UNIVERSITY OF LONDON

BSc Computer Science

CM3070 FINAL PROJECT
FINAL PROJECT REPORT

Task Manager Mobile Application

Name: Shanath S/O Rajendran
UoL ID: 210470552
GitHub: https://github.com/ShanTheShan/FYP

Contents

Introduction

The template chosen for this final project, is the Project Idea Title 1: Task Manager Mobile App from CM3050 Mobile Development.

The idea for this project comes from my own personal experience, where I have downloaded task manager applications in the past, but never consistently used them because the application was either not engaging enough, or it was aesthetically unappealing.

Why is this important?

We live in a fast-paced world, most of us reside in cities. City life is stressful. We have lots of responsibility and obligations to fulfil on a regular basis. We might have multiple places to be in a day. Thus, we need a way to track and manage our tasks. Coupled with the fact that we have too many distractions in the modern world, due to rapidly emerging technology, like social media and games, it can be hard to stay focused on our responsibilities.

The objectives for this final project are as follows:

- Define the scope of the project
- Conduct a literature review to support the development of the project
- Build a full stack task manager mobile application for Android
- Have my target demographic review the application, and ascertain if it outperforms what's currently out on the market.
- Publish the mobile application to the Google Play Store

The deliverables for this final project are as follows:

- An initial preliminary report showcasing the chosen project template and the path ahead.
- The source code for the mobile application.
- A final report detailing the entirety of the project, including the full development, application testing, user feedback compilation and Google Play Console upload process.
- A video demonstration showcasing the mobile application at work.

The following sections will comprise of literature reviews to support the need of having a time management and project management tool. Followed by an overview of the intended audience, technologies to be used, wireframes that will be tested by survey respondents, and a Gantt Chart outlining the path ahead in completing this final project, finishing off with a prototype implementation of the discussed features.

Introduction: 338 words

Literature Review

In today's fast-paced, digitally distracting world, effective time management and goal-setting are crucial for students and adults. This literature review informs the development of a task manager application by exploring three key areas: **Time Management Techniques**, **Goal Setting and Visualization** and **Comparative Analysis of Productivity Apps**.

Time Management

Nasrullah and Khan from the University of Peshawar and Gomal[1], conducted a study to evaluate the strengths and weaknesses of time management practices on student academic performance. Recognizing the importance of proper time management in today's competitive environment[2], the study focused on time management mechanisms and goal setting among students from Qurtuba University of Science and Technology, Pakistan. Surveys assessed students' short-term and long-term time management capabilities. The data analysis included descriptive statistics and correlation analysis to explore the relationship between time management practices and academic performance.

The results showed a positive correlation between time management and student's academic performance. It revealed that there is a connection when it comes to academic performance and stress reduction, when utilizing time management techniques such as immediate goal setting, to-do lists and time log usage through short range planning. However, the sample size chosen was rather small, only a total of 120 students, lacking diversity of participants from various regions. Therefore, care has to be taken as it may not be wise to generalize the findings to all student populations globally.

A study by Albulescu and colleagues from the University of Timisoara, Romania[3], explored the effectiveness of micro-breaks on performance. They conducted a meta-analysis spanning over 30 years, examining whether taking micro-breaks during tasks enhances performance without negatively affecting health.

Out of 4868 case studies drafted, 22 studies were finalized and narrowed down for the analysis. The participants chosen in the selected studies were healthy individuals which included both students and young adults. The measurements that were monitored in this analysis were vigor, fatigue or performance. Vigor being an individual's inclination to carry on with the task at hand, even when challenges present themselves, while fatigue relates to how tired the individual is. The performance measurements relate to whether the individual was able to perform his or her task with high degree of accuracy and precision. Tasks participants had to perform were classified into three categories. Creative, clerical and cognitive tasks. All 22 studies had a control group as a comparison, such as those with micro-break and those without.

The collective findings of the chosen studies concluded that micro-breaks do indeed improve the performance and vigor of the individual whilst reducing fatigue for clerical and creative tasks, though there was barely any improvement for cognitive intensive tasks. The data uncovered that those who took breaks that were longer than 10 minutes, generally had better performance.

However, there were several key factors that had to be considered for further study. The duration of the micro-break could not be decisively agreed upon by experts in the field. The tasks in which an individual partakes prior to the break, also mattered greatly, as well as the activity the individual partakes during their break. An improvement to this analysis could have been taking the findings of the conducted studies, and perform a new bespoke experiment, comparing the results of different intervals of rest period between control groups. I also uncovered that the countries of these studies were not mentioned, only that they were taken in English. Perhaps a comparative study could be made between individuals from different regions of the world, like Asia and Europe, to see if there were any noticeable differences in performance when micro-breaks were given. It shared a similar weakness from the first study reviewed, where diversity was lacking. But I do agree with the findings that micro-breaks are needed in order for an individual to remain productive, as based off the research above, an individual's performance is tied to how well he or she manages their time and workload.

This literature review analyzed two studies from 2015 and 2022, emphasizing the importance of time management today. The findings prove that the techniques of short-term time management through immediate goal setting, paired with micro-breaks, improves a person's ability to manage their time better, and be more productive. This supports the need for a task manager app incorporating the 52/17 productivity technique, which aids individuals in focusing and planning study or work schedules with timed work and breaks.

The Effectiveness of Goal Visualization

Cheema and Bagchi[4] explored how visualizing goals impacts an individual's sustained effort towards achieving objectives. Their study showed that intuitive, short-term goals are perceived as easier to reach, especially when broken into sub-tasks. Visualization was found to make goals appear nearer and increase effort and efficiency in pursuing them, highlighting the power of visualization and its effectiveness when goals are clearly defined, such as envisioning a finished painting being created from a blank canvas.

A total of five studies were conducted to assess the effects of goal visualization.

The first study took 68 Olympic swimmers and had them swim 30 laps, each of which was a 100m. The measurements used was to take the difference in swim times of the first 50m, when the swimmer was facing away from the finish line, and the last 50m, when the swimmer was able to see the finish line, hence, able to visualize the end goal. The results revealed that the time difference decreased as the swimmer neared the finish line, showing a positive influence on the swimmer when he was approaching the goal, despite the presence of fatigue.

The second study had 79 students sustain grip pressure for 130 seconds, split into two groups. One group viewed a horizontal progress bar, while the other saw a stopwatch at 30-second intervals. Results showed that the group with the stopwatch exerted significantly less pressure towards the end, whereas those with the progress bar maintained more uniform pressure. This suggests that visualizing goal proximity helps sustain effort.

The third study took 183 undergraduates and tasked them to save $750 for a fabricated vacation to Europe. There were two groups, one that could easily visualize their savings with a horizontal bar that was shaded depending on how much was already saved, 30% or 70%, whilst the other control group was only given textual representation of the savings amounts. The measurements used was a scale given to the participants to assess their commitment to reaching the required amount[5]. The results revealed those given the bar that made visualizing the amount left to save up, committed greater effort to saving than those that had a difficult time visualizing.

Just off these three studies, participants with goal visualization, like a progress bar, outperformed those without it, in endurance tests. These findings suggest that visualizing goals boosts effort and motivation, supporting the feature a project management tool with visual feedback, such as a progress bar, to track progress and remaining tasks.

However, the studies conducted above do leave room for question as to the factors that could contribute to how likely an individual is able to outperform another, given that both has access to easy visualization tools. For example, the undergraduates whose grip strength were tested, could house bias to the strength of the individual. A student with phenomenal grip strength and endurance could exert the most amount of willpower and willingness to complete the test, regardless of what goal visualization tool he was given. Thus, it is important to note an individual's characteristics and personality might be a contributing factor when to comes to accomplishing goals.

Comparison of Existing Applications

The following is a review of similar applications that have already been published on the Google Play Store. These competitors were selected for analysis due to their popularity for Android Devices.

Microsoft To Do[6] is a to-do list mobile application developed by Microsoft for Android and IOS devices(Figure A1). The analysis of the application are as follows:

1. Users must sign in or create an account to use the application, which may drive them to competitor apps if they don't wish to log in.

2. The app lacks a tutorial, making it hard for users to discover all features.

3. The application's aesthetic could be improved. The contrast between the black top bar and white main container, affects readability of the text, and contrast of colors is one of the key areas in GUI design[7].

4. Wang and colleagues[8] noted that students face significant digital distractions. A focus tool to help manage time and minimize these distractions, as discussed in the Time Management section, would be beneficial.

5. A study done by the U.S. Department of Labor[9] showed that 78% of the population, spent their days engaged in household activities while 44% of the population were engaged in work/work related activities. This indicates users have many tasks to accomplish. The application lacks a project management tool to meet this need.

Google Tasks[10] integrates seamlessly with other Google services like Gmail and Google Calendar, allowing users to create, view, and manage tasks across multiple platforms (Figure A2). The analysis of the application are as follows:

1. Google allows users to access basic to-do list features without signing in. Users can log in to sync tasks with Gmail and Google Calendar.

2. Has no tutorial, making it difficult for users to uncover all the available features.

3. Google has adopted the flat UI approach. This provides a clean and easy interface to navigate.

4. A tool that could aid students in focusing and managing their time, as discussed in the Time Management section, could prove fruitful.

5. No tool for users to undertake and manage large scale projects, and houses no way to visualize the progress being made.

6. Does not allow users to upload and attach images as part of their notes.

Tasks by Pocket Brilliance[11]. A review of this application reduces the bias of only analyzing task manager applications from already established technology giants(Figure A3).
The analysis of the application are as follows:

1. It uses minimal API such as calendar, syncs the application and its data across multiple devices. It does not require users to log in. However, it requires users to purchase the premium version of the application, in order to synchronize across devices.

2. This application does not collect or share user's data, a good practice to entice users to use their application, as data collection from corporations is rampant in today's world[12].

3. Has a poor tutorial that only activates once users start to interact with the application. It does not give the option to replay the tutorial in the settings page, which is extremely cluttered.

4. Adopted a flat UI approach.

5. No tool that could aid students in focusing and managing their time, as discussed in the Time Management section.

6. No tool for users to undertake and manage large scale projects, and houses no way to visualize the progress being made.

Key Takeaways

The findings from the analysis, highlighted in the "Carry Over" column, will be integrated into my application to enhance user experience and minimize inconveniences. Additionally, the literature review on time management and goal visualization supports using techniques of immediate goal setting, micro-breaks, and progress visualization via a progress bar, for the project management tool and focus timer tool.

Name of Application	Good Practices	Bad Practices	Carry Over
Microsoft To Do	- Allows customization via images	- Forcing users to sign in - Bad design principle - No tutorial - No focus tool - No project tool	- Include focus & project tool - Include a tutorial - Do not force users to sign in - Select proper design principles
Google Tasks	- Great UI design choice, flat principle	- No tutorial - No focus tool - No project tool - Can't attach images	- flat principle and minimalism design - Ability to attach images
Tasks by Pocket Brilliance	- Minimalize use of 3rd party API - No ads or data collection - Minimalist and flat UI approach	- No focus tool - No project tool - Overwhelming settings page	- Use 3rd party services only when necessary - Do not collect data or house advertisements - Adopt a minimalist approach

Literature Review: 2020 words

Project Design

Domain & Users

The domain of the project falls under productivity, and the mobile application being built in this project is intended for students and adults. I intend to develop, test, and deliver an Android application to the Google Play Store. The reason I will not be developing for IOS devices, and attempt to publish it into the Apple App Store, is due to the fact that I do not own any Apple devices, particularly an iPhone. Therefore, I am limited by my hardware.

Justification of Selected Features

The modern world is increasing in terms of responsibilities, and digital distractions. A task manager application that employs time management techniques, has shown to improve student's academic performance, whilst goal visualization, has shown to help adults such as athletes and managers, to exert more effort to reach their end goal, based on the current literature.

Therefore, to address time management, the task manager application will house a timer, employing the 52/17 technique[13]. An individual will time themselves to work for 52 minutes and rest for 17 minutes. In regards to goal setting and visualization, the application will house a project management tool that will allow users to undertake large sets of tasks, whilst being able to visualize their progress, via a horizontal progress bar, providing motivation to see the project through to its end, as discussed in the literature review. The application must also house the ability to use images as notes. Analysis of existing application has also shown tutorials, or the lack thereof, emphasizing the need for a proper tutorial, guiding users to all the features available to them.

Selected Features based off the Literature Review
Project Management Tool - Allow creation, management and prioritizing of tasks, subtasks and attachments, using a dynamic relational database and APIs, as well as relevant native and external libraries, as well as providing good visual feedback through the use of animations
Focus Tool - timer to allow users to manage their time spent working and resting that provides good visual feedback through the use of animations
Checklist - to-do list with calendar integration API that will dynamically update based off a relational database
Notes - note section that gives users the ability to store any text and images through accessing the device's camera and gallery, and will dynamically update based off a relational database
Tutorial - Guide users on all the features available to them via an overlay/modal

Technology & Methodology

The chosen technology to build this mobile application is React Native[14] and Expo[15]. React Native is a JavaScript framework developed by Meta[16], with the aim of allowing developers to use a single language, in this case JavaScript, to develop applications for a particular platform or operating system using the platform's own tools and languages.

Expo is a framework and platform built around React Native that simplifies mobile app development, building, and deployment. It provides a managed workflow with a suite of pre-configured libraries and APIs, enabling me to start building apps quickly without dealing with native code configurations. The Expo CLI and Expo Go app streamlines development, allowing for instant previewing on physical devices. Furthermore, the Expo EAS cloud service will be used to build and deploy the application. React Native based mobile application, can also run on IOS devices, giving this project the room for upgradability to Apple devices, should one come into my possession for testing, in the future. Testing will be done using the Jest Framework[17].

The key APIs and libraries that were used to build this application are as follows:

- expo
- SQLite
- react
- react-navigation
- react-native
- react-native-async-storage
- react-native-reanimated
- react-native gesture handler
- react-native-tableview-simple
- jest

For a full list of libraries used, refer to the package.json file in the source code.

The software development for this project utilized Sprints[18] and User-Centered Design (UCD)[19], both methodologies stemming from Agile software development. I selected these approaches for their effectiveness in addressing the project's requirements and final product characteristics. Sprints are repeatable stages within a software development cycle, while UCD focuses on the needs, preferences, and limitations of end-users through active feedback and iterative testing.

Given that the task manager mobile application prioritizes user experience over complex technical features, UCD enables a better understanding of users and allows for iterative improvements based on their feedback. This approach ensures the application consistently meets users' needs and prevents biases from my own assumptions about the application's delivery.

Design Structure

The initial user flow diagram (Figure A4) was created to dictate how users are supposed to navigate the application and its core features. Subsequently, the relational database schema (Figure A5) was designed containing four tables. `Projects`, which houses the id of the project, its name and progress. `ProjectDetails`, which is referenced to the `Projects` table via the foreign key `projectId`, that allows the querying of all tasks of that particular project. The `Notes` table, for the creation of notes and their attached images, and lastly, `Todos`, that had columns to know which date it belongs to in the calendar, the todo itself, and whether it was completed or not.

