SOUTHERN
ADVENTIST UNIVERSITY
School of Computing

Gamified Task Manager

Marc BUTLER
EMAIL: marcbutler@southern.edu

supervised by
Willard MUNGER, PhD.
Robert ORDÓÑEZ, M.S.

May 5, 2022

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Gamified Task Manager

MARC BUTLER

***Abstract*—In this paper we aim to remove the difficult structuring of tasks through a task management app that is to be developed in the winter of 2022. The target audience in this work will consist of college students and will be tested or evaluated in terms of a how ease the UI is to interpret. We cover multiple task management applications that are available on the Google Play Store. Additionally our team has researched development tools, psychological nature of students employing a task management system, and the effects of smartphone notifications.**

I. Introduction

A. Motivation

Imagine the stress a college student in Computer Architecture and Calculus II courses will have during finals week. The only method to pass classes with an acceptable grade is to have reasonable discipline.

Studies show a high correlation between increased mobile phone usage and unsatisfactory grades [1]. Furthermore, excessive notifications often distract college students [2]. Many time management solutions exist on an array of platforms to aid in the correction of this issue, but these applications lack cohesion, integration with other third party tools, or an easy-to-use interface.

The Pomodoro technique has users spend about twenty-five minutes studying and the next five minutes taking a break during each Pomodoro session [3]. It is because of this students will be more focused when they come back to study. This effect promotes study efficiency and time management.

Our new cutting-edge Pomodoro scheduling mobile app will mend the situation. The application will be able to shred through study time and revamp study efficiency all through gamifying a variable timer. All distracting notifications will be disabled and the user will be rewarded through his or her great concentration with an aesthetically pleasing UI trophy to be added to his or her trophy case. This trophy case will store the user's trophies with each containing a memo of what task the user had accomplished.

B. Problem Statement

College students want to study efficiently and waste less time preparing for exams. Existing solutions do not provide aesthetic animations, time management techniques, or are not exciting–according to the ratings. This can be seen in **Table 1** Therefore it is valuable for college students with smartphones to utilize our task management system: a gamified mobile application implementing features such as the Pomodoro technique.

With this application we implement the Pomodoro timer and a task management system to solve disorganization and poor time management which are two of the leading causes of college student class failure. Our proposed application is targeted at college students primarily, but should be available to anyone with at least an Android device.

II. The Setting

A. Apps on the Market

Some of the more popular habit or scheduling managing apps on the Google Play Store include *Forest* [4] and *Habit Forest* [5]. The *Forest* app has won the award for the 2018 Google Play Editors' Choice Top Productivity App. It touts the ability to grow a tree when a task is completed through gamification. A user can set reminders, custom phrases, and use popular timer options such that utilize the Pomodoro technique. The *Habit Forest*, as the name suggests, allows the creation of habits while also supporting reminders to keep up good habits. Much like *Forest*, this application has "trees" that will sprout into foliage denoting a habit.

Among other smartphone apps pertaining to task management that were discovered on the Google Play Store are *Habatica* [6], *Todolist: To-Do List & Tasks* [7], and *LifeUp: Gamification To-Do & Tasks List | Habit RPG* [8]. A more comprehensive list of apps and their properties can be viewed in **Table 1** which additonally include *Do It Now: RPG To Do List. Habit Tracker. Planner* [9], *Pomodoro* [10], *Epic To Do List* [11] and *Level up Life* [12].

Our fields in how we determine a great task management, study, or Pomodoro timing application include 13 fields:

1) **Rating:** A Rating from the Google Play Store
2) **Cost:** A cost in $USD
3) **Pomodoro Timer:** If the application has a Pomodoro timer
4) **Animations:** Are there fantastic animations?
5) **UI:** In regards to the look and interaction of the application
6) **Tutorial:** Is there a tutorial that shows how to use the application and what is the quality?
7) **Skills:** Represents a category in which a user may place a task to improve on; programming, swimming, hiking, etc.; Is it available?
8) **Attributes:** Represents statistics such as strength, charisma, and intelligence
9) **Customization:** Can the theme of the app, tasks colors, or even the font can be edited?
10) **Statistics / Progress:** Is there a way to show user progress in task completion? Is it graphical?
11) **Bad / Good Habits:** Is there a management of creating new good habits or working to drop bad habits
12) **In-Game Currency:** An incentive in the form of virtual coins, gems or other denominations for users to stay motivated with purchases from an in-game shop; earn through task completion

APP	*Habitica*	*Todolist*	*Do it Now*	*Pomodoro*	*Epic-to-do list*	*Level up Life*	*LifeUp*
Rating	4.3	4.5	4.4	4.7	4.7	4.1	4.6
Cost	Free	Free	Free	Free	Free	Free	$2.99
Pomodoro Timer	No	No	No	Yes	No	No	Yes
Animations	Minimal	Minimal	Minimal	Yes	Yes	Minimal	Yes
UI	Pleasing	Pleasing	Average	Simplistic	Style	Average	Very
Tutorial	Yes	No	Yes	No	Yes	Yes	Yes
Skills	Yes	No	Yes	No	Yes	Yes	Yes
Attributes	Yes	No	Yes	No	Yes	Yes	Yes
Customization	Yes	No	Yes	No	Yes	Minimal	Yes
Statistics / Progress	Some	Yes	Yes	Yes	Yes	Yes	Yes
Bad / Good Habits	Yes	No	Yes	No	Yes	Maybe	Yes
In-Game Currency	Yes	No	Yes	No	Yes	No	Yes
Gamified	Yes	No	Maybe	No	Maybe	No	Maybe

TABLE I: The most popular task management apps on the market

13) **Gamified:** Is the application playable or does it have a entertaining atmosphere? Do user choices and task completion affect the state of a game?

