# SE491 - Final Report
# Group 13

Amanda Bishop a7bishop

Giselle Sherese Ramlal gramlal

Dimitar (Mitko) Atanassov datanass

Jui Shah j44shah

# Table of Contents

# Glossary of Terms

*Course Completion Survey* - A survey containing question and answer pairs following the successful completion of any course, meant to gain insight into the user's (see definition below) opinion of the course content delivery and quality.

*Entrance Survey* - A survey containing question and answer pairs following a successful user sign-up, meant to assess the user's (see definition below) starting level of knowledge.

*Friend* - Any other User (see definition below) that is part of a specific user's friend list, allowing both Users to interact through our app.

*Nav Bar* - The navigation bar is located on the lowest portion of the screen, allowing the User (see definition below) to navigate to different screens.

*User* - The individual interacting with the app

# Abstract

As students arrive at university, it is apparent that there is a disparity between the knowledge of basic life skills across different populations. These skills will not be taught during their university courses, however, they are still important for students to learn. Many people learn these skills from those around them, such as parents or older siblings, and yet part of the novelty of this new environment is that they are now away from this familiar support system and have to learn by themselves for the first time. Additionally, according to a Forbes article, people spend an average of 1,300 hours on social media every year. These hours are often spent mindlessly scrolling, being unproductive and ultimately, deteriorating their mental health.

Team Girlboss aims to bridge this gap by delivering content centered around learning life skills in a low-commitment, highly engaging lesson format. We are providing our users with engaging content in a lesson format targeted towards our users' individual learning styles while preserving the social aspect users crave on social apps.

Our algorithm will recommend courses based on users’ topic preferences and, unlike existing platforms, we will feature personalized learning plans, expert content, and engaging competition through friend-based challenges.

# Requirements Specification

## Functional Requirements

| ID | Specification | Necessity | Description |
|---|---|---|---|
| FR1 | Account Creation | Essential | Users are able to create an account with credentials that are validated |
| FR2 | Account Sign In | Essential | Users are able to log in with their previously created credentials that are validated |
| FR3 | Account Log Out | Essential | Users are able to log out of their account |
| FR4 | Saved Login Credentials | Non-essential | Users who have previously logged into their account are automatically signed in |
| FR5 | Display Entrance Survey | Essential | The entrance survey is displayed to the user immediately after an account is created |
| FR6 | Utilize Entrance Survey Results | Essential | The user's response to their entrance survey will customize their learning experience |
| FR7 | Profile Page Progress | Essential | Users are able to view their current progress on the Profile page |
| FR8 | Adding Friends | Essential | Users are able to add friends |
| FR9 | Removing Friends | Non-essential | Users are able to remove friends |
| FR10 | Searching Friends | Essential | Users are able to search for friends by username |
| FR11 | Begin Course | Essential | Users are able to start courses |
| FR12 | Different Lesson Formats | Non-essential | Users are able to access lessons in various media formats |
| FR13 | Course Progression | Essential | As a user completes lessons their progress in that course updates |

| | | | |
|---|---|---|---|
| FR14 | Finish Course | Essential | When a user has completed all lessons the course is completed |
| FR15 | Redo Course | Non-essential | The user is able to redo a lesson they have completed |
| FR16 | Searching Courses | Non-essential | Users are able to search for courses |
| FR17 | Filtering Courses | Non-essential | Users are able to filter courses by rating and course duration |
| FR18 | Displaying Lesson Quizzes | Essential | Every lesson has a matching lesson quiz that will be displayed to the user after completing the lesson |
| FR19 | Multiselect Question Functionality | Essential | Multiselect questions allow the user to select multiple answers |
| FR20 | True or False and Multiple Choice Question Functionality | Essential | True or false and multiple choice questions only allow the user to select one answer |
| FR21 | Matching Question Functionality | Essential | Matching questions allow the user to match a term to a definition |
| FR22 | Lesson Completion | Essential | Users cannot progress to the next lesson until they have passed the current lesson quiz |
| FR23 | Displaying End-of-Course Quizzes | Essential | Every course has a matching end-of-course quiz that will be displayed to the user after completing the last lesson in the course |
| FR24 | Course Completion | Essential | Users cannot complete the course until they have passed the end-of-course quiz |
| FR25 | Display Course Completion Survey | Essential | The course completion survey is available to the user after the course is completed |
| FR26 | Update Course Badges | Non-essential | Users obtain a course badge when they have passed the end-of-course quiz |
| FR27 | Display Friends' Activity | Essential | Users are able to see their friends' activity |

| | | | |
|---|---|---|---|
| FR28 | Display Completed Challenge | Essential | When a user completes a challenge this accomplishment is reflected on the Friends Feed for each of their friends |
| FR29 | Display Completed Course | Essential | When a user completes a course this accomplishment is reflected on the Friends Feed for each of their friends |
| FR30 | Update Points from Lessons | Essential | The user's points total will update when they have completed a lesson |
| FR31 | Update Points from Challenges | Essential | The user's points total will update when they have completed a challenge |
| FR32 | Update Course Challenges | Essential | The user's course challenge progress will update when the user has completed a course |
| FR33 | Update Lesson Challenges | Essential | The user's lesson challenge progress will update when the user has completed a lesson |
| FR34 | Update Points Challenges | Essential | The user's points challenge progress will update when the user has earned points |
| FR35 | Update Add Friends Challenges | Essential | User's add friend challenge progress will update when the user has added friends |
| FR36 | Update Friend-Based Challenges | Essential | The user's friend-based challenges will update the progress on their friend's document as well |
| FR37 | Update Ranking | Essential | The user's ranking is updated when the user has earned points |
| FR38 | Update Leaderboard | Essential | The Leaderboard page reflects changes in the user's points progress |
| FR39 | Minimum Number of Challenges | Non-essential | Users must always have at least 4 current challenges |
| FR40 | Like/Unlike Friends' Posts | Non-essential | Users are able to like and unlike their friends' posts on the Friends Feed |
| FR41 | Home Page Navigation | Essential | Users are able to navigate to the other pages from widgets on the Home page |

| | | | |
|---|---|---|---|
| FR42 | Resource Page Navigation | Essential | Users are able to access Resources from the Home page |
| FR43 | Nav Bar Navigation | Essential | Users are able to navigate to any of the 5 main pages using the tabs in the Nav Bar |

## Non-functional Requirements

| **ID** | **Specification** | **Necessity** | **Description** |
|---|---|---|---|
| NFR1 | Supportability | Essential | Growth Hub must be usable for both Android and iOS users |
| NFR2 | Course Content | Essential | All course content is created by experts |
| NFR3 | Accessibility | Essential | Growth Hub abides by the A and AA Accessibility Standards |
| NFR4 | Updating Data | Essential | All data is updated in real time |

