

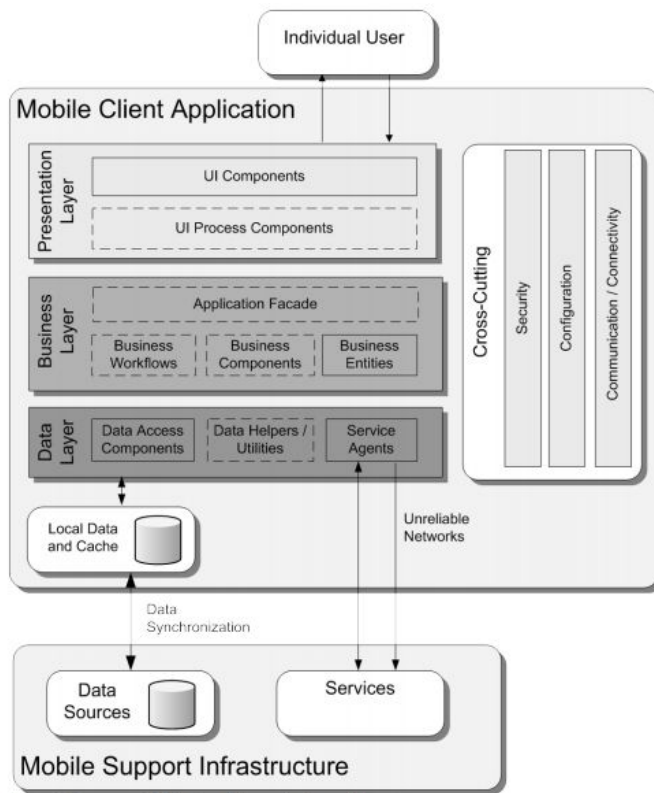
SWEN 325 Assignment 1 Report

Gabriel Tennent

300393699

Overall application architecture:

I tried to design my mobile application using the guidance of the microsoft - "Mobile Application Architecture Guide" linked in the lecture slides. This design splits the application into three primary parts, the presentation layer, business layer, and data layer. Each layer is independent and is only aware of and communicates with the neighbouring layers, which in turn helps prevent coupling. The three layers are unique in their purpose and paired together provide the functionality needed for a mobile application. Each layer being unique and independent in turn makes modifications to each layer easier, because you can modify a specific aspect of the mobile application without affecting others.



For example, if you wanted to modify the data layer and introduce a new storage system to the application (which I did) you do not want to have also modify the presentation layer to account for this change in the data layer. Using this design application you will change the way you are accessing the data and even where the data is stored but the data being provided will still be the same and in turn will still be presented the same way without changes. If you were not using this application architecture changing the data layer could affect the presentation of the data and in turn lead to huge amounts of time being wasted on adjusting the code for the presentation as aspects of the presentation were dependant on the previous data system. Ideally I wanted as much of the code to not be dependant on other pieces of code as possible.

The application I designed was a rich client application, meaning that the data and business services were located locally on the users device itself. I tried my best to follow the architecture described above and designed my application around presentation layer, business layer, and introduced two data layers one being on a firebase server and the other being stored locally.

The presentation layer is what the user sees and interacts with on their mobile device. This was made up of the HTML, and CSS files inside of each page of the application in ionic. When designing this layer I tried to take into account who would be using my application and the environment they would be using it in. With this in mind I made the application navigation self explanatory and simple by using buttons that described their purpose through simple images. Visually I tried to make the application simple and easy to understand by reducing clutter and having lots of white space which in turn guides the eyes towards the aspects of the application that displays the relevant information. Information inputted by the user through interaction with the presentation layer was passed through to the business layer of the application which then handled it accordingly.

My application controller was the business layer of the application architecture. This layer was made up of the TS files inside of ionic. These files were used to send and retrieve data from the data layer therefore creating the necessary connection between the data and presentation layer of the application. This connection was possible because the business layer formatted the data according to the layer requiring it. Allowing for data to be displayed in the presentation layer through viewable objects that are suitable for the users experience and stored in the data layer in a manner that is ideal for the storage components being used.

The data layer was composed of the data access components, data utilities, and service agents. My application used the Ionic storage component and Firebase application. The user information and login was linked through the Firebase application, and integrated with a guard to prevent certain pages of the presentation layer from being accessed without logging in first. Information relevant to the habits the user has inputted and is tracking was stored in the Ionic storage component. Both the storage components were modified using their own services that then worked with the .TS files to provide the relevant information

Description on external component:

The external components I used were the Firebase storage system and Ionic storage system. The firebase was integrated into my application using the Firebase ionic plugin and the Ionic storage system was integrated using the Ionic storage plugin. The firebase component just stored information on the user's account details, allowing for logging in and logging out. This was integrated with a guard which prevent users from accessing the applications features without logging in. The account management aspect of the presentation layer communicated with its .TS files (business layer) which then took relevant information and passed it through to the firebase inside of the data layer. The logging in and out validation was clarified using service pages and guards that didn't modify the presentation layer at all helping insure the independence of each layer of the application.

