Tech Feasibility Draft

10/2/2025

Team Hive Thrive

Team Lead	**Elijah Sprouse**	**ebs233@nau.edu**
Member	**Andrew Velez**	**ajv359@nau.edu**
Member	**Benjamin Levine**	**bjl348@nau.edu**
Member	**Latisha Talayumptewa**	**lt537@nau.edu**

Mentor | Md Nazmul Hossain

Sponsor | Dr. Okim Kang

Contents

1. Introduction

About 73% of 16- to 24-year-olds suffer from loneliness. No doubt, Generation Z is the generation suffering the most from loneliness. This is a unique issue that leaves researchers scratching their heads. Some blame the global pandemic, while others blame an increase in digital communication. Many think it's a combination of factors. As this generation grows up to become functioning members of our society, social habits and patterns will pass from them to the next generation. Unfortunately, those patterns might include this "loneliness epidemic." This means that not only does this issue affect the social and mental health of teenagers and young adults everywhere, it may spread to countless future generations to come; and time is of the essence to mitigate it.

Many people have devised creative yet lacking solutions. One could claim that while social media was created with the intention of bringing people together, it only takes them further apart by creating less reason for physical interaction. In other words, people are more inclined to communicate by sending a quick text message rather than making time to meet in person. Video calls are another, more recent solution devised to bring people together. In the last decade, it has become an increasingly reliable way to communicate; even professionally. However, while it is an effective communication tool, technology should not be used as a substitute for human interaction; it should enhance and encourage it.

Our vision is a progressive web application that acts as a healthy lifestyle coach for struggling teenagers and young adults. The product will provide dynamic textual feedback based on user reports (i.e., users will be rewarded for healthful habits). In addition to this, the product will use gamification to ensure an engaging experience for a technologically savvy generation. Finally, this product will encourage users to practice healthful habits, like taking a 15-minute walk, by taking care of a virtual pet that visually changes according to recent progress.

This unique approach has the potential to change the lives of Generation Z everywhere, and thus, the world. While some solutions incorporate some aspects of our product, none execute all of them in an effective way. The gamification element will make an experience that keeps the user coming back for more. Finally, with live feedback and a visual representation of the user's progress via their virtual pet, our progressive web application will be a powerful, positive routine that users won't have to force themselves to interact with like a chore.

Since we have established that our product is necessary and impactful, it should be appropriate to discuss the technicalities of our project. At this current stage in development, it is imperative to evaluate the upcoming challenges we expect to face, define alternatives, and select alternatives based on a logical and practical analysis. In this Technological Feasibility Analysis document, we have identified several fundamental challenges and solutions, along with reasoning for each, in the following sections.

2. Technology Challenges

1. Mobile Application Development
    - We will need to evaluate the possibility of developing a wrapper for the existing code to make it suitable for Android and iOS app stores
    - We must examine how it may affect the project timeline, since new project features are the highest priority.
2. User Authentication and Security
    - We will need a secure way for users to log into the application and access/manage their accounts.
    - As this project involves collecting adolescent health data, security is a crucial issue.
3. Data Storage and Management
    - We will need a reliable database that is secure and highly accessible to store user credentials, user health data, and gamification progress.
4. Notification System
    - We need a way to deliver timely, reliable reminders while avoiding overloading users.
5. Gamification System
    - We will need to be able to integrate interactive elements like avatars, rewards, and progress tracking to keep users engaged.
6. Offline Usability
    - We will need a way to allow access to the application offline while maintaining its functionality.
7. UI/UX Improvements
    - We will need a seamless, modern interface, equipped with a responsive design that allows for easy interactions and a quality experience.
8. Interactive Teaching
    - We will need to create a feature (either using AI or not) to give feedback and teachable content to users based on their results.
9. AI Integration (Stretch Goal)
    - We will need to evaluate the possibility of integrating AI into the virtual coaching system to provide personalized responses and feedback.
10. Application Scalability
    - We will need to evaluate if the application can be scaled to a massive amount of users, and what that amount is.
11. User Testing
    - We will need to implement a testing system for the application while features are being developed.

3. Technology Analysis

The primary technological challenge that we will face in this project is the development of a mobile application. In addition, we face the challenge of making notifications occur in such a way that users can rely on them in the case they forget to check in, while also not overwhelming their inbox. A stretch goal that we have which would be quite the challenge is using AI to virtually coach users based on their results. This could possibly take more time than we actually have for the project due to our experience level, so this is still a stretch goal. Below, we will analyze each of these challenges:

3.1 Mobile App Development

The Issue

While our client wants to maintain the web application, she is also extremely interested in making it accessible via the Android and iOS app store. The existing PWA is functional and accessible already online, but making it available through these venues could improve usability and engagement for our target audience: teenagers and young adults. A native mobile application would also allow us to implement features such as push notifications and offline functionality in a way that is a lot simpler, effectively solving those other challenges. To balance feasibility and performance, we are exploring the possibility of using a wrapper to make our current PWA into a mobile app.