# Design Documentation

## Architecture Overview

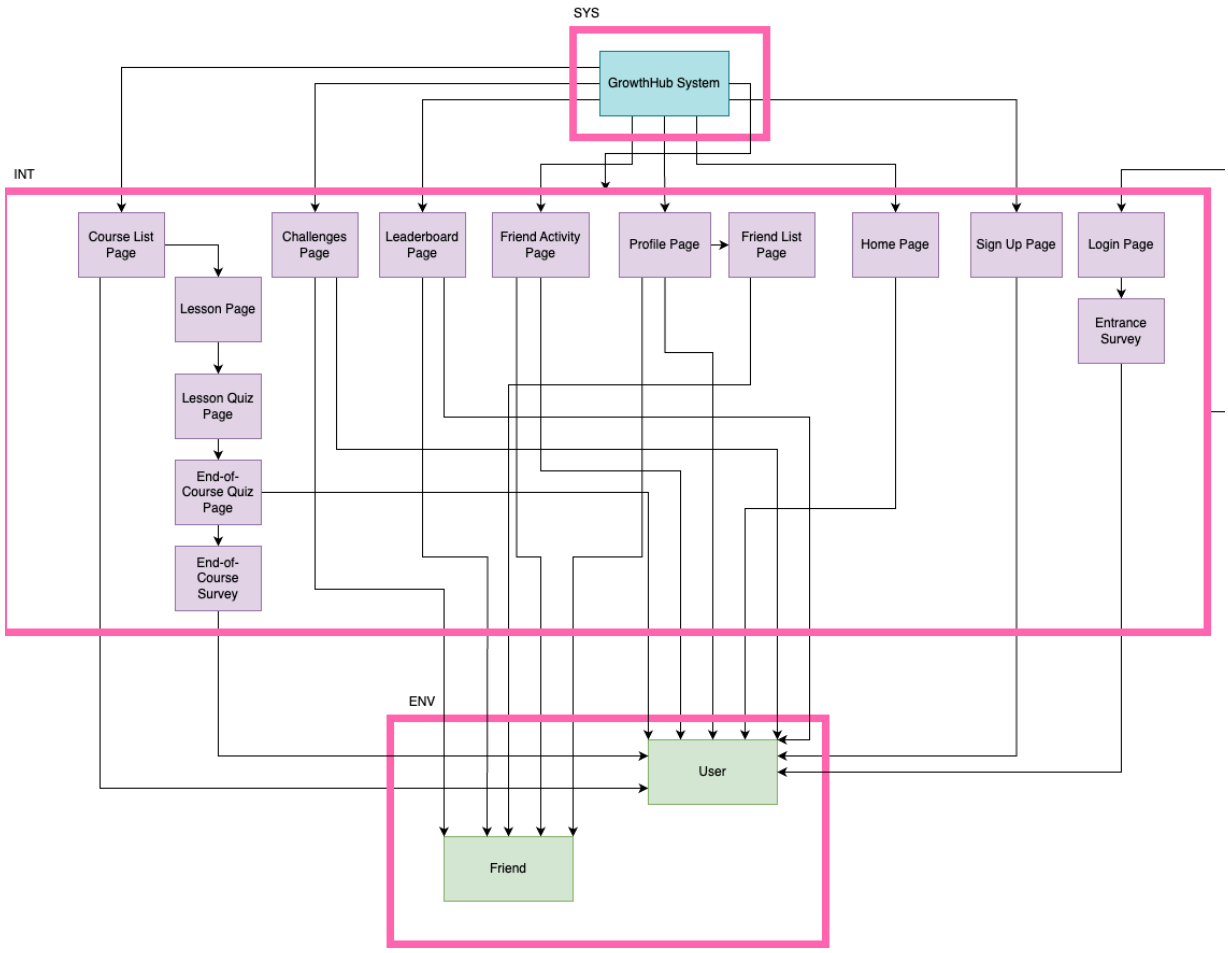


Figure 1: System Context Diagram

## Design Principles

Our development approach was anchored in user-centered design principles, recognizing that university students have limited time and varying levels of life skills knowledge. We prioritized creating an intuitive interface that accommodates the user's different learning styles and preferences, conducting multiple rounds of user surveys and initial testing with our target demographic to ensure the app truly addressed their needs. The responsive UI design ensured our platform functions seamlessly across mobile devices, regardless of screen size and

operating system. This flexibility was essential in delivering the "low-commitment" experience promised in our mission, allowing users to engage with content whenever and wherever they have a few spare moments that might otherwise be spent mindlessly scrolling social media.

Security by design and accessibility-first principles guided our technical implementation throughout development. Understanding that our target users might share potentially sensitive information about their skill gaps, we implemented robust privacy controls and data minimization practices. The accessibility first approach ensured that all students, regardless of abilities, could benefit from our life skills content—employing clear typography, sufficient colour contrast, and screen reader compatibility. These principles directly support our mission to bridge knowledge gaps across different student populations, creating an inclusive, robust platform.

## Component Design

### Home Page

Our Home page serves as the landing page for our users. We decided to incorporate widgets displaying the user's current progress that direct the user to other pages on the app. These widgets serve to encourage our users to explore the rest of the app to see where this information is coming from. As well, we display several progress bars, as shown in Figure 2, indicating the user's progress in courses and challenges. This is shown to incentivize our users to continue with their learning to further their progress and feel more accomplished.

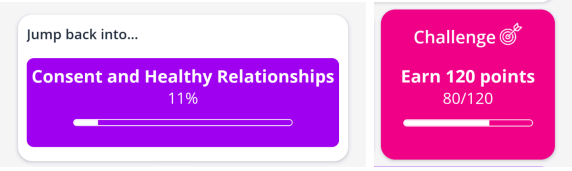


Figure 2: Home Page progress bar widgets

### Challenges/Leaderboard

Our Challenges and Leaderboard pages both serve to facilitate friendly competition between the user and their friends. Inspired by other learning apps as well as game apps, we decided to implement this friendly competition as a method to incentivize our users to continue to use the app. We felt that establishing this competition, and in the process, gamifying our app, would make the product feel more like a game for our users than a serious task where they are learning content. We highlight the user's progress in their challenges, as shown in Figure 3, and their ranking compared to their other friends, as displayed in Figure 4.


Figure 3: Challenge widgets

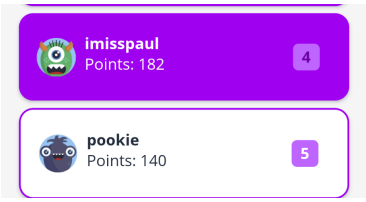


Figure 4: Leaderboard widgets

## Friends Feed

Our Friends Feed page serves to hold users accountable to completing courses and challenges. When a user completes a challenge or a lesson this accomplishment is reflected on their friend's Friends Feed pages, as shown in Figure 5. By sharing this information with the user's friends, we are able to encourage the user to continue accomplishing these goals since they are aware that their friends can see how well they are progressing.


Figure 5: Friends Feed post widgets

## Courses

Our Course List page is designed to encourage users to jump into a course. This is done by providing our users with a description of what each course consists of as well as providing our users with information on the course duration and rating, as displayed in Figure 6. The rating is updated when other users fill out the Course Completion survey and rank the course. This informs the user how much other users enjoyed the course. This will encourage the user to start popular courses since they will want to know why the courses are well-liked.


Figure 6: Course Duration and Rating

### Profile

Our Profile page also serves to encourage our users to continue their progress. This page displays the user's current ranking and current courses, aimed to encourage the user to continue progressing through their courses. Additionally, we display course badges, which the user earns by completing courses. We decided to display these badges, as shown in Figure 7, to give the users a sense of accomplishment when they finish a course and further the gamification by making the user feel that they have earned an item that not all users have.

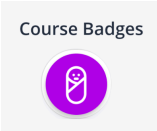


Figure 7: Course Badge widget

## Data Models

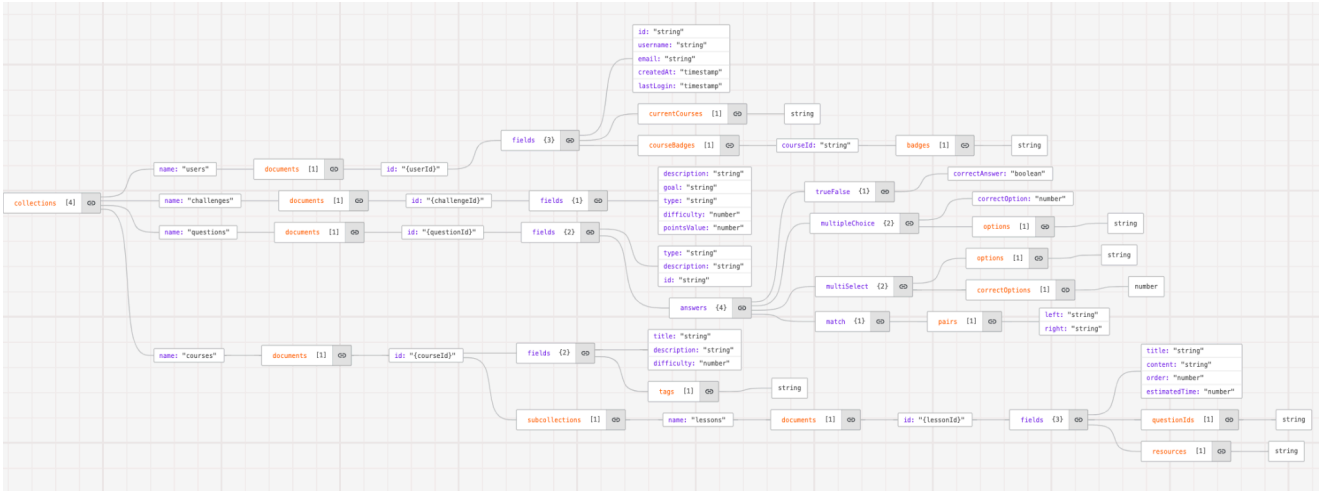

Figure 8: Database Schema Diagram

# UI/UX Design

## Wireframes

Our design process began with comprehensive wireframing that directly mapped to each functional requirement in our application. We created low-fidelity wireframes for all key user journeys, including:

- User onboarding and preference selection
- Course discovery and browsing
- Lesson navigation and completion
- Challenge participation and friend interactions
- Profile management and progress tracking

The wireframes served as skeletal frameworks that allowed us to validate the information architecture and user flows before committing to visual design decisions. We conducted early usability testing with these wireframes to identify and address navigation issues and content organization concerns.