React Navigation[20] is a core library for modern mobile applications, enabling the creation of multiple components and pages with controlled navigation. It uses a stack data structure, following the Last In, First Out (LIFO) principle, where the most recent item is removed first. The user navigates through the bottom tab, with each tab containing its own page, and the pages are navigated via a stack (Figure A6).

The project tab tool, encompasses not only the React Navigation library but also APIs from Expo as mentioned in the Technology & Methodology section, that give access to the Android Software Development Kit(SDK).

The project tool feature will include an overview page showing project titles and their progress based on completed tasks. Tapping a project navigates to a details page where users can create and view tasks, subtasks, and notes. The SQLite API will be utilized to perform CRUD operations to manage storing and displaying project information, tasks, subtasks, notes, and images on a local database. Because this is done locally, users will not require an internet connection (Figure A7).

The task/note creation page of the project tool will contain a text input for users to enter the task, as well as multiple additional cells below to provide more useful functionalities such as deadline for the task, subtasks for the main task, and reminder to complete the task by the due date, and camera option to attach images.

To set a deadline, a Calendar library will be used for date selection, which is then stored in the local SQLite database. The Expo Notifications API will send reminders if set by the user. For image capture or import, the Expo Camera and ImagePicker APIs will access the device's camera and gallery, storing the images in the SQLite database.

The focus timer tool will feature a circular progress bar to display the remaining time, emphasizing goal visualization and visual feedback, as supported by the literature review. Below the timer, start and stop buttons will initiate the countdown. To prevent the device from sleeping and ensure continuous visual feedback, the KeepAwake API will be used, allowing the user to stay motivated by seeing how much time has elapsed and how much remains (Figure A8).

The literature review on goal visualization and time management, along with analysis of existing apps, justifies the project and focus tool. Adults have a myriad of tasks to accomplish on a daily basis, as cited from the US Department of Labor study. As technological integration grows, students seek better tools for academic tasks, despite the consensus that the tools available to them

could be improved upon[21]. Thus, the project management tool and its features cater and address the needs of my target audience of students and adults, providing a means to manage tasks, projects, and productivity techniques such as time management.

Low & High-Fidelity Wireframes

Low and high-fidelity wireframes were developed for use in the first round of user-based testing in the Development phase, as well as to gather initial feedback on the design and prototype features (Figure A9, A10). Certain design choices were backed up by the literature review such as the horizontal progress bar at the top of the `project details screen`, as well as the circular progress bar in the Focus Tool screen, to promote goal visualization.

Usability is an important aspect in the design of this application. To ensure a smooth experience, especially visually, the UI must possess simplicity and clarity such that the interface is kept as simple as possible, whilst avoiding clutter. For research shows that majority of mobile users prefer a minimalist approach in regards to the interface[22]. Examples of this design choice and UI are the bottom navigation tabs, buttons and page title, so that users know exactly where they are and what they can do, in terms of pressable elements.

I gathered user feedback on the wireframes through Google Surveys. Seven students and staff from the Singapore Institute of Management (SIM) participated, providing valuable insights for our target audience of students and adults (aged 19-40).

The survey was conducted on site and in person, and no personal data was collected, for privacy purposes. The overall sentiment was neutral to positive (Figure A11). This small survey test has shown that the project and the intended application that is to be developed, is on the right track, with no negative feedback as of yet, on any crucial features.

Test Plan

An Excel worksheet was created to document the test plans before development began. The test cases were designed using a black box testing approach, which is why they were written before any code was developed. This ensured there was no bias or prior knowledge of the application's internal workings, at least before the start of the second sprint, as the application and its features needed iteration and I would become familiar with its internal structure. The test cases were to be executed externally by an anonymous third party to ensure that the app's core features of the project tool, focus tool, notes, to-dos, and settings options, functioned as intended.

The test plan housed six columns to document the test cases:

1. Test Description: Explaining the purpose of the test.
2. Execution Steps: Detailing how the user is supposed to operate the feature.
3. Expected Result: What the outcome should be.
4. Actual Result: The observed outcome.
5. Pass/Fail: Indicating whether the test passed or failed.
6. Response Time: Measuring how long the system took to respond.

Application Assets

The necessary icons and vector images were sourced free of charge from Expo Icons and flaticons[23]. Refer to Figure A12 for the initial application icon and splash screen.

Gantt Chart

A Gantt chart was created to outline the roadmap for this final project (Figure A13).

The key milestones are as follows:

- Completion of Design Phase – June 7th
- Submission of Preliminary Report – June 11th
- Completion of First Sprint – July 13th
- Submission of Draft Report – July 22nd
- Completion of Second Sprint & Development Phase – August 14th
- Submission of Final Project, in its entirety and end of Delivery Phase – September 9th

Design: 1853 words

Implementation

Environment Setup

During the development phase, a local Expo project was created on my machine. Initially, Expo Go was used to view and develop features. Later, the application was built natively for Android using the Expo CLI, running it on my physical device for a more accurate performance representation and debugging. Finally, the project was moved to Expo EAS to build the production version and prepare it for the Google Play Store, as Expo EAS handles the key signing on behalf of the user. The project directory housed all the relevant folders and sub-folders, to modularize the codes (Figure B1). Android Studio was utilized to emulate different Android devices, to ensure the application's layout was responsive.

First Sprint

The first task was to create the React Navigation bottom tab, which would house all the necessary screens for the application. A custom navigation component, situated inside the navigation folder, had to be coded using React components, in order to house the necessary screens in each stacks, as well as customizing the styling and settings icon that would be displayed on the top left of the screen(Figure B2). This was wrapped in a React Context[24] to allow states to passed down the application hierarchy, allowing the styling of the pages to change based on the selected theme(Figure B3). In order to set the header styling and color to match the theme selected by the user, custom React hooks were created to change the header background and text color, housed appropriately in the hooks folder(Figure B4). Next, the front-end was coded for all the respective screens within each tab. Once the responsive UI was tested via emulators, the focus shifted to implementing the back end of the application.

The SQLite database was initialized and the tables seen in the schema was made with SQL syntax `CREATE TABLE` and exported as a component, enabling the relevant screens to access the tables and perform necessary CRUD operations (Figure B5). To implement the project management tool's visual feedback via a horizontal progress bar, as discussed in the Literature Review and Justification of Selected Features, the following logic was coded. Several React states were created to track the total number of tasks created by the user, the number of tasks left, and a counter for completed tasks. When a user marked a task as completed, that task would be removed from the database, and the progress bar would increment by subtracting 1 (representing 100% completion) from the current number of tasks left, divided by the total number of tasks created by the user. This ensures that the progress value dynamically adjusts as users add or complete tasks throughout the project's lifecycle. The progress value is also stored in the 'progress' column in the 'Projects' table of the relational database. This allows the user to see the project's progress from the home screen and have it update dynamically.

Furthermore, in the task creation screen of the project tool, to enable the user to add a reminder to complete a certain task in a project, push notifications had to be enabled. To register for push notifications on Android, via the Expo workflow, the project had to be first initialized via EAS, which was done in the Environment Setup. At the time of writing, Expo required Firebase Cloud

Messaging (FCM) V1 in order to enable push notifications on Android. Thus, a project was created on Firebase[25], a backend cloud computing services by Google. Through the Firebase project, the credential key needed to register the application with Expo, as well as the google services file was generated and configured into the android native module, as well as the Expo Project account credentials settings for the mobile application.

The current state of the application at the end of the first sprint can be seen in Figure B6. At the end of the sprint, user testing was conducted on the application, and feedback was gathered. Refer to Appendix E for a complete documentation of user feedback.

Second Sprint

Based off user feedback gathered from the first sprint, the application icon was first redesigned (Figure B7). Then, a revised user flow (Figure B8) was drafted to enhance the app's navigation by introducing a new screen that allows for adding reminder notifications to to-dos and creating project comments, addressing key features requested by users. New low and high-fidelity wireframes (Figure B9, B10, B11) were then created to incorporate the new elements, and finally, the relational schema was modified to reflect these changes and accommodate the new features (Figure B12).

Firstly, users expressed the desire to view project tasks that had already been completed, as well as individual project comments. The current design did not support this, as tasks were immediately deleted upon completion, and there was no option to add comments. To address this, the updated wireframe for the project details screen now includes accordions for viewing uncompleted tasks, completed tasks, and project comments. These three elements are touchable sections labeled "Uncompleted," "Completed," or "Comments," which toggle the view of the accordion, displaying the relevant data inside. This new design meets users' needs by allowing them to view project tasks in their entirety. The accordion element is scrollable to accommodate any length of tasks the user creates. To implement this feature, the React Native Reanimated library was used. The library had an accordion example that fitted my needs, but only after it was modified and refactored [26].

Referring to Figure B13, the custom accordion tab was split into four separate components, `AccordionItem`, `Item`, `Parent` and `AccordionTouchable`, held inside the file `customAccordion.js`. This was done for two reasons. One, so that I could customize the accordion to house the React Native components and properties I needed it to, and two, so that it was modularized and could be easily unit tested. The `AccordionItem` was the component responsible for creating smooth opening and closing transitions of the accordion using Reanimated hooks. The `Item` component was what was going to be rendered inside them, with customs properties such as content and current theme to render the content and styling appropriately. The `Parent` component is the parent of the both the `AccordionItem` and `Item` which houses them inside a view component. The `AccordionTouchable` was a touchable that on press, would toggle the visibility of the Accordion and animate a change in the background color via the useAnimatedStyle hook.

However, upon building the application, I had three problems. One, the scrollView that was being rendered inside as an item of the accordion, couldn't be scrolled. To solve this issue, I had to manually pass in a property into the `Item` component that sets the scroll height, and round the layout height view of the animated `AccordionComponent` as it was rendering the height of the entire screen. Secondly, when one accordion was pressed, the two other accordions opened as well. To solve this, three individual states were created and passed as properties into the `Parent` component. Lastly, declaring the component three times created rendering issues that was solved by passing a unique key as a property to the `Parent` component. This is how one of the biggest UI element changes was implemented. The Todo screen was redesigned, as delving into the library of the Calendar API,` react-native-calendars`, showed that the intended approach to redesign the Calendar to make it Expandable via their `ExpandableCalendar` component could not be done, as there was a significant performance drop in terms of frames per second. The GitHub page of the library, showed that many developers had opened issues regarding the performance of the `ExpandableCalendar`, which were not fixed or addressed. Thus, I had to redesign the wireframe, that would use a touchable that had a calendar icon at the top of the screen instead, to close or expand the Calendar, giving users more space to see their tasks, as per their feedback.

The relational schema saw the addition of two new tables, `ProjectSubTasks` and `ProjectComments`, as well as modification to existing tables, `ProjectDetails` and `Todos`. The project comments created would be queried and rendered in the `Comments` accordion mentioned above. The `ProjectDetails` was modified because user feedback stated users wanted the ability to mark individual subtasks as completed, a feature not present in the initial design of the application. An oversight on my part. The `ProjectSubTasks` would relate to the `ProjectDetails` table through the foreign key `parent_task_id`. This way, it ensured that every unique task created in a project, would have its set of sub tasks, that can be checked if it's completed via the table column `completed`. Referring to Figure B14, an SQL left join query was utilized to fetch each task of a project, along with all its sub tasks, if they exist. However, an issue was that in order to render the data into the accordions, a singular array of objects was needed. The SQL query returned multiple arrays and objects for a task that had multiple sub tasks. Thus, the function `reformat` was made, to reduce the multiple arrays into one singular array of objects for each task, containing all its sub tasks. This was done via the JavaScript Array Reduce method.

There were many bug fixes and minor changes made to the application as well. A changelog was utilized to document the changes made. Refer to Appendix E for the full list of changes made to the application.

Google Play Store

A developer account was created on Google Play Console and I named the application, TaskTracker 9000, under the developer alias, GammaDigital (Figure C1). There are five stages to the publishing an application to Google Play Store. Setting up the app in the store listing, the internal testing, closed testing, open testing and production access.

The store listing set up consists of giving your application a name, description, as well as a feature graphic and screenshots(Figure C2). Several questionnaires had to be filled as well, such as the target audience and if application contains ads, which it does not. An important step was setting up the privacy policy of the application, which can be viewed here.

Internal testing required me to test the application in beta, which was completed and feedback gathered at the end of the First Sprint. Subsequently, closed testing required me to test my application for 14 consecutive days with 20 active testers. Google recently implemented this new feature that, according to feedback from developers who have experienced the process, has made uploading applications to the Play Store more difficult and less accessible. I experienced this myself as I successfully tested my application with 20 testers, comprising of friends, acquittances, classmates and family members for 14 consecutive days and gained the ability to request for production access. Unfortunately, Google denied my request for production access, stating that my applications needed further closed testing for an additional 14 days (Figure C3). Google did not explicitly state why my testing did not meet their requirements, even though I completed the 14 consecutive days of close testing with 20 testers as per their policy. My testers provided feedback to improve the application, fixing of minor bugs and as well as verifying the responsiveness and the fact that the app did not crash once. I suspect that my request was denied because my testers may have uninstalled the application during the 14 days, as they were simply honoring my personal request.

At the time of writing, the additional 14 more days required for closed testing to unlock production access meant the application won't be published on the Play Store before the course deadline. Although the project timeline was aligned with the Gantt chart, this unexpected setback of 24 total days of closed testing from Google has disrupted the schedule.

The application is still in a fully production ready state, and the APK file can be downloaded and installed from the GitHub source code under the master branch, or through this Google drive link.

Implementation: 1983 words

Evaluation

Unit Testing

As mentioned in Technology & Methodology, Jest was used to perform unit testing. Unit testing here was considered white box testing, as the internal codes were known and understood by the tester, me. The tests were preconfigured to run on Android only, to align with the objectives and limitations of the project. Test cases were written for modularized code within the `components`, `constants`, `context`, and `hooks` folders. The test files for each of these modules were organized in their respective `__tests__` folder, following common testing practices. The components were tested to ensure that their logic was performing as expected, such as time conversion, and that they were rendering correctly without the UI changing unexpectedly, through the use of snapshots. Certain unit tests failed in the first sprint such as the custom modals rendering with the wrong styling, but these issues were fixed and by the second sprint, all test cases passed as the application moved to production (Figure D1).

System Testing

To evaluate the application developed in the first sprint, a system wide component testing was done using the Test Plan created in the Project Design. For both sprints, the tester that tested the application via a local device APK installation, were two separate students from SIM above the age of 18. The anonymous individuals were used as testers as they had no prior knowledge of the internal workings of the application, and were part of the domain of users the application was intended for, providing an unbiased ground for testing. The first system-wide test at the end of the initial sprint resulted in several failed test cases within the Project Management Tool, the Notes creation, and the Settings. Failed test cases include the date and time still being selected when the user cancelled the date time picker for selecting a project deadline and reminder in the `Task Creation Screen`, and the light theme reverting back to dark upon application restart (Figure D2), to name a few. During the second sprint, a revised system test plan was created to cover all the newly added features, as well as test cases for application behavior and navigation that were missed in the first system-wide test. The first and second system-wide tests were organized into version one and version two, separated by tabs at the bottom of the Excel sheet.