Desired Characteristics

The ideal approach should maintain or improve the efficiency and accessibility of our existing codebase, while enhancing the user experience through capabilities that can only be achieved through a mobile app. The application should:

- Provide a smooth, responsive interface that feels natural on mobile devices.
- Allow for easy deployment to both Android and iOS app stores.
- Use mobile features such as notifications and storage.
- Be developed in a timely manner to allow for PWA feature development.

Alternatives

There are three main options to choose from. The first is continuing developing the PWA without a wrapper, and optimizing it further for mobile browsers. The second solution would be to develop a completely new application from the ground up using a mobile app framework like Flutter. Finally, we can choose not to develop for a mobile application at all, since it is not necessarily a requirement, though it is desired.

Analysis

Continuing development as a PWA would save time, but limit critical features, such as the ability to send push notifications and access device features (i.e., a step tracker for tracking if a user takes a walk). Building a fully native app would allow for best performance and full access to mobile device capabilities but would require significantly more time and experience than our team has at the moment. Using a wrapper provides a balanced approach, which would allow us to reuse our existing codebase while giving access to mobile features. Most wrapper frameworks are well-documented and very compatible with PWAs, so this would be practical for a small development team such as ours. This approach would be a good balance between functionality, performance, and feasibility. Additionally, if we continue to develop the PWA as planned, we could always reassess feasibility for whether or not we want to develop the wrapper.

Chosen Approach

We plan to develop the PWA as intended. When we are satisfied with the features developed, we can assess the time remaining for the development process; then, if time permits, we can begin developing the wrapper.

Proving Feasibility

Using a wrapper to convert our PWA into a mobile app is a proven and practical approach. Frameworks such as Capacitor and Apache Cordova are widely used and well-documented. They allow developers to integrate existing web applications into mobile environments without restructuring the entire project, which fits perfectly within our case. Our current PWA design already supports responsive layouts, so most of the groundwork for mobile deployment is already in place. This compatibility would significantly reduce the risk of integration issues. Additionally, the process of wrapping a PWA does not require an entirely new development cycle, but we'll have to configure, test, and check against app store requirements. We can accomplish this in our existing project timeline once the core web features are complete. To confirm this approach, we can conduct small-scale trials to ensure that features like notifications, offline access, and database interactions function properly in the wrapped environment. With proper testing and documentation, we are confident that this approach can be executed confidently and efficiently.

3.2 User Authentication and Security

The Issue

Because our product involves collecting sensitive data from users, like their personal information and health-related data, we need to employ a secure authentication and data protection system. Users must be able to safely create and manage their accounts while trusting that we store and handle their information responsibly. Without a proper authentication and security system, the entire project could potentially be compromised.

Desired Characteristics

Our authentication and security system should provide:

- A simple yet reliable login and account creation process.
- Strong password protection, like complexity requirements and encryption (i.e., hashing).
- Secure storage of user data.
- Token-based authentication so we can improve session security.
- Scalability to handle many users at the same time without compromising performance.
- An application that complies with data protection laws.

Alternatives

We have identified several significant options for implementing authentication. One approach is to build a custom authentication system from scratch, which allows full control over the process but increases complexity and might increase security risk. Another approach could be to use third-party authentication services, like Firebase Authentication or Auth0. These services provide built-in security features such as encrypted storage, password hashing, and optional multi-factor authentication.

Analysis

	Firebase Authentication	Auth0
Ease of Integration	X	
Customization		X
Most Supported Authentication Methods		X
Security Features		X
Scalability	X	X
Pricing	X	
Developer Experience	X	
Compliance		X
Documentation and Support	X	
Best Suited for Our Project	X	

Creating a custom authentication system would give us flexibility, but would also require advanced security expertise and continuous maintenance to prevent any vulnerability. Using a third-party provider would reduce security risks and simplify our implementation. Most existing services are well-documented and reliable, and would integrate well with our PWA.

Chosen Approach

Our approach will focus on implementing a secure, token based authentication system that protects the users credentials and effectively manages sessions. Users will be able to register, log in and manage their accounts through a secure process that encrypts any sensitive information before storage. The system will validate login attempts, issue authentication tokens for active sessions, and ensure users only access their own data. We also plan to include password requirements and recovery options to enhance security.

Proving Feasibility

Implementing a token-based authentication system using a pre-established service like Firebase Authentication is achievable and reliable in our project's timeframe. Firebase provides detailed documentation, built-in encryption, and compliance with online security standards and laws. This will help us ensure the protection of sensitive user information without the need for allocating our time and resources to implementing a custom security solution. Because Firebase integrates easily with JavaScript applications, we can connect it directly to our PWA and use the mobile wrapper later with minimal modifications. Their service also offers tools for user account management, secure password hashing, session tracking, and more. These all align with our desired characteristics. During implementation, our team will conduct testing with simulated user accounts to confirm that authentication tokens are being used correctly, that data is being transmitted securely, and that accounts can be managed by their users without error. These steps will help us to confirm our system can safely handle user data.