Although these applications are splendid in what they do, they lack the core functionality of allowing users with various options. This is explained by reviewers of the application who have claimed that there was no way to pause and resume a task and the inability to control the sound and music for the *Forest* app [4]. The current challenges that *Habit Forest* face are the lack of options to re-edit or categorize habits with additional high severity bugs such as notifications not showing up or the incapacity to distinguish AM or PM time.

B. Explored Research

Among some of our research that was conducted:

1) **Exploration of mobile trackers [13]**
2) **The effects of smartphone notifications on users [14]**
3) **Mood-monitoring Apps [15]**
   These are among the subset of resources that center the focus on mobile applications. Our research has also focused on sociological and psychological topics:
4) **How the Pomodoro Technique can reduce procrastination in Grad students [16]**
5) **Time management for students [17]**
   Finally, we explored implementation methods using frame works and tools through:
6) **Research in NoSQL and SQL Databases for Mobile Applications [18]**
7) **Firebase[1] for Android development [19]**

*1) **Exploration of mobile trackers**:* In our research of the development of mobile trackers, we had determined that there is an initial phase that includes idea generation, research and development, testing, analysis, and a roll out phase [13].

*2) **The effects of smartphone notifications on users**:* From research on the effects of notifications we had determined that most notifications, are considered unimportant and actually distracting to the user [14]. This is largely determined by the content of the notifications. An example of bad notification content is overly wordy or abstract. Purportedly, there is a correct time to deliver notifications to the user [14]. People are more likely to be distracted during a nine-to-five work day, during a school day, or while sleeping. Between these times users are more likely to be distracted during work [14]. Another point discovered within our research was that notifications are generally useful. The closing of notifications does not mean that the notification did not help the user. That does not go without saying that notifications can still be intrusive.

*3) **Mood-monitoring Apps**:* Our analysis of mood-monitoring apps consists of discoveries on what users look for in an application. This is supported by reviews that a user would make. Purportedly, over one-third of reviews are centered around accessibility [15]. Simplicity was one of the major components that makes an app accessible especially in regard to a mood tracker app. Another esteemed component in what users desire in an application is flexibility. Users would want to be able to customize their experience [15]. For us, this means granting the ability to modify themes of our app. Aside from these components, user requests are essential to the growth of the app [15]. Multiple users would provide feature request in their reviews. This is evidence that a community between the developer(s) and the users is important and will be the driving force behind our studying application. Our community is generally targeted at college students with smartphones so it is logical to communicate with college students on this project.

*4) **How the Pomodoro technique can reduce procrastination in grad students**:* In regard to graduate students, a research study was done on affects of a Pomodoro timer on student procrastination. In the study, it was determined that student procrastination was negatively correlated to the GPA of students [16]. However, when the students were introduced to an online task management timer, their late work would decrease, greater focus was achieved, and motivation was instilled in the students.

*5) **Time management for students**:* A psychological topic that was explored was time management of students. Students that study harder do not necessarily translate their difficult efforts to academic performance [17]. Prioritization is key to spend time appropriately with a given task. This task

[1]https://firebase.google.com/

prioritization should be included in our task management application. Specifically tasks can be broken up into urgency and importance to determine priority. A simple rule of thumb that was found was if it cannot be done in two minutes set a scheduled time for it or set it for later [17]. This kind of task scheduling or task priority placement can be represented in a priority queue with the high priority tasks ready to be taken out first in our application. Users will be able to complete tasks more efficiently this way.

6) ***Research in NoSQL and SQL Databases for Mobile Applications****:* Finally our research led us to the various tools and frameworks that could be used in our development of our application. We specifically look at databases and storage options for a mobile device that could pose issues. Because mobile devices are smaller they will have smaller resources and thus the processing power, memory, and storage are not the same as on a desktop computer. This is to say that no "hard disk techniques" can be used [18]. Exploring our options, we had determined we could use MongoDB and SQLite. The criticism against MongoDB, as it is a NoSQL database system, is whether or not a schema can meet the proper user requirements [18]. In other words, normalization of databases may be more difficult in terms of designing a database. MongoDB uses a JSON in terms of querying whereas SQL has its own version syntax for querying.

7) ***Firebase for Android development****:* Aside from these differences, a third option known as Firebase offers a storage solution on the cloud [19]. Firebase is backed by Google allowing fast cross-platform development [20]. For our application, Firebase may answer most of our needs while reducing complexity.

III. Proposed Project

A. Description of the proposed solution, approach, or methodology

The method we will use to develop the Pomodoro study application is to select proper frameworks for development. These frameworks shall include the front-end, the back-end, and a data storage solution. Because of this we have chosen to research React Native, Firebase, and SQLlite.

B. Delineation of major tasks

These major tasks can be separated through experimentation with React Native, Firebase, and SQLite. During experimentation of React Native we plan on testing how well animations will be implemented. Specifically we will make small prototypes of the trophies that could be earned in the application. React Native must be able to handle trivial tests such as fast rendering. In regards to a database solution, we will be testing SQLlite and Firebase in terms of speed in retrieving user data, ease with data migration, and accessibility. In order to add a local storage, the same tests would be applied to React Native's Async Storage library. The project should also contain the basics of a timer and will be developed in Javascript and React Native libraries. To create a gamified experience the Unity Game Engine may be incorporated within this project.

C. Description of the expected final deliverables

The project can expect to have task creation, task management, and task review while also containing vibrant animations, and an ability to customize. Task creation consists of being able to provide a name, a description, and the current date. Task management simply allows the user to rearrange the tasks in a priority queue and editing of a task if it has not been committed. Task review will include the ability to visit previous tasks so that the user would be able to reflect on their work with useful statistics such as time spent on a task or concentration level. The application should include energetic animations that provide a refined aesthetic that complement the apps functions. User customability is very important in this application as it should allow the user to customize the color palette of a task, the boards they work on, and the overall theme of the app.

