Software Design Specification

Version 1.0

Mental Health App

By:

Daniel Bornemann
Joshua Breininger
Phi Duong

Table of Contents

1. Introduction

1.1 Purpose

This document describes the design of the implementation of the system. We will use Java, Android's native coding language to implement a UI which the user can navigate to log and reference information. We also will use sqlite to create a relational database to store the user's information as well as retrieve it. Our goal is to make this interface streamlined and as easy to use as possible.

1.2 Scope

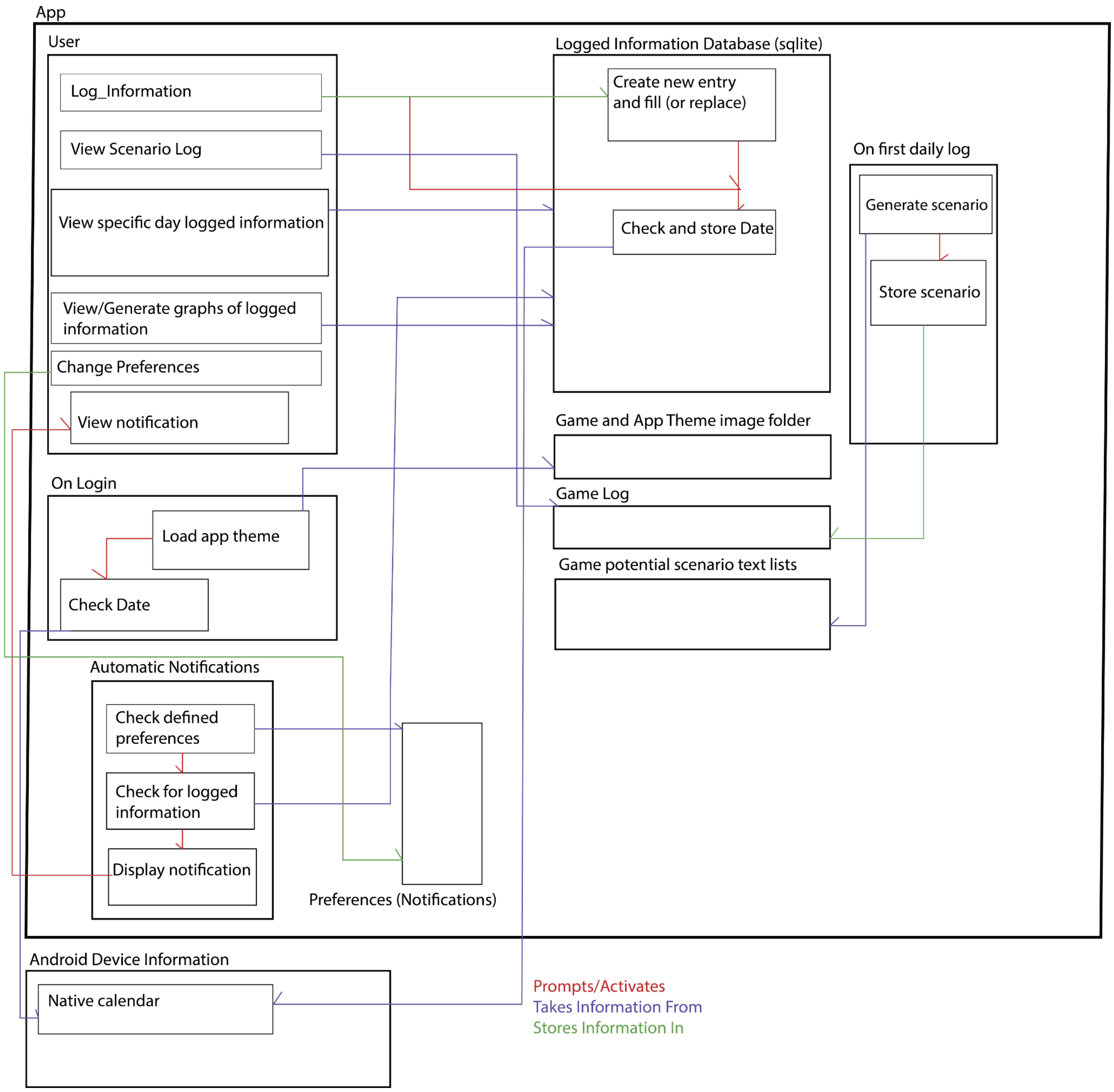
The project is that of an Android application where a user can log requested basic information about themselves, being diet, exercise and mood information. The app will provide graphical and direct ways to view previously logged information for the user, with the goal of providing some self monitoring functionality to individuals who are not motivated to keep track of and log detailed information about their diets and exercise habits. The implemented game to incentivize usage will also be simple and streamlined to not serve as a barrier in itself to quick efficient usage.