3.3 Data Storage and Management

The Issue

Our product involves the use of obtaining data retrieved from the users. It is necessary to contain the following user information:

- User credentials: Username, password, and other information to prove the user's identity to gain access to the application.
- User health data: We want to be able to analyze the users health data to provide feedback and guidance back to the user, as well as using the information to allow us to better understand the health of our users.
- Gamification process: As we are planning to implement a gamification system into our application, it is essential to store the users progress they made during their interaction with the games.

Desired Characteristics

Our ideal database solution would offer scalability, security, reliability, and effective performance to ensure the containment of data can be stored and retrieved safely and efficiently.

- Scalability: Achieving scalability allows to handle increasing quantities of data submissions without compromising performance or availability.
- Security: Encompasses the protection of sensitive data through a variety of measurements to promote data confidentiality, integrity, and safety.
- Reliability: Ability to access the database at all times, along with ensuring the consistency of data is available.
- Performance: This is typically the bottleneck. It is in consideration on how to optimize our database to be able to handle continuous accessing of the database and maintain our applications performance.

Alternatives

As we are considering our desired characteristics for a database, as well as considering our client’s former team developers database, we are approaching the utilization of relational databases. Relational databases continue to remain highly relevant. As compared to other technologies like NoSQL databases and cloud platforms, SQL database management systems continue to be at the center of majority data operations. According to the “2025 Stack Overflow Developer Survey”, 58.2% of developers report to use PostgreSQL, and MySQL follows in second with 39.6%. These reports demonstrate that SQL continues to remain an active tool in

database management. Possible approaches include using the following databases: PostgreSQL and MySQL.

Analysis

In the evaluation of PostgreSQL and MySQL, we evaluated each based on the above criteria and through the process of compare & contrast. Both are open-source, relational database management systems, each allowing for the organization of data into tables with rows and columns and support relationships between tables.

Features	PostgreSQL	MySQL
Better Scalability	✓	
Better Security	✓	
Better Reliability	✓	
Better Performance	✓	
Better Usability		✓
Better Data Flexibility	✓	
Better Concurrency	✓	
Better Cloud Support	✓	

Chosen Approach

As we had weighed the similarities and differences of each database, we have concluded to choose PostgreSQL. It is deemed to be a secure, reliable approach and fits the needs of our project. Our application will store personal health and wellness data, so it's ideal to require a database that contains information safe and accurate. PostgreSQL is known for being stable as it protects data, even when something goes wrong and follows a set of rules to ensure the correctness of data. Another reason is it works well for both small and larger applications. It’s capable of connecting data such as user accounts, progress checking and check-ins, ideal for our project. Overall, PostgreSQL gives us a strong, secure, and flexible foundation to build our application on.

Proving Feasibility

PostgreSQL has been proven to perform well in scalable, secure web applications, making it an appropriate choice for our project. It is fully ACID-compliant, ensuring data reliability and consistency, and supports encryption for secure data storage. Its compatibility with various programming languages and cloud hosting services also makes it flexible for both local testing and production environments.

We plan to implement PostgreSQL alongside an ORM (Object Relational Mapping) tool such as Prisma or Sequelize, which simplifies communication between the database and our application. This approach will allow us to manage data models more easily and maintain security through controlled queries. To ensure reliability, we will test the database with mock data entries representing user credentials, health data, and gamification progress. Load testing will help us confirm that the system can handle concurrent requests without performance issues. Through this combination of secure architecture, controlled testing, and proven database technology, we can confirm the feasibility of PostgreSQL for our application's needs.

3.4 Notification System

The Issue

To keep users coming back, we plan to use notifications to gently push the user towards their goal, without overwhelming them with spam.

Desired Characteristics

Using push notifications, we should give users the ability to customize the time of their notifications or have it set to a “random” time. For instance, maybe a user gets out of school at 2:45, so a notification sent around that time would be the best chance to remind them to check in on the app. Also, if people really don’t want daily notifications, but still have notifications enabled, we should send a reminder notification after a week of no activity. Our notifications should not be forceful on the user, or seem like we NEED them to get on the app immediately. Instead, a gentle reminder to think about their health for a few minutes on the app would work much better.

Alternatives

Alternatives to this solution could be email notifications instead of push notifications. This is less likely to get through to the user base we are going for, as most teens don’t check their email on a regular basis. In addition, our notifications could be sent straight to junk or spam, essentially not even doing anything. This is not ideal, so email notifications are definitely not the best alternative.

Analysis

In our evaluation, push notifications are a much better option for user engagement as well as other results. With email notifications, chances are the user will not even receive them on time and might miss a daily check-in they would have done had they gotten a real notification. Also, emails seem more like spam, so they might be disregarded altogether.

