working on, we used virtual device emulators on top of the Expo CLI framework to responsively and efficiently test the results of our code. The backend system for the application, which is developed by a different cohort, has not been defined as of the time of this writing, so much of the experimental design and testing was limited. However, by speculating potential examples of callbacks and data handling methods, we prepared a flexible interface that could adapt to various backend implementations. This approach ensures that once the backend services are

established, integration will be streamlined for functional testing and iteration. Due to the focus of the application that we were assigned to develop, there is not necessarily an amount of experimental testing of substance, because our work was more centered on the construction of clearly defined functional requirements and already established designs that were expected of us.

## 3.1 Experimental Considerations

We had to consider the potential structure of Posts and Reviews in regards to whether or not likes and comments should be included in their respective objects. When we were creating the post and review object, we asked the question: Should the number likes and the comments be included with the objects? We asked this question because the number of likes and comment data of a post or review is constantly variable, while other aspects are not, leading to concerns with data scalability and server calling efficiency.

## 3.2 Dataset

A post can have any permutation of including or not including the number of likes and the comment data, contained within it. Attributes that are not included would have to be retrieved from a server and appended to the corresponding post or review object.

## 3.3 Results and Analysis

If the number of likes and comment data is included with a review object, there are reduced server requests enhancing the responsiveness of the application. It will reduce the delays that come with API calls and join operations when fetching the like and comment data. This may provide a better user experience, as they will see updated like counts and comments immediately after a review loads in their device. However, the payload size is increased, as each request carries all comments and likes, which could lead to longer loading times in the first place.

The time complexity for retrieving posts or reviews with both likes and comments included is O(n), where n is the number of posts to display. The practical performance can degrade as the number of comments increases due to larger data volumes having to be processed. Alternatively, with likes and comments being fetched with separate API calls, the time complexity is effectively O(3n), when only likes and comments are considered for each post.

As a result, a more practical and balanced approach was chosen for our project, which is to include the like count with each post and review, but make a separate API calls to fetch comments for posts. This method was chosen because although the two attributes are variable,

the scalability of comments are significantly more sensitive than the scalability of the number of likes, which is comparably negligible. The time complexity for our system is O(n + m) where n is the number of posts and m is the total number of comments across these posts. In the end, the trade off that was accepted was to experience faster general loading times and a more balanced database, for more potential inconsistency. This will provide a good balance of user experience while handling scalability for comments.

# 4 Legal and Ethical Practices

## 4.1 Legal Considerations

- Our project is a social media app that will be used for a business. The code belongs to that business and will be used however they see fit. However, there could be problems caused by the overlapping of unsuitable content. The app may unintentionally contain or post media that belongs to other entities. For example, YouTube was sued by Viacom because many of their clips were uploaded to YouTube. There are other cases similar to this one, so although the app is primarily for business, there should be consideration on whether there is a need for extra caution.

## 4.2 Ethical Considerations

- Our application stores the users' information such as their emails so that users are able to create accounts. In addition, it may collect data about the user, such as their interests and location. There is a security concern if the owner of the app decides to use that data for other reasons without the user's knowledge or if there is a security breach. Unauthorized or excessive data collection is a major concern in today's world as people's data gets sold without their consent. If the data gets leaked, users will have to face problems such as identity theft.

# 5 Effort Sharing

Contributions will be generally defined by the project files each member has worked on for the project.

# 6 Conclusion and Future Work

Our project, The Red Book, successfully sets the groundwork that establishes a platform that integrates social networking with professional insights, providing a unique blend of features tai-

Table 1: Effort sharing

| Team size | Joint efforts | Kevin Lee | Billy Plantin | Jean Shin | Xavier Lodge | |
|---|---|---|---|---|---|---|
| 4 | J (≈11%) | I.1 (≈60%) | I.2 (≈20%) | I.3 (≈6%) | I.4 (≈3%) | - |

J ≈ 4/35 files≈ 11%
I.1 ≈ 21/35 files ≈ 60%
I.2 ≈ 7/35 files ≈ 20%
I.3 ≈ 1/35 files + graphics ≈ 6%
I.4 ≈ 1/35 files ≈ 3%

lored to enhancing user engagement and community building. Throughout our development, the project has addressed technical challenges related to user interface design, data management, and creating a framework for future iterations. The Red Book has the potential to affect users and influence society by creating a transparent and informed communities. By enabling users to share genuine experiences and insights about their organizations, it contributes to a more informed decision-making process for their peers and spreads a culture of personalized knowledge.

As the project moves forward, a backend team is expected to expand on the application's capabilities by refining data handling and server interactions to ensure scalability and performance. Additionally, it is expected for a frontend team to continue enhancing the user experience by creating more flexible features and improving user interactions. Future work will involve extending the application's features to include more personalized content, and enhanced privacy and accessibility settings. Some specific features for future work on the frontend that we could not achieve in the time that we had are: The Symposium (livestreaming service), form validation, application settings, security and privacy settings, and visual updates to fit the true aesthetic of the application and what it stands for.

# References

"Core Components and Native Components · React Native." Reactnative.dev, reactnative.dev/docs/intro-react-native-components.

"Expo Go." Expo Documentation, docs.expo.dev/get-started/expo-go/.

"Hello React Navigation | React Navigation." Reactnavigation.org, reactnavigation.org/docs/hello-react-navigationsummary. Accessed 7 May 2024.

Pallavikhedle. "Exploring Navigation Solutions in React Native: Expo Router vs. React Navigation." Medium, 1 Mar. 2024, medium.com/@pallavi8khedle/exploring-navigation-solutions-in-react-native-expo-router-vs-react-navigation-37c270d45a7b. Accessed 7 May 2024.

"Understanding the Legal Issues for Social Networking Sites and Their Users." Https://Www.findlaw.com/Le Law-Practice/Understanding-The-Legal-Issues-For-Social-Networking-Sites-And.html, FindLaw.

"Core Components and Native Components · React Native." Reactnative.dev, reactnative.dev/docs/intro-react-native-components.

"Expo Go." Expo Documentation, docs.expo.dev/get-started/expo-go/.

"Hello React Navigation | React Navigation." Reactnavigation.org, reactnavigation.org/docs/hello-react-navigationsummary. Accessed 7 May 2024.

Pallavikhedle. "Exploring Navigation Solutions in React Native: Expo Router vs. React Navigation." Medium, 1 Mar. 2024, medium.com/@pallavi8khedle/exploring-navigation-solutions-in-react-native-expo-router-vs-react-navigation-37c270d45a7b. Accessed 7 May 2024.

"Understanding the Legal Issues for Social Networking Sites and Their Users." Https://Www.findlaw.com/Le Law-Practice/Understanding-The-Legal-Issues-For-Social-Networking-Sites-And.html, FindLaw.