1.3 Definitions and Acronyms

- UI: User Interface. The graphical elements the user can see and interact with.
- Sqlite: Relational database system used for storing related information.
- Java: Object oriented programming language used by Android.
- Localdate object: A data structure supplying information about a saved date.

2. Architecture

2.1 System Architecture Diagram


App
User
Log_Information
View Scenario Log
View specific day logged information
View/Generate graphs of logged information
Change Preferences
View notification
Logged Information Database (sqlite)
Create new entry and fill (or replace)
Check and store Date
On first daily log
Generate scenario
Store scenario
Game and App Theme image folder
Game Log
Game potential scenario text lists
On Login
Load app theme
Check Date
Automatic Notifications
Check defined preferences
Check for logged information
Display notification
Preferences (Notifications)
Android Device Information
Native calendar
Prompts/Activates
Takes Information From
Stores Information In

2.2 Modules and Methods

2.2.1 General Functions:

Listed in this section are functions that are used in multiple modules with the same functionality.

- Check_Date():
  - Used to obtain the current date from the Android device. Obtains a Java time object and converts it into a localdate object for easier use.
  - Input: None
  - Output: Localdate object of the current date and time.

2.2.2 UI_Handler:

This module will be used to set up the app's UI by calling other modules with user input.

- generateTheme():
  - Sets a value deciding what theme the app will use based on the month, so each time a function that relies on the month is called does not need to parse the date again.
  - Input: Localdate object containing the date.
  - Output: An integer value corresponding to a month’s index.

- displayUI():
  - Contains the code used to visually set up and run the UI and buttons. Uses different graphics based on the month. It also calls the gameFirstDisplay() function.
  - Input: Integer value corresponding to the current month’s index.
  - Output: Integer representing success or failure/error. Display of UI.

- displayGameScenario():
  - Interacts with the Game_Daily_Update module to use its gameDisplay() function to show the last logged scenario.
  - Input: None
  - Output: Integer representing success or failure/error.
- checkNotifications():
  - Checks if there are any notifications to view, and displays any in a pop up based on information stored in the database, the date, and user preferences.
  - Input: Localdate object containing the date.
  - Output: Integer representing success or failure/error. Display of a pop up in the UI with information regarding the notification.

- logDailyInformation():
    - Calls Logging_Daily_Info to log user given information.
    - Input: None / Button press
    - Output: Integer representing success or failure/error.
- viewUserPreferences():
    - Calls the Preference_Handler for the user to view and change preferences.
    - Input: None / Button press
    - Output: Integer representing success or failure/error.
- viewGameLog():
    - Shows a pop up containing the list of all game scenarios seen so far by the user.
    - Input: None / Button press
    - Output: Integer representing success or failure/error. Pop up containing all scenarios in a text file holding all game logs.
- viewUserGraphs():
    - Calls User_Graph_Handler to direct flow to that module for graph UI.
    - Input: None / Button press
    - Output: Integer representing success or failure/error.

2.2.3 Logging_Daily_Info:

- This module displays three successive pop ups requesting user information and saves that information to the sqlite database.
- moodLog():
    - Creates a pop up asking for mood from one to ten using a slider.
    - Input: User input from slider, localdate object containing the date.
    - Output: Integer representing success or failure/error. Input saved into sqlite database.
- foodLog():
    - Creates a pop up asking for food groups the user has eaten that day with a choice button list.
    - Input: User input from list, localdate object containing the date.
    - Output: Integer representing success or failure/error. Input saved into sqlite database.
- exerciseLog():
    - Creates a pop up asking for exercise amount in minutes in a text box.
    - Input: User input from text box, localdate object containing the date.
    - Output: Integer representing success or failure/error. Input saved into sqlite database.