Chosen Approach

	Targets Youth	Engagement	Implementation
Push Notifications	Appears in the notification center, could be customized.	Customized logo and text can attract users to come back to app (example Duolingo)	Requires mobile app development and notification system. Needs to adhere to

			app distribution guidelines.
Others (Email, SMS)	Youth tend to not check email as much as messages and notification centers.	Could get lost in an inbox.	Very simple, could work through Knock API (already set up).

Pros of push notifications include higher user engagement and more daily check-ins. This is what our project plans to bring to the project, so it seems like a good choice. A con is that this requires a mobile app framework that we may not be familiar with. This would make us do extra work to learn how to use it before implementing it, but if the outcome benefits the project so much, it is surely worth it. For email notifications, the pros is that it is much easier to implement. This would be a quick implementation that may already be set up from the previous team, but the cons are the lack of user engagement that would come from it. We have discussed a few ways this could be seen as a negative in the previous paragraphs.

Proving Feasibility

The current implementation of the application has a notification system using Knock API. A script called sendReminders.js schedules the notifications. Notifications are currently sent to emails due to the fact we are still a web app. In addition, the notifications can be edited by us through Knock so that we can still change stuff how we want it. If we do end up going for a mobile application, we will most likely abandon this current approach and use push notifications as it is more efficient and convenient, while also having the ability to engage more users.

3.5 Gamification System

The Issue

To keep users engaged and encourage them to build positive habits, our application will incorporate a gamification system. Our system will aim to transform the process of building and maintaining healthy routines into a far more enjoyable and rewarding experience. By linking real-life behaviors to a virtual, visual representation, users will have a more tangible representation of their progress. This feature is especially important to our target audience, as Generation Z users are technologically savvy and responsive to interactive designs.

Desired Characteristics

Our gamification system should:

- Provide a clear and rewarding connection between user actions and in-app progress
- Include interactive elements such as a virtual pet(s), rewards, levels, and progress tracking.
- Motivate our users through positive reinforcement, rather than pressuring or penalizing users.
- Be visually appealing to keep users engaged.
- Be easy to update, maintain, or expand to include new activities or challenges

Alternatives

One approach is to design a simple reward system that grants points or badges for completing healthful tasks. Another approach could be to incorporate a more complex system involving a virtual pet that reflects user progress and habits. A third alternative is to combine the best of both worlds, integrating rewards, progress tracking, and a visual representation of progress via a pet in a cohesive environment.

Analysis

	Points-Based System	Virtual Pet System	Combined System
Quicker Implementation	X		
User Engagement		X	X
Long-Term Retention		X	X
Visual Appeal		X	X
Technical Complexity	X		
Scalability and Expandability			X
Emotional Connection		X	X
Maintenance Requirements	X		X
Best Fit			X

A basic points-based system would be pretty easy to implement, but might not sustain long-term user interest. Users could easily become bored and drop the app. A virtual pet system would definitely provide a far more engaging and emotionally-connected experience, since users would be able to see their actions reflected through their pet's appearance and behavior. However, this would require more consideration; we would have to design, animate, and program this pet. Combining both systems into a unified gamification system would ultimately create a great balance. Users would be able to earn points that translate into the growth and well-being of their virtual pet, reinforcing positive behavior and keeping the user participating and motivated.

Chosen Approach

We ultimately decided on the balanced approach: creating a virtual pet that works hand-in-hand with the points system.

Proving Feasibility

Our chosen gamification system, which would combine a points-based structure with a virtual pet, is both feasible and scalable. Our system's logic can be implemented using simple database relationships and animations in the frontend that will respond to user progress. Every user action, like completing a check-in, will trigger updates to both the user's points and the virtual pet's visual attributes, which can be managed through the PostgreSQL database. We can achieve the visual and interactive aspects of the pet using lightweight frontend animation libraries such as Lottie or, better yet, CSS-based animations. These technologies are well-suited for web environments and also work great on mobile. The incremental nature of this development process would also allow us to test as we go, perhaps starting with a static pet before gradually adding animations, tasks, and reward feedback. Because all of these technical components are supported by existing frameworks and tools, we can confirm with confidence that this chosen approach will be feasible within the scope and timeline of this project.

3.6 Offline Usability

The Issue

As our web application is currently designed as a PWA, and will be in the process of a mobile application, it will need to be ensured that users can still access the application and record their check-ins offline. Many Gen Z users rely on mobile devices and may not always have a strong or stable internet connection.

Desired Characteristics

Our ideal solution would enable the user to connect to and use the application in an offline setting. Through this offline ability, the application could store essential pages and content so when the user is offline, they are able to access them and continue to input their check-ins or updated health data. When offline, the app should temporarily store the data and when the user reconnects to the internet, the application will automatically save and update in the main database. We plan to utilize offline usability to enable the application to be more convenient, accessible and reliable.

Alternatives

We have identified various approaches to integrate offline usability into our PWA application. The first approach is a simple option, which is to make certain parts of the application (such as the homepage, forums, etc.) available offline through caching. This stores certain pages on the user's device so they can load without the internet. Another approach would use local storage for offline entries. This would allow users to fill out check-ins offline, save their data on their device, and send to the server once an online connection has been established. The last approach would be to use Service Workers, which is a small background helper for PWAs that can serve cached pages, can tell when you're offline, and queue data to send later.