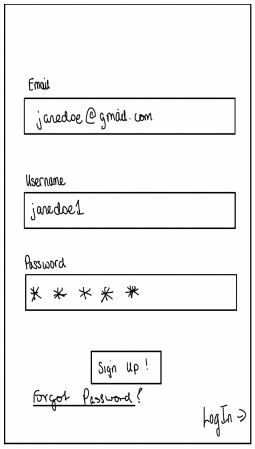


Figure 9: Sign Up Wireframe

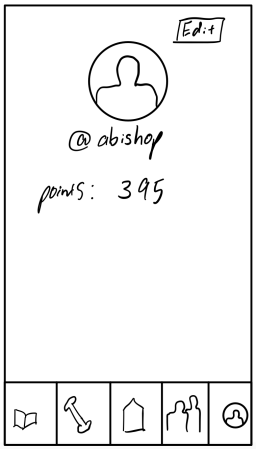


Figure 10: Profile Wireframe

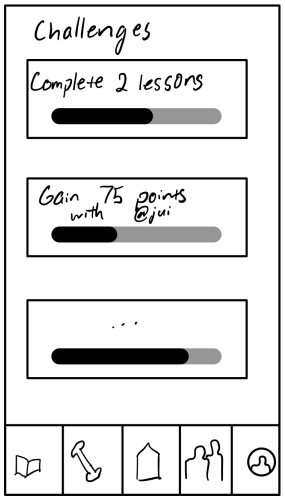


Figure 11: Challenge Wireframe

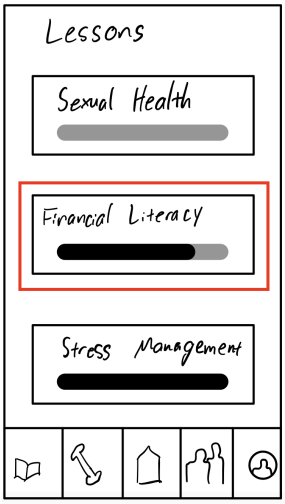


Figure 12: Lessons Wireframe

## High-Fidelity Designs

After validating our wireframes, we translated them into high-fidelity mockups using Figma. This phase involved:

1. Applying our colour scheme at the time consistently across all screens
2. Implementing our profile page, home page, leaderboard and courses page
3. Creating components such as cards and buttons elements
4. Designing state variations for all interactive elements

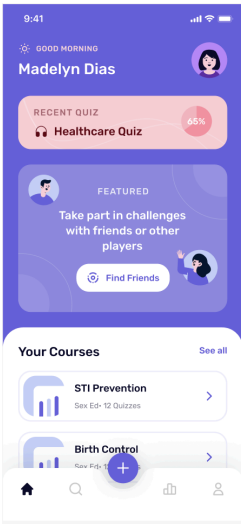


Figure 13: Home Page Figma Design

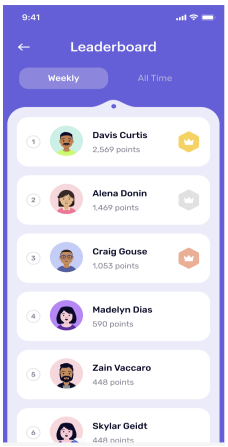


Figure 14: Leaderboard Figma Design

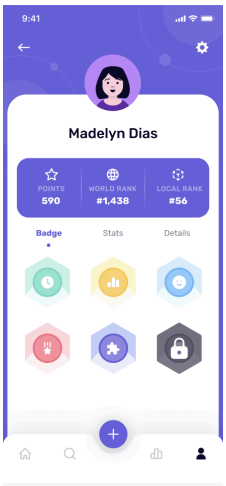


Figure 15: Profile Figma Design

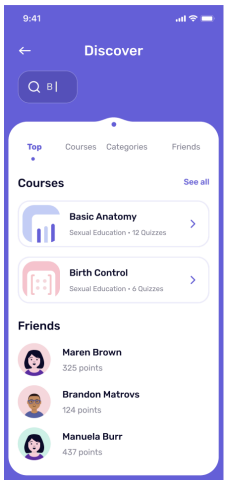


Figure 16: Courses Figma Design

## Layout Design Decisions

The layout was structured to support both learning and social engagement. Some examples of the necessary layout decisions we made include:

- We implemented a bottom navigation bar for primary features, keeping it accessible for one-handed mobile use
- Course content was designed with a card-based UI to allow for easy scanning and selection
- Lesson screens utilized generous white space (background: #F5F5F9) to improve readability and focus

These layout decisions directly supported our goal of creating "low-commitment, highly-engaging" content by making navigation intuitive and reducing cognitive load during the learning experience.

### Responsive Design Implementation

The theme configuration demonstrates our commitment to responsive design through carefully calculated spacing units and component sizing. All measurements were based on a consistent 8-pixel grid system (SIZES.base: 8), creating visual rhythm and ensuring proper scaling across devices. Button and input field dimensions were standardized (buttonRadius: 8, inputHeight: 46) to create a coherent interface that adapts seamlessly to different screen sizes.

### Colour Scheme and Visual Identity

The colour scheme was strategically selected to appeal to our university student demographic while supporting our app's educational purpose:

- Primary Colour (#F20089): A vibrant pink that brings energy and excitement to the platform, helping transform what could be perceived as "boring" life skills content into something engaging and contemporary.
- Secondary Colour (#A100F2): A rich purple that complements the primary pink while adding depth and sophistication. We used lighter (#C066FF) and darker (#7000A9) variations to create visual hierarchy within the interface.
- Tertiary Colour (#46C6B9): A calming teal that balances the high-energy primary and secondary colours, used for success states and completion indicators.

This colour palette was carefully considered to create a vibrant yet accessible interface that would stand out from traditional educational platforms while maintaining readability and reducing eye strain during extended learning sessions.

### Icon System and Visual Elements

Our icon system was designed to be both functional and visually appealing:

- We used outlined icons with consistent stroke weights for navigation and actions
- Filled icons were reserved for active states and emphasized features
- Custom icons were created for specialized features like Challenges where pre-existing standard icons did not accurately represent the feature. Eg. Medals, Trophies, and Arrow in Bullseye.

Badges were designed as circular achievements with gradient fills using combinations from our defined gradient palette (primary, secondary, etc.). Each badge visually represented specific life skills accomplishments, reinforcing the gamification aspect of our platform.