2.2.4 Game_Daily_Update:

- This module governs all information and UI elements associated with the game.

- gameGenerate():
    - Accesses the text lists containing game scenario elements and selects elements randomly to create a random scenario based on theme/date.
    - Input: Integer representing theme to select, and game scenario element text lists.
    - Output: Object containing string with game scenario and string paths of the graphical elements used in scenario.

- gameSaveScenario():
    - Saves scenario message, and paths of graphical elements to a file for Display.
    - Input: Object containing string with game scenario and string paths of the graphical elements used in scenario.
    - Output: Integer representing success or failure/error. Rewritten file of current saved scenario for display.

- gameLogScenario():
    - Appends the string (game scenario) along with a string of the date to the text log of scenarios experiences.
    - Input: Localdate object of the date, String of game scenario.
    - Output: Integer representing success or failure/error. Modified file of all scenarios.

- gameDisplay():
    - Sets up the graphics on the UI dedicated to the game, using the same scenario and graphics from the last time information was logged.
    - Input: File containing the string of the last scenario message, and string locations/paths of the graphical files used for the scenario's art.
    - Output: Integer representing success or failure/error. Displays last logged information date's game scenario in the UI.

2.2.5 Preference_Handler:

- This module shows the user's preferences in a menu and allows the user to edit them.
- displayPreferences():
    - Shows a menu with a list of preferences with editable text boxes filled with the changeable values.
    - Input: Current preference values from text file.
    - Output: Integer representing success or failure/error. Displays menu with preferences.

- changePreferences():

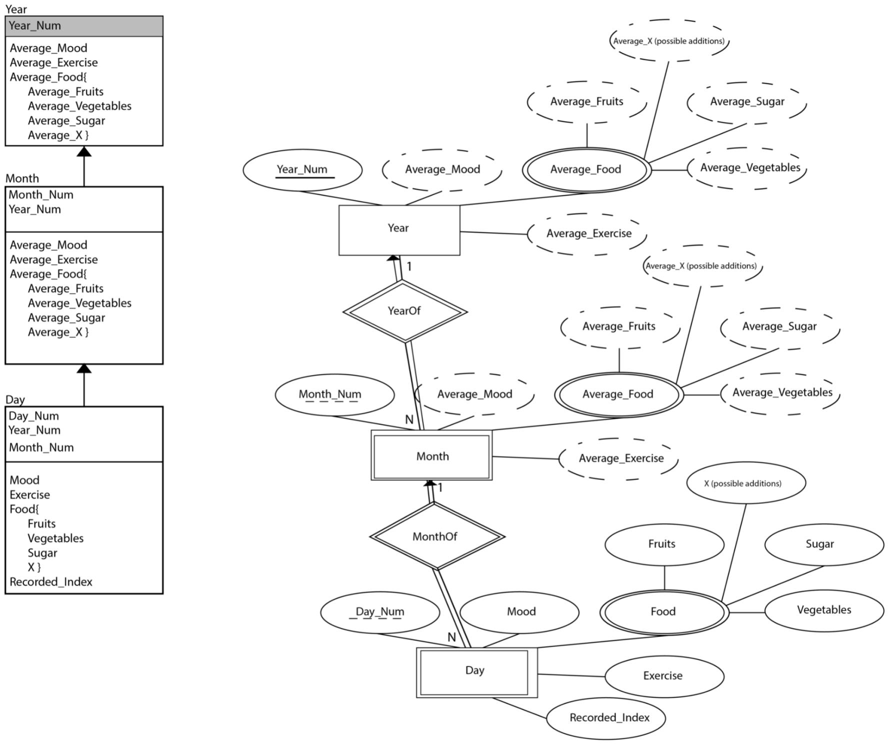
  - Edits the saved file preferences if the editable text boxes shown/run by displayPreferences() are edited.
  - Input: Values from editable text boxes.
  - Output: New preferences saved to file.

