User/Developer Manual

Version 1.0

Journal Hero App

By:

Daniel Bornemann
Joshua Breininger
Phi Duong

Table of Contents

1. User Manual

1.1 Disclaimer

This app, although intended to help self monitoring and to push for good health choices is not a replacement for mental health professional help and does not claim any medical authority. Please seek professional and qualified mental health help if you find yourself struggling, and be aware of the limitations of this app and its intent.

1.2 Controls

All interactions in the app can be done using the Android touch screen. The app is implemented with this in mind as the user controls, so traversing the app should be done with the buttons presented rather than sliding or scrolling.

1.3 Main Menu

1.3.1 Calendar

There is a calendar that shows the current day of the week, where by pressing arrows on the week can let you select a different day. Clicking on the calendar window itself will open a pop up containing a full monthly calendar which also can select a day. Selecting a day will place it as the current day in the calendar window at the top of the main menu. With this, you can then click on the adventure log button to view what input was placed on that day by the user. However, any day that isn't the current day cannot be edited with the all in one button.

1.3.2 Background

The background of the app changes every two months to one of six backgrounds, swapping in the same order. There also is a visual of the player character in the background, which remains unchanged.

1.3.3 Adventure Log

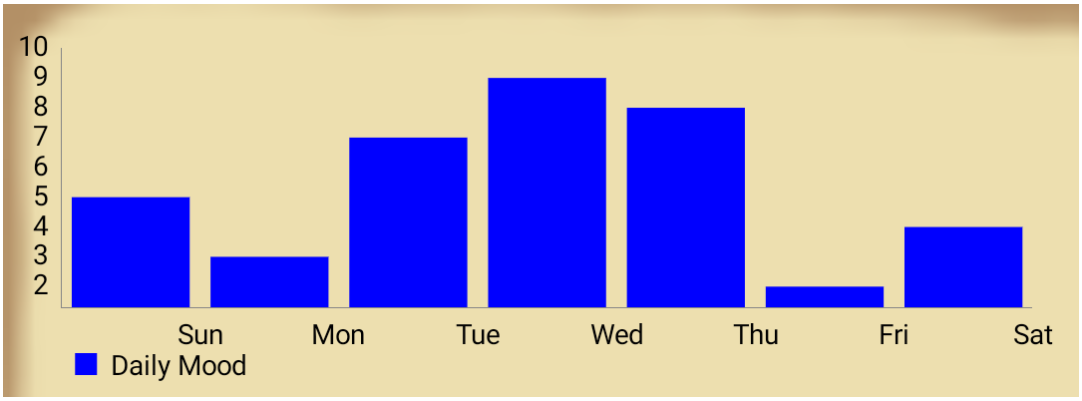
Opening the adventure log via its button creates a pop up screen for whatever day is selected in the calendar and prints out what has been inputted for it. This by default is the current day, but provides a way for the user to see whatever has been inputted for a day or the current day in case they need to edit it.

1.3.4 Settings

The settings menu presents a few text boxes which allows for the user to decide how many days in a row no exercise, no vegetable, no fruit or the presence of high sugar foods in logged input the app will wait before triggering a reminder about each topic. The notifications are sent using the native Android notification system and should be dismissed in the same way. Each text box opens the Android number pad, with a single digit given. A 0 will outright disable the relevant setting, being fruit, vegetable, sugar, and exercise respectively. To edit the settings, click on the respective text box containing for example “Edit Fruit.”

1.3.5 Graphs

The graphs button creates a pop up containing a daily, weekly, monthly and yearly graph of mood, exercise, and diet logged information. These graphs take the shape of bar graphs and can be examined by the user using the pinch functionality on the Android screen to zoom further into the graph to view it closer. Through buttons, the user can swap between viewing the graphs.



1.3.6 Log All-In-One

The log button creates three separate pop ups that appear one after another. The first pop up asks for the user to rate their mood from one to ten. The user can input this number via a slider, where they press the circle on the line and slide their finger to place it to what number they like. A small indicator appears to state what number the circle currently represents where it is. This data is stored into a database for future reference for the user. The user has an “ok” button to save that data, or a skip button to refrain from inputting mood data for that day. By default the mood value is five, and if skipped the number is entirely absent and does not participate in the graph or averages calculations.

The second pop up asks the user to log what food groups they have eaten that day. Like the first pop up, it contains ok and skip buttons with the same functionality, along with the third pop up. This pop up has three circles along with a label and a small image depicting what the circle represents - fruit, vegetables, and high sugar foods. The user should click the circle to designate which of those food groups they have eaten that day. The user can also click a circle to remove the indication that they ate that group if they change their mind.