Final Results

To evaluate the success of the project and the application, and to determine if it has achieved the objective of outperforming current market offerings, an experiment was conducted with the target audience.

A research conducted by healthcare professionals revealed key methodologies for designing and conducting questionnaires[27], and found that face to face interview provides higher merits like clarification of questions, thus the evaluation plan was carried out on campus grounds of the Singapore Institute of Management, comprising of students and working adults above the age of 18. Once consent was granted, participants were split into three groups. One used Microsoft's application as seen in the literature review, the other Google, and the third, my own application. They were given a fabricated project of renovating their home, and have to lists down all the given tasks and notes necessary to accomplish the renovation.

Once they had inputted all the tasks and marked them as completed, I had them fill out the User Experience Questionnaire(UEQ)[28], to see which application had the best assessment score. To remove any bias, the third group of respondents did not know that the application was developed by me, until the questionnaire was completed. No personal data such as names, age or email addresses were collected, ensuring no ethical breach. The assessment scores were manually collated and keyed into a data analysis tool in the form of an Excel spreadsheet, provided on the UEQ source page. The sheet would automatically calculate the mean and standard deviation of all items and their respective scores, that was provided by the survey respondents and display the results in the form of a bar chart. The bar chart consists of 5 labels. Attractiveness, perspicuity, efficiency, dependability, stimulation and novelty. Figure D3, D4 and D5 are the overall results for TaskTracker 9000, Google Tasks, and Microsoft To Do respectively. The results show that users found my application, TaskTracker 9000 to be the best, followed by Google, then Microsoft in last place. Both Microsoft and Google scored extremely poorly in terms of Novelty and Stimulation, in contrasts to my application. However, Google's perspicuity was slightly better, but only by an extremely small margin. For reference, Figure D6 was the worst assessment survey result for TaskTracker 9000.

In summary, evaluation of the project as a whole are as follows:

- Define the scope of the project: The project scope for the application was designed appropriately, making it achievable within the given deadline.

- Conduct a literature review to support the development of the project: The in-depth literature review conducted gave supportive evidence for the use of techniques such as goal visualization via the progress bar of the Project Management Tool, and the use of micro breaks for time management, in the Focus Tool. The analysis of competitor application aided in the design choices made during the design of the application.

- Build a full stack task manager mobile application for Android: A full stack task manager mobile application was built and bundled for Android devices. It tracks projects, visualizes progress, and includes a focus timer tool. Users can log tasks and notes with image attachments, subtasks, and set reminder notifications and deadlines. The app also integrates a calendar API for marking tasks as completed. The UI is easy to use, with a replayable tutorial to guide them through the features. All information is stored locally on the device, without the need for internet connection, making offline usage possible. Objective successfully achieved. However, despite developing the UI from scratch using core React Native Components to display my skills learned from the Mobile Development module, perhaps popular design libraries could have been used instead, improving the user experience.

- Have my target demographic review the application, and ascertain if it outperforms what's currently out on the market: The evaluation plans carried out achieved this objective. The experiment was conducted with the actual target audience, ensuring that the scores were relevant and directly applicable to the intended users of the application and it was grounded in methodologies recommended by professionals for conducting questionnaires, including the use of the UEQ, adding credibility to my evaluation plan. However, the single interaction on the day the evaluation plan was carried out, might not fully capture the long-term user experience, including aspects like user retention.

- Publish the mobile application to the Google Play Store: This was the only objective that couldn't be achieved. The new policy implemented by Google has revealed that requiring the help of professional services like paid testing, would be required in order to satisfy the need for 14 consecutive days of active testing from 20 users, which as a solo developer, I found difficult to accomplish solely through personal requests from friends and family, as 20 is a rather large amount of people to source and there are factors that I cannot control or impose upon them to fulfil this personal request.

Evaluation: 1202 words

Conclusion

This final project has required me to apply the full breadth of skills and knowledge I have acquired throughout my Computer Science degree. I utilized my testing and debugging skills to ensure the application functions correctly and efficiently. Wireframing and iterative development were essential in designing user-friendly interfaces and continually refining the application based on user feedback. Understanding and implementing the software development lifecycle allowed me to manage the project's progress effectively from conception to deployment. Additionally, I gained hands-on experience in development and production builds through the use of version control, ensuring the application is robust and ready for real-world use. My experience with Google Play Console has given me invaluable insights when it comes to pushing an application to the Play Store, which will aid me in the future in my career, should clients have this objective in mind. This comprehensive application of my education has solidified my expertise and prepared me for future challenges in software development.

Further work that could be explored is gamifying the focus tool. If the user is able to successfully stay productive for the set duration, he or she is rewarded with a token. A collection or gallery of tokens could be created and rendered. However, should the person exit the application or reset the timer, the token would be lost or destroyed. After some research, React Three Fiber[29] seemed capable of achieving this further work. It’s a library that supports 3D animation and graphical interface with React Native. Performance comes in mind when considering this further work, and optimization would then take precedence over user interaction. Furthermore, the 3D models would have to be manually created through a computer graphics software tool such as Blender[30]. If the application's performance is subpar, switching to Unity[31] would be necessary, which would require a complete overhaul of the app. In terms of scalability, the application could migrate to Supabase[32]. It’s an open source database infrastructure, that can support relational databases as well authentication. However, this should only be considered, if the production application scales to the point where a large number of users demand for features such as device syncing, allowing tasks and notes to be transferred to one personal device to another through an account, which is currently not possible, with the current offline local storage system.

Conclusion: 385 words

Total word count of all 6 sections: 7781

(excluding cover page, table of contents, references, and appendices)

References

[1] Shazia Nasrullah and Muhammad Saqib Khan. 2015. Retrieved August 14, 2024 from https://www.researchgate.net/publication/313768789_The_Impact_of_Time_Management_on_the_Students'_Academic_Achievements

[2] S.N. Razali, M.S. Rusiman, W.S. Gan, and N. Arbin. 2018. The impact of Time Management on students' academic achievement. Journal of Physics: Conference Series 995 (April 2018), 012042. DOI:http://dx.doi.org/10.1088/1742-6596/995/1/012042

[3] Patricia Albulescu, Irina Macsinga, Andrei Rusu, Coralia Sulea, Alexandra Bodnaru, and Bogdan Tudor Tulbure. 2022. "Give me a break!" A systematic review and meta-analysis on the efficacy of micro-breaks for increasing well-being and performance. PLOS ONE 17, 8 (August 2022). DOI:http://dx.doi.org/10.1371/journal.pone.0272460

[4] Amar Cheema and Rajesh Bagchi. 2011. The effect of goal visualization on goal pursuit: Implications for consumers and managers. Journal of Marketing 75, 2 (March 2011), 109–123. DOI:http://dx.doi.org/10.1509/jm.75.2.109

[5] Patrick M. Wright and K. Michele Kacmar. 1994. Goal specificity as a determinant of goal commitment and goal change. Organizational Behavior and Human Decision Processes 59, 2 (August 1994), 242–260. DOI:http://dx.doi.org/10.1006/obhd.1994.1059

[6] Microsoft. To do list and task management app: Microsoft to do. Retrieved August 14, 2024 from https://www.microsoft.com/en-sg/microsoft-365/microsoft-to-do-list-app

[7] Figma. 5 key UI design principles- and how to use them. Retrieved August 14, 2024 from https://www.figma.com/resource-library/ui-design-principles/

[8] Chih-Hsuan Wang, Jill D. Salisbury-Glennon, Yan Dai, Sangah Lee, and Jianwei Dong. 2022. Empowering college students to decrease digital distraction through the use of self-regulated learning strategies. Contemporary Educational Technology 14, 4 (September 2022). DOI:http://dx.doi.org/10.30935/cedtech/12456

[9] Anon. American Time Use Survey – 2023 Results . Retrieved August 14, 2024 from https://www.bls.gov/news.release/pdf/atus.pdf

[10] Google. Google Tasks. Retrieved August 14, 2024 from https://play.google.com/store/apps/details?id=com.google.android.apps.tasks&hl=en_SG&gl=US&pli=1

[11] Pocket Brilliance Limited. Tasks: To do list & planner – apps on google play. Retrieved August 14, 2024e from https://play.google.com/store/apps/details?id=com.tasks.android&hl=en_SG&gl=US

[12] Samuel Chapman. 2024. Big-Tech Data Collection: Protect Your Info 2024. (March 2024). Retrieved August 14, 2024 from https://www.privacyjournal.net/big-tech-data-collection/

[13] Lark Editorial Team. 2023. Unveiling the 52-17 rule: A guide to boosting productivity. (December 2023). Retrieved August 14, 2024 from https://www.larksuite.com/en_us/topics/productivity-glossary/52-17-rule#what-is-the-52-17-rule-in-the-context-of-productivity?

[14] React Native ed. React native. Retrieved August 14, 2024 from https://reactnative.dev/

[15] Expo.Retrieved August 14, 2024 from https://expo.dev/

[16] Meta. Social Metaverse Company. Retrieved August 14, 2024 from https://about.meta.com/

[17] Jest.Retrieved August 14, 2024 from https://jestjs.io/

[18] Simplilearn. 2024. Agile Sprint in software development: Definition, process, & roles. (August 2024). Retrieved August 14, 2024 from https://www.simplilearn.com/agile-sprint-article

[19] Ekaterina Novoseltseva.Retrieved August 14, 2024 from https://usabilitygeek.com/user-centered-design-introduction/

[20] React Navigation.Retrieved August 14, 2024 from https://reactnavigation.org/

[21] Eden Dahlstrom and Jacqueline Bichsel. 2014. ECAR Study of Undergraduate Students and Information Technology, 2014 (October 2014). DOI:http://dx.doi.org/http://dx.doi.org/10.13140/RG.2.1.3030.7040

[22] Mudita Sandesara et al. 2022. Design and experience of mobile applications: A pilot survey. Mathematics 10, 14 (July 2022), 2380. DOI:http://dx.doi.org/10.3390/math10142380

[23] flaticon.Retrieved August 14, 2024 from https://www.flaticon.com/

[24] React. CreateContext. Retrieved August 14, 2024 from https://react.dev/reference/react/createContext

[25] Google.Retrieved August 14, 2024 from https://firebase.google.com/

[26] React Native Reanimated. 2024. Accordion: React native reanimated. (August 2024). Retrieved August 17, 2024 from https://docs.swmansion.com/react-native-reanimated/examples/accordion

[27] S. Roopa and MS Rani. 2012. Questionnaire designing for a survey. The Journal of Indian Orthodontic Society 46 (October 2012), 273–277. DOI:http://dx.doi.org/10.5005/jp-journals-10021-1104

[28] Andreas Hinderks, Jörg Thomaschewski, and Martin Schrepp. User experience questionnaire. Retrieved August 14, 2024 from https://www.ueq-online.org/

[29] React Three Fiber. Introduction. Retrieved August 26, 2024 from https://r3f.docs.pmnd.rs/getting-started/introduction

[30] Blender Foundation. About Blender. Retrieved August 26, 2024 from https://www.blender.org/about/

[31] Unity. Unity real-time development platform: 3D, 2D, VR & AR engine. Retrieved August 26, 2024 from https://unity.com/

[32] Supabase. The Open Source Firebase Alternative. Retrieved August 28, 2024 from https://supabase.com/

Appendices

Appendix A

Appendix A consists of images related to the initial design of the project.