2.2.6 User_Graph_Handler:

- This module displays graphs associated with logged information based on UI input from the user.
- displayGraphMenu():
  - Shows a menu containing a default graph showing logged information with several toggle buttons that can show and hide different measurements (mood, food groups, exercise) and change axis unit sizes.
  - Input: Information from sqlite database.
  - Output: Integer representing success or failure/error. Displays menu with toggle buttons.

- refreshGraph():
  - Edits the graph based on toggle buttons.
  - Input: User input on buttons. Information from sqlite database.
  - Output: Displays new graph based on buttons pressed.

2.3 ER Diagram

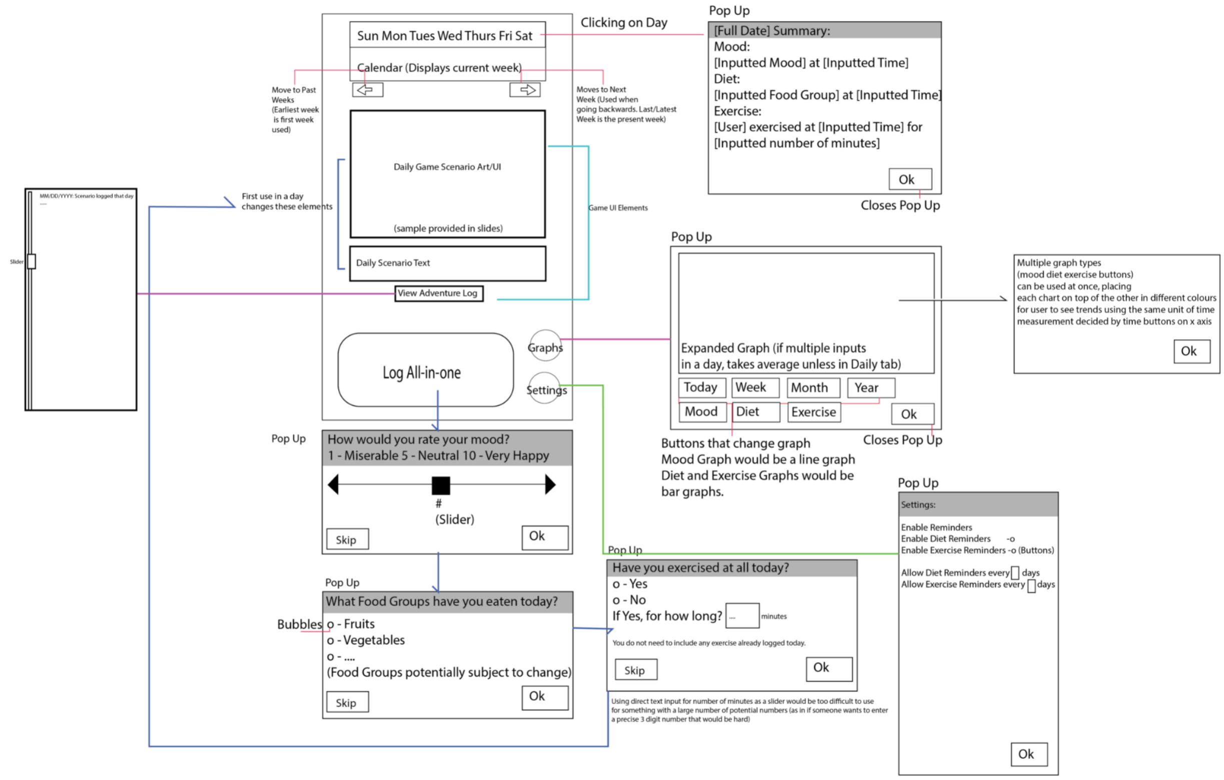


Recorded_Index is the "counter" of how many days have had information recorded for reference by reminders and the game.

The design of our database is made primarily to accommodate how information needs to be fetched by date. Since the graphs made for the user need to be able to know broad information and trends for the options with long periods of time being measured, we decided that using specific entities for the larger periods of time would be useful and save time while also not filling up too much storage space. The larger entities contain averaged information about the logged data in order to save running time so that when the data is being given to the user the calculations do not need to be done each time they request the information.