The final pop up asks the user to log the number of minutes spent exercising that day, again with the option to skip doing so. The input here is to click on a small text box, which pops up with the native Android number pad only keyboard. In order to prevent extremely massive numbers which could impact the database, only three digits are allowed. The maximum value of three digits is well over what a human could exercise for in a day, so there shouldn’t be a reason to go above that. Once the final pop up is closed, the app transitions to the game and scenario section.

The All in One button can be repeated as many times as the user wishes - it will replace the current day's logged information if it is already stored. Skipped values will not overwrite previous submissions of that day, so if you skip on your second log each value will be the same

as the original log. Using the All in One button a second time in the day will not generate a new scenario however.

1.4 Scenarios

The first time the user logs information in a day, a “scenario” appears to the user. The player character is depicted in a new screen with some kind of game art representing an entity or object, and text describes some kind of scenario that occurred between the player character and the object, with the intent of being something to read every day to create the effect of the player character experiencing a daily life while the user does. These scenarios are randomly selected, although new scenarios become available depending on the user’s input. The scenario is also logged into the adventure log for future reading. Scenarios can be repeated if they are randomly selected again.

1.5 Game

The battle system is accessible from the main menu via the clashing swords button in the center of the screen. On the screen displays the player art and one of the 9 enemies, along with a blue bar above the player and a red above the enemy. The blue bar represents the player's mana, and the red bar represents the enemy's health. There also is a yellow bar and number next to the player denoting how close they are to leveling up and what their level is. After a user uses the all in one log button, the day "refreshes" and the user receives a turn to do damage to the enemy. They can either use magic, which uses up all their mana, or a basic attack. Magic attacks do more damage than basic, however each enemy type has different resistances. One enemy may take less magic damage than another, so the user can make decisions and prioritizations based on what they are battling. When their turn is up, the buttons to attack vanish until a day passes and they use the all in one again. Defeating an enemy presents the user the next time they click on the battle screen a new randomly selected enemy, as well as granting exp. Mana is recovered gradually by logging each day and viewing a scenario. Damage buffs or an extra turn are given if the user logs information that include good choices, like exercising or eating fruits/vegetables. No punishments are given for any kind of input, and no rewards are given for what the user inputs for mood. There is a button with a book as its icon that when pressed moves to the last scenario shown to the user.

1.6 Notifications

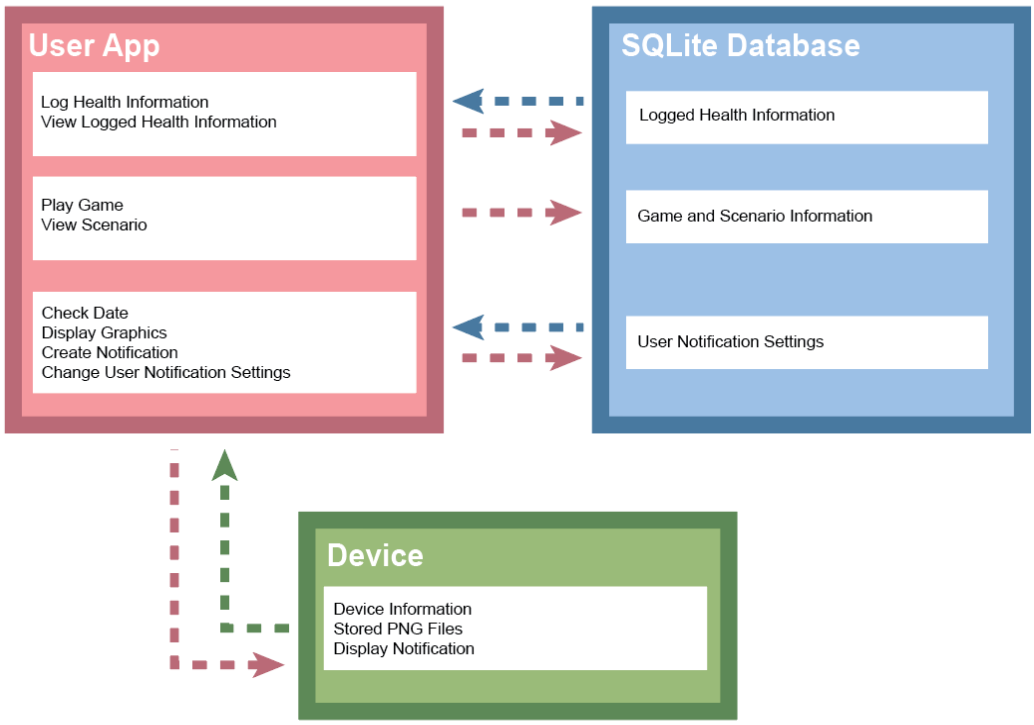
Depending on the logs given by the user, the app may send notifications through the Android notification system to suggest good choices. These can include neglecting to exercise, eating fruits and vegetables too infrequently or eating high sugar foods too frequently. The number of days that the notification considers a trend can be edited by the user, or disabled outright in the Settings menu if the user finds them too pressuring. These notifications act as general suggestions rather than strict advice, and can be dismissed through the native Android system as well.