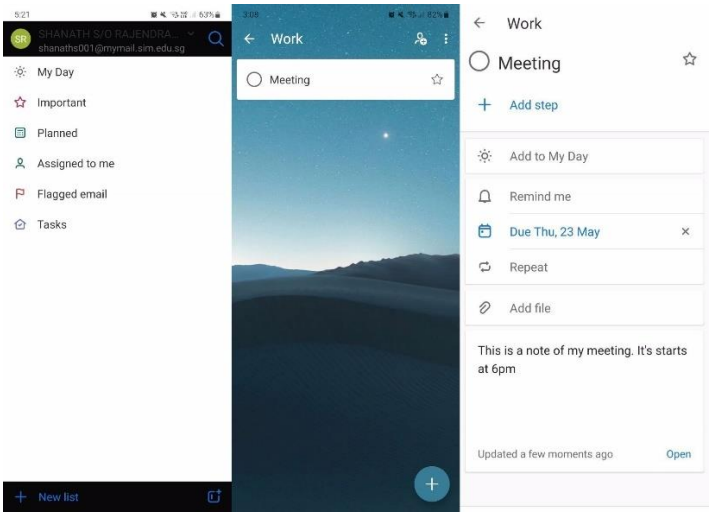


Figure A1Microsoft To Do

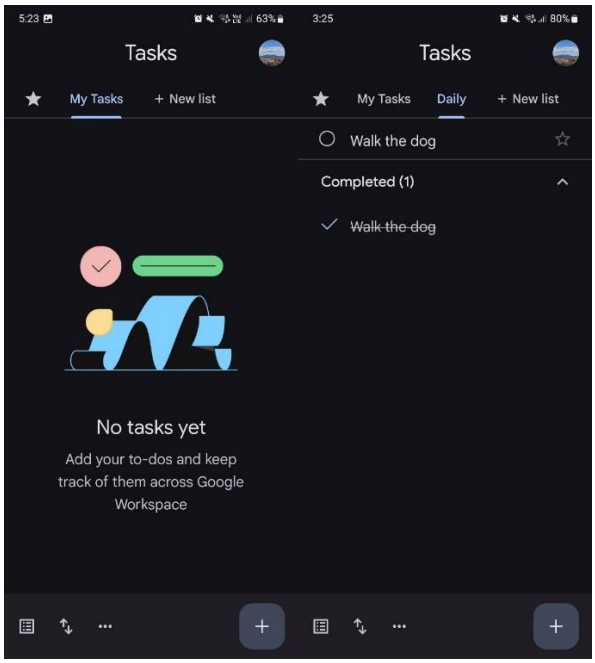


Figure A2 Google Tasks

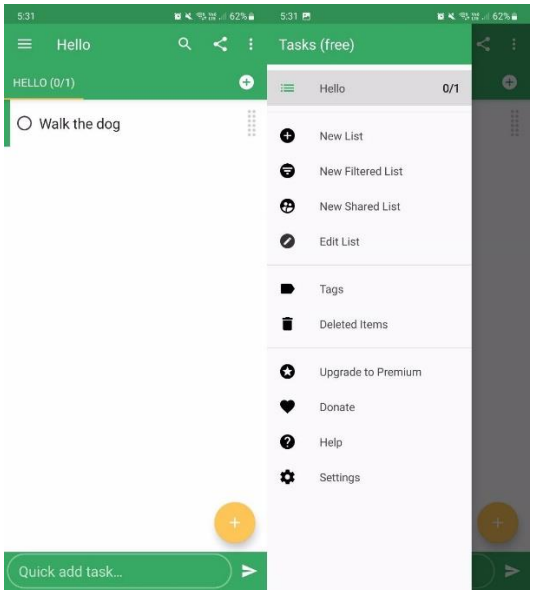


Figure A3 Tasks by Pocket Brilliance

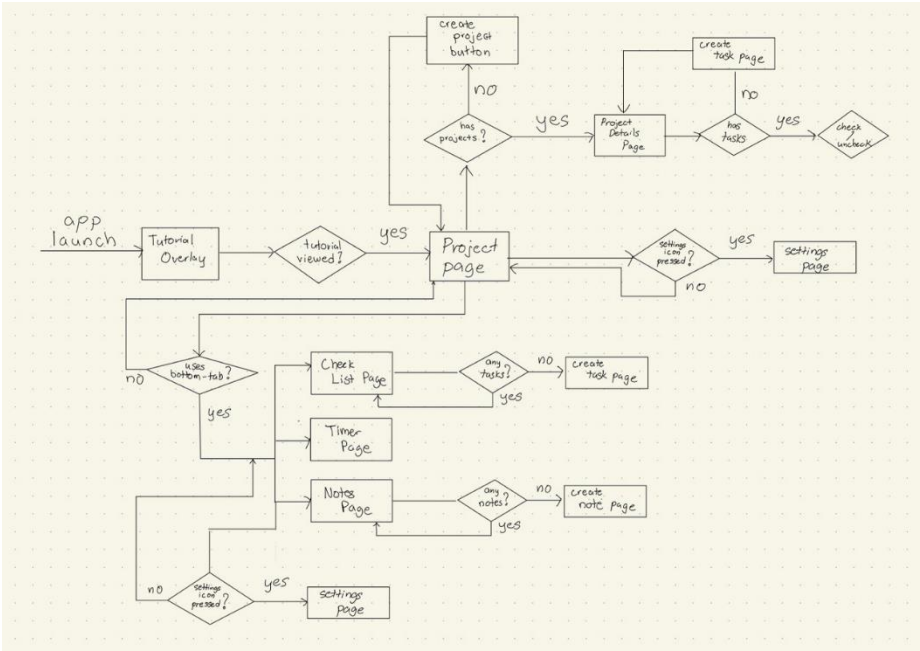


Figure A4 User Flow Diagram

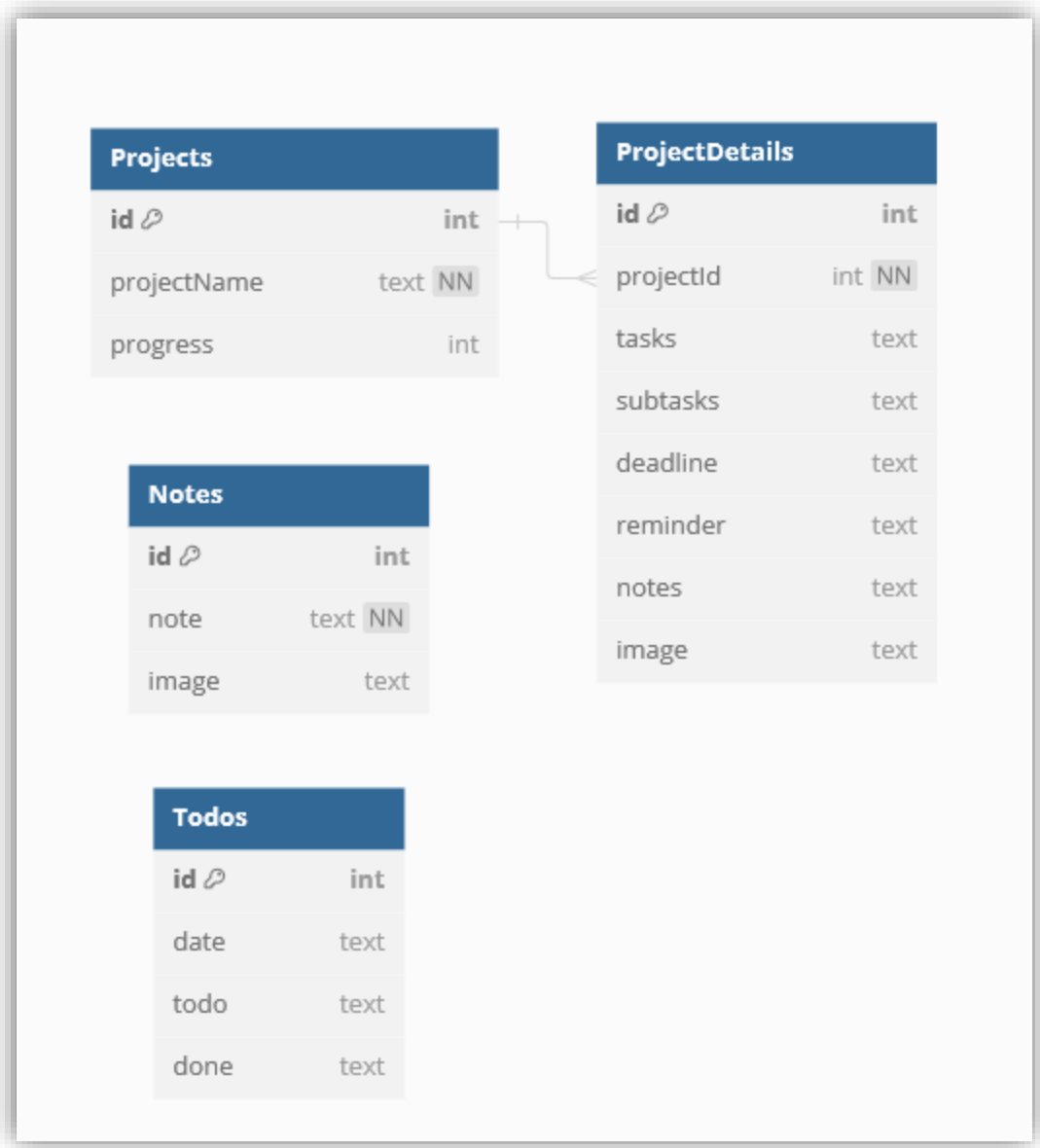


Figure A5 Relational Database Schema

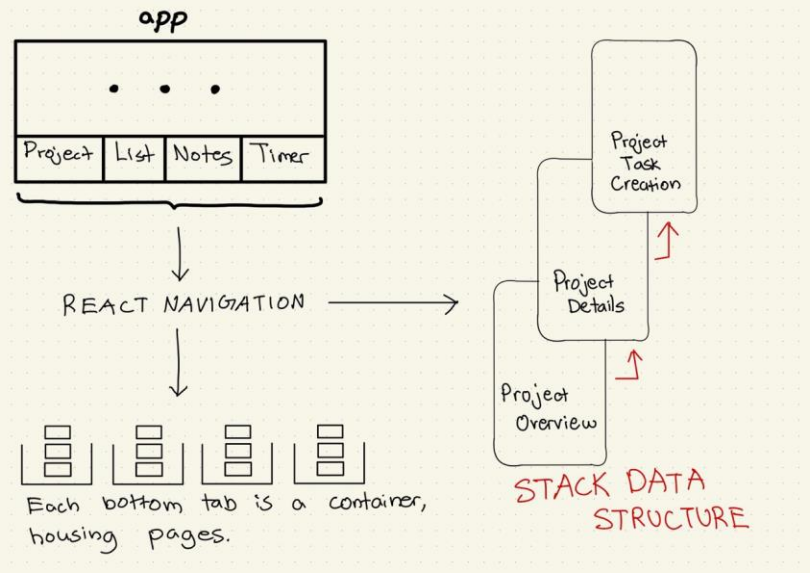


Figure A6 React Navigation Overview

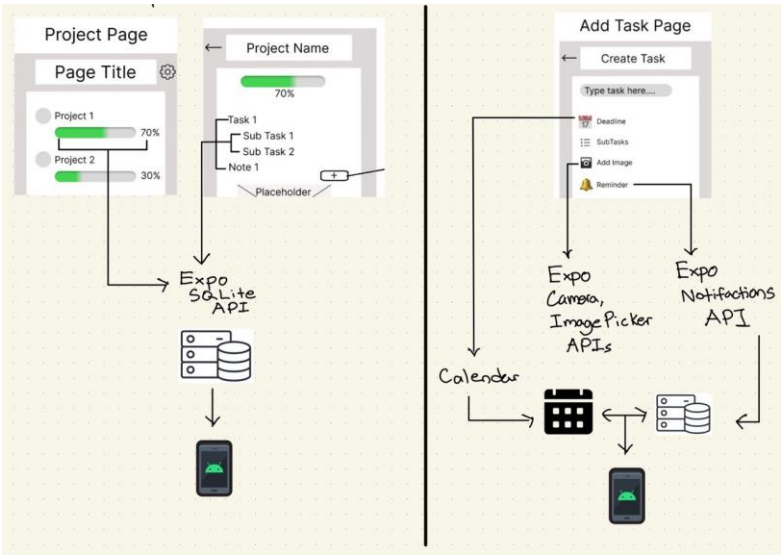


Figure A7 Project Tab Structure

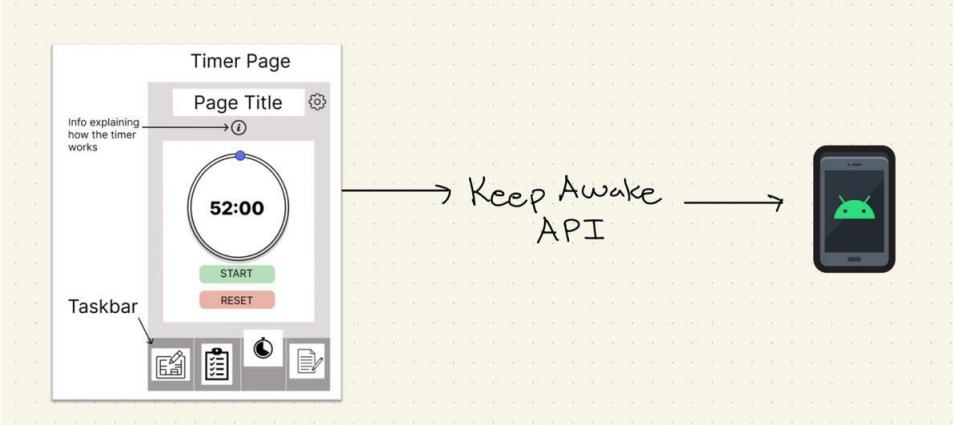


Figure A8 Focus Tool Overview

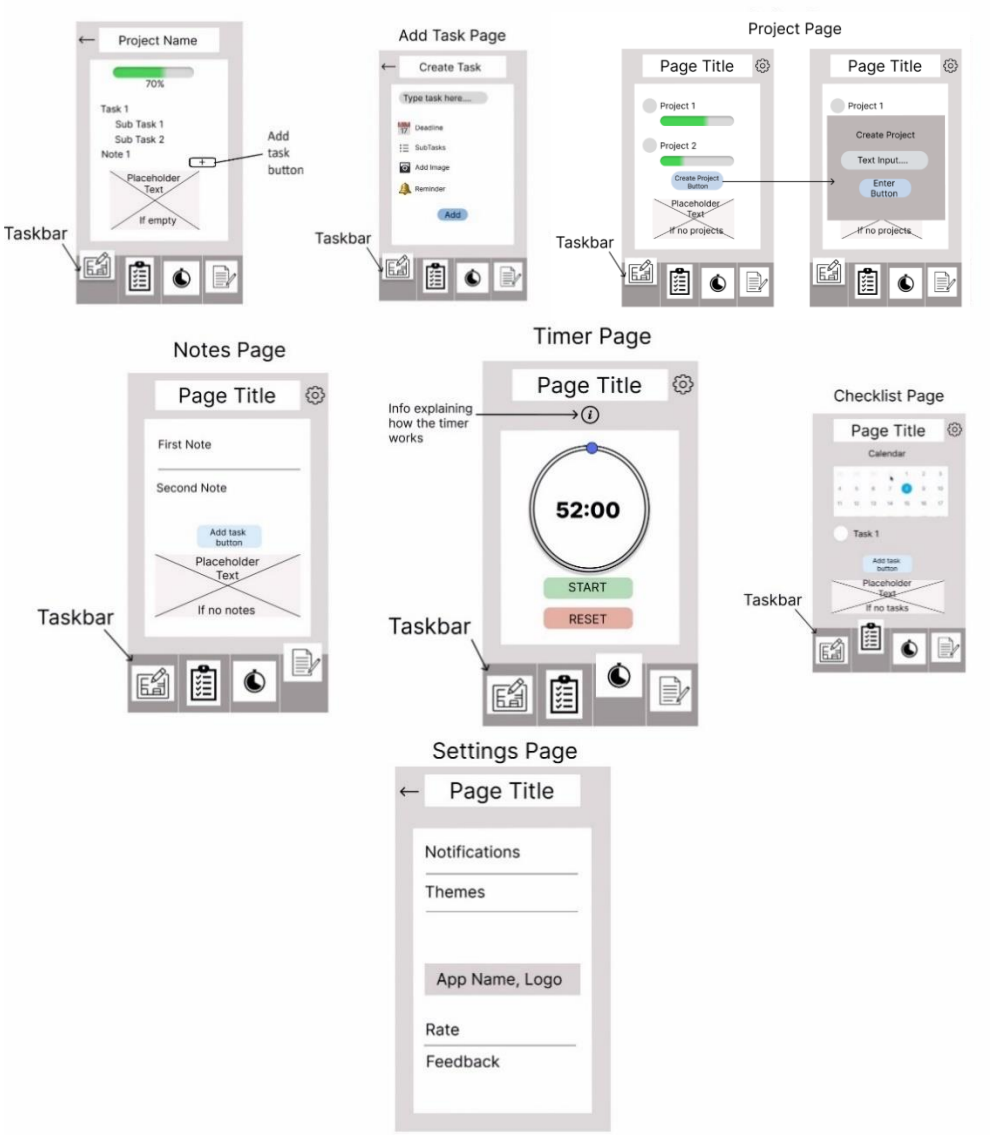
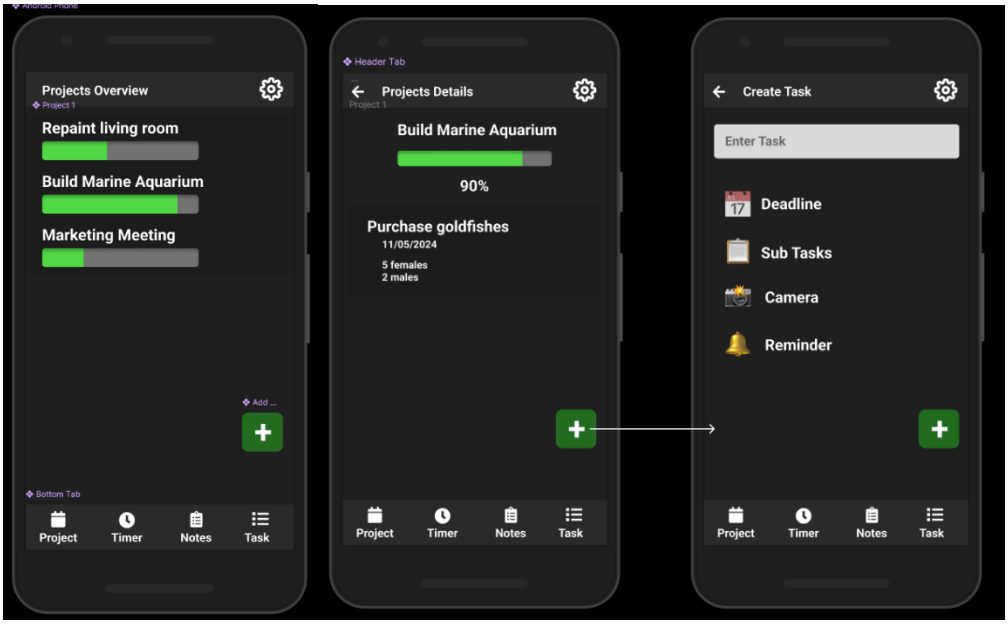