Analysis

In the analysis of the approaches for offline usability, each was evaluated through not only their uses, but also their impact on the user and the objective of our health application. Caching static content is an easy approach to improve speed and reliability, however users would only be allowed to view data, not submit new data. Local storage for offline entries allows users to keep daily streaks and progress through checking in offline, however it requires a sync strategy. Service workers would allow the application to feel more real. The approach allows it to load offline, show recent data, accept new check-ins, and can auto-sync later on. It combines fast offline loading with reliable sync-later ability. However, this approach requires more moving parts.

	Static Caching	Local Storage	Service Workers + IndexedDB	Mobile Wrapper
Offline Loadability	X	X	X	X
Offline Check-Ins		X	X	X
Automatic Synchronization			X	X
Display Offline / Sync Status			X	X
Best Fit			X	X

Chosen Approach

Our offline plan will follow two paths that share the same data model and API calls.

- If the mobile application wrapper is not completed:
  We will implement offline usability in the PWA using Service Workers for caching and IndexedDB for a local write queue. The app will open and function with previously cached screens, accept new check-ins offline, and sync them to the backend when a connection is available. The interface will show clear status messages for offline, queued, and synced states.
- If the mobile application wrapper is completed:
  We will keep the same Service Worker and IndexedDB pattern but add wrapper capabilities to improve reliability. We will use the wrapper's network status API to detect connectivity more accurately, and the wrapper's file system to store larger cached assets if needed. On Android, we will schedule periodic sync on app resume and at safe intervals. On iOS, we will sync on app launch and resume, and after significant time gaps. In both cases, the user experience and data flow will remain consistent with the PWA path.

This dependency ensures we can deliver a solid offline experience even without the wrapper, while allowing quality improvements once the wrapper is in place.

Proving Feasibility

Both paths are feasible because they rely on technologies that fit our current stack and skills, and they share the same core logic.

- Feasibility without the wrapper:
    1. Use a Service Worker to pre-cache the application shell and core routes with the Cache Storage API.
    2. Store pending check-ins in IndexedDB with an id, payload, and timestamps.
    3. On reconnect, a simple sync loop will post queued items to the backend and remove them locally on success.
    4. Validate by loading once online, switching to airplane mode, submitting a check-in, then reconnecting and confirming the server update.
- Feasibility with the wrapper:
    1. Reuse the same Service Worker and IndexedDB logic inside the web view.
    2. Use the wrapper's network information to gate sync attempts and show accurate status.
    3. Trigger sync on app resume and at safe intervals that respect platform limits.
    4. Validate by repeating the offline tests above, then add app lifecycle tests such as backgrounding and resuming, and short bursts of flaky connectivity.

Because both paths share the same schema, API endpoints, and user messages, we can start with the PWA implementation and add wrapper enhancements later without rewriting features. This staged plan keeps our work aligned with the mobile wrapper timeline while guaranteeing that offline usability is delivered in either case.

3.7 UI/UX Improvements

The Issue

The current implementation has some design features that we would like to change, with some of the buttons not looking as great as we can get it.

Desired Characteristics

We want our UI to be very appealing to our users so that they continue to use our product. In addition, we want all pages to blend together with each other. We do not want one page to have vastly different UI and design choices than the others. The different aspects of our project should all flow into one overall project in the eyes of all users. This is a common design choice used in a multitude of different softwares, and we plan to carry it on to ours as well.

Alternatives

An alternative would be to split the different sections into different design and UI paths. The main reason this would be a possible thing we would do is that it could help a user know where they are on the site based on simply the style choices. However, there are many more cons to this than there are pros. A main con is that it would be more unnecessary work for us as a team, which would lead to less time spent on the actual project's goals. Anything that adds little to no actual benefits but takes time away from the main goals of the project is a big negative.

Analysis

The reason keeping UI consistent throughout works is simply how the UI is supposed to work. The goal of UI and your design is to make the user experience both simple enough to understand while also looking nice. If you switch between UI choices between pages, there is a good chance it will look quite bad while also leading to issues with the user experience and getting confused throughout the site.

Chosen Approach

We will adopt a consistent design system that includes a shared component library, a unified color and typography scale, and clear spacing rules. All screens will follow the same navigation patterns and layout grid so the app feels familiar as users move between features. Components such as buttons, inputs, cards, and dialogs will be designed once and reused across the application to ensure visual consistency and reduce development time. We will define design tokens for colors, font sizes, spacing, and radii to keep styles predictable and easy to adjust. Accessibility will be treated as a core requirement, including sufficient color contrast, keyboard navigation, and descriptive labels. We will create lightweight wireframes to validate structure, then apply the design system for high fidelity screens. This approach provides a cohesive experience that supports engagement and makes future updates faster and safer.