D. Software and Hardware Requirements

The app will be run at least on Android phones, but as React Native is being used, it may be available for iOS. Software tools used to build this application will be a personal computer with access to a visual studio IDE, terminal access, either firebase or SQLlite, and javascript libraries. Any dependencies should be included in the packaged app.

E. Use Case Diagram

Upon looking at the user requirements, we have need to design a use case diagram to display the users abilities within the application. In **Fig. 1** we can see that the user has four major abilites within the app. Each of these bubbles are high level abilities a user can do. Specifically, the CRUD Timer and CRUD Task can be broken down into: creating timers and tasks, reading timers and tasks (the user will be able to clearly see the text they created), updating their timers and tasks they have created, and finally being able to delete their timers and tasks.

The Change Theme ability allow the user to change color of the application or task and timer UI display. The Customize Board ability will allow decoration of these boards, theming of the board, styling, and management of display of the board(s).

F. Project Timeline

The tasks fall into five sections: UI Skeleton, Database solution, full customibility of the app, completed animations, and fix remaining bugs to deploy to the Google Play store. **Fig. 2** shows an overview of what is expected in the development phase of this application. Each of these bubbles is approximately 20 hours in length. These are rough estimates and depending on external factors these may change during development with constraints such as time, software limitation, or hardware restriction.

IV. Testing/Evaluation

At deployment, the targeted group will be subset of students at Southern Adventist University. This subset of students could range from 10 to 15 students.

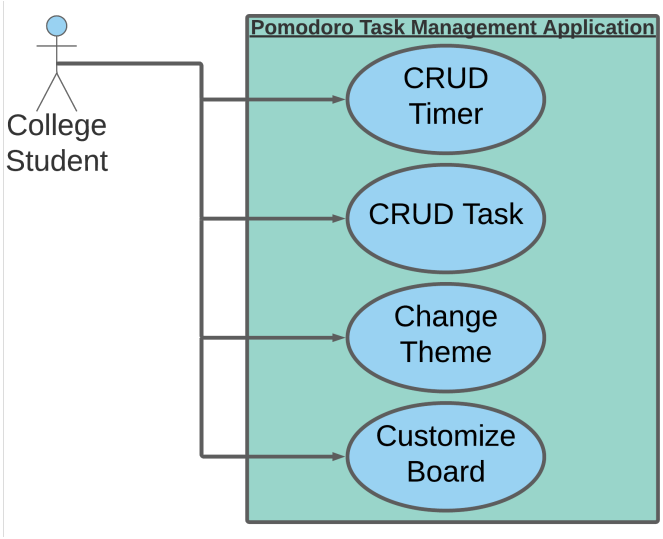

Fig. 1: Note that that CRUD will denote the ability to create, read, update, and delete operations–essentially it allows the user to manage with full capacity.

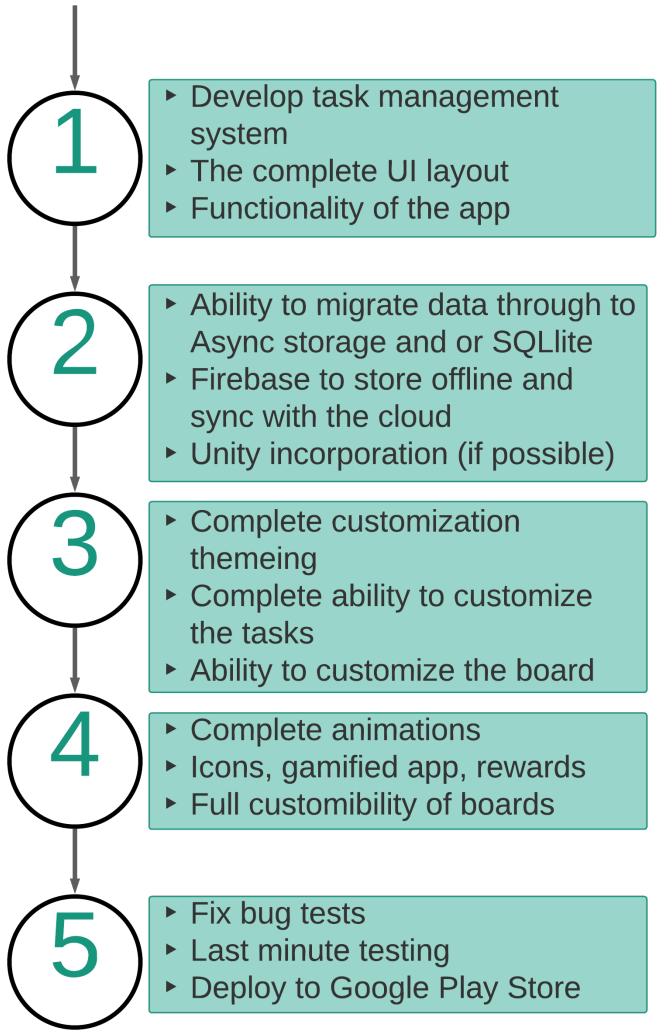

Fig. 2: Task management application estimated development timeline; approximately 20 hours each.

A. *Expected Outcomes Based on Project Requirements*

The expected outcomes on this project are users will be able to create customizable, editable tasks in which users can set specified times in which fluid animations are played. The app will have smooth transitions in the navigation portion of the app. The trophies–completed-tasks–will have a navigation page in which previous tasks can be viewed.