### Accessibility Considerations

Accessibility was integrated throughout the design process:

- Colour contrast ratios exceed WCAG 2.1 AA standards for text readability
- Interactive elements have appropriate sizing (minimum touch targets)
- Visual feedback mechanisms were designed for all interactive states
- Text scaling was supported with a multiplier value of 2

# Algorithms and Logic

## Updated Challenge List Generator

The Updated Challenge List Generator serves as a key engagement mechanism in our platform, creating personalized challenges that encourage students to apply their life skills knowledge in practical scenarios.

```
// Updates the user's currChallenges
export async function updateCurrUserChallenges(uid: string) {
    const userRef = doc(db, 'users', uid);
    const userSnap = await getDoc(userRef);
    if (!userSnap.exists()) {
        console.log('[updateCurrUserChallenges] No such user');
        return false;
    }

    const currChallenges = userSnap.data().currChallenges;
    for (const challenge of currChallenges) {
        await completeChallenge(uid, challenge.challengeID);
    }

    const updatedCurrChallenges = await getCurrChallenges(uid);
    const completedChallenges = userSnap.data().completedChallenges || [];
    if (updatedCurrChallenges.length < 4) {
        const challengesRef = collection(db, 'challenges');
        const challengesSnapshot = await getDocs(challengesRef);

        const allChallenges = challengesSnapshot.docs.map((doc) => {
            const data = doc.data();
            return {
                id: doc.id,
                ...data,
                isSolo: data.isSolo
            };
        });

        for (let i = 0; i < 4 - updatedCurrChallenges.length; i++) {
            const availableChallenges = allChallenges.filter(challenge => {
                const inCurrent = updatedCurrChallenges.some(
                    (curr: { challengeID: number }) => curr.challengeID === Number(challenge.id)
                );

                const inCompleted = completedChallenges.some(
                    (comp: { challengeID: number }) => comp.challengeID === Number(challenge.id)
                );
                return !inCurrent && !inCompleted;
            });

            if (availableChallenges.length === 0) {
                console.log('[updateCurrUserChallenges] No more available challenges to add');
            }

            const randomChallenge = availableChallenges[Math.floor(Math.random() * availableChallenges.length)];
            const challengeID = Number(randomChallenge.id);
            if (randomChallenge.isSolo) {
                await addChallengeToUser(uid, challengeID, "");
            } else {
                await addFriendChallenge(uid, challengeID);
            }
        }
    }
}
```

Figure 17: Updated Challenge List Generator code

## Quiz Generator

The Quiz Generator creates adaptive assessments that test knowledge retention and application across different learning styles.

```
// Get end of course quiz ie. a random question from each lesson in the course
export async function getEndOfCourseQuiz(courseId: string): Promise<number[]> {
  try {
    const courseRef = doc(db, "courses", courseId);
    const courseSnapshot = await getDoc(courseRef);

    if (!courseSnapshot.exists()) {
      throw new Error("Course not found");
    }

    const lessonsRef = collection(courseRef, "lessons");
    const lessonsSnapshot = await getDocs(lessonsRef);

    const endOfCourseQuiz: number[] = [];
    for (const lessonDoc of lessonsSnapshot.docs) {
      const lessonId = parseInt(lessonDoc.id, 10);

      // Get questions from the lesson using getLessonQuestions
      const questions = await getLessonQuestions(courseId, lessonId);

      // Pick one question randomly from the lesson, if available
      if (questions.length > 0) {
        const randomIndex = Math.floor(Math.random() * questions.length);
        endOfCourseQuiz.push(questions[randomIndex]);
      }
    }
    return endOfCourseQuiz;

  } catch (error) {
    console.error("Error getting quiz questions:", error);
    throw error;
  }
}
```

Figure 18: Quiz Generator code

This algorithm supports personalized learning by:

- Ensuring comprehensive coverage of lesson material
- Creating varied assessment experiences

## Recommendation Algorithm

Our Recommendation Algorithm suggests personalized courses based on user preferences.

```
// Get course list (with optional filters)
export async function getCourseList(uid: string, filters: QueryConstraint[] = [], sortOptions: SortOptions | null = null) {
  try {
    const userRef = doc(db, 'users', uid);
    const userSnap = await getDoc(userRef);
    const userSurvey = userSnap.data()?.survey["2"] || [];

    const coursesRef = collection(db, "courses");
    let q = query(coursesRef, ...filters);
    const querySnapshot = await getDocs(q);

    const courses = querySnapshot.docs.map(doc => ({ id: doc.id, ...doc.data() }));
    courses.sort((a, b) => {
      const indexA = userSurvey.indexOf(a.id);
      const indexB = userSurvey.indexOf(b.id);

      if (indexA === -1 && indexB === -1) return 0; // Both not in userSurvey, keep original order
      if (indexA === -1) return 1; // A is not in userSurvey, move down
      if (indexB === -1) return -1; // B is not in userSurvey, move down

      return indexA - indexB; // Sort based on userSurvey order
    });

    return courses;
  } catch (error) {
    console.error("Error getting courses:", error);
    throw error;
  }
}
```

Figure 19: Recommendation Algorithm code

## User Progress Algorithm

The User Progress Algorithm tracks and visualizes skill development across different life skill domains by updating points, challenges and the leaderboard in real time.

```
// Updates the challenge progress for a user based on the action type
export async function updateChallengeProgress(uid: string, type: string) {
    try {
        const userRef = doc(db, 'users', uid);
        const userSnap = await getDoc(userRef);

        if (!userSnap.exists()) {
            console.log('[updateChallengeProgress] No such user');
            return false;
        }

        const userData = userSnap.data();
        if (!userData.currChallenges || !Array.isArray(userData.currChallenges)) {
            console.log('[updateChallengeProgress] User has no current challenges');
            return false;
        }

        const currChallenges = [...userData.currChallenges];
        let updatedAny = false;

        for (let i = 0; i < currChallenges.length; i++) {
            const challenge = currChallenges[i];
            const challengeData = await getChallengeById(challenge.challengeID);

            if (challengeData && challengeData.type === type) {
                currChallenges[i] = {
                    ...challenge,
                    progress: (challenge.progress || 0) + 1
                };
                if (!challengeData.isSolo && challenge.friendUid) {
                    await updateFriendChallengeProgress(challenge.friendUid, challenge.challengeID, currChallenges[i].progress);
                }

                updatedAny = true;
            }
            if (challengeData && challengeData.type === "POINTS") {
                currChallenges[i] = {
                    ...challenge,
                    progress: (challenge.progress || 0) + 10
                };
            }
        }

        if (updatedAny) {
            updatePoints(uid, "POINTS");
            await updateDoc(userRef, { currChallenges });
            return true;
        }
    } catch (error) {
        console.error(`[updateChallengeProgress] Error updating challenge progress:`, error);
        return false;
    }
}
```

Figure 20: User Progress Algorithm code

The progress algorithm provides value through:

- Multi-dimensional progress tracking beyond simple completion
- Personalized feedback on strengths and growth areas
- Gamified elements (badges, milestones) to increase motivation

## Security Design

Our application implements a robust authentication system leveraging Firebase Authentication to ensure secure user identity verification while maintaining a frictionless user experience. This approach allows us to:

- Support authentication method of user email/password.
- Implement secure password policies with minimum strength requirements
- Maintain secure session management with JWT (JSON Web Tokens)

Firebase Authentication was selected specifically because it offers industry-standard security practices while reducing implementation complexity, allowing us to focus development resources on our core educational features. The authentication flow was designed to minimize friction during onboarding, recognizing that excessive security barriers could deter students from engaging with the platform.

## Design Trade-offs

### Platform Selection

We carefully evaluated multiple platform approaches for our life skills learning application:

1. Web-Only Application
    - Advantages: Universal access across devices, simpler development process, no app store approval requirements
    - Disadvantages: Limited offline capabilities, reduced performance on mobile devices, no push notifications
2. Native Mobile Application
    - Advantages: Superior performance, full device integration, offline functionality, push notifications
    - Disadvantages: Platform-specific development (iOS/Android), higher development costs, app store restrictions

We ultimately selected a React Native mobile approach based on several factors directly tied to our target audience and app objectives:

1. Usage Patterns: University students predominantly use mobile devices during short breaks, precisely the moments when they might otherwise engage in unproductive social media scrolling. Our mobile-first approach allows us to capture these micro-learning opportunities.

2. Engagement Features: The challenge and social components of our application benefit significantly from mobile-specific features like push notifications, which help drive engagement and completion rates.
3. Performance Requirements: Interactive lessons and gamified elements require responsive performance that native or near-native approaches provide better than web applications.