Upon logging in the user has access to the pages that provided the application functionality of tracking habits. The Ionic storage component was used to store information processed on these pages. The data layer that made a connection between the storage component and the business layer was made using a storage service. This service provided all of the functionality for storing and retrieving data from the component, and had the relevant methods for finding specific information inside of the storage component. The storage service only spoke to the .TS files of the pages that it provided information for, to help ensure the independence of each layer and prevent modification of the storage component from affecting the presentation layer. A problem I faced when implementing the Ionic storage component was getting the .TS files to retrieve the relevant information using the services accessible without reloading the page.

Ionic framework - Advantages/Disadvantages:

I enjoyed using the ionic framework and found it reasonably easy to use. It was sometimes difficult to find up to date tutorials as there has been a new release of Ionic 4 relatively recently. Sometimes I would be halfway through a tutorial only to realise they are using an out of date framework and methodology of thinking is no longer applicable.

Advantages:

- Easy to emulate. Since you can emulate the entire application through the command line with “ionic serve” very simple and useful. Using visual code I could also install an extension that allowed me to emulate the application through there.
- Generating pages and services/guards was really simple and could also be done through the command line. It was amazing it generated all of the files required for a page.
- The initial structure of the filing provided by ionic when you generate a new blank application makes it easy to grasp what is front end and back end. This in turn made the architecture of the application simple to implement.
- The native components of the application are adaptable to multiple platforms and still maintain a very professional feel. They represent what you would expect of a professionally developed application specific for IOS.
- A lot of great icons to choose from for buttons. There is a range of already graphically sound image options for your buttons.
- Though Ionic has just relatively recently deployed Ionic 4, there docs for it are easy to follow and even provide visual examples for you alongside an example of code that you could also implement to achieve the same result.

Disadvantages:

- I found it very difficult to know if there is a bug and then work out where the bug is when working with the HTML side of the ionic framework. There is no indication of where the bug lies inside of the HTML code and it instead just does not render the page. Since an entire page may not be visible because of a lower case letter not have reference to where this letter is made it very difficult to fix issues lying in the HTML code.
- I also had a lot of trouble learning the “promise” system that has come with Ionic 4. This change is very significant to the development of applications as it is how you retrieve data from the storage utilities. This was emphasised as since Ionic 4 was only deployed

recently the online tutorials for this were very basic and only covered simple basis. Which made it possible to learn but difficult to implement into your own application that is dealing with more complex objects.

- I spent hours trying to work out how to refresh a page data from the data layer without reloading the page. This may be a me problem, but all references online pointed to out of date methodology that was in existence during Ionic 3.
- Run time errors were sometimes very difficult to locate as the error messages that came with them were very basic. Sometimes they referred you right to the problem but on other occasions it was hard to work out what is not working.
- The online community for Ionic seemed rather small and there was little resources available for solving application braking issues. This was emphasised even more by how out of date the majority of resources were and meant you had to search for solutions for a long time or spend hours working out the issue without a resource to refer to.
- When I did finally an online resource that was relevant to my issue, it normally involved deploying more packages into application which made it hard to track everything being imported.

The majority of issues (disadvantages) I faced with Ionic were because I was new to the framework and there is little online to help beginners at this stage of Ionic 4's deployment. With this in mind once learning the basics of the framework it was a really enjoyable experience and with experience it would be a fantastic framework for quickly producing professional multi-platform applications.

3 UX Decisions:

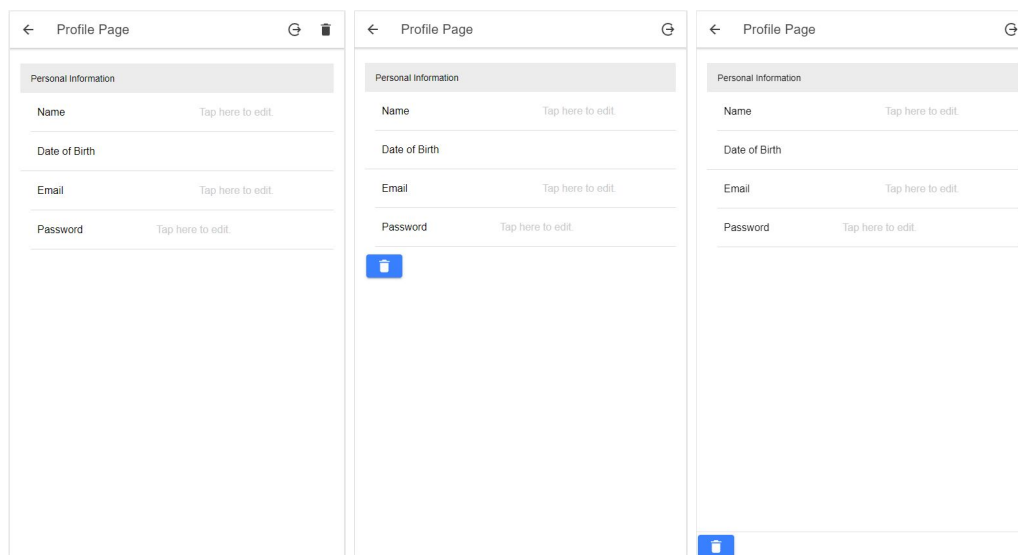
1st UX design decision:

The first UX design decision I had to make was about where to locate the buttons on the screen. This may not seem very important but in some cases it could have had terrible consequences on the application if button placement was not thought out. Though this design decision affected all of my pages, it was most clearly seen in the profile page where a single button “trash button” can wipe all of the information on habits the user has inputted.