B. *Description of the Specifics*

There are five tests with their given acceptance constraint. These are the five different characteristics that the user will be test on:

1) **Number of misclicks**: 0-2 misclicks are acceptable
2) **Time to create a task**: time to create a task without a description should take no more than 5 seconds
3) **Time to get to goal state**: time to navigate through all navigation windows to the goal state in the app should take more than 30 seconds
4) **Steps taken to get to the goal state**: if n is the number of goal states it should take only n steps to get to goal state
5) **User retention**: User retention should include the user stays on the app for 60 seconds and interacts with the app at least every 5 seconds

Each of these constraints will be averaged on each of the 10 to 15 participants. That is to say the average of misclicks of these 10 to 15 college students is X. If X does not meet each of their corresponding constraints, such as $X > 2$ misclicks, then X, the average, must be repeated until it meets the given constraint.

C. *Method of Evaluation*

In its simplest form, these specifics could be evaluated with a timer, pen, and paper. A human actor, a college student, would use the app against these specifics where our team would count the number of misclicks and steps where time can be recorded through a simple digital timer on a smartphone. The secondary option is to use external software to count the misclicks and time such as maze.design.

V. Implementation

A. *Updated Problem Statement*

In order to become more inclusive of the proposed audience, a new problem statement was established. This extended problem statement's audience was broadened to handle anyone with desire to complete a task, aside from only college students. Essentially, every busy person would like to spend less time on repetitive tasks and complete tasks efficiently. Existing solutions fail by not effectively gamifying a task management application. Quite simply, they have been deemed monotonous according to the ratings of the Google Play store as seen in [12].

B. Determined Solution

The proposed solution, Gamified Task Manager, in this paper solves all of these issues. In terms of task management the solution enables the user to speed through any added tasks by the user. This task handling process occurs over various screens in the application which can be seen in the provided state diagram in **Fig. 3** Theses 4 screens represent high-level states in the Gamified Task Manager.

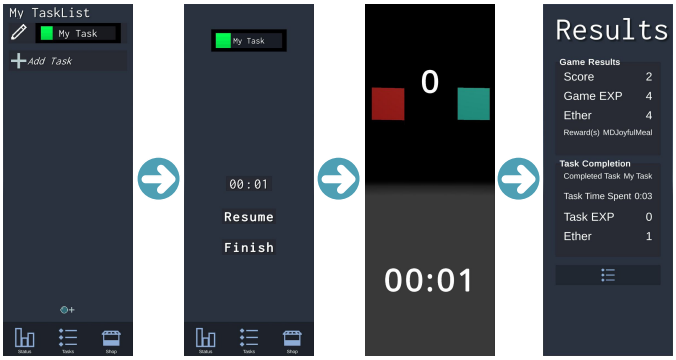

Fig. 3: Various screens that represent the flow of the architecture known as the Pomodoro Game Switch (PGS).

C. Technologies and Updated Method

Due to integration issues with Android Studio and Unity as a library[2] the technologies of this paper have changed. One reason for this is that building in gradle raised various issues. These issues include lack of proper dependencies as well as version inconsistencies. Another reason for this change in development tools is because Unity as a Library, at the time of development, was no longer actively maintained. Specifically, this tool, at the time of this work, was last edited in May, 2021. Aside from this lack of upkeep with the continuously growing versions of Unity, the tool had several inconsistencies in the documentation–specifically relating to gradle.

Instead of using Unity as a library inside of a React Native application, it was decided that Unity itself would handle all of the front-end. This proposed another issue: access to OS related functions of a user's device. React Native (and potentially Expo), as the name suggests, would have allowed OS level operations on Android and iOS devices. Since this work had to be migrated to Unity, various Unity packages were used instead to access OS operations such as time and date and notifications. One such package known as Voxel Busters Cross Platform Essential Kit enabled these abilities[3]. Another Unity related package that was utilized was the Graph and Chart package[4]. As the name suggests, this package saved time in creation of charts for screens of the application such as the status-screen.

For diagramming, Lucid Chart[5] was utilized for its flexibility in designing class hierarchy and dependencies. The greatest reason for this tool was the rapid ability to visualize the database connections with the C# scripts in unity. In turn, this allowed for the dataflow that was the backbone of the application.

This data was handled via Firebase's Real Time Database. Additionally Firebase provided authentication through Firebase Authentication. The Real Time Database is a NoSQL database that stores string data in the form of JSON. This database management system had reasonable documentation and support for Unity. In terms of Authentication, users are allowed to sign-in with Google–provided that the user has a Google account on their device.

For version control, GitHub was originally considered. However, Unity's built in source control system, Collaborate, was employed for its direct integration with Unity and ease of use. As of April of 2022 Unity has deemed Collaborate deprecated. Despite this pitfall, Unity allowed migration to Plastic SCM[6]–another source control system. This was the final choice for version control in this work.