2. Developer Manual

2.1 Tools Used

The app is written primarily in Java with XML files for the UI. It is made for the Android Pixel 3 so formatting errors may occur when being used in devices with different screen sizes.

2.2 System Architecture



On the userland of the system are the various menus and buttons. Most of the system is built here, as the only necessary backend is the SQLite database. Some interaction with the actual device is made, as the png files that contain the various graphical elements in the app are stored directly in the filesystem. There also is some checking of datetime information from the device itself as well as interactions with the Android notification system to display reminders. In terms of the database, all values subject to change are stored inside of it along with enemy stats and logged information. More detail on the database is covered in the database section. Since the SQLite database already existed for logging information, the developers preferred to store information in the database in special tables rather than directly in the filesystem, as the database should be faster while also being small enough in size, as not many bytes are required for a small amount of attributes of very few rows.

2.3 Source Files

2.3.1 Java Files

MainActivity.java

This file serves as the main for the entire program, acting as the driver for the app. MainActivity includes the initialization of the app in the onCreate() function, handlers to switch to and from xml files, which include all of the popups, and is heavily connected to the DataBaseHandler.java file.

DataBaseHandler.java

This file contains all of the SQLite statements including the setup of the database and main different SQLite methods which are used to insert and retrieve data from the database.

2.3.2 XML Files

activity_main.xml

This file contains the xml for the start screen of the app. Some key components are the calendar, adventure log button, log-all-in-one button, graphs, settings, and the battle button. An important thing to note is that all background art is present and their visibility is set programmatically in MainActivity.java according to the current month.

popup_adventure_log.xml

This file contains the daily information that was logged for the current day as indicated by the calendar on the home screen.

popup_adventure_scenario.xml

This file contains the randomly generated daily scenario that is presented to the user after logging their daily information.

popup_calendar.xml

This file contains the fullscreen view of the calendar that is displayed on the home screen of the app.

popup_disclaimer.xml

This file contains the popup that is shown when the app is started which is a disclaimer stating that the creators of the app are not medical professionals and a phone number to the National Suicide Prevention Lifeline.

popup_exercise.xml

This file contains the popup for entering the number of minutes exercised into an EditText. The input is limited to a numpad and 3 characters.

popup_food.xml

This file contains the popup for entering food groups eaten that day, including fruits, vegetables, and high sugar foods. The inputs are radial buttons that can be checked or unchecked. There are 3 hidden radial buttons corresponding to each of the 3 inputs which are used to control the checking and unchecking.

popup_graphs.xml

This file contains the popup for displaying the graphs for the day, week, month, and year for mood, exercise, and food types.

popup_mood.xml

This file contains the popup for entering the mood on a scale from 1 to 10. The input is a slider.

popup_settings.xml

This file contains the settings for the phone, including reminders for eating fruit, vegetables, not eating high sugar foods, and a reminder to exercise. These settings default to 5 days and can be disabled by entering 0 for them.

2.3.3 Miscellaneous Files

manifests

This folder contains the Android manifest that is used by Android for metadata regarding the app.

java

This folder contains all Java code used in the app.

res

Contains folders of different images used in the app.

raw

This folder contains the images used throughout the app including backgrounds and icons.

Grade Scripts

This folder contains the gradle scripts that are required to build the app.

2.4 Methods/Functions

public void openCalendarMenu()

This method opens the calendar popup (popup_calendar.xml). This method is called when the calendar is pressed on in the main screen.

public void openDisclaimer()

This method opens the disclaimer popup (popup_disclaimer.xml). This method is called when the app is started.

public void openMoodMenu()

This method opens the mood popup (popup_mood.xml). This method is called when the log-all-in-one button is pressed on the home screen.

public void openFoodMenu()

This method opens the food popup (popup_food.xml). This method is called after the mood menu is closed.

public void openExerciseMenu()

This method opens the exercise popup (popup_exercise.xml). This method is called after the food menu is closed.

public void generateScenario()

This method opens the adventure scenario popup (popup_adventure_scenario.xml). This method is called after the exercise menu is closed.

public void openSettingsMenu()

This method opens the settings menu popup (popup_settings.xml). This method is called when the settings button is pressed on the home screen.

public void openGraphsMenu(boolean addPopup)

This method opens the graphs menu popup (popup_graphs.xml). This method is called when the graphs button is pressed on the home screen. The method takes in addPopup as a boolean which is used to decide if the user is initially opening the graph menu popup.

public void loadDayBarChartData(String setting)

