Technical Documentation: Project Amiga

Alex Welsh, Anna Mola, Anthony Fiddes (PO), Caroline Yoon
Julia Levine, Mario Becerra Aleman, Rahul Nair (SM)

Second Draft: November 28, 2020

Introduction

This technical documentation aims to explicate the process, features, and technologies that were used during the development of our semester-long project submission for the *CS 370: Software Engineering Practicum* course.

Project Amiga was designed to be a mobile application that assists college-aged individuals with a friendly, user-centric design that enables logging and mood tracking to visualize historical progress of mental health. The intention is that users will be able to connect their log entries to real-world events in their lives. In turn, this empowers users to identify and control stressors and stress-reducers, and manage their mental health in a more proactive way. Such "journalling" can also be shared with medical providers and loved ones to further evidence trends in mental health over time.

Each section will explain, in more technical detail, each titular feature, and will walk through the design, development, and execution/integration of said feature into the total application.

1. Agile, Technical Beginnings, and How We Work
2. Front- and Back-End: Design and Development
3. Logging: Keeping Track of Moods
4. Stats Page
5. Gamification with Friends and Achievements
6. Profile Details
7. Miscellaneous Technical Details

Agile, Technical Beginnings, and How We Work

This section will explain how the team aligned on a common objective and began executing towards it using the Agile methodology of software development. It will also discuss how *Project Amiga*'s technical workings first started.

Agile

At the beginning of the term, after the course's Agile exam and team formation, the team met to align on a project idea. After having unanimously agreed on the mental health-focused *Project Amiga*, the team was unable to identify a formalized "business" entity in the scope of our course. Therefore, the team itself became the business stakeholders, and the Product Owner and Scrum Master worked in tandem with "business" to decide on an overall "project vision."

Crucial to our team's successful execution and determination of project scope was our use of our Agile board. Hosted on Microsoft's Azure DevOps ("ADO") platform, the board and linked product backlog served as a transparent way for the team to visualize current and future work items, align on project scope, and understand who was working what to facilitate load management. It became the responsibility of the Scrum Master to establish and lead team ceremonies, such as Backlog Refinement, Team Retrospectives, and Daily Stand-Ups, to ensure the team was in constant alignment, dealt with minimal dependencies, and moved forward together.

As is common at the outset of real-world software development projects, the team held an initial meeting to populate work items for the entirety of the term in something akin to a *Program Increment* refinement session. This initial session of backlog population served as a foundation upon which the development timeline of the project was created. The team determined the features necessary for the final deliverables, prioritized said features, estimated total development times for each, allocated each to one of ten sprints, and discussed the medium of delivery for the final project. What came of this session was a robust product backlog, which informed a rough development timeline.

One of the key Agile guidelines the team agreed upon was to keep the board up-to-date with updates in PBI status, scope, and definition of done. The Scrum Master took special care to enforce these rules while challenging the team to maintain a constant sprint velocity of one feature/sprint. This meant that one large work item (usually a collection of three or more PBIs or 15+ effort points) was expected to be completed each sprint. This ensured that the team remained motivated and completed agreed-upon work to an agreed-upon standard.

Techstack

The first three sprints were dedicated to the completion of "SPIKE" product backlog items (PBIs), which laid the foundation for important decisions that were made further along during the project timeline. One major SPIKE item was the determination of the techstack used; after the team aligned that *Project Amiga* should be a mobile app, there was doubt as to which technology should be learned and leveraged to best execute the project vision. The leading candidates were Flutter, Xamarin, and React Native. During the first three sprints, the team split into Design and Development sub-teams; Design was tasked with using Adobe Xd and open-source images to create a mock-up design of what the app would look like on both Android and iOS, while the Development team split each member to a techstack and had a reasoned discussion to determine a techstack for the long-term execution of the project. More detail on this process will come in the "Front- and Back-end Design" section.

Working Together

Every week saw the team complete slides in preparation of a Sprint Review session to the larger course audience, and the team would take that time to also complete code reviews. Using GitHub for Source Code Management, the team submitted Pull Requests (PR) with the intent of committing feature branch changes into the mainstream **master** branch. Team members would congregate to read through the PR contents together, and then execute the code on their local machines, testing for errors and bugs on simulated and real-world iPhone and Android devices. Fixes would be implemented as needed, and upon approval, the code would be merged into **master**, ensuring stable, shippable, and buildable code at all times.

Front- and Back-end Design/Development

This section will give a high-level explanation of design and development in front- and back-end sectors of *Project Amiga.*

Design

As previously referenced, the initial split of the team into Design and Development sub-teams led to the creation of mock-ups of app structure and user workflow regardless of final techstack. Tools used were **Adobe Xd**, **coolors.co**, and several **open-source libraries** to create vivid designs with functional buttons, headers, menus, and modals for interaction. The ability of Xd especially was valuable in sharing these functional mock-ups as detached, mini-web apps for use by the Development team in their creation of the final deliverable. Importantly, the Design phase was conducted with particular attention to the features determined at the beginning of the project. A key component of mapping out user workflows was to imagine how users would transition from one feature to the next;

for example, a user would open the app, log their day's mood, and transition to the **Stats** page to view a day-over-day progression. Such workflow modelling and feature development provided vital blueprints for the Development team to follow as the app matured. The Design team transitioned into the Development team by the end of Sprint 4, with all of the team working on Development in some capacity by Sprint 5.

Development

Development was the other sub-team that came out of the initial division of labor at the outset of the project. As such, the first task of the Development team was to determine which techstack to use for the duration of the project. Initially, a lot of momentum around "newer" technologies, such as Flutter, motivated discussions, but the team decided that the final techstack should...