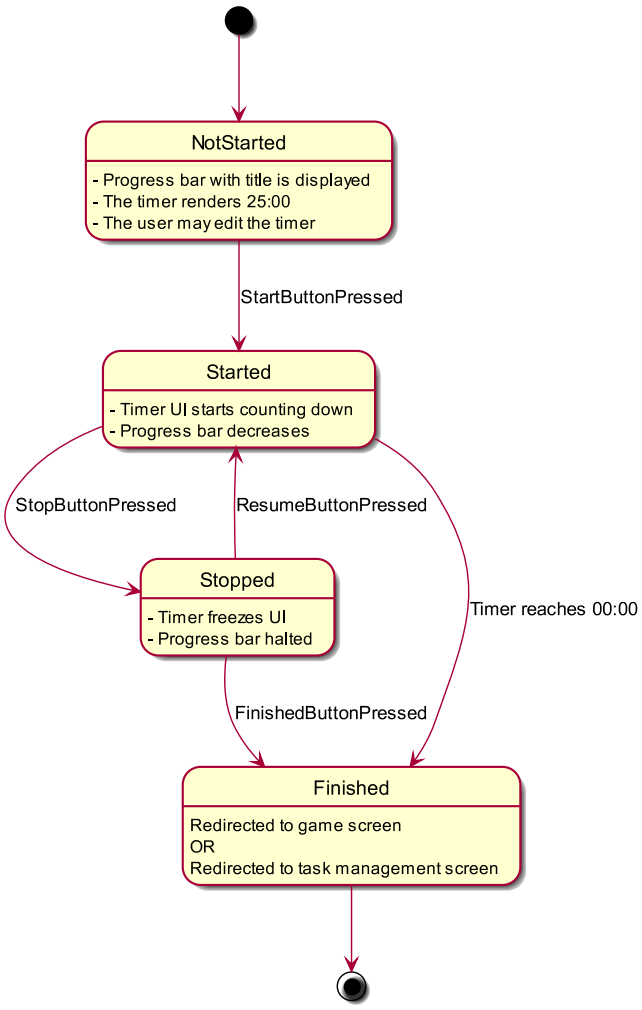

Fig. 4: The 4 states of that occur on the timer screen.

[2]https://docs.unity3d.com/2019.3/Documentation/Manual/UnityasaLibrary.html
[3]https://assetstore.essentialkit.voxelbusters.com/
[4]https://assetstore.unity.com/packages/tools/gui/graph-and-chart-78488
[5]https://www.lucidchart.com/pages/
[6]https://www.plasticscm.com/

VI. Results

A. Application Capabilities

The user may activate the task into its timer mode. Timer mode has 4 states as shown in **Fig. 4** the user can specify the duration of the task, which at default is set to 25 minutes. The user may then press a start button in which the timer counts down until reaching 0 minutes and 0 seconds or until the user presses a stop button. A progress bar denotes the remaining time over the initial time. The stop button pauses the timer in which the UI and back-end variables have ceased updating their values. At this state, the user may press either the resume button or the finish button. The resume button simply resumes the allows the timer values to be updated again as well as enables the progress bar to decrement. The finish button, on the other hand, sends data to a singleton class which contains a static field denoting user task data, specifically called kanbanData. Additionally, the finish button will redirect the user to the kanban screen and close the timer screen.

If the timer had reached 00:00, the user will be redirected to a mini-game screen. The mini-game screen is simply a Unity scene that has its own hierarchy and is distinct from the main application. Multiple "game scenes" have been created that each contain a mini-game, a 5 minute timer, and a score counter. These mini-games are typically managed by an instance labeled "Game Manager" but all use a "game timer script" that will end a game after the five minute game timer is ended. This game timer is started as soon as the scene is loaded and ends when the timer runs out. After the timer runs out the user is directed to a results screen.

Fig. 5: The task screen which displays the users task and game statistics.

This results screen receives the user's score according to the game and displays game related data related to the score. Additionally, the screen will display a tasks completed card in which user data related to the task completed is rendered. Finally, the user may press the "kanban section" to navigate back to the kanban screen where more processes may be viewed.

User data can be seen on the status screen, as represented in **Fig. 5**, which may be navigated via the navigation bar at the bottom of the screen. Screens such as the kanban screen and the status screen can be navigated easily for this. On the status screen, the user may view task history in the form of a time chart and game related data according to "level" and "experience".

The kanban screen is a Unity UI panel that holds the data of a task lists. The kanban has one tasklist available to users by default as well as an "add task list" screen. The kanban displays one task list at the full resolution of the devices screen at a time. However, the user may traverse the kanban by scrolling horizontally to a given task list. The "add task list" screen is always the last screen no matter how many task lists get added to the kanban. This "add task list" screen adds task lists by placing tasks at the previous index of the last element in the kanban, i.e. the "add task list" screen's index.

Like the kanban containing multiple task lists, each task list may contain 0 or more tasks (in Unity, these are called task cards). More tasks can be added via a "add task" button within the task list–similar to the kanban containing a "add task list" screen. These tasks are instantiated, every time presses the button, in a vertical format under each task list within a given window. If the number of tasks extend past the given window's mask then it will simply not be rendered unless the user scrolls vertically on the task list's window.

Fig. 6: The Task Card prefab that has two main buttons.

Each task has two buttons as seen in **Fig. 6**. The left most button on the task is the edit button which has a small graphic of a pencil. Tapping on this will allow the user to edit the task data of this task by rendering another screen known as the edit screen. The edit screen will allow text input and native OS time and date input in the form of a calendar. The user may then close the task editor in which the UI and data for the edited task will be updated and sent to Firebase. The second button of a given task displays the title of the text which is renamed after utilizing the task editor screen and a progress bar denoting the time spent on the task. If this button is pressed, the kanban screen will be deactivated and the timer screen will be populated with data.

In terms of data flow of the solution, various classes have been involved in sending and receiving transactions. A small portion of the application's data flow can be witnessed in **Fig. 7**. Data flow begins with the user authenticating with Google on the first screen. Data at this point is either retrieved from a previous session if the user has existed before or created for the first time. This data is simple information pertaining to the user's profile image, user name, and other identify-