Figure A9 Low Fidelity Wireframe



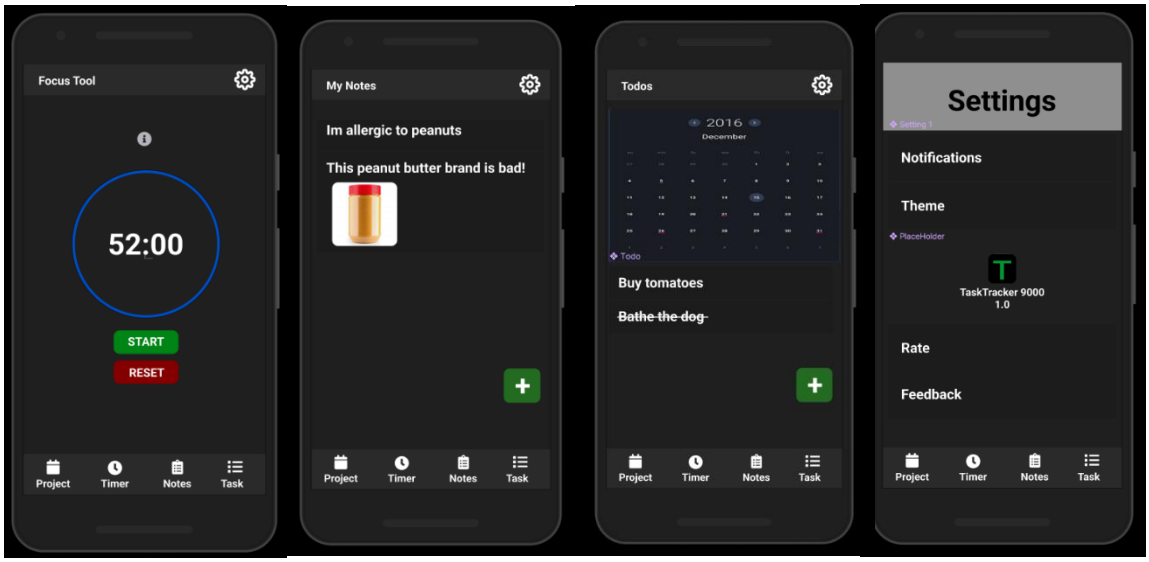


Figure A10 High Fidelity Wireframe

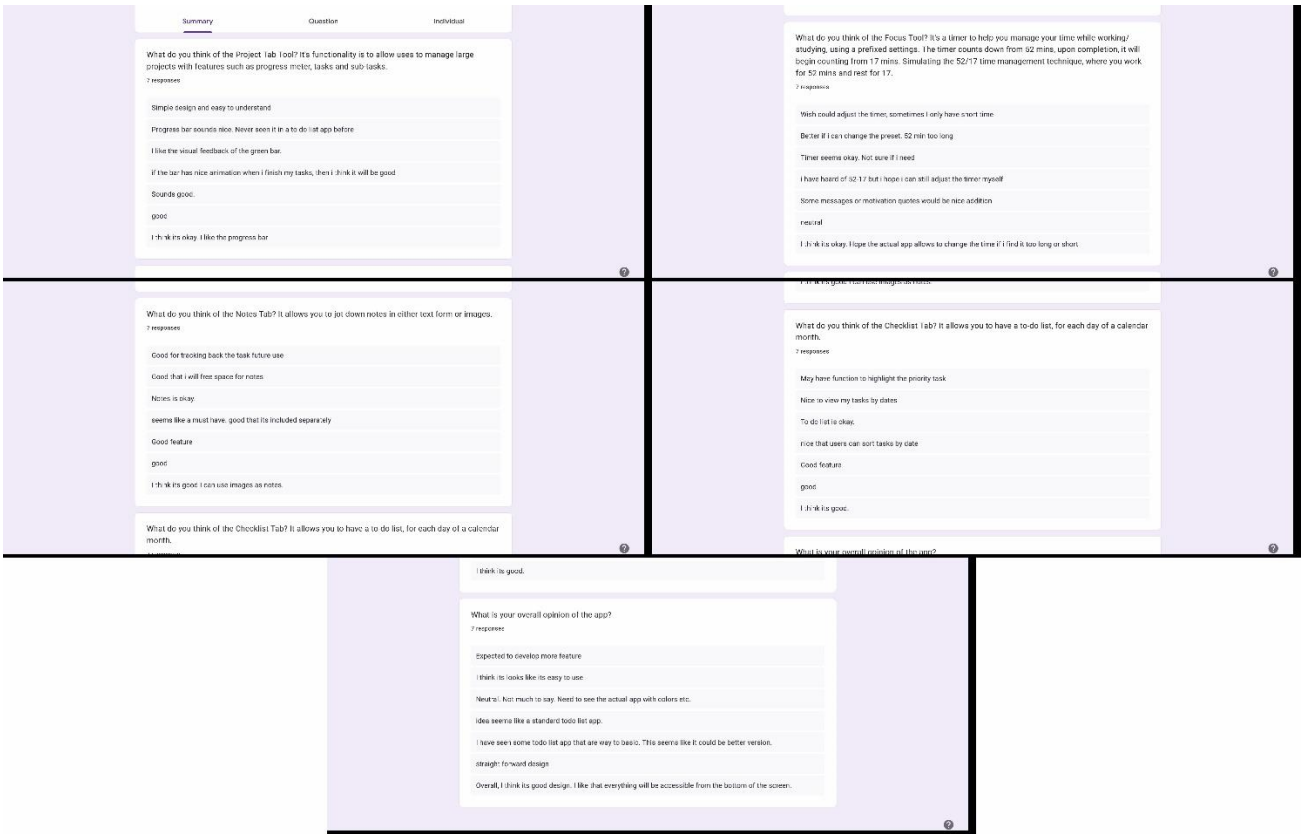


Figure A11 Google Survey

Figure A12 Notification icon, app icon & splash screen

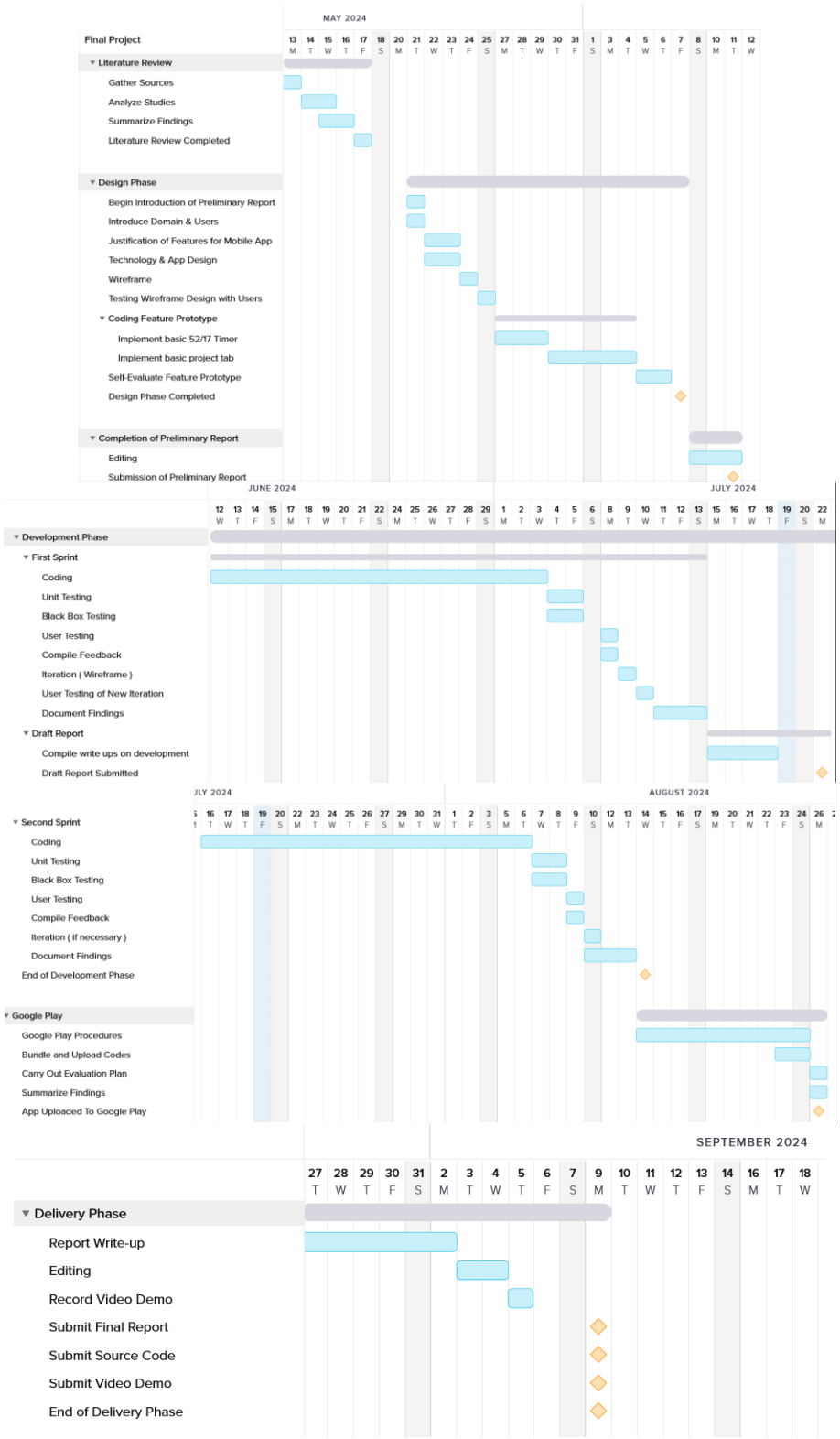


Figure A13 Gantt Chart

Appendix B

The contents of this Appendix include the code snippets and iteration of the database schema, user flow, wireframes and application icon that were mentioned in Sprint sections.

```
TaskTracker 9000
  ---| android
  ---| assets
  ---| components
  ---| constants
  ---| context
  ---| hooks
  ---| navigation
  ---| screens
  App.js
  app.json
  babel.config.js
  eas.json
  package.json
  package-lock.json
```

Figure B1 File Directory

```
import { React, useContext } from "react";
import { TouchableOpacity, Text } from "react-native";
import { NavigationContainer } from "@react-navigation/native";
import { createStackNavigator } from "@react-navigation/stack";
import { createBottomTabNavigator } from "@react-navigation/
bottom-tabs";

import HomeScreen from "../screens/HomeScreen";
import TimerScreen from "../screens/TimerScreen";
import RenderCamera from "../screens/CameraScreen";
import ProjectDetails from "../screens/ProjectDetailsScreen";
import TodoScreen from "../screens/TodoScreen";
import ProjectTaskScreen from "../screens/TaskCreationScreen";
import NotesScreen from "../screens/NotesScreen";
import NoteCreationScreen from "../screens/NoteCreationScreen";
import SettingsScreen from "../screens/SettingsScreen";

import useHeaderBackground from "../hooks/headerBackground";
import useHeaderTitle from "../hooks/headerTitle";

import { Entypo } from "@expo/vector-icons";
import { Foundation } from "@expo/vector-icons";
import { FontAwesome5 } from "@expo/vector-icons";

import { themeContext } from "../context/themeContext";

// navigation stack object
const ProjectStack = createStackNavigator();

//a stack for each bottom navigation
function ProjectStackScreen({ navigation }) {···
}

const TimerStack = createStackNavigator();

function TimerStackScreen({ navigation }) {···
}

const TaskStack = createStackNavigator();

function TaskStackScreen({ navigation }) {···
}

const NotesStack = createStackNavigator();

function NotesStackScreen({ navigation }) {···
}

const Tab = createBottomTabNavigator();

//hiding the settings tab, but still navigable
export default function CustomNavigator() {···
}
```

Figure B2 Custom Navigator Component

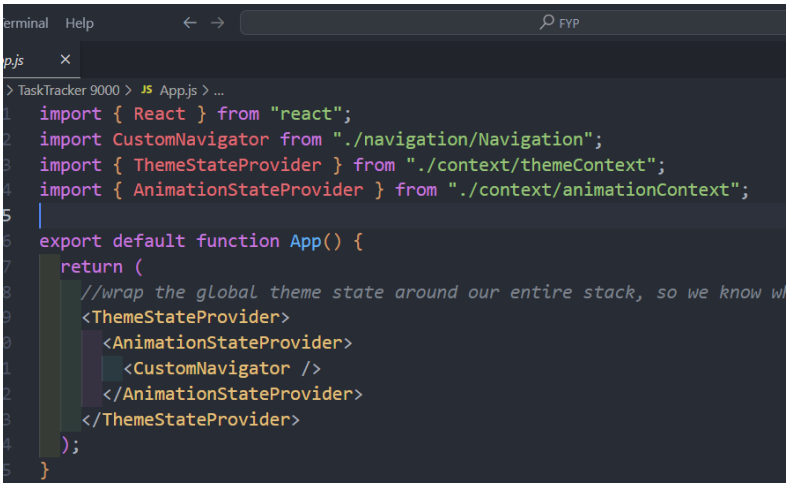


Figure 13 Global Context

Figure B4 Header Custom Hooks

```
import * as SQLite from "expo-sqlite";

//intiatelize SQLite DB
const db = SQLite.openDatabaseSync("userData");
// DROP TABLE IF EXISTS Projects;
// DROP TABLE IF EXISTS Notes;
// DROP TABLE IF EXISTS ProjectDetails;
// DROP TABLE IF EXISTS Todos;
const intiatelizeDatabase = () => {
  try {
    //database persists across restart
    db.execSync(`
          CREATE TABLE IF NOT EXISTS Projects (
          id INTEGER PRIMARY KEY,
          projectName TEXT NOT NULL,
          progress INTEGER
          );

          CREATE TABLE IF NOT EXISTS ProjectDetails (
          id INTEGER PRIMARY KEY AUTOINCREMENT,
          projectId INTEGER NOT NULL,
          tasks TEXT,
          subtasks TEXT,
          deadline TEXT,
          reminder TEXT,
          notes TEXT,
          image TEXT,
          FOREIGN KEY (projectId) REFERENCES Projects (id));

          CREATE TABLE IF NOT EXISTS Notes (
          id INTEGER PRIMARY KEY AUTOINCREMENT,
          note TEXT NOT NULL,
          image TEXT
          );

          CREATE TABLE IF NOT EXISTS Todos (
          id INTEGER PRIMARY KEY AUTOINCREMENT,
          date TEXT,
          todo TEXT,
          done TEXT
          );

          `);

    console.log("database.js has ran...");
  } catch (err) {
    console.log(err);
  }
};

intiatelizeDatabase();

export { db };
```