Proving Feasibility

Our chosen UI system will work very similarly to the previous implementation, with some tweaks to make it even more refined and user friendly. As we have all taken classes that have had us practice design and UI, we all have experience with the chosen approach as well. To look even further into the feasibility, our current team website is using this approach, so there is no doubt that our approach to UI/UX will be very feasible and expected of the team.

3.8 Interactive Teaching

The Issue

To be a lifestyle coach, our application has to have some sort of interactive teaching. Without it, it won't be able to be a good lifestyle coach.

Desired Characteristics

Our ideal solution would be to include coaching as a feature based on the user's results. In addition, we plan to have links to resources to get additional information and teaching from other sources (e.g. UnitedWay).

Alternatives

An alternative we have proposed is integrating AI to take the unique results of each user and finding the best way to coach them individually. As of now this is a stretch goal and may not make the final project, but this will be touched upon more in the AI Integration section.

Analysis

The reason the solution we chose works better than the proposed alternative is the fact that our alternative is based on a stretch goal. We really would like to be able to integrate AI into this project to get even more personalized coaching to each user, but none of us have experience with LLMs. If possible, it would be super ideal, but as of now it is a stretch goal, leaving us with a still great way of interactive teaching.

Chosen Approach

We will implement a rules-based interactive teaching feature that provides personalized feedback based on user input and progress. This will include written advice that encourages users to maintain positive habits or improve where needed, along with links to reputable sources such as health organizations or community support pages. This system will create a guided, supportive environment for users while remaining achievable for our team's technical capabilities.

Proving Feasibility

Our chosen approach is fully feasible within our current development scope. Implementing rule-based feedback can be done through basic conditional logic using data already stored in the database. Each condition can trigger specific responses or resource links, allowing us to create meaningful teaching interactions without relying on complex algorithms.

We can store these responses in a structured format within our database and update them as new topics are added. For example, if a user consistently skips a daily check-in, the system could respond with a short motivational message and a link to an article about habit formation. This system requires only moderate programming effort and no third-party dependencies, which keeps it reliable and secure.

Because this approach uses familiar web development methods and allows incremental testing, our team can confidently implement it within our existing timeline. It also provides a

strong foundation for future enhancements, such as AI-based coaching, once we gain more experience and technical resources.

3.9 AI Integration (Stretch Goal)

The Issue

As a stretch goal, our team is exploring the possibility of integrating artificial intelligence into our application to enhance personalization and engagement. The goal is to allow our system to act as a virtual coach that provides feedback and suggestions that are unique based on user input and data. This integration could create a more dynamic and meaningful user experience by tailoring responses to each user's individual case. However, due to the complexity of AI

development and the limited timeframe of our project, this feature is unfortunately an advanced goal; only to be implemented if our core features are completed successfully.

Desired Characteristics

If implemented, the AI integration should:

- Analyze user data to provide relevant and personalized feedback.
- Maintain consistent tone and accuracy across all responses.
- Operate without increasing processing requirements too much.
- Keep user privacy and protect sensitive data.
- Adapt over time through updates or added datasets.

Alternatives

One alternative is to design a rule-based feedback system that uses conditional logic rather than full AI processing. This approach would emulate AI, but would provide pre-written responses triggered by specific inputs or keywords or user progress. Another alternative is to integrate a lightweight machine learning model that can adapt its responses over time but remains limited to a smaller dataset. The final option, which is the most advanced, involves incorporating a fully-trained AI model capable of processing language and real-time feedback generation.

Analysis

A fully developed AI system would definitely provide the most engaging and adaptive experience. However, it would also demand extensive data, computing resources, and time to develop. A much simpler rule-based or hybrid system could still achieve our main goal of delivering personalized feedback while remaining feasible within our current scope. By collecting user data patterns over time, we could design an AI system in the future that builds on the existing foundation once more resources do become available.

Chosen Approach

Adopt a rules-based coaching engine with templated feedback, then optionally add a lightweight assistive step later.

- Phase 1: Build a rules-based “Coach Engine.”
    - Define input signals from existing features: recent check-ins, streaks, step counts or activity flags, mood entries, and goal completion.
    - Score these signals into simple tiers, for example Low, Medium, High.

    - Map each tier to clear feedback templates that we write and review with the sponsor.
    - Personalize templates with safe variables such as first name, current streak, last check-in time, and pet status.
    - Store all templates and mappings in the database so they are easy to update.
- Phase 2 (optional, only if time permits): Add an assistive wording pass.
    - Use a hosted text service to lightly rewrite our selected template for tone and clarity.
    - Keep strict prompts and guardrails. Do not send raw health details. Do not store user data externally.
    - Fall back to the original template if the service is unavailable or returns low-quality text.
- Guardrails and quality:
    - Keep all coaching factual, supportive, and non-clinical.
    - Add a review checklist for bias, accessibility, and age-appropriate language.
    - Log which rule fired and which template was delivered so we can audit results.