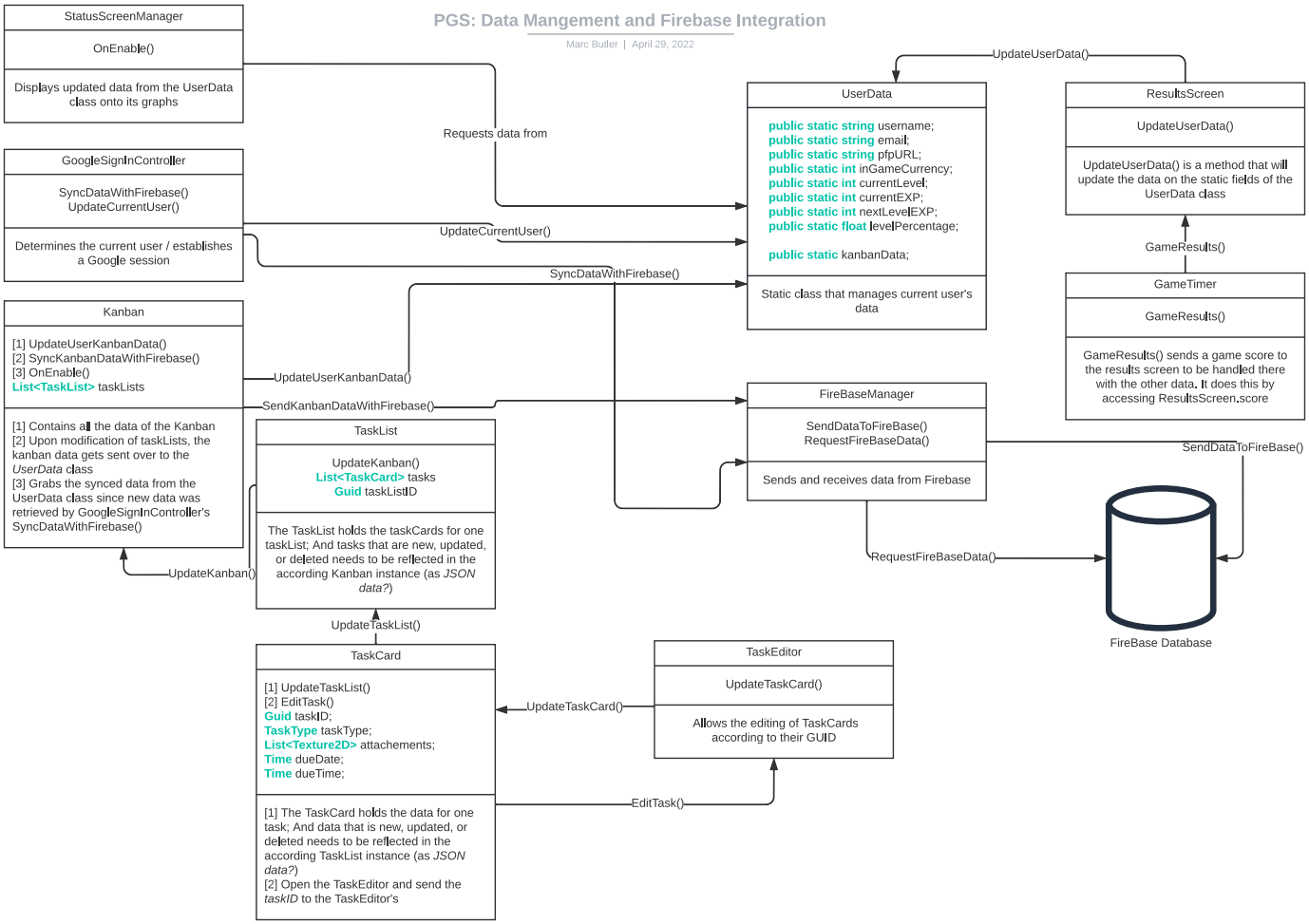

Fig. 7: The data architecture that shows the interfaces between various c# classes and firebase.

ing information. The GoogleSignInController class handles the bulk of this work through the UpdateCurrentUser() and SyncDataWithFirebase() methods.

FireBaseManager handles retrieving, posting, updating, and deleting user data–especially in the form of task data. It is through this module that the CRUD operations described in **Fig. 1** can be marked as fulfilled.

The UserData class that acts like a singleton to store user data that needs to be accessed by other classes. Furthermore, it contains static fields that can be interacted with via other classes. Most importantly, this class contains task data and other vital information of the user.

Mutliple classes such as the ResultsScreen, StatusScreen, TaskEditor, and Kanban rely on the UserData class. For instance, the UserData class's field for storing kanban data is modified whenever a user creates, updates, or deletes a task. Each task is represented by a TaskCard class which sends data up to a higher level of abstaction–specifically its parent TaskList. Similarly, each TaskList passes its self up to the Kanban. This is accomplished by serializing the data, placing it in C# lists, giving it to the UserData class, and finally posting these changes to Firebase's Real Time Database.

B. Outcome of the Specifics

Upon testing the description of the the specifics, a makeshift desk was established with a poster promotion that incentivised participants on the campus of Southern Adventist University with donuts to test the proposed solution. The poster provided students with information of the problem statement, the proposed solution, the methodology, and future work. Additionally, the poster supported a scannable, Quick Response (QR) code that enabled students with Southern Adventist University orgaization email with a 5 question Google Forms[7].

Google Form inqury consisted of the following questions:

1) Without any additional information, what do you believe this app was designed to do?
2) What features did you like?
3) On a scale from 1 to 5, where 1 is very difficult and 5 is very easy, how difficult was the app to use?
4) How effective, on a scale from 1 to 5 do you believe the app was at achieving the task it was designed to? (where 1 is not effective and 5 being very effective)
5) How responsive does the app feel? (on a scale from 1 to 5, where 1 is not effective and 5 being very effective)

[7]https://www.google.com/forms/about/

The total number of participants for this study was 11. Out of the 11 students that had participated in the study, only 9 of them had successfully completed the survey. Out of these 9 points of data 7 of the participants claimed that the application was designed to manage tasks. 4 of them had stated that their favorite feature of the application was the game or gamification, 3 of them had claimed the UI as a favorite feature–2 of them specifically appreciated the taskbar UI, 1 had mentioned timing as favorite feature, and 1 had stated they were uncertain of a favorite feature. In terms of quantive metrics, the highest rating in terms of application difficulty was a 5 indicating the highest amount of ease according to the survey. The lowest number was 2 which represented a moderately difficult application. Application effectiveness's highest rating was given a 5 by 1 participant, meaning that the application was highly effective, where the lowest scores were 3 given by 4 of the participants, denoting a moderate level of effectiveness. The highest score in terms of responsiveness was given a 4 by 5 students who had claimed an above average rating for responsivness of the application where the lowest score was given a 2 by one participant denoting a below average responsive design.