### Database Architecture

We evaluated multiple database architectures:

1. Traditional Relational Database (PostgreSQL/MySQL)
    - Advantages: ACID compliance, structured data with enforced relationships, mature ecosystem
    - Disadvantages: Less flexibility for evolving content structures, potential scaling complexity
2. Document Database (Firestore)
    - Advantages: Flexible schema, natural mapping to JSON objects, built-in real-time capabilities
    - Disadvantages: Less rigid data integrity, eventual consistency model
3. Hybrid Approach (Relational + Redis)
    - Advantages: Structured core data with high-performance caching
    - Disadvantages: Increased system complexity, synchronization challenges

We selected Firestore (non-relational document database) based on several factors:

1. Content Evolution: Our educational content structure frequently evolves as we develop new life skills modules. Firestore's schema-less nature allows us to iterate quickly without costly migrations.
2. Real-time Features: The social and challenge aspects of our application benefit from Firestore's built-in real-time updates, enabling instant reflection of friend activities and challenge progress.
3. Scaling Considerations: Firestore's automatic scaling aligns with our goal of serving a growing university student population without manual database management.
4. Development Velocity: The document model accelerated development by allowing a closer mapping between application objects and storage structures.

While we acknowledged the trade-off in losing some of the rigid data integrity of relational systems, we mitigated this through application-level validation and careful security rule design in Firestore.

### Data Synchronization

For social features, we implemented real-time synchronization to enhance engagement, while lesson content uses on-demand loading with offline caching to optimize performance and data usage—a critical consideration for student users who may have limited data plans.

## Future Enhancements

As the goal for our app was to customize the learning experience for each user, we would like to gear our future enhancements towards furthering this goal. One of the main changes we would want to implement is to create more dynamic user interactions between the user and their friends. This would be accomplished through more interactive challenges. Currently, the friend-based challenges are created with a random friend of the user. We would like to implement the functionality for the user to select the friend to complete the challenge with them. This would help the user feel more engaged since they are able to slightly customize the challenges. Additionally, we would like to implement a feature to allow the user to challenge their friends. This challenge would act differently from a friend-based challenge because instead of the user working with their friend to complete a challenge, they would be competing against their friend to see who completes the challenge goal first. Whoever completes the goal first would be the person to complete the challenge and earn the points from that challenge. This would encourage friendly competition amongst users, keeping the users motivated to use the app.

To expand on this goal to customize the learning experience for the user, we want to customize the lessons and quizzes based on the user's learning styles. As a future enhancement, we would take the user's answers from the entrance survey and change the content they receive as a result. We would prioritize their preferred question format for the lesson and end-of-course quizzes. In addition, we would like to develop lessons in other media formats, such as audio lessons. With these other lesson formats, we would be able to customize the lessons better for our users to support their preferred learning style.

In addition, a non-technical enhancement we would like to accomplish would be providing our users with more content on a wider range of topics. Since we want to provide our users with expert-created content, we felt that for the period of our capstone course that it was best to focus on one topic to have courses created for. Because of this, we partnered with the Shore Centre to create this course content related to sexual health. However, since our app is centered around teaching life skills, we would like to feature other courses as well on topics such as financial literacy and time management.

# User Manual

## Installation Instructions

First, users must download Expo Go from their system's app store.
Then, users can use the QR code in Figure 21 to run the app:

Figure 21: Expo Go QR Code

# Use Cases

## Use Case Diagram

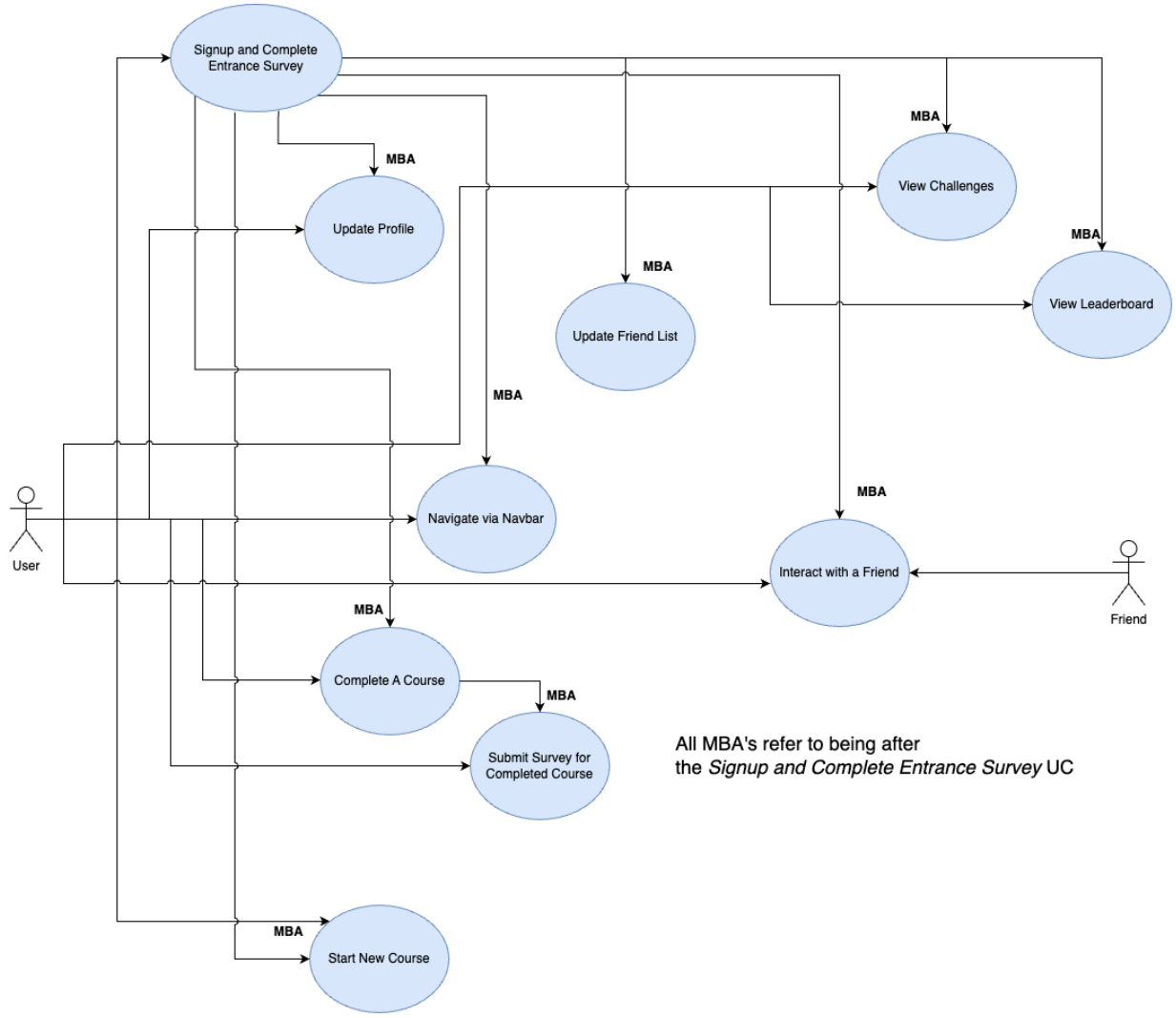


Figure 22: Use Case Diagram

## Assumptions, Exceptions and Variations

### Assumptions

1. Users have an active account and provide valid, unique credentials to log in.
2. Users understand the logical mapping of questions and answer fields and thus will input their answer in the appropriate answer field.
3. Users complete the entrance survey immediately after sign-up.
4. A user must be authenticated to access certain screens like the profile, friends feed and lessons.
5. The entrance survey is the same for all users.
6. Users have access to their profile information at all times.
7. Users are connected as friends to see each other's activities.
8. Users can redo a course after completing it.
9. Users complete lessons in sequence and only take the quiz at the end.
10. All system updates reflect in real time.
11. Users' profile data remains consistent and intact across sessions.
12. All course materials are accessible to users without technical issues.