Proving Feasibility

Our chosen approach for AI Integration is feasible because it builds on technology and methods that are already within our team's current skill set. A rules-based coaching system can be implemented using conditional logic and templated feedback, both of which are familiar concepts in standard software development. This approach does not require any advanced AI training, model tuning, or external hosting, which keeps the workload manageable and the data secure.

We can design and test the logic directly within our existing application structure using JavaScript or a similar language. This allows us to create dynamic responses that adjust based on user input or progress, effectively simulating an intelligent system. The logic can be refined over time through testing and user feedback, making it adaptable to future improvements.

In addition, this method provides a solid foundation for future AI integration. Once the rules-based system is complete, we can gradually introduce lightweight enhancements such as simple natural language generation through secure APIs, if time and resources allow. This phased approach ensures that our application remains fully functional even if we cannot complete the advanced AI goal. By prioritizing a rules-based system, we ensure that our application remains achievable, safe, and scalable within the scope of our current knowledge and project timeline.

3.10 Application Scalability

The Issue

When we get to the point of release for our product, we have to look towards the ways it will be supported as we continue on. With our product set to be immediately available to the youth of Northern Arizona, we need to make sure that we can handle large amounts of users. Scalability will be essential as we do not know how many people we will need to handle until they need to be handled.

Desired Characteristics

For our product, we want to be able to scale at whatever rate is needed for success. One such way to measure scalability is how many people can concurrently be using the application at once. As we picture users at least checking in daily, there will definitely be a need to address this measure. Another way to measure this is through the offline usability that we touched on earlier. Offline should still function as the same application without features that directly require a connection. Also, we want scalability of our backend and database, so that we are not overwriting previous data if our expectations are exceeded.

Alternatives

It is hard to think of alternatives to handling scalability, since you either handle it or you don't. If you don't, users will not use your product which will force you into going back and testing for it. Regardless of what approach you want to do initially, eventually you will have to test for scalability and work to improve it. It makes the most sense to do it before you have a user base so that when you do get one, it will only have to be expanded instead of implemented.

Analysis

A product with a good basework for scalability is very needed in the software sphere. Without it, an otherwise solid project can fail as soon as consumers begin to find it. Testing scalability of multiple levels of the product, such as concurrent users and the database, is a make or break for the overall success. Testing must be done to make sure our application is scalable, and at what rate it is. If this rate becomes not enough, we should also know ways to make it more scalable as it grows.

Chosen Approach

The approach we will choose is to test for scalability as we are doing our user tests. It depends on the exact types of tests that we will do based on if we are using a web app or mobile app, but either way there will be validation checks and implementation done to improve and expand scalability.

Proving Feasibility

While we will not be able to directly test scalability until our product is in the testing stages, we did some research on how testing scalability in a web app usually works. The testing is most likely different for a mobile app so there could be differences in how we go about it judging on how our final product is. Scalability testing validates and checks the performance of software, increasing load and putting pressure on the system. Our goal of the tests will be to

evaluate risks of higher loads, understanding of limits and how these limits are reached, and the strengths in our software. Using these strengths as a base, areas that are limiting scalability can help be solved and lead to higher ceilings. In addition, the scalability tests will be done in multiple scenarios, so as to not miss anything and make sure we can deliver the best product.

3.11 User Testing

The Issue

User testing is an essential step in ensuring that our application meets the expectations and needs of its intended audience. Since our goal is to create a positive, engaging experience for teenagers and young adults, it is critical that the system is both intuitive and enjoyable to use. Conducting user testing will allow us to identify usability issues, gather valuable feedback, and make improvements before the final release. Without user testing, we risk overlooking issues that could negatively affect user engagement, satisfaction, and overall success of the product.

Desired Characteristics

Our user testing process should:

- Involve participants that accurately represent our target demographic (teenagers and young adults).
- Evaluate both usability and overall experience, including layout, navigation, and feature clarity.
- Include structured feedback methods such as surveys, observation, and task-based testing.
- Be iterative, allowing us to make improvements and retest as the application evolves.
- Prioritize accessibility, ensuring that all users can comfortably navigate and interact with the system.
- Respect participant privacy and follow ethical guidelines when collecting feedback or data.

Alternatives

One approach is to perform in-person user testing sessions, where users interact with the application under direct observation. This allows for real-time feedback and a deeper understanding of user behavior. Another option is to conduct remote testing sessions, where participants use the application on their own devices and complete a feedback form afterward. A third alternative is to release a small public beta version of the application to a limited group of users who volunteer to test the system and provide feedback.

Analysis

In-person testing provides detailed qualitative feedback, but it requires more time and coordination. Remote testing is more flexible and can reach a wider group of participants, though it may yield less detailed observations. A limited beta release would allow us to test the application in real-world conditions, but it also introduces potential risks if major issues are discovered after release. A combination of these methods would provide the most comprehensive feedback while balancing time and resource limitations.