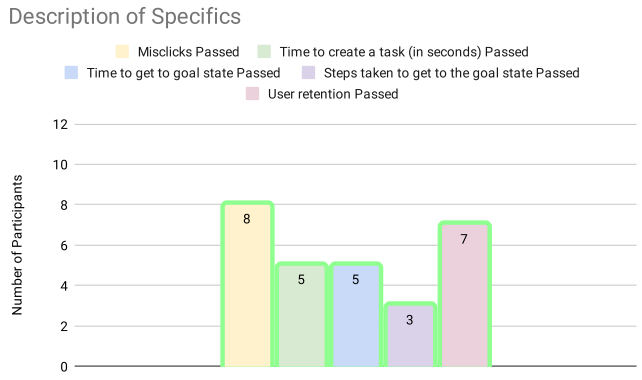

Fig. 8: The results of the study of 11 participants with 5 metrics.

Aside from the Google Form inqury, UI tests were provided as stated in Description of the Specifics subsection under the Testing/Evaluation section. A total of 11 student participants were selected in the study. Out of these 11, 8 of the participants passed the acceptable number of misclicks criteria. A full representation of this can be seen in **Fig. 8** as a yellow column. 5 of the pariticapants had managed to create a task in under 5 seconds which denotes either too strong metric requirements, the user was destracted during this phase, or the UI was not enticing enough to garner a interaction. A full chart can be seen in **Fig. 8** as a green column. 4 of the participants were able to get to the results screen–the given goal state–where the rest of the 7 participants had failed due to a timer bug issue. This issue was found and resolved as mentioned later in this paper's testing section. 3 of the participants had passed the steps taken to get to the goal state metric where the threshold was 6. One participant failed by exceeding the threshold with a value of 11. The other 7 failed due to inability to reach the goal screen. The threshold value, 6, was determined by accounting for the number of screens that may be navigated to in current build. This includes: the status screen, the task mangement screen, edit screen, timer, minigame, results screen. A complete chart can be seen in **Fig. 8** as a blue and purple column. 7 of the participants passed the user retention metric where the other 4 had failed due to distractions of outer sources.

C. Testing

Automated testing was performed via Unity Test Framework where manual testing was conducted in a spreadsheet. Because testing singleton classes from C# with Unity's Testing Framework proved to be inefficient. Nonetheless, the automated tests deal with intialization of the task management screen as various errors are were discovered here. These tests consist of Navbar, TaskCard, and Timer tests for our application. The Navbar test ensures that the correct number of elements are intialized under the Navbar prefab. The TaskCard tests ensure that TaskCards and TaskLists are initalized with the correct children. This is to say that the Task Lists should contain a single Add Task Button. Likewise the Kanban is ensured to contain the default number of Task Lists where one is the default Task List and the other is a Add Task List.

Provided in the Assets directory are manual tests that were recorded in an spreadsheet. The form consists of a high-level title of the test column, a expected result column, an actual result column, and if the test had passed. Various tests were recorded throughout the project timeline which ensured test driven development (TDD). These tests cover subjects of user interface (UI), data integrity, and CRUD operations.

D. Documentation

In order to build the Gamified Task Mangement solution, certain criteria must be met and steps must be done in the correct order. Installation of Unity Hub is required; the last version of Unity Hub used utilized is Unity Hub 3.3.1. Navigate to Unity's projects directory, which should be titled "UnityProjects", by default. This screen can typically be entered by pressing the drop-down of the open button in Unity Hub under the projects tab. Unzip the provided Unity project file and place it in this directory. A reload of Unity Hub may need to occur. The recommended editor version for the Pomodoro Game Switch is 2020.3.26f1. Which may be edited under the editor version column. If Unity crashes upon starting the editor, reloading the editor and attempting to load the project a second time will typically resolve the crash.

Upon loading the project the user should navigate to the TaskManagerScreen scene in the project window stored in the path of Assets/TaskManagerScreen. After double clicking to open the TaskManagerScreen, ensure that the game view is on a variant of portrait display by pressing the game window and selecting a portrait size under the resolution background. If Unity editor does not provide default portrait sizes, it is recommended to create a custom portrait size such as 1080px x 1350px.

In order to run the application, simply press the play button at the top-center of the screen. Whenever the TaskEditorScreen is enabled its prefab icon and text will appear at full opacity under the screen canvas in the project hierarchy. To disable and re-enable this, press this icon in the project hierarchy and make sure the "is active" check-box is unticked beside the title in the inspector window. This operation is crucial to close the TaskEditorScreen as in Unity's editor does not handle closing window in the game. This is to say that naturally disabling this TaskEditorScreen can only occur when built to an Android device as of now. For this the user would have to press the native back button on his or her Android device. Building to Android device is also required for the mini-game screens which only accept finger tap-input on Android devices rather than Unity editor mouse-clicks. Additionally, building to an Android device is required for syncing data with Firebase as Google Authentication is not supported in Unity's editor.

If a build to an Android device is absolutely necessary, a keystore is required and is held by the projects owner. Once the keystore has been acquired, navigate to File > Build Settings. Ensure that the selected platform is Android. A Unity restart may be required to install the packages needed to handle building on Android such as necessary SDKs. The SDKs can be installed via Unity Hub for a particular project. After the platform has been selected, navigate to Player Settings > Player > Publishing Settings. Enter the provided keystores and close the project settings window. Insert a USB cable– connecting an Android device. Finally, build with the "Build And Run" button.