Figure B5 SQLite database file

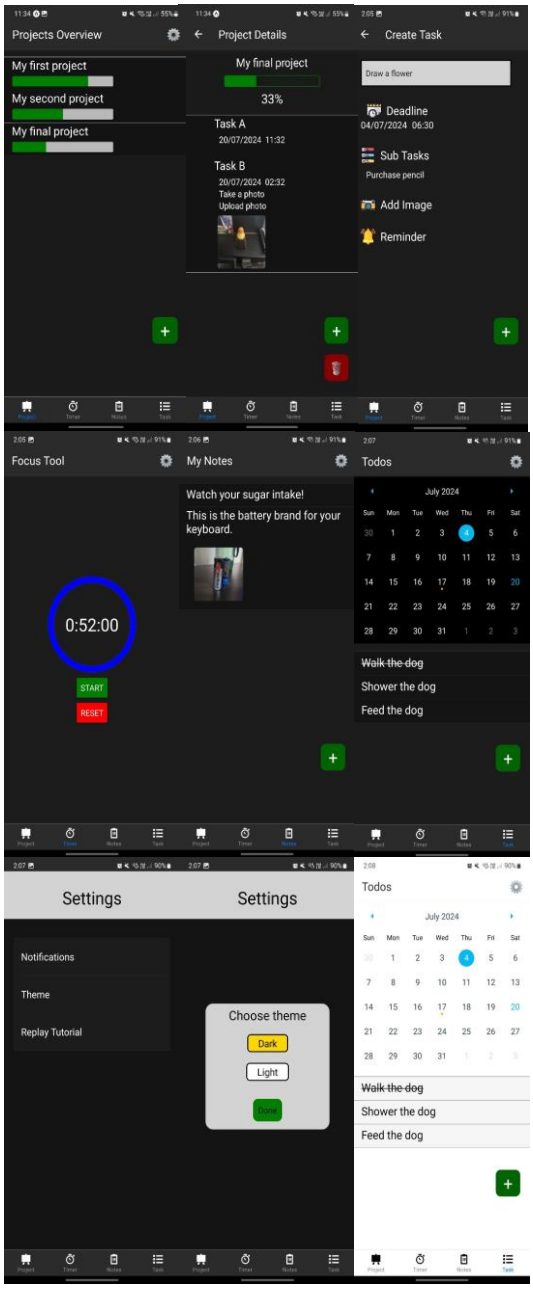


Figure B6 First Sprint Live App

Figure B7 Iterated App & Notification Icon

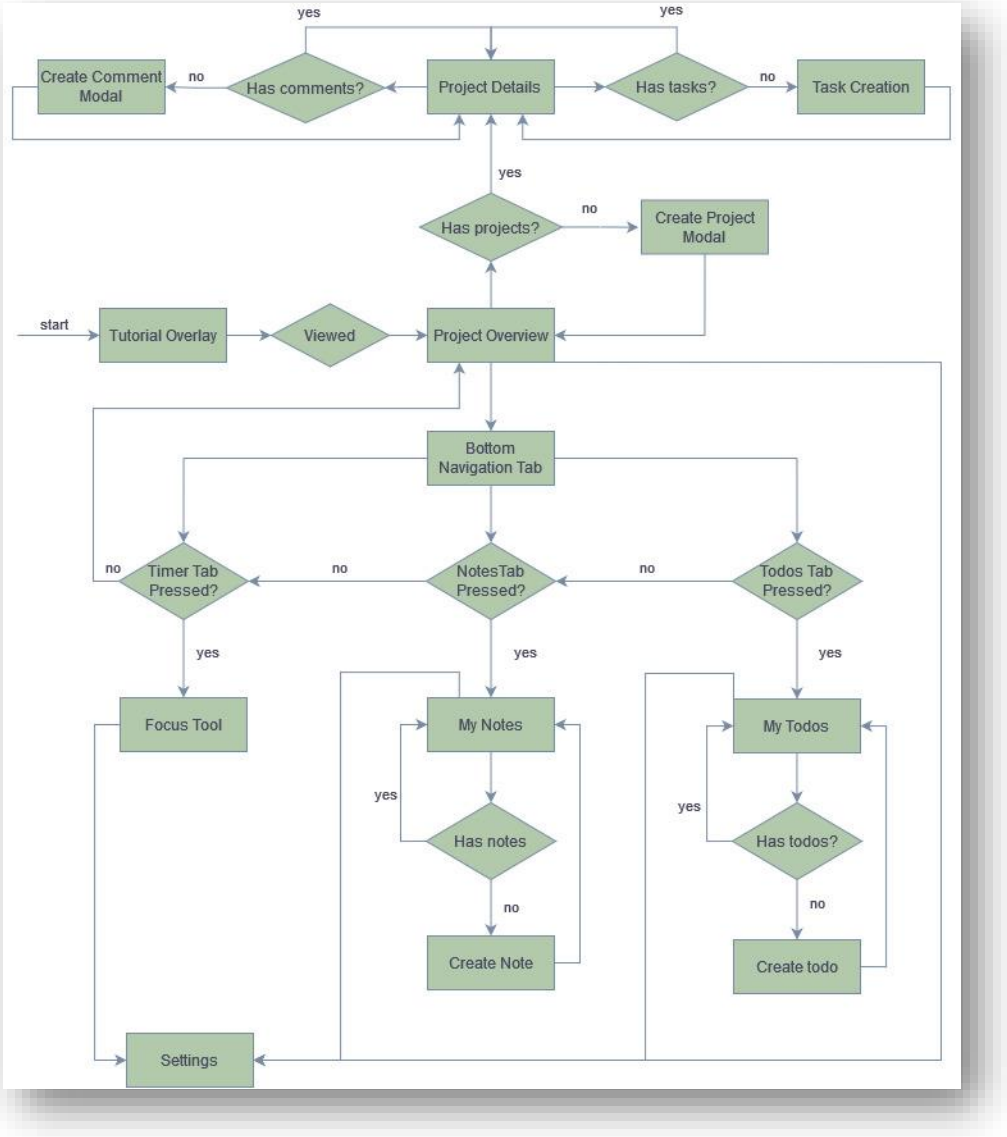


Figure B8 Iterated User Flow Diagram

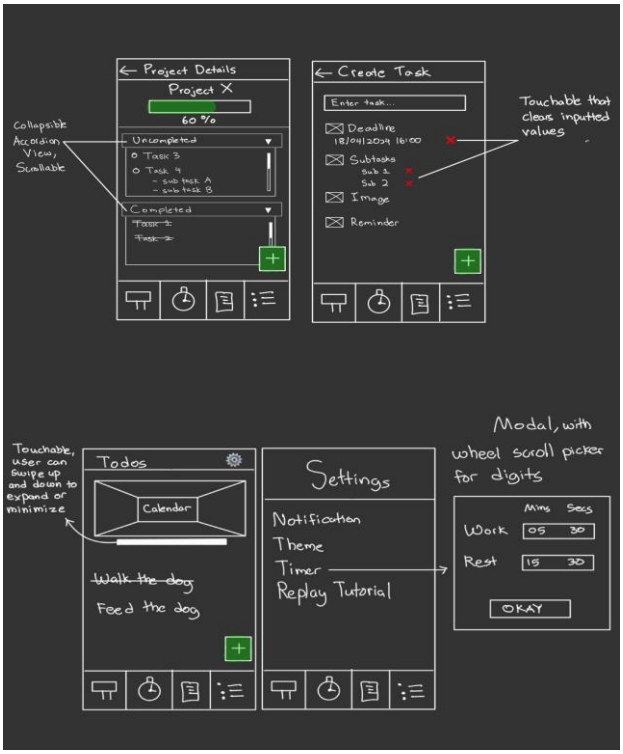


Figure B9 Low Fidelity Iterated Wireframe 1

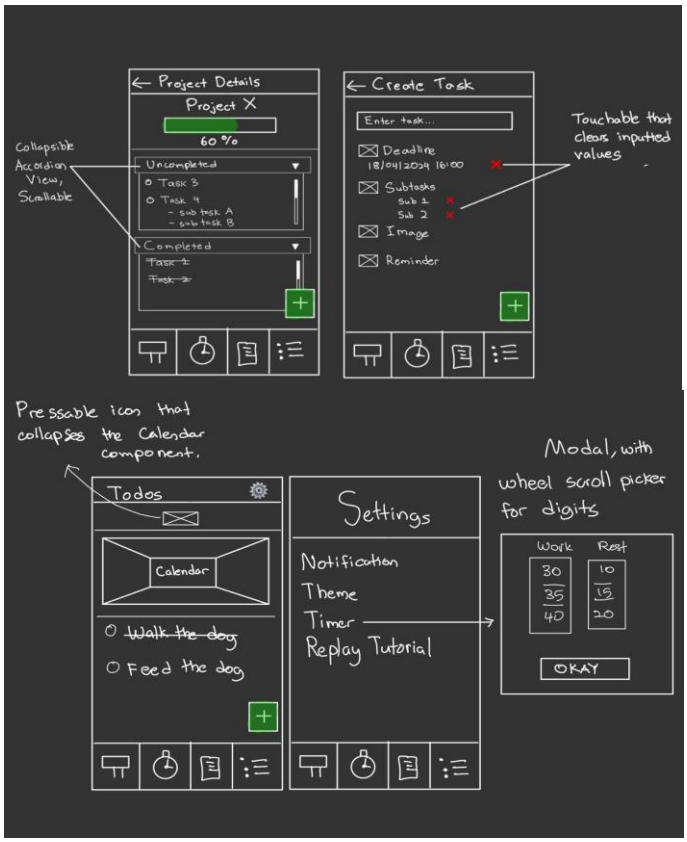


Figure B10 Low Fidelity Iterated Wireframe 2

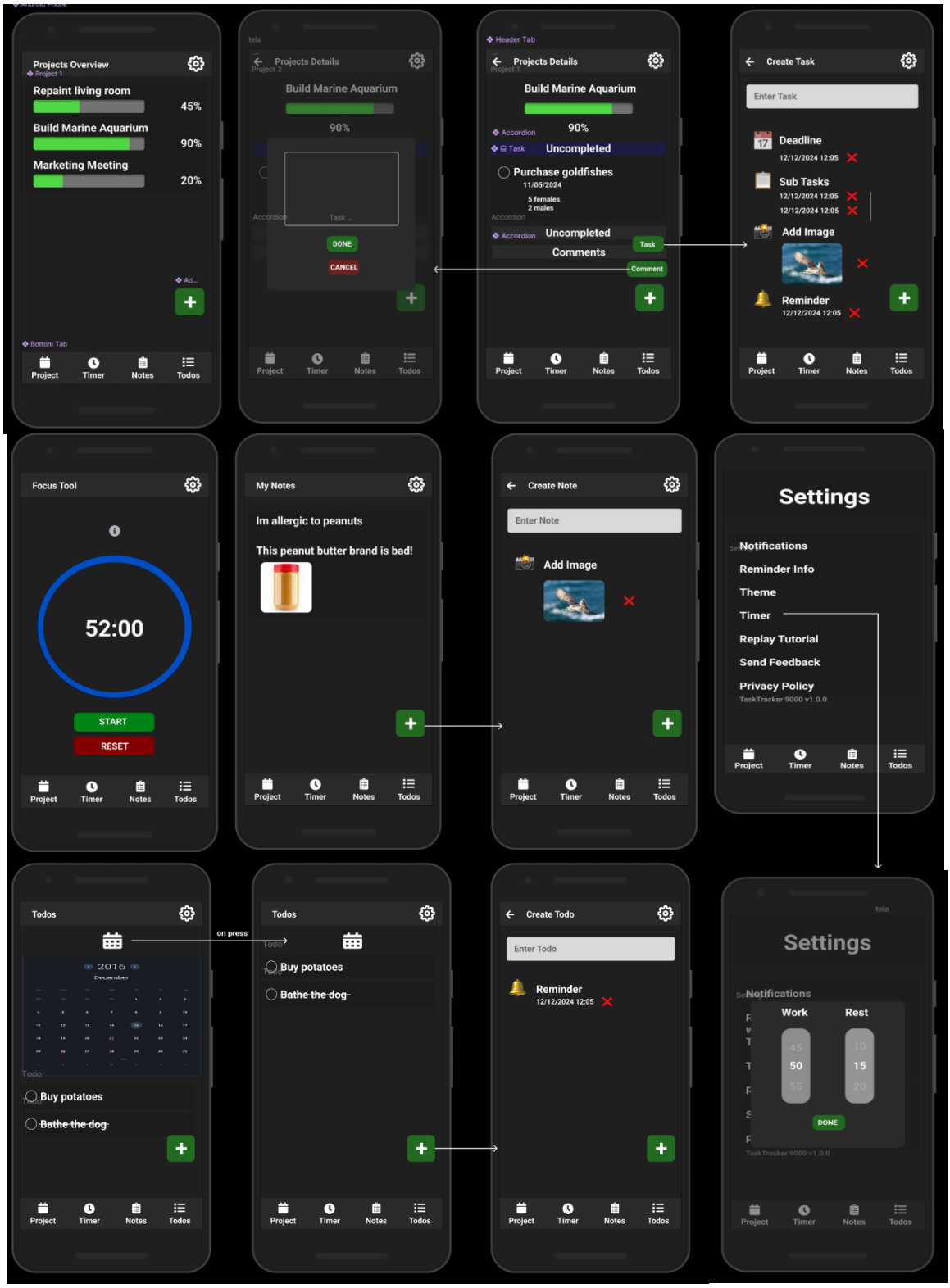


Figure B11 Iterated High Fidelity Wireframe

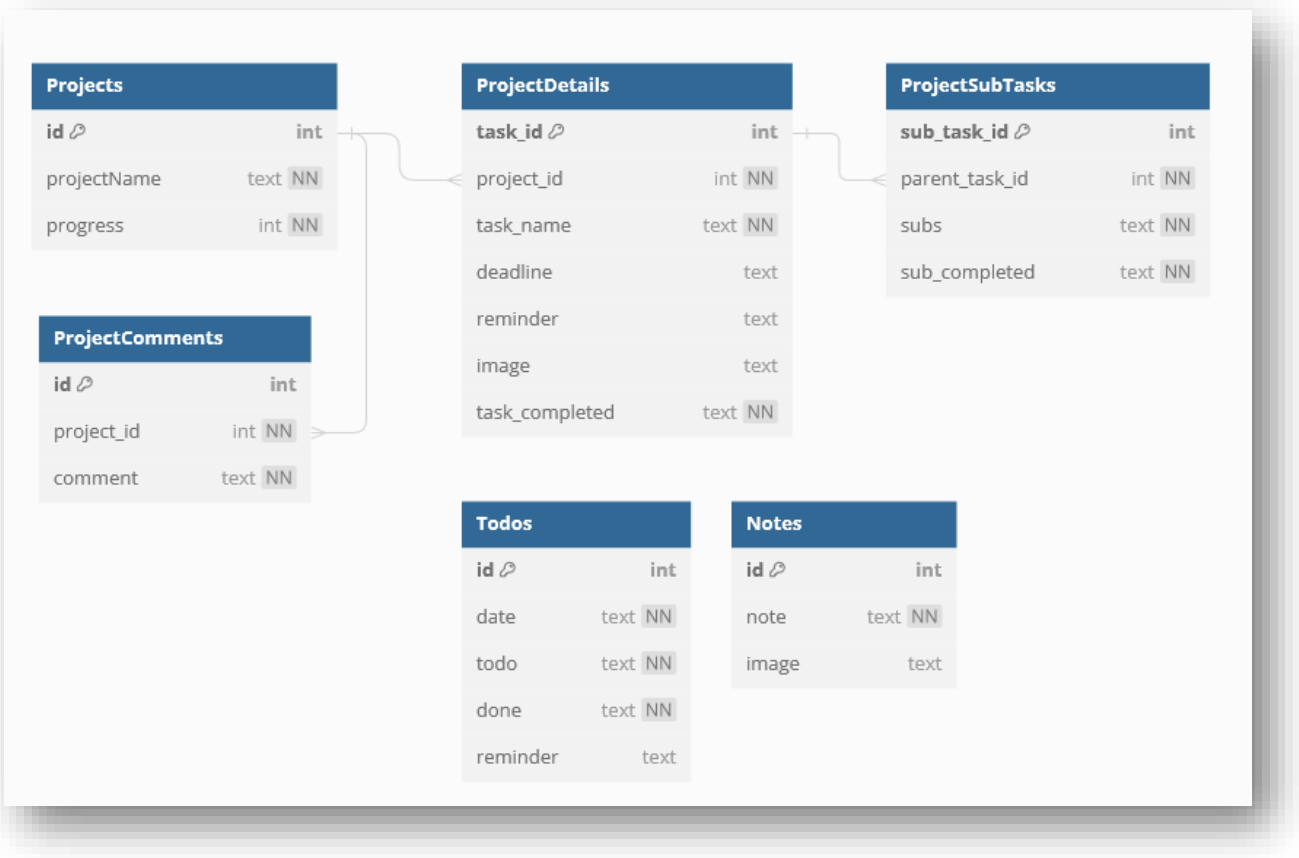
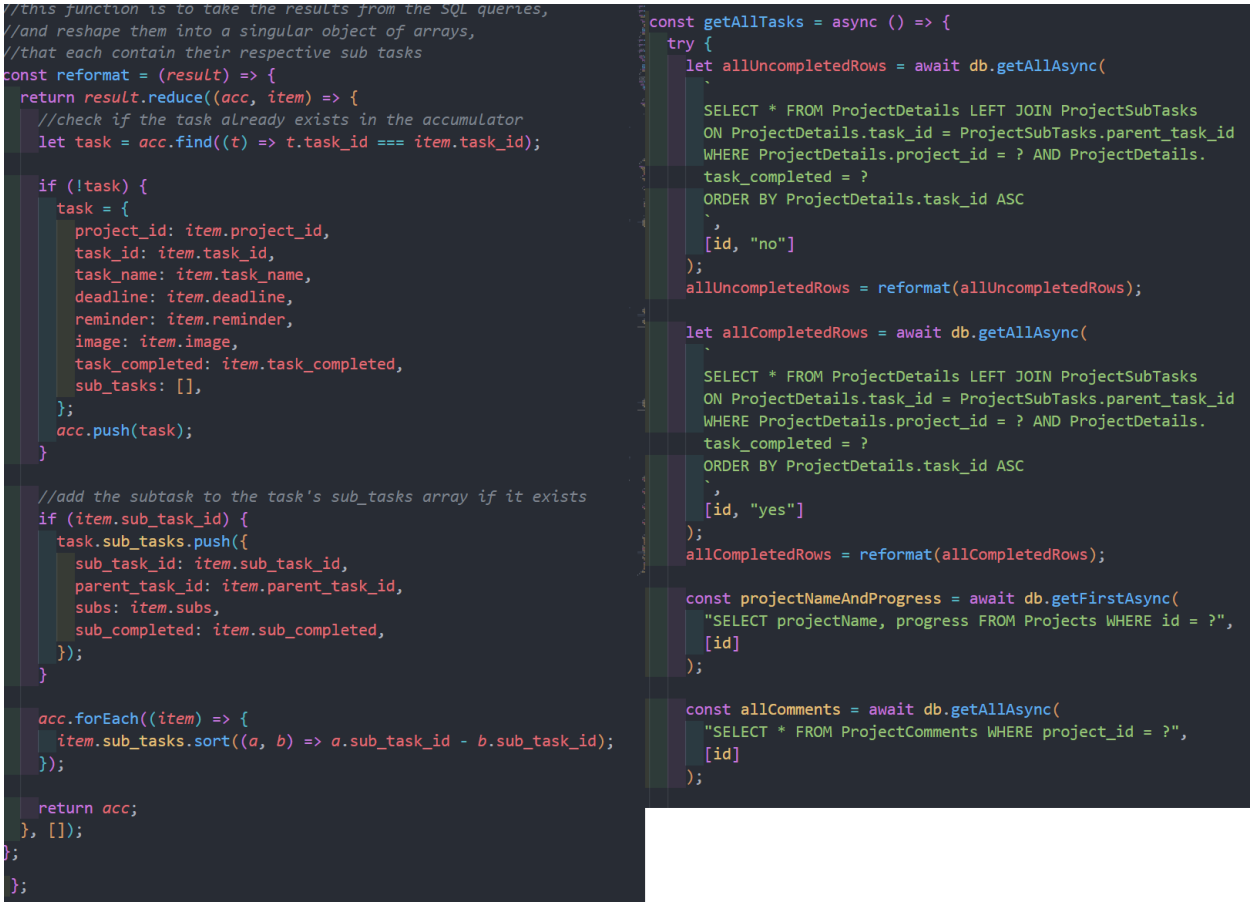


Figure B12 Iterated Relational Database Schema

```
export function AccordionItem({ isExpanded, viewKey, children,
style, duration = 200 }) {
  const height = useSharedValue(0);

  const derivedHeight = useDerivedValue(() =>···
  );
  const bodyStyle = useAnimatedStyle(() => ({···
  }));
  return (
    <Animated.View key={`accordionItem_${viewKey}`} style={[styles.
    animatedView, bodyStyle, style]}>
      <View
        onLayout={(e) => {
          height.value = Math.round(e.nativeEvent.layout.height);
        }}
        style={styles.wrapper}
      >
        {children}
      </View>
    </Animated.View>
  );
}

/**···
export function Item({ content, scrollHeight, currentTheme }) {
  return (
    <ScrollView···
    </ScrollView>
  );
}

export function Parent({ isOpen, AccordionComponent, ItemComponent,
uniqueKey }) {
  return (
    <View style={styles.parent}>
      <AccordionComponent isExpanded={isOpen} viewKey={uniqueKey}>
        <ItemComponent />
      </AccordionComponent>
    </View>
  );
}
```

```
export function AccordionTouchable({ onPress, currentTheme, text }) {
  const progress = useSharedValue(0);

  const animatedStyle = useAnimatedStyle(() => {
    return {
      backgroundColor: interpolateColor(progress.value, [0, 1],
      ["#2B2B2B", "#003366"]),
    };
  });

  return (
    <TouchableHighlight
      onPress={() => {
        onPress(), (progress.value = withTiming(1 - progress.value,
        { duration: 200 }));
      }}
      style={styles.buttonStyling}
    >
      <Animated.View style={[styles.animatedWrapper, animatedStyle]}>
        <Text
          style={
            currentTheme === "dark"
              ? { color: "white", fontSize: 15, paddingVertical: 3,
              paddingRight: 5 }
              : { color: "black", fontSize: 15, paddingVertical: 3 }
          }
        >
          {text}
        </Text>