3 GUI

3.1 Full UI


Sun Mon Tues Wed Thurs Fri Sat
Calendar (Displays current week)
Clicking on Day
Move to Past Weeks (Earliest week is first week used)
Moves to Next Week (Used when going backwards. Last/Latest Week is the present week)
Daily Game Scenario Art/UI
(sample provided in slides)
Daily Scenario Text
View Adventure Log
Game UI Elements
First use in a day changes these elements
MM/DD/YYYY: Scenario logged that day
Slider
Log All-in-one
Graphs
Settings
Pop Up
[Full Date] Summary:
Mood:
[Inputted Mood] at [Inputted Time]
Diet:
[Inputted Food Group] at [Inputted Time]
Exercise:
[User] exercised at [Inputted Time] for
[Inputted number of minutes]
Ok
Closes Pop Up
Pop Up
Expanded Graph (if multiple inputs in a day, takes average unless in Daily tab)
Today
Week
Month
Year
Mood
Diet
Exercise
Ok
Closes Pop Up
Buttons that change graph
Mood Graph would be a line graph
Diet and Exercise Graphs would be bar graphs.
Multiple graph types (mood diet exercise buttons) can be used at once, placing each chart on top of the other in different colours for user to see trends using the same unit of time measurement decided by time buttons on x axis
Ok
Pop Up
How would you rate your mood?
1 - Miserable 5 - Neutral 10 - Very Happy
#
(Slider)
Skip
Ok
Pop Up
What Food Groups have you eaten today?
Bubbles
o - Fruits
o - Vegetables
o - ....
(Food Groups potentially subject to change)
Skip
Ok
Pop Up
Have you exercised at all today?
o - Yes
o - No
If Yes, for how long?
minutes
You do not need to include any exercise already logged today.
Skip
Ok
Using direct text input for number of minutes as a slider would be too difficult to use for something with a large number of potential numbers (as in if someone wants to enter a precise 3 digit number that would be hard)
Pop Up
Settings:
Enable Reminders
Enable Diet Reminders -o
Enable Exercise Reminders -o (Buttons)
Allow Diet Reminders every □ days
Allow Exercise Reminders every □ days
Ok

Our GUI is made to be relatively simple, although it contains many options for the user to obtain information. It should be clear to the user what each function does, and cycling through every functionality of the application should be smooth and relatively quick to not create any barriers to entry. Our log input pop ups are made to be easily accessible depending on the data types requested to further smoothen this process.

Notifications

Free to Use Stock Image

You have not logged eating any fruit for [] days in a row

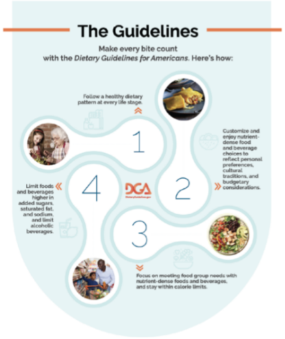
Ok

Free to Use Stock Image

You have not logged eating any vegetables for [] days in a row

Ok

You have logged eating high sugar foods for [] days in a row



The Guidelines

Make every bite count with the *Dietary Guidelines for Americans*. Here's how:

1 Follow a healthy dietary pattern at every life stage.

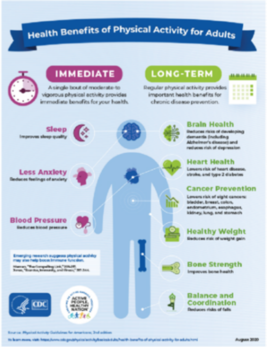
2 Customize and enjoy nutrient-dense food and beverage choices to reflect personal preferences, cultural traditions, and budgetary considerations.

3 Focus on meeting food group needs with nutrient-dense foods and beverages, and stay within calorie limits.

4 Limit foods and beverages higher in added sugars, saturated fat, and sodium, and limit alcoholic beverages.

Ok

You have not logged any exercise for [] days in a row



Health Benefits of Physical Activity for Adults

IMMEDIATE: A single bout of moderate-to vigorous physical activity provides immediate benefits for your health.

LONG-TERM: Regular physical activity provides important health benefits for chronic disease prevention.