Chosen Approach

We have not exactly chosen which approach we will do for testing, but we plan on deciding before our next client meeting. The two options are publishing our findings publicly and letting more people join tests, or doing it in house and keeping the data to ourselves.

If we go with the option of publicly sharing our results, we need to obtain permission from NAU. The way that we would have to do this is through an IRB protocol. This protocol will take quite a bit of time to draft, and even after submitting will take some time for review before being approved. Therefore, we have to be able to commit to it well before our testing occurs.

This is one reason we have not decided on it yet, we want to have the best option available to us with as much information as possible before choosing.

Our second option is to do our testing informally, most likely relying on our client to ask for some help from her students or others. This informal testing is not allowed to be publicly shared without the permission of NAU and the approved IRB protocol. This option can still yield good results, but if our client wants us to formally publish our findings and how we work through them, this is not a possible route to go.

Either way, our chosen approach for user testing will combine both in-person and remote testing methods. We plan to begin with small in-person sessions where users from our target demographic will be asked to complete specific tasks, such as registering an account, performing a daily check-in, or interacting with their virtual pet. During these sessions, we will observe how users navigate through the app and take notes on points of confusion, difficulty, or positive feedback.

Following these initial sessions, we will distribute the application for remote testing to a slightly larger group. These participants will use the application on their own devices and submit structured feedback forms that measure usability, engagement, and satisfaction. This two-phase approach allows us to first identify major design or functionality issues through direct observation, and then confirm improvements with broader input from remote users.

This method ensures that we can refine the user experience iteratively and make data-driven design decisions. It also minimizes risk, since major issues will be caught early before the application is released to a wider audience.

Proving Feasibility

This approach is fully feasible within our current project timeline and technical capabilities. Conducting small-scale in-person and remote tests requires only coordination, documentation, and communication tools that are already available to our team. For in-person testing, we can recruit volunteers from our university community, ensuring that participants represent our target demographic of teenagers and young adults. Testing can take place in campus computer labs or other controlled environments, allowing us to observe users while maintaining consistency in device and network conditions.

For remote testing, we can use simple feedback tools such as Google Forms or survey platforms to collect data on user experience. We can also ask participants to record brief notes about their experience, including any confusion or technical issues they encounter. This information will help us pinpoint usability problems and identify improvements needed before launch.

By combining both in-person and remote testing, we can balance depth and reach in our evaluation process. Each testing phase can be completed within one to two weeks, depending on availability and participation rates. Because this approach uses common tools, requires no special equipment, and relies on iterative feedback, our team can confidently carry it out within the scope of the project.

4. Technology Integration

Technology Integration:

To bring everything together, our goal is to build a simple connected system where each part works smoothly with the other. All the smaller solutions such as authentication, data storage, notifications and gamification will link up through one main structure that keeps the app secure, reliable, and easy to expand later.

Overall Setup:

- Frontend (Mobile App / PWA): The main interface the user sees. This handles check-ins, tracking progress and the virtual pet. It will also support offline use by caching key data and syncing it.
- Backend / API: Acts as the middleman between the app and the database. Manages logins, stores updates and sends notifications when needed.
- Authentication Layer: Verifies users and controls access to personal information and data using secure login tokens.
- Database: Keeps all user information, health data, and gamification progress protected and organized.
- Notification System: Sends reminders or alerts based on the user's activity level or chosen setting.
- Gamification Engine: Tracks progress, updates the virtual pet, and rewards consistent activity.
- Interactive Teaching Module: Gives users personalized tips and feedback. This could later connect to AI if we expand the system.

Flow:

When a user logs in, the backend confirms their identity and pulls their saved data from the database. The frontend displays their progress and pet updates. When they log new activity, that data will be sent to the backend, stored securely, and used to update the streaks and rewards. Notifications remind users to check back in, and if they go offline their actions will be saved locally and synced once they reconnect. This system keeps all systems connected while still staying flexible allowing for future features to be implemented such as AI integration or full mobile support without major changes.

5. Conclusion

The loneliness epidemic affecting Generation Z presents an urgent and complicated challenge; one that demands a both empathetic and innovative approach. Our project seeks to address this by promoting genuine, healthful habits through an engaging, gamified experience. By merging behavioral science principles with accessible technology, our product will help young users across the world build better routines, strengthen their social and emotional well-being, and ultimately improve their quality of life.

This feasibility analysis explores our technical considerations in making this vision possible. We examined critical challenges such as secure user authentication, effective data management, and reliable notification systems, while also evaluating possible paths to gamification, offline usability, user experience design. We also considered advanced opportunities like AI integration and interactive teaching. Each component contributes to a plan for a successful software that aligns with the needs of teenagers and young adults.

Through careful research, collaboration, and structured analysis, our team determined that this project is technically feasible within the scope of our current resources and strengths. Moving forward, our next steps include finalizing smaller details. With the continued support from our client and mentor, Team Hive Thrive is confident that we can carefully build a system that doesn’t just work efficiently, but also makes a meaningful contribution to the lives of our users.