      </Animated.View>
    </TouchableHighlight>
  );
}
```

Figure B13 Custom Accordion Code Snippet



```
//this function is to take the results from the SQL queries,
//and reshape them into a singular object of arrays,
//that each contain their respective sub tasks
const reformat = (result) => {
  return result.reduce((acc, item) => {
    //check if the task already exists in the accumulator
    let task = acc.find((t) => t.task_id === item.task_id);

    if (!task) {
      task = {
        project_id: item.project_id,
        task_id: item.task_id,
        task_name: item.task_name,
        deadline: item.deadline,
        reminder: item.reminder,
        image: item.image,
        task_completed: item.task_completed,
        sub_tasks: [],
      };
      acc.push(task);
    }

    //add the subtask to the task's sub_tasks array if it exists
    if (item.sub_task_id) {
      task.sub_tasks.push({
        sub_task_id: item.sub_task_id,
        parent_task_id: item.parent_task_id,
        subs: item.subs,
        sub_completed: item.sub_completed,
      });
    }

    acc.forEach((item) => {
      item.sub_tasks.sort((a, b) => a.sub_task_id - b.sub_task_id);
    });

    return acc;
  }, []);
};

};
```

```
const getAllTasks = async () => {
  try {
    let allUncompletedRows = await db.getAllAsync(
      `
      SELECT * FROM ProjectDetails LEFT JOIN ProjectSubTasks
      ON ProjectDetails.task_id = ProjectSubTasks.parent_task_id
      WHERE ProjectDetails.project_id = ? AND ProjectDetails.
      task_completed = ?
      ORDER BY ProjectDetails.task_id ASC
      `,
      [id, "no"]
    );
    allUncompletedRows = reformat(allUncompletedRows);

    let allCompletedRows = await db.getAllAsync(
      `
      SELECT * FROM ProjectDetails LEFT JOIN ProjectSubTasks
      ON ProjectDetails.task_id = ProjectSubTasks.parent_task_id
      WHERE ProjectDetails.project_id = ? AND ProjectDetails.
      task_completed = ?
      ORDER BY ProjectDetails.task_id ASC
      `,
      [id, "yes"]
    );
    allCompletedRows = reformat(allCompletedRows);

    const projectNameAndProgress = await db.getFirstAsync(
      "SELECT projectName, progress FROM Projects WHERE id = ?",
      [id]
    );

    const allComments = await db.getAllAsync(
      "SELECT * FROM ProjectComments WHERE project_id = ?",
      [id]
    );
```

Figure B14 SQL Query Code Snippet

Appendix C

Process of uploading the application to the Google Play Store.