Aside from building the application, various game scenes can be explored in the Assets/Scenes directory. C# scripts are found under Assets/Scripts, Unity prefabs can be found under Assets/Resources/Prefabs, and testing can be found in the Assets and Assets/EditorTests paths. Manual testing is in the Assets directory and requires a spreadsheet software to open the spreadsheet or alternatively be viewed as read only as a PDF. Automated tests found in the Assets/EditorTests must be run by activating the Navbar, TimerPanel, and KanbanPanel GameObjects. Unity Test Runner can be found in Window > General > Test Runner. All tests may be run at once or individually.

VII. Conclusion

After 100 hours of research, the work was adjusted to meet more general goals. This is to say that the problem statement was adapted to meet a broader group rather than college students that wanted to find an efficient method of studying. Consequently, a new solution and method for resolving the new problem statement was developed. These changes, in turn, resulted in a shift in technologies used in this work. Finally, upon developing the application, future plans to improve the application have been prepared.

Future work requires that an in-game shop which interacts with the user's in-game currency can be navigated from the navigation bar. Additionally, the status screen should have the capability to track the users tasks over weekly, monthly and possibly yearly periods. This data would be rendered on the TaskProgress graph. Other information concerning the Skills graph and HabitProgress should be researched and supplemented with an appropriate frontend and backend. Advertisements and other monentization options will be researched involving the user's in-game currency. Finally, publishing to the Google Play store will be followed by review from Google requirements.

References

[1] R. Nelson, *College classroom policies: Effects of "technology breaks" on student cell phone usage and grades*, 2020.

[2] N. Johannes, H. Veling, T. Verwijmeren, and M. Buijzen, "Hard to resist?: The effect of smartphone visibility and notifications on response inhibition," *Journal of media psychology*, vol. 31, no. 4, pp. 214–225, 2019.

[3] L. Scroggs, *The pomodoro technique*, [Accessed] on 12/6/2021, 2021. [Online]. Available: https://todoist.com/productivity-methods/pomodoro-technique.

[4] Seekrtech, *Forest - stay focused, be present*, [Accessed] on 10/13/2021, 2021. [Online]. Available: https://forestapp.cc/.

[5] P. DESIGN, *Habit forest - habit tracker, plans, goal tracker*, [Accessed] on 10/13/2021, 2021. [Online]. Available: https://play.google.com/store/apps/details?id=com.pionestudio.treeofhabit&hl=en_US&gl=US.

[6] I. HabitRPG, *Habitica*, [Accessed] on 11/9/2021, 2021. [Online]. Available: https://habitica.com/static/home.

[7] ©. D. Inc., *Todoist: To-do list & tasks*, [Accessed] on 11/9/2021, 2021. [Online]. Available: https://todoist.com/.

[8] AyagiKei, *Lifeup: Gamification to-do & tasks list | habitrpg*, [Accessed] on 11/9/2021, 2021. [Online]. Available: https://play.google.com/store/apps/details?id=net.sarasarasa.lifeup&hl=en_US&gl=US.

[9] T. Lozovyi, *Do it now: Rpg to do list. habit tracker. planner*, [Accessed] on 11/9/2021, 2021. [Online]. Available: https://play.google.com/store/apps/details?id=com.levor.liferpgtasks&hl=en_US&gl=US.

[10] O. Upon, *Pomodoro*, [Accessed] on 11/9/2021, 2021. [Online]. Available: https://play.google.com/store/apps/details?id=com.onceupon.pomodoro&hl=en_US&gl=US.

[11] A. Kolmachikhin, *Epic to-do list — rpg planner with reminders*, [Accessed] on 11/9/2021, 2021. [Online]. Available: https://play.google.com/store/apps/details?id=com.onceupon.pomodoro&hl=en_US&gl=US.

[12] ©. 2. lvluplife, *Level up life*, [Accessed] on 11/9/2021, 2021. [Online]. Available: https://play.google.com/store/apps/details?id=com.lvluplife.lvluplife&hl=en_US&gl=US.

[13] Z. Zainol and A. A. Ramli, "Let's study: Mobile tracker app for study group," *Global Business and Management Research*, vol. 10, no. 3, p. 1237, 2018.

[14] A. Visuri, N. van Berkel, T. Okoshi, J. Goncalves, and V. Kostakos, "Understanding smartphone notifications' user interactions and content importance," *International journal of human-computer studies*, vol. 128, pp. 72–85, 2019.

[15] E. Widnall, C. E. Grant, T. Wang, *et al.*, "User perspectives of mood-monitoring apps available to young people: Qualitative content analysis," *JMIR mHealth and uHealth; JMIR Mhealth Uhealth*, vol. 8, no. 10, e18140, 2020.

[16] K. Almalki, O. Alharbi, W. Al-Ahmadi, and M. Aljohani, *Anti-procrastination online tool for graduate students based on the pomodoro technique*, 2020.

[17] F. Bast, "Crux of time management for students," *Resonance*, vol. 21, no. 1, pp. 71–88, 2016.

[18] M. Fotache and D. Cogean, "Nosql and sql databases for mobile applications. case study: Mongodb versus postgresql," *Informatica Economica*, vol. 17, no. 2, pp. 41–58, 2013.

[19] A. K. S, *Mastering Firebase for Android Development : Build Real-Time, Scalable, and Cloud-enabled Android Apps with Firebase*. Birmingham: Packt Publishing, Limited, 2018.

[20] G. LLC, *Firebase*, [Accessed] on 11/9/2021, 2021. [Online]. Available: https://firebase.google.com/.