This method loads daily bar chart data into the bar charts in the graphs menu. This method is called in the openGraphsMenu function. The method takes in “setting” as a String which is used to determine what data is displayed. The data is then put into a graph that displays the inputs for the week.

public void loadWeekBarChartData(String setting)

This method loads weekly bar chart data into the bar charts in the graphs menu. This method is called in the openGraphsMenu function. The method takes in “setting” as a String which is used to determine what data is displayed. The data is then put into a graph that displays the average weekly inputs for the month.

public void loadMonthBarChartData(String setting)

This method loads monthly bar chart data into the bar charts in the graphs menu. This method is called in the openGraphsMenu function. The method takes in “setting” as a String which is used to determine what data is displayed. The data is then put into a graph that displays the average monthly inputs for the year.

public void loadYearBarChartData(String setting)

This method loads yearly bar chart data into the bar charts in the graphs menu. This method is called in the openGraphsMenu function. The method takes in setting as a String which is used to determine what data is displayed. The data is then put into a graph that displays the average yearly inputs for the last 7 years.

public void openAdventureLog()

This method opens the adventure log (popup_adventure_log.xml). This method is called when the view adventure log button is clicked on the home screen.

public void updateCalendarView()

This method updates the calendar’s week label to match the current date or the date that is selected. This method is called when the app is started and when the calendar is clicked on.

public void onCreate(Bundle savedInstanceState)

This method opens the home screen (activity_main.xml). This method is called when the app is opened and uses savedInstanceState to hold information that was previously stored in the app. All notifications are handled in onCreate since they are displayed when the app is started.

private void createNotificationChannel()

This method registers the notification channel with the app, allowing for notifications to be triggered. This method is called in the notification section of onCreate.

protected void onResume()

This method resumes the app from a paused state. This method is called when the app is resuming from a paused state (not in foreground).

private void saveMood()

This method saves mood information to the database. This method is called in the openMoodMenu method when mood information needs to be saved.

private void saveFood()

This method saves food information to the database. This method is called in the openFoodMenu method when food information needs to be saved.

private void saveExercise()

This method saves exercise information to the database. This method is called in the openExerciseMenu method when exercise information needs to be saved.

private void saveSettings()

This method saves settings information to the database. This method is called in the openSettingsMenu method when settings information needs to be saved.

public void updateDateTime()

This method updates the calendar to the current date. This method is called in the onCreate method.

public void changeDayByIncrement(int incr)

This method updates the current day on the calendar by the integer incr. This method is called in the onCreate method.

private Integer getGraphColor(String type)

This method returns the color of the graph displayed in the graphs popup depending on what type it is (mood, exercise, etc). This method is called in every bar chart related method.

2.5 Database Summary

The database contains all logged information and all information generally subject to change, such as current health in an enemy. Each table has some kind of id, either through specific attributes or a special attribute made specifically for that purpose. Day table and month table information is removed from the database when a year rolls over to prevent bloating.

Player

This table contains relevant information about the player in the context of the battle system. The current exp count, the amount of mana they currently have as well as the max, how much damage they do both normally and magically, as well as the number of turns they have access to at the current moment.

Enemylisttable

The enemy table contains information about each type of enemy. Each enemy has health, max health, identifying information like name and id along with its magic and physical resistances. A positive resistance decreases damage taken in that damage type, and a negative increases damage taken.

Daytable

This table is the foundation of the logged information part of the database. Its key is the day number, month and year. It also contains the logged information about a single day - how much exercise was done, if high sugar, fruit and vegetables food groups were eaten, and what their mood rating was.

Monthtable

The month table contains average information and has its month number and year as a key. It also contains averages, like exercise, diet attributes and mood, along with the counts for each value so that the mean can be properly calculated, while skipping days where no information is logged.

Yeartable

The year table is the last table, and functions similarly to the month table except for the broader context of a year. The table's only key is the year number, and contains just the averages of that year along with their counts.

2.6 Expanding

Being limited by time, we could not fill the game and scenarios with as much content as we would have liked. Scenarios, being some text associated with the image, can be fairly easily be added into by inserting a row containing the relevant information and scenario into the scenario table in the database. It would then be a part of the scenarios list immediately, and could be called randomly when a scenario triggers. We only could put in a limited amount of scenarios due to the time it takes to write some kind of interesting prompt, so expanding the app could be as easy as adding a new png to the right folder and those insert statements. Expansions to the game could be to add new kinds of attacks - we considered adding a "stamina" heavy physical attack to have a second meter like mana but does physical damage, so a player could optimize damage against an enemy. Alternatively, enemy's could be given a turn per day to create a true battle system. To do so, one would need to add rows to the player table to give damage values and meter and the meter's max to save the new attacks, while adding a new button to trigger that attack. Adding new enemies is relatively simple and would require a new row in the enemy list table with its stats, along with new art.