Sleep: Improves sleep quality

Less Anxiety: Reduces feelings of anxiety

Blood Pressure: Reduces blood pressure

Brain Health: Reduces risks of developing dementia (including Alzheimer's disease) and reduces risk of depression

Heart Health: Lowers risk of heart disease, stroke, and type 2 diabetes

Cancer Prevention: Lowers risk of eight cancers: bladder, breast, colon, endometrium, esophagus, kidney, lung, and stomach

Healthy Weight: Reduces risk of weight gain

Bone Strength: Improves bone health

Balance and Coordination: Reduces risks of falls

Emerging research suggests physical activity may also help boost immune function.

Ok

graphic provided by dietaryguidelines.gov

https://www.dietaryguidelines.gov/resources/downloadable-graphics

graphic provided by cdc.gov

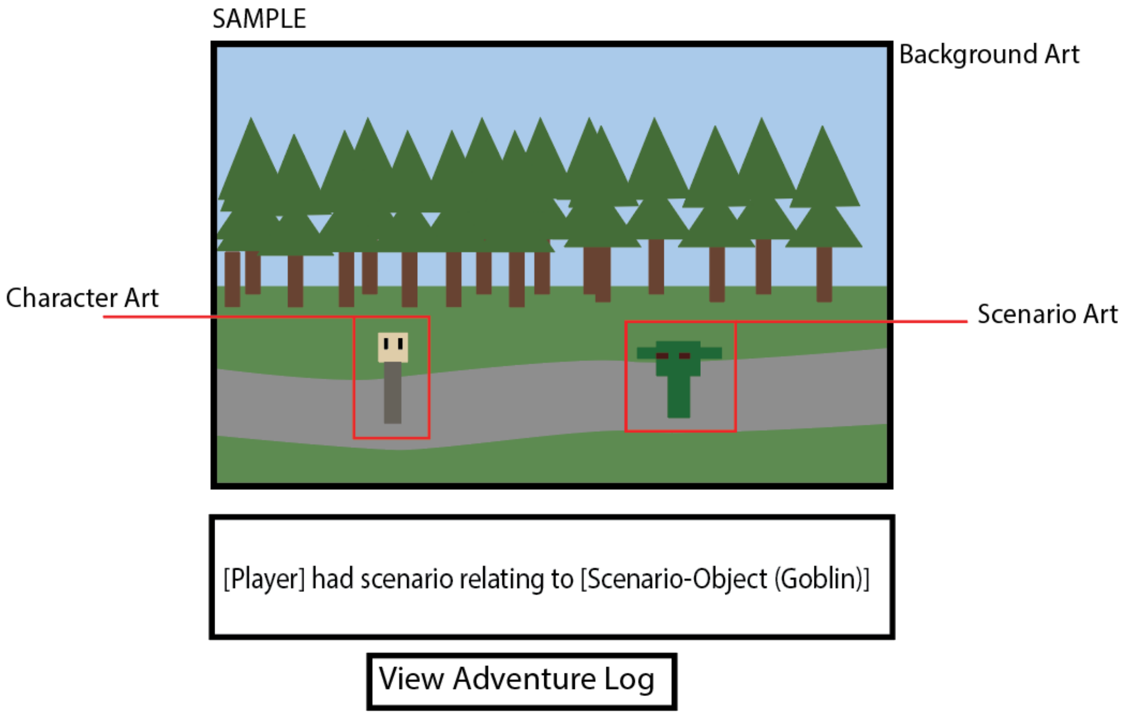
https://www.cdc.gov/physicalactivity/basics/adults/health-benefits-of-physical-activity-for-adults.html

Note: If copyright is an issue then can be replaced, although as they are obtained from .gov it should be fine as it appears that government documents/websites/etc are public domain.

We likely would add more notifications depending on the end food groups chosen, currently focusing on vegetables/fruit and sugary foods due to our Reference article.

Note: Our notifications don't focus on portions or specifics as that would require logging much more specific and lengthy information, and there are many apps that can do this as if someone was motivated to log their diet that extensively they likely would just use those. Increasing the specificity of the diet or exercising logging would increase the barrier of usage, and we want the reminders to still be useful/clear to someone not motivated enough to track and log the specifics of their diet.

3.2 Sample Game UI



[Player] had scenario relating to [Scenario-Object (Goblin)]

These UI samples are examples of what the end UIs would be like functionally, and are not representative of the graphical quality and visuals.