README FILE

Maya Harvey, Gaya Tarcar, Jonathan Affeld, Janelle Rudolph
CS 147 Autumn 2022

Prototype Link

Please download the .zip file from the website to obtain the prototype files. To see how to build, please check Operating Instructions.

General Overview

The lucIDLy app is intended for use at any time, but at least once daily, as users are required to complete the *Daily Survey* that can be accessed from the home screen. In addition to completing the *Daily Survey* questions, users are able to access trend-over-time data and visit available friends in this prototype. Users will also be able to obtain rewards from checking in and observe how the app might display specific health insights.

Design Tools

We constructed our high-fidelity prototype using React Native, React, and Expo Go. Since lucIDLy is an app with a mobile format (even when accessed from the plush toy's screen), we were able to develop an iPhone UI for lucIDLy with React

Native. This gave us the advantage of an easily and realistically testable prototype. All graphics for the prototype were constructed inside Figma using shapes and text and imported into React Native.

Though lucIDLy is intended to be for use with all mobile platforms (such as Android, etc), we only modeled for an iOS platform with React Native. This means that users of different mobile operating systems might not experience a 100% accurate prototype to what their use of the app would really look like.

Operating Instructions

Note -

To operate, download the .zip file and unzip the package. In the Terminal, use the cd command to navigate into the package, then run npm i, then npm start. You will need to have node installed already. This will build the prototype.

Users are able to return to the Garden/Home Screen at any time by clicking the upper left bear icon.

Login/Onboarding

- Upon loading, users must input a username and password.

Garden/Home Screen

- Click the *Garden Inventory* button at the bottom of the screen to view user inventory, a scrollable page.
- Click the *Daily Survey* button in the top middle of the screen to begin Task 1 and complete the Daily Survey.
- Click the small button of a graph to the left of the *Daily Survey* button to begin Task 2 and view health trends over time.
- Click the small button with user silhouettes to the right of the *Daily Survey* button to begin Task 3 and interact with friend users.

Task 1 - Record one's current mental and physical well-being

- Click the *Daily Survey* button in the top middle of the screen to begin Task 1 and complete the Daily Survey.
- After interacting with the questions on the screen by clicking the buttons prompted by the text, click *Next* on the bottom of the screen to progress or *Back* to go to the previous page.
- Upon completion of the survey, users will be presented with a reward screen and

prompted to click the upper left home button.
- User answers are stored and not refreshed upon clicking daily survey again to preserve realistic functionality.

Task 2 - Compare and track mental and physical wellbeing over time

- Click the small button of a graph to the left of the *Daily Survey* button to begin Task 2 and view health trends over time.
- Users can view two hard-coded graphs for physical and mental health.
- Users can scroll to view a hard-coded summary of survey answers from previous days that updates based on survey answers.

Task 3 - Make others aware of how one is feeling

- Click the small button with user silhouettes to the right of the *Daily Survey* button to begin Task 3 and interact with any of 3 friend users with varying statuses.
- Clicking the *pig* icon will take users to an invitation to visit the *pig's* garden. Users may accept or decline the invite.
- Accepting will take users to the pig's garden, where users can return back to friends or click the upper right chat icon to view the chat with the pig.
- Declining will return users to the main friend page.
- Doing either will mark the invite as read, eliminating the invite from the main friend page.
- Clicking the *bunny* or the *koala* icon will take users to the *bunny's* or the *koala's* status page, where users can click a button to invite them over, or click the other button to ask to visit.

Limitations

For our Hi-fi prototype, we were unable to implement every single small feature of the app, despite implementing all the tasks' main functionality.

We did not implement the harvest functionality, as we determined this did not contribute to the tasks we prioritized in our prototype. Instead, the only collectible users are able to view is the coin collection page upon finishing the survey.

In addition, users are not able to physically submit chat text that they have typed into the chat page, though they are able to type and draft out a message.

Meanwhile, on the health-over-time screen, users are not able to view a live graph based on their real-time survey answers - the graph is a hard-coded image. In addition, we decided to remove the specific health insights pages as we determined that it was not directly conducive to our second task of viewing health trends over time.

On the world-view screen where users are able to view friends and their statuses, the user is unable to actually go see the bunny and the mouse's gardens, only ask to be invited to visit or ask them to visit. The only garden available to visit is accessible from the pig's invite message. Furthermore, once at the pig's garden, users are only able to view the garden and return to previous pages or to access the chat feature, not explore the garden further.

The bear screen where users are able to tap locations of discomfort has only the hardcoded stomach option available to tap.

A couple buttons lack functionality and pages that they redirect to, as they do not directly contribute to the task performances. For example, the button to invite contacts does not lead anywhere and is just in the background as an example.

Wizard-of-Oz

Health Insights/Graph: Due to the lack of historic use, there is no past data to draw from - as such, the prototype 'magically' analyzes supposed past health data to give the user a graph based on their history, without demonstrating how this occurs.

Rewards: Since there is no past user data to demonstrate the 'progress' that this user might have made as reasoning for why they are receiving specific rewards, we have acted as the algorithm that will determine the quality and quantity of rewards that the user receives from checking in.

Friend Status: Users are unable to manually set their own status. However, on the world-view screen, we are still able to view other friends' statuses even with skipping this step. This is so that users can understand the results of this feature and how it might impact friend visit requests without having to do anything on their end.

Login: Upon opening the app, users are prompted to input a username and password to create an account. This information is not actually stored anywhere other than the username being used to address the user throughout the remainder of the app.

Graphs: Graphs would update after the user submitted the survey depending on their response. When we tried implementing the only graph package for React Native, we found that it crashed when new props were appended to it. In response, we instead created 10 different graphs with different possible outcomes depending on what the user inputted for their daily survey. This way, the graphs were already stored and we could just pull up whichever graph matched the survey information.

Hard-Coded

User information: As there are no users on the app, historic user data has been hard-coded, except those pertaining to recent survey answers. All friend data - habitats, current status, etc. has been hard-coded so that there are friends to interact with.

Inventory: As there is no historic use by the current user, all objects in the inventory must be hard-coded.

Health Insights/Graphs: Again due to the lack of historic use, the graphs page has been fabricated and hard-coded to provide an example, though the historic answers reflect what the user has actually put in.

Survey Questions: The answer choices for survey questions have been hard-coded, such as the bear only having the stomach as the option to select for discomfort. Furthermore, there have been emotions and feelings provided already for survey questions, though users can add more.

Inventory and Chat Page: The inventory was non-clickable because adding seeds and goods to the garden was not part of our tasks. Also, our chat page was non-interactive because we did not have another user on the other side of our interface.