### Exceptions

1. Do not provide valid, unique credentials to create a new account if they do not already have an account. If an account exists, exceptions can occur through invalid login attempts, account lockouts, or password reset requirements preventing access.
2. Users who are non-native speakers, or have cognitive challenges, or are unfamiliar with the typical question-answer paradigm may misinterpret field labels, leading them to enter responses in the wrong fields.
3. Users might skip the survey if the app allows it, or they may delay completion if they lose connection and reach an exceptional state somehow
4. Admin user privileges enacted from the developer's end can bypass the user authentication stage, to troubleshoot client issues for example.
5. The app may require a slightly more personalized entrance survey based on language, region, user demographic which could lead to different questions for different users.
6. Client apps that disconnect from the Internet will not be able to query and get the latest profile information until the session reconnects or refreshes.
7. A user will no longer see the activity of a Friend if that Friend user has already removed the User from their Friend list.
8. Users are unable to do courses again after completing them.
9. Technical issues allow the user to jump directly to the quiz for a given lesson without going through the content/questions associated with it.
10. There may be sessions with errant network latency, caching resulting in outdated information until the app reconnects or refreshes.
11. A data sync error or database issue might cause profile data to temporarily or permanently go out of sync across sessions.

12. If there are issues serving a video lesson, the textual transcript for the lesson will be provided instead as a compromise.

### Variations

1. The lesson flow might vary between lessons with single page vs. multi-page lessons.
2. Lesson content will vary (text-based or video format)
3. Lesson quizzes and end-of-course quizzes lengths will vary.
4. Users may choose between solo or collaborative challenge completion.

## Core Functionality

### Entrance Survey

Upon setting up their account, the user will be prompted to fill out an Entrance Survey before using the app. The Entrance Survey, as displayed in Figure 23, prompts users to answer questions about their learning preferences and adjusts their learning experience accordingly. For example, the topics that the users select will be recommended to the user by appearing above the rest of the courses in the course list.

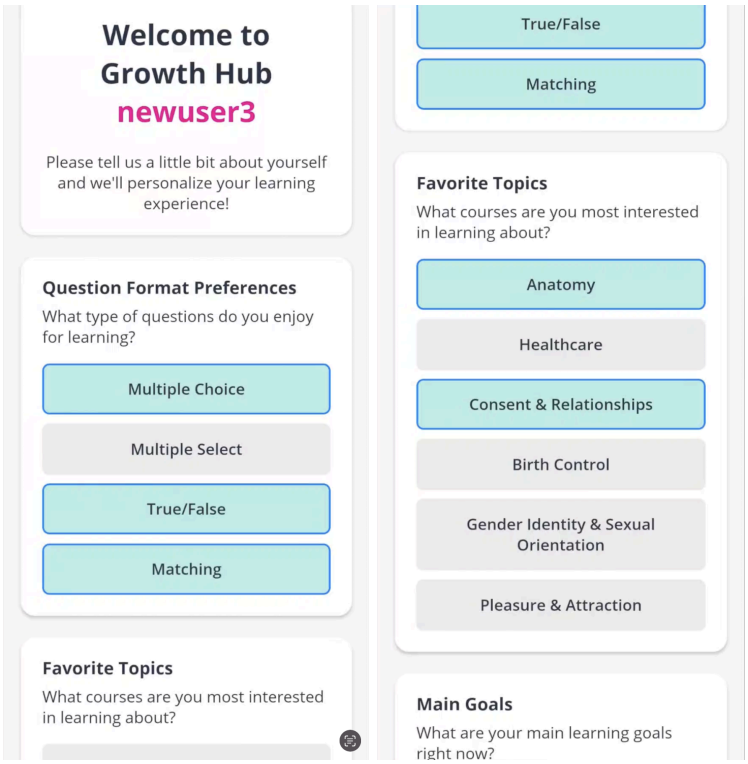


Figure 23: Entrance Survey

## Home Page

When the user enters the app, they are first directed to the Home page, as shown in Figure 24. The Home page displays the user's current progress, allowing them to navigate to the corresponding pages. The "Jump back into…" card pulls a random course from the user's list of current courses and allows the user to enter their current lesson in that course. The "Your Ranking" card displays the user's current leaderboard ranking, and the "Challenge" card pulls a random challenge from the user's list of current challenges. Both of these cards link the user to the Leaderboard/Challenges page so the user can see their progress. The "Friends Feed" card pulls the most recent post from a random friend of the user and links to the Friends Feed page. Lastly, the "Resources" card links to a page of healthcare resources in the KW region that include external links.

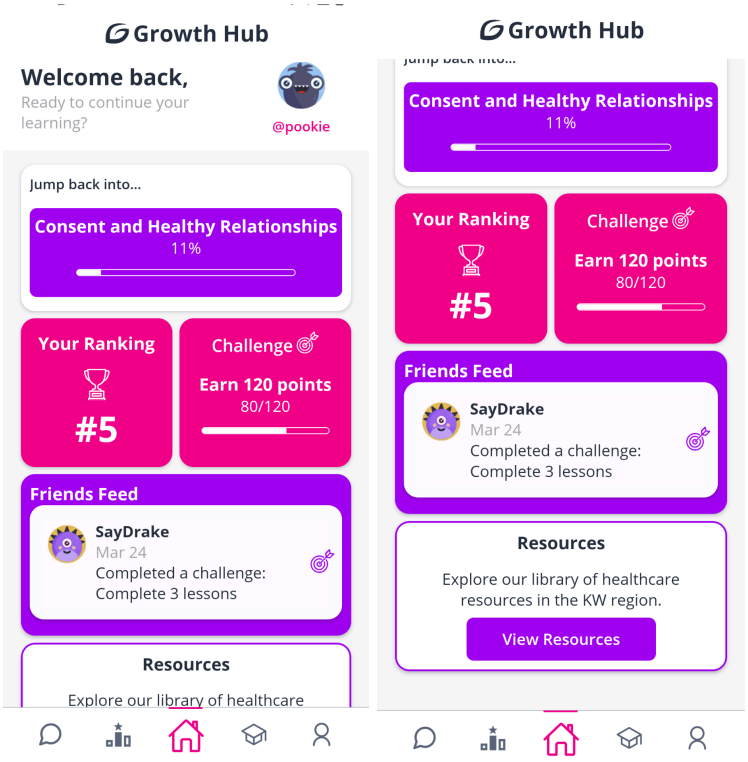


Figure 24: Home Page

### Friends Feed

The Friends Feed, as shown in Figure 25, allows users to see updates about their friends' progress. When a user completes a challenge or a course this will be displayed as a post on their friends' Friends Feed. Then the user is able to like or unlike these posts.

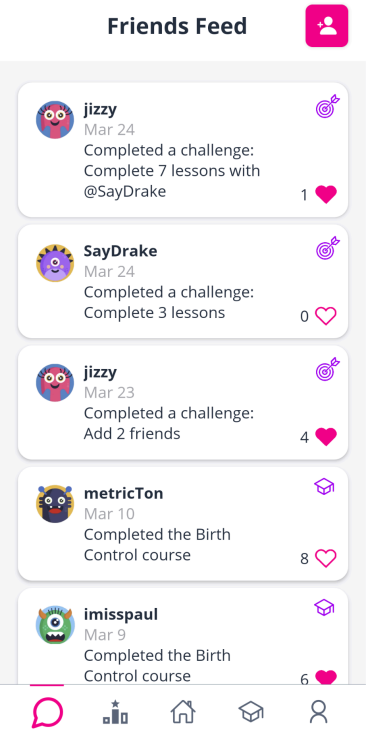


Figure 25: Friends Feed

### Challenge Page

The Challenge page, as displayed in Figure 26, allows the user to view their progress in their current challenges as well as see their completed challenges. There are different categories of challenges, lessons, courses, points, and add friends. As well, the user is given both solo and friend-based challenges, allowing them to accomplish a challenge alongside their friend to add accountability and camaraderie.

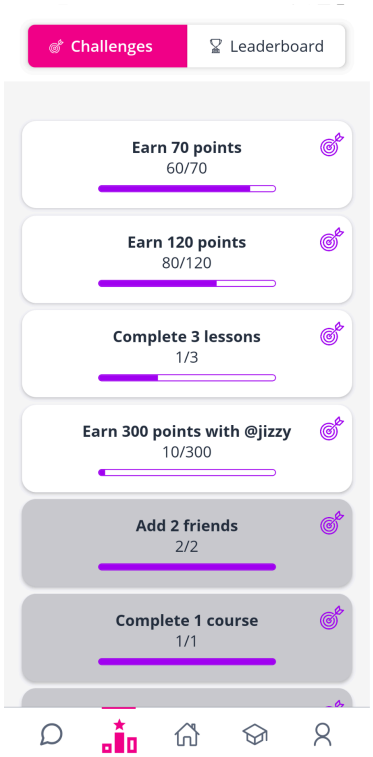


Figure 26: Challenges Page

## Leaderboard Page

The Leaderboard page, as displayed in Figure 27, allows users to view their current ranking compared to their friends. This leaderboard updates when a user has earned points and changes the ranking accordingly.