- be cross-platform.
- require a short learning curve.
- execute quickly and easily upon the user's device.
- have robust libraries for integration into back-end techstack
- have lots of documentation.

After Sprint 3, the determination was made to move to React Native. It represented the best "learning opportunity" for the team, as many expressed an interest in learning JavaScript, HTML, and CSS. The other pre-requisites fulfilled, the next big step to take was to make determinations on how to build out the back-end.

Establishing a MERN stack (MongoDB, ExpressJS, React, Node) meant that several key components would need to be instantiated, but this quickly was struck down due to the limited time available for the project's execution. As such, an out-of-the-box, back-end as a service (BaaS) offering was sought after, and the free *Spark* tier of **Google Firebase** provided an excellent solution that seamlessly integrated with React Native.

Thanks to the noSQL nature of Firebase's **Cloud Firestore** tool, the team realized significant time-savings from avoiding the instantiation of SQL-style database schema, routes, and key relations that the MERN stack would have necessitated. Instead, **Firestore** established data hierarchically, starting with sub-collections at the root and creating documents as branches. Each document nested customizable fields underneath it, and this structure enabled robust, but compute-efficient sub-collections to be created for users (which contained friends, achievements, and streaks) and log (moods, journal entries, and mood slider values). With a front-end development framework complemented by a back-end to store and. manage the resulting user data, development of our core features could mature during Sprints 7, 8, and 9.

Logging: Keep Track of Moods

The main technical function of *Project Amiga* involved the writing and saving of logs. Users should be able to open the app, login, land on the home screen, write a journal entry, visualize historical log metrics, and close the app after completed. This simple workflow was split into multiple steps that took several sprints to complete. Their technical details are explained further as follows:

- login: handled by **Firebase** auth, the team designed a login page and "create user" login form for new users and password resets. Importantly, the usernames were all moderated by email address, and functionality for "usernames" or "handles" was deemed unnecessary.
- navigation: handled by React Native navigation.
- journal entry: this was the most complex part of the home screen and arguably, of the app as a whole. The front-end components are visualized by LogItem and LogModal, and the back-end storage of these logs reads/writes them in the aforementioned **Firestore**. Users wrote their entry, were able to select specific "mood words" to describe their entry. Here again, users slide a 0 - 100 slider to quantify their emotions on a numerical scale for easy comparison.

- stats: visualizing mood over time was a key feature, enabled and visualized by plotting the scale percentile values against the given date.

With the workflow of the user visualized, it was important to note that all these entries are visible. This means that users can show their progress to a licensed mental health professional, a loved one, or most importantly, themselves for personal self-reflection. By helping democratize this information, the user can feel encouraged to find connections between how they feel and how far they have come as members of the mental health community.

Visualization in Stats

Stats is a core function of *Project Amiga* that enables the user to view progress of mood percentile over time. The graphs generated use create a smoothed plot that visualizes mood percentile, selected on the log from the home screen, against time. View windows show day, week, and month iterations, given enough data exists to populate those views. A calendar below the graph displays a calendar that displayed each logged date in color, creating a colorful graphic that can also be used to show which days had logs and how the user felt on each of those days. Logically, there is only one log entry permitted per day. To promote continued use of the app, **Stats** page also includes small buttons for **Streaks** and **Achievements**.

Gamification: Friends, Streaks, and Achievements

Gamification features in mobile apps have made significant changes in the way users interact with each other and with the contents of the app itself. For example, leaderboards in Fitbit and Apple Health have "gamified" performance, encouraging use of the app as well as affecting desired behaviors - in this case, diet and exercise - onto the app's userbase. Similar gamification strategies have not yet been implemented for mental health apps as of yet, but *Project Amiga* adds light gamification by including streaks and achievements, and allowing friends to view each other's achievements and streak values.

Streaks are a very simple method of encouraging continued use. Each day logged increases the streak count by 1, and each subsequent day thereafter continues to increment that streak value. The user's longest-ever streak is stored in **Firebase** as a part of the user profile, and is shared with friends. Furthermore, progressively longer streaks that exceed Fibonacci numbers (1, 2, 3, 5, 8, 13, etc.) will unlock achievements. The total length of the longest streak as well as the number of achievements will be visible to friends.

As a team, we struggled with an important ethical consideration: how would **Project Amiga** enable friends - sharing personal information- while maintaining the integrity of the users' very personal information; mental health logs are understandably very private, but the team still believed gamification would be an important method of user retention. Therefore, the published level of gamification with simple visualization of Streaks and Achievements was deemed appropriate to share with friends, and should the user desire to share more, they can do so by sharing their mobile devices as needed.

Profile Details

The team leveraged open-source tools to create Avatars for each user to select. The idea was to enable the user to select a cartoon depiction of themselves to use in the app while also maintaining a visible image for other users to connect to them when utilizing the Friends feature of the app.

Additionally, the app stores birthday and email address as personal information upon the creation of a new account. This is done to give a friendly "happy birthday" to the user (still in progress) as well as to enable the back-end authentication structure via **Firebase** using email/password authentication.

Testing and Feedback

The team used close friends, loved ones, classmates, peers, and other teammates to test out **Project Amiga**. These audiences ranged in age from 18 to 49, and included a span of demographics. Users liked **Project Amiga**'s clear and simple design, large text, "calming" colors, and user-friendly interface. Some of the feedback received was addressed live, during project execution in other sprints, or remain as to-do tasks for implementation as time permits for the team members.

Conclusion