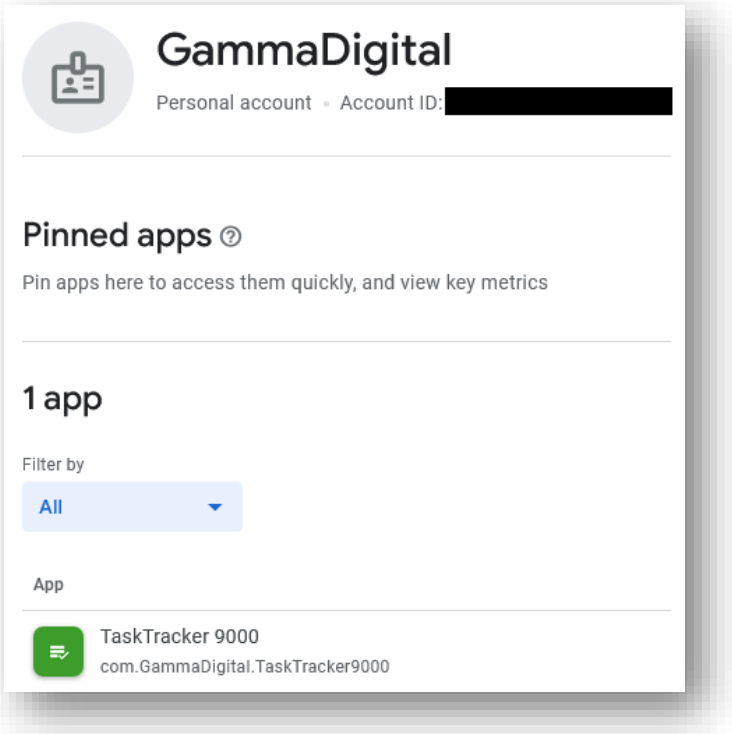


Figure C1 Google Play Console

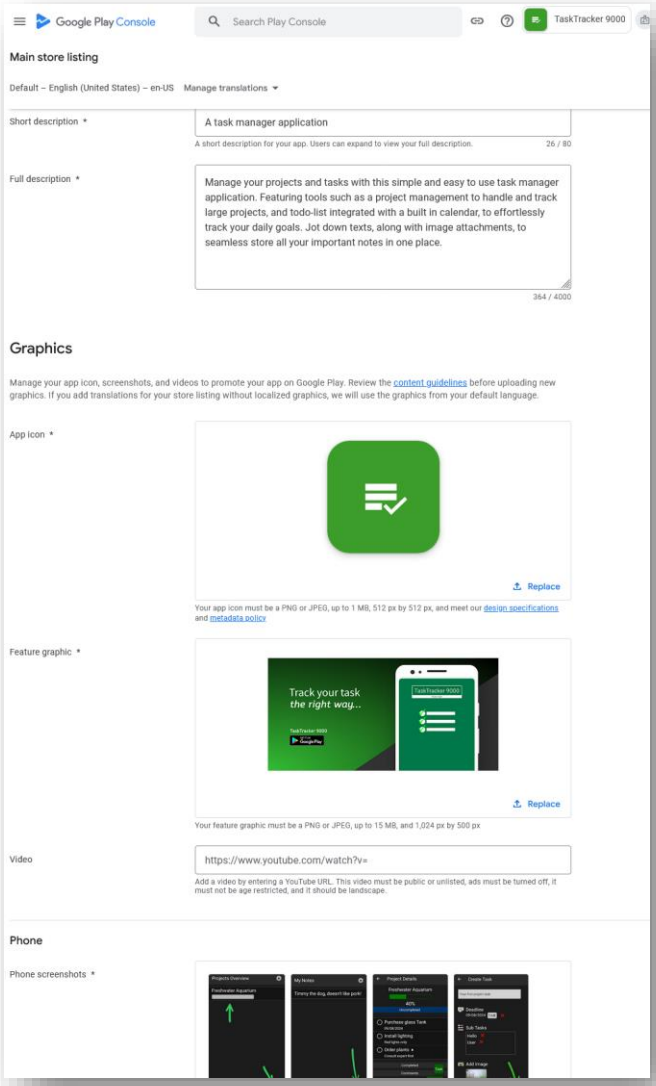


Figure C2 Store Listing

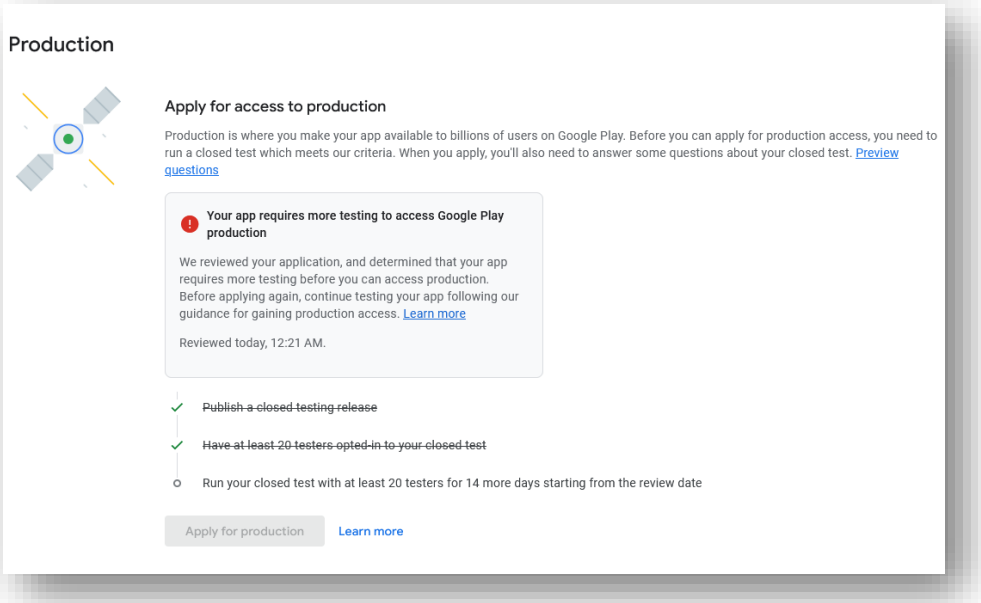


Figure C3 Production Access Dashboard

Appendix D

Evaluation and testing screenshots.

```
PASS  Android  constants/__tests__/push.test.js
PASS  Android  constants/__tests__/database.test.js
PASS  Android  context/__tests__/themeContext.test.js
PASS  Android  hooks/__tests__/headerBackground.test.js
PASS  Android  hooks/__tests__/headerTitle.test.js
PASS  Android  hooks/__tests__/statusBar.test.js
PASS  Android  context/__tests__/animationContext.test.js
PASS  Android  constants/__tests__/carouImages.test.js
PASS  Android  constants/__tests__/timePreset.test.js
PASS  Android  components/__tests__/customPlaceHolder.test.js
PASS  Android  components/__tests__/customShapes.test.js
PASS  Android  components/__tests__/customCarousel.test.js
PASS  Android  context/__tests__/timerContext.test.js
PASS  Android  components/__tests__/customModal.test.js
PASS  Android  components/__tests__/customCells.test.js
PASS  Android  components/__tests__/customTextValidator.test.js
PASS  Android  components/__tests__/customAccordion.test.js
PASS  Android  components/__tests__/customButtons.test.js

Test Suites: 18 passed, 18 total
Tests:       59 passed, 59 total
Snapshots:   15 passed, 15 total
Time:        4.051 s, estimated 13 s
Ran all test suites.
```

Figure D1 Jest Tests Results

TaskTracker 9000 Black Box Testing

The green `+` button on the bottom right of the project details screen must navigate the user to the project task creation screen.	Tap the green `+` button on the bottom right of the device, at a project's details screen.	User is brought to the Task Creation screen.	As expected	PASS	Immediate
The datetime calendar library must launch when the `Deadline` text is pressed.	Tap the `Deadline` text or icon.	The calendar view launches, and the user can select a date.	User can select a date, but when if canceled, the date is still selected and rendered	FAIL	Immediate
The datetime calendar library must launch when the `TIME` text is pressed.	Tap the `TIME` text next to the selected date.	The time picker view launches, and the user can select a time.	User can select a time, but when if canceled, the time is still selected and rendered	FAIL	Immediate
Ensure the creation of a sub task modal launches.	Tap the `Sub Tasks` text or icon.	The create a sub task modal renders.	As expected	PASS	Immediate

Figure D2 System Test Results

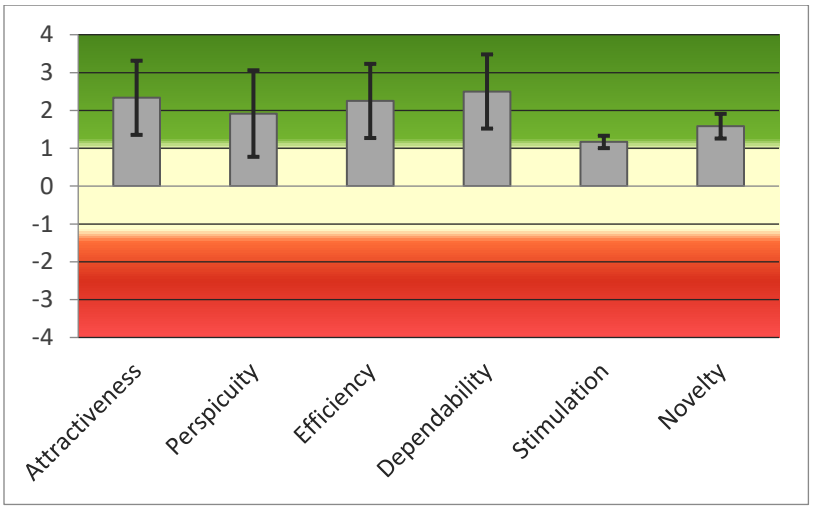


Figure D3 TaskTracker 9000 UEQ Results

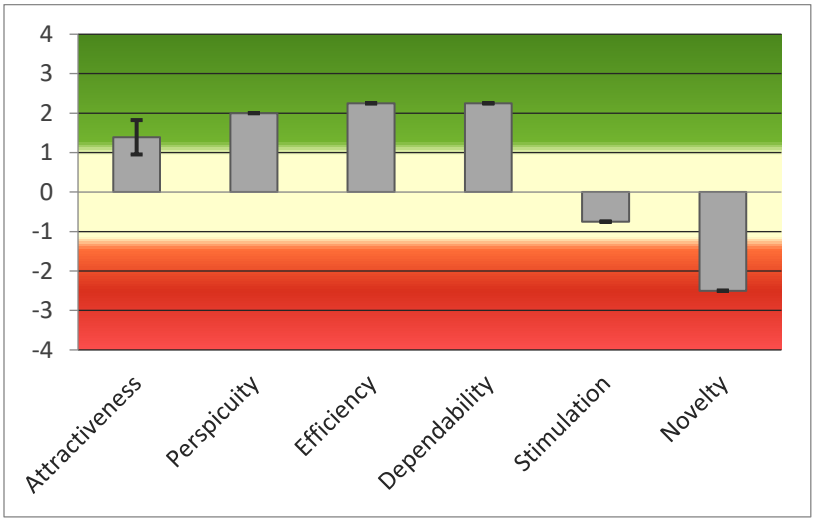


Figure D4 Google Tasks UEQ Results

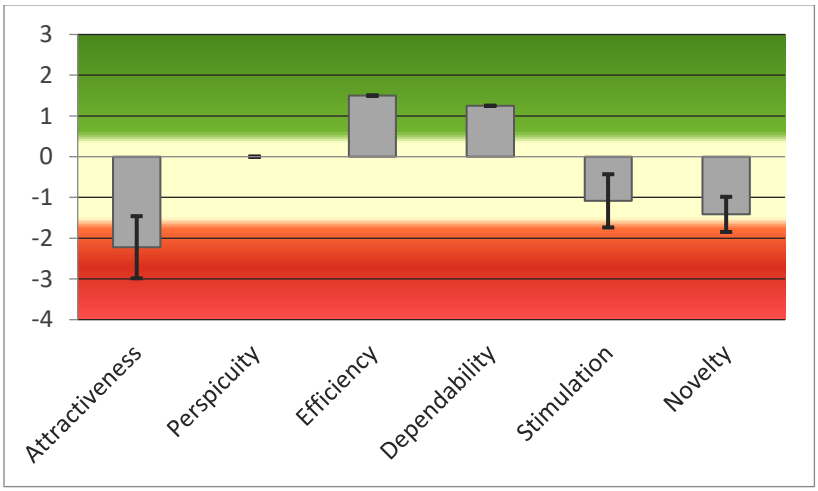


Figure D5 Microsoft To Do UEQ Results

Please assess the product now by ticking one circle per line.

	1	2	3	4	5	6	7		
annoying	○	○	○	✓	○	○	○	enjoyable	1
not understandable	○	○	○	○	✓	○	○	understandable	2
creative	○	○	✓	○	○	○	○	dull	3
easy to learn	○	○	✓	○	○	○	○	difficult to learn	4
valuable	○	○	✓	○	○	○	○	inferior	5
boring	○	○	○	✓	○	○	○	exciting	6
not interesting	○	○	○	○	✓	○	○	interesting	7
unpredictable	○	○	○	✓	○	○	○	predictable	8
fast	✓	○	○	○	○	○	○	slow	9
inventive	○	✓	○	○	○	○	○	conventional	10
obstructive	○	○	○	○	✓	○	○	supportive	11
good	○	✓	○	○	○	○	○	bad	12
complicated	○	○	○	○	✓	○	○	easy	13
unlikable	○	○	○	○	✓	○	○	pleasing	14
usual	○	○	○	○	✓	○	○	leading edge	15
unpleasant	○	○	○	○	✓	○	○	pleasant	16
secure	✓	○	○	○	○	○	○	not secure	17
motivating	○	✓	○	○	○	○	○	demotivating	18
meets expectations	○	✓	○	○	○	○	○	does not meet expectations	19
inefficient	○	○	○	○	✓	○	○	efficient	20
clear	○	○	○	✓	○	○	○	confusing	21
impractical	○	○	○	✓	○	○	○	practical	22
organized	○	○	✓	○	○	○	○	cluttered	23
attractive	○	○	✓	○	○	○	○	unattractive	24
friendly	○	○	✓	○	○	○	○	unfriendly	25
conservative	○	○	○	✓	○	○	○	innovative	26

Figure D6 UEQ Lowest Score TaskTracker9000

Appendix E

This appendix houses the changelog and user feedback gathered throughout the project

User Feedback

Below is a compilation of the feedback gathered from the face-to-face user testing.

- The application icon needed rework
- Long pressing the cells to delete it was preferred, users did not like a delete icon to delete the project or cell
- Many wanted the ability to view already finished project tasks, instead of deleting them away instantly
- Users wanted to have notes for each project as well
- An accidental tap on any task or todo cell would immediately delete it or mark it as done, with no way to undo or revert it
- Some liked the 52/17 preset
- Most preferred to be able to manually set the timer
- Most appreciated the ability to add images as attachments.
- Users couldn't delete an input when creating a project task in the task creation screen, users did not appreciate reloading the page to re-correct their false input.
- Todo and project tasks that were completed should have a greyed-out color, in addition to the strikethrough.
- Users felt it was difficult to differentiate one note/task/todo cell from another
- Some users highlighted a notice should be displayed if task creation was successful or not
- App logo and version should be placed after all settings cells, not in the middle
- Having 00:52:00 or 52:00 in the timer would make more sense than 0:52:00
- Users felt timer could be bigger
- The placeholder text for Todos in the Task tab should display the future dates when days that were not today was pressed
- Users noticed a white flash when app finishes loading upon launch
- In the task creation screen, deadline and reminder element would still input a date, even when the datetime picker was canceled
- The app did not save the theme state, and would revert back to dark from light
- Users wanted a different color dot for days that had completed and uncompleted todos
- Users found the grey background of the Settings title in the settings screen revolting
- Users wanted the ability to view images that were attached to tasks/notes in an enlarged state as it was too small to view the details of the image
- Users found that long strings of text would go offscreen at the line break

Changelog

All notable changes to this project will be documented in this file.

[Second Sprint]

Added

- accordion tabs for completed and uncompleted tasks in the project details screen
- added new table in SQLite DB, ProjectComments
- added comment accordion tab, to simulate notes, but for each project
- added long press on cells to delete
- spring text animation for successfully created tasks, and for missing values in input or reminder
- double tap notes cell to edit them
- floating action buttons to add comment or task
- added button to toggle Calendar opening and closing in Task tab
- Reminder Info Cell now in Settings, informing user about battery optimization

Fixed

- completed tasks would not persist across app restart, state was empty
- project tasks would reappear when the project was deleted and recreated
- fixed tutorial modal being laggy after pagination was added
- fixed tutorial carousel and pagination not being responsive
- fixed white screen appearing on app launch, instead of splash screen
- fixed theme defaulting back to dark on restart when light mode was set
- fixed tableview simple library Cell separator being missing
- fixed text out of bounds on Todo screen

Modified

- slightly increase height and width of text inputs in modals
- added circular touchable to complete tasks instead of touching the entire cell
- increased size of all elements in the focus timer tool screen
- placeholder text in Task tab, shows the date when there’s no todo on selected date
- updated tutorial images
- change Task tab name to Todo, header is now My Todos
- pressing an image in a Cell now opens a show image modal

[First Sprint]

Added

- redid app logo designed, did not fit properly
- added marked dots to calendar, to signify which dates have todos
- pagination to tutorial carousel

Fixed

- fixed application crashing during development build launch
- calendar background would not change with theme, only when app relaunches
- fixed a bug where application would crash when returning from camera screen
- fixed issue where if camera permission was denied at the notes screen, hitting the back arrow would navigate to the task creation screen of the project tool instead
- tutorial would always run on app launch

Modified

- replaced emojis in camera screen with vector icons
- modified placeholder screen when camera permission denied
- cells now have a long press to delete the cell instead
- increase touchable width to encompass the entire cell, instead of just the text
- added multiline to text input

Removed

- red delete button to delete project or tasks