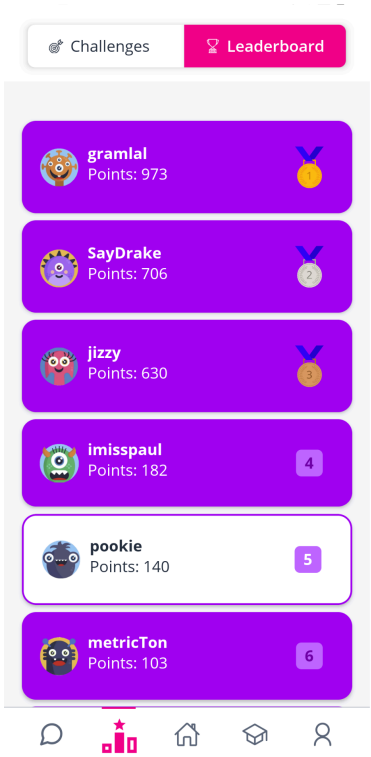


Figure 27: Leaderboard Page

## Course List Page

The Course List page, as shown in Figure 28, displays all the courses available to the user. The list will adjust which courses are displayed at the top based on the user's preferred courses that they stated in the Entrance Survey. Users are able to search for courses and filter the list of courses based on course duration and rating, as shown in Figure 29.

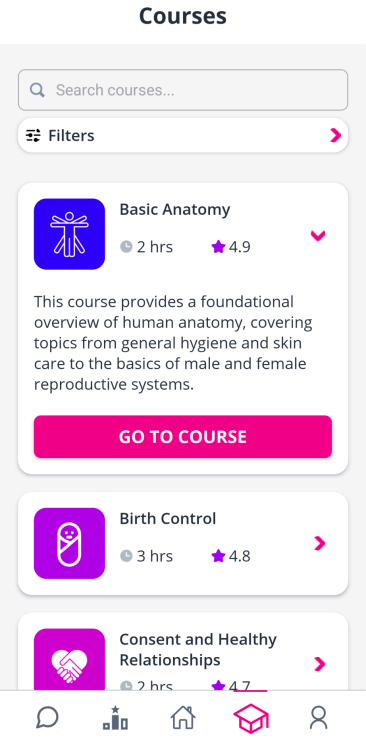


Figure 28: Course List Page

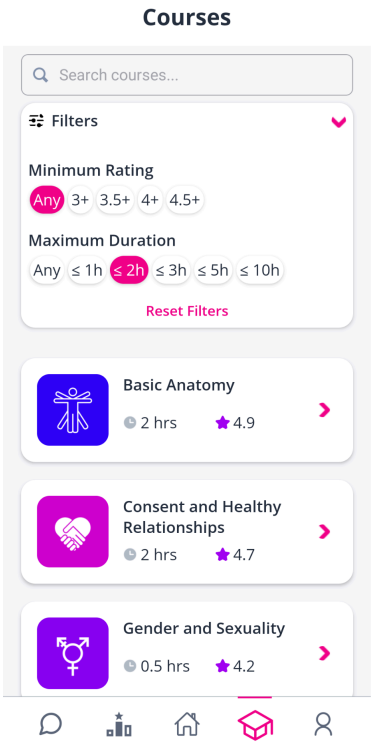


Figure 29: Course List Page Filtered

## Lessons and Quizzes

The Lesson page displays content for each lesson in the quiz. The lessons can be text-based, as shown in Figure 30, spanning multiple cards if needed or video-based. Once the user has completed a lesson they are able to start the lesson quiz. This quiz will have questions of a variety of types such as multiple choice, true or false, multiselect, and matching, as displayed in Figure 31 The user can only complete the lesson if they pass the quiz, as shown in Figure 32. Once the user has completed all of the lessons in a course, they will be prompted to complete an end-of-course quiz. This end-of-course quiz will consist of one random question from each lesson in the course. After the user has passed this quiz they will be able to complete the course and achieve a course badge.

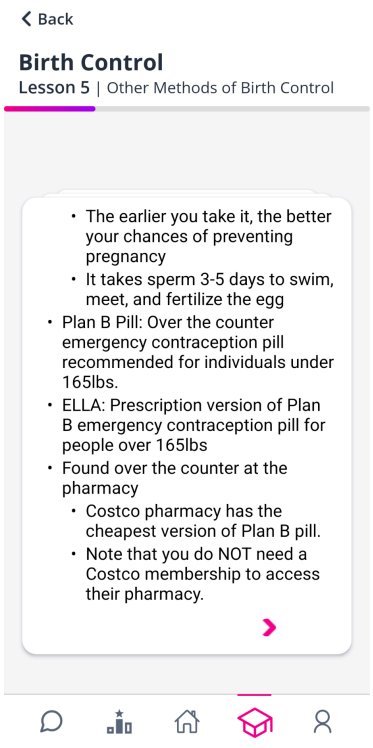


Figure 30: Lesson Page

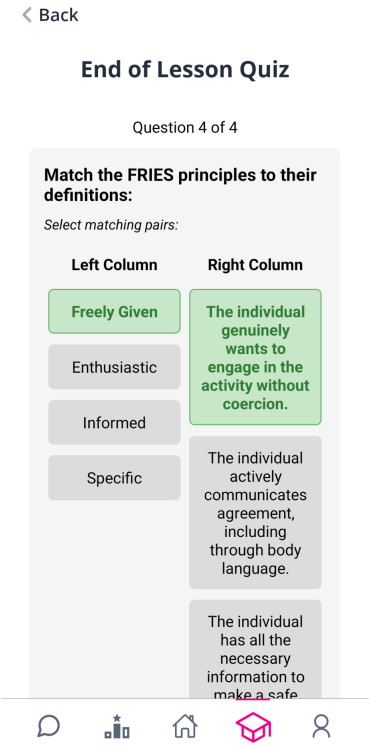


Figure 31: Matching Question

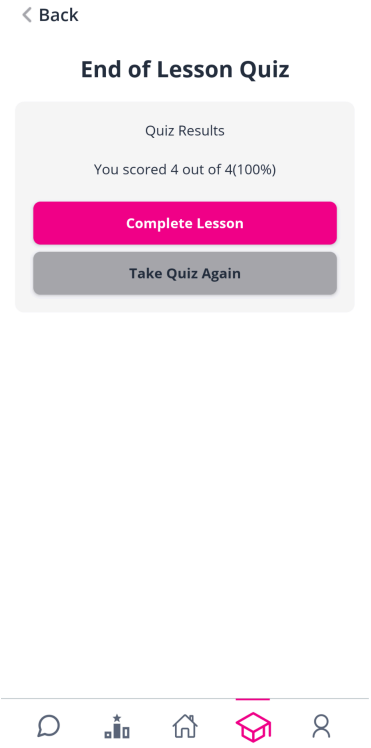


Figure 32: Quiz Completion

## Course Completion Flow

Once a user has passed the end-of-course quiz, they will be prompted with a pop-up, displayed in Figure 33, informing them that they have completed the course and earned a badge for completing the course. The user is also able to fill out the Course Completion survey, as shown in Figure 34. This survey allows the user to rate the course, updating the course rating that is displayed on the Course List.

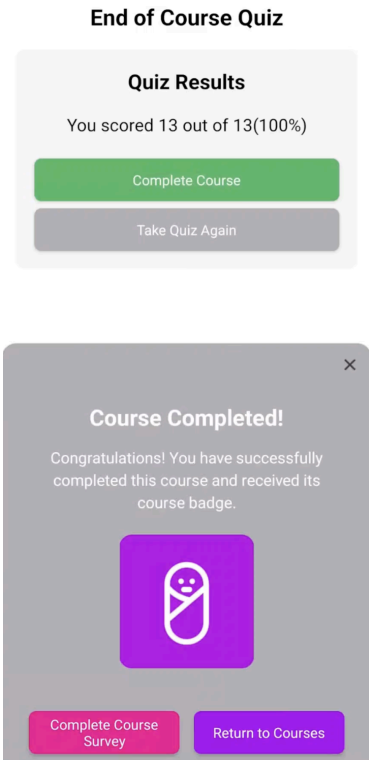


Figure 33: End of Course Pop-up

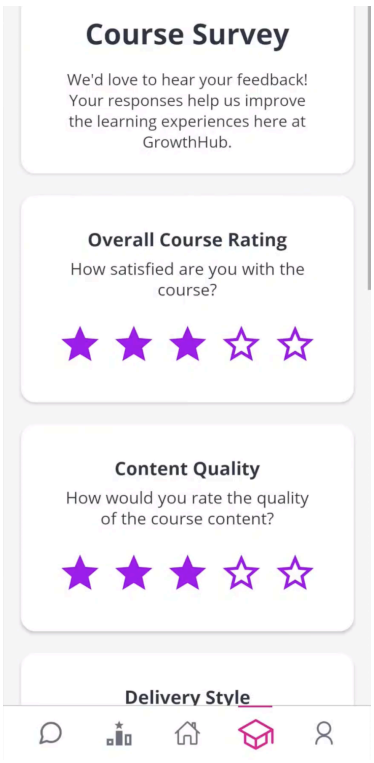


Figure 34: Course Completion Survey

## Profile

The user's Profile page, as shown in Figure 35, displays the user's information such as their username, ranking, and number of friends. The "Ranking" button directs the user to the Challenges/Leaderboard page, and the "Friends" button directs the user to the Add Friends page, as displayed in Figure 36 This page allows the user to search for friends to add as well as remove their current friends or look at their profile. The user's Profile page also displays the user's current courses and badges they have earned for completing a course.

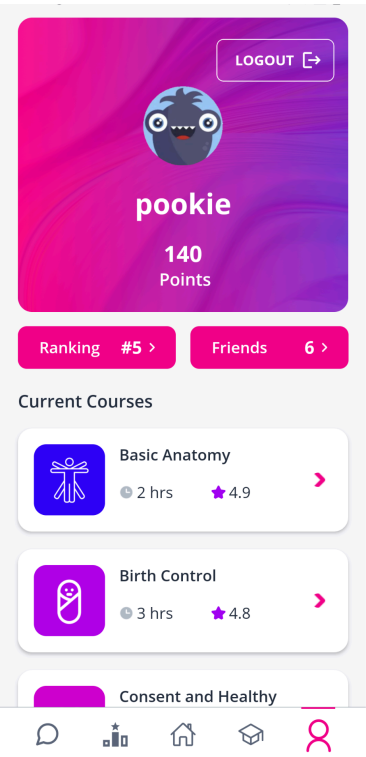


Figure 35: User Profile Page

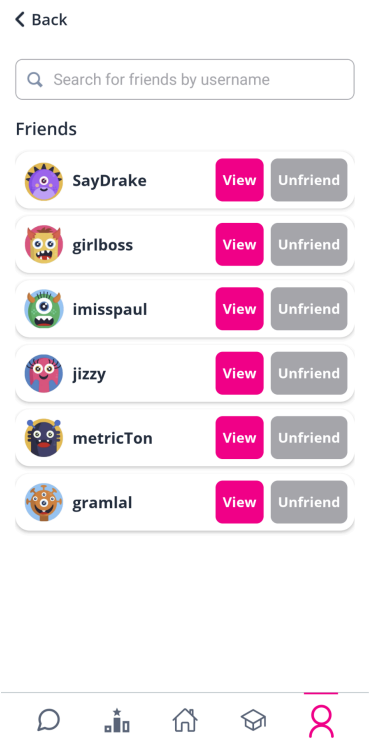


Figure 36: Add Friend Page

# Deployment Information

## Tech Stack

For our app, we used React Native and Node.js for our front-end development and TypeScript for our back-end development. We also used Expo Go to enable a quick and easy app setup and allow us to host our app. This also allowed us to set up our app so it can be used on both Android and iOS devices. Another reason why we used Expo Go was because it has built-in tooling that can be used easily in React Native. We chose to use TypeScript as well because it is the recommended language to use for React Native as they work well together.

## Hosting Environment

We hosted our app on Expo Go, allowing users to run our app from the Expo Go app directly. All of the data for our app was stored on Google Firestore which allowed us to easily store and retrieve data. Expo Go provides direct support for Google Firestore's Firebase services, allowing them to be easily implemented. Most importantly for our product, Google Firestore's real-time database updating works efficiently with Expo Go's reactive framework. This allowed us to update the data being displayed to the user, such as their ranking or course progress, seamlessly.

# Verification and Validation

## Testing

When developing our app, we made sure to continuously test as we were developing. Every feature or group of queries that we developed was contained in its own branch and would only be merged with the main branch once it had been thoroughly tested. This allowed us to ensure that each feature worked and that it would not affect the rest of the app.

Once all our features were implemented, we completed white box testing to ensure that our requirements were fulfilled. This consisted of examining each requirement and testing the related features to ensure that these requirements were met. Additionally, after the core features were implemented, we initiated regression testing. This was done by adding additional features, such as video lessons and matching questions, that were not needed for the base functionality of the app. Once we added in these features, we tested both the new feature and the original features, text-based lessons and other question types, to ensure that these new changes did not affect the original features.

Lastly, we tested our UI to ensure that it abided by the Web Content Accessibility Guidelines (WCAG). We ensured that our background and text colours had high readability and that our text was organized with clear headings. This increased the readability of our app making it more accessible for users. Additionally, we maintained consistent navigation elements throughout our app by ensuring the Nav Bar was visible on every page and that back buttons remained visually consistent.

## Bug Tracking

After all our features were implemented, we conducted thorough testing of our app, logging the bugs we found. These bugs were logged with pictures and videos of the bug as well as where they were found. Then, we each resolved our bugs and iterated on this process until we could no longer find any bugs to fix.

# Individual Contributions

## Amanda

- Wrote the technical report
- Created the final demo video
- Created the symposium demo video
- Created the symposium poster
- Built the Friends Feed page
- Built the Leaderboard page
- Built the Challenges page
- Implemented the functionality of the Home page
- Created and implemented video lessons
- Implemented functionality to update user's points and challenges
- Implemented functionality to keep track of user's progress in courses
- Connected Course Completion survey to database, updating the course ratings
- Created queries related to the user profile, challenges, courses, friends feed, and friends
- Created lessons and quiz questions for the Birth Control and Pleasure courses
- Conducted extensive debugging and fixed the found bugs

## Giselle

- Wrote the technical report
- Created the final demo video
- Created the symposium poster
- Created the UI for the Home page
- Set up database and collections structure
- Built custom components for each different question type featured in quizzes (Multiple Choice, True or False, Multiple Select, Matching)
- Built the Course List page
- Built the Lesson Quizzes
- Built the Course Quizzes
- Implemented the navigation functionality to connect the Course List, Lesson, Quiz, and End-of-Course pages
- Implemented functionality for individual question verification
- Implemented functionality for lesson and course quiz verification
- Created queries related to courses, questions, friends, and submissions
- Implemented functionality for the course badges on Course pages
- Created lessons and quiz questions for the Consent and Healthy Relationships course and Resources content.
- Conducted extensive debugging and fixed the found bugs

## Mitko

- Implemented the Profile page and Friend Profile page

- Implemented the Log-in/Sign Up page and connected it to Firebase Authentication
- Built the Add Friends page
- Built the written Lesson page
- Built the Nav Bar
- Created queries related to submissions
- Implemented functionality for course badges on Profile page
- Created lessons and quiz questions for the Basic Anatomy course

### Jui

- Created the Entrance Survey and began implementing the functionality
- Created the Course Completion survey
- Created a Resources page
- Created queries related to courses
- Created lessons and quiz questions for the Gender and Sexuality and Healthcare courses