The first option was to locate the button next to the other logout button at the top of the screen like all of the other buttons on the application used for navigation. But since the buttons at the top of the screen throughout the application are used for navigation, I thought that having a clear all button up there might confuse users and they might accidentally wipe all their data. They could also accidentally wipe all of their data if they were in a rush and tried to push the logout button but accidentally tapped the trash button.

The second option was to locate the button in the center of the screen where it is easily visible for the user and is not close to the logout button. But this option provided its own problems since the user is modifying their information so close to this button they might accidentally tap the trash button or push it out of curiosity or thinking it will wipe their personal information.

The third and final option I went with was at the footer of the page. This option was the best in this case as it meant the button would not accidentally be pushed when trying to interact with the page in another manner. It also indicates to the user that this button deletes useful information (is not a button to press for fun) as it is at the bottom and gives a menacing feel. Since the button is the only colored object on the page, it still made it easy to see for users actually wanting to clear their habits.



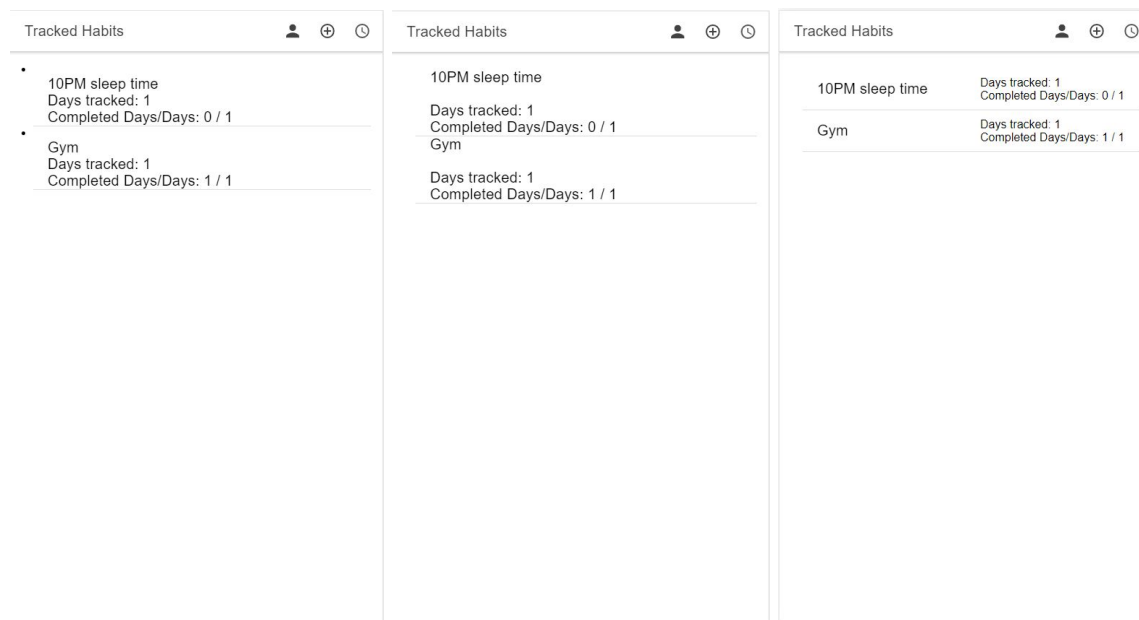
2nd UX Design Decision:

The second UX design decision I made was regarding the displaying of habits being tracked. I knew from the beginning I wanted the habits to be listed on the homepage of the application, but this list has to be clear and easy to understand without thinking much.

The first approach was by implementing a simple ` ` object which is a HTML list element. But this approach lead to bullet points being added to the page for each habit which were necessary as it was already self explanatory that it was a list and increased the page Clutter.

The second approach involved putting the Li html element inside of a `<ion-List>` element which removed the bullet points for the listing of the habits. This approach made the home page look a lot cleaner which was a step in the right direction but the information associated with the habits was still to bold and caught your eye when looking through a large list of habits for a specific one. This made finding habits amongst a large list tedious.

The third and final option that I went with was moving the information on the habits across the screen relative to the habit name by putting inside its own `<ion-content>` block. This made the list a lot clearer when dealing with large lists of habits and removed the tedious action of finding a specific habit amongst a large list. It also made the information associated with each habit a lot more clear as instead of having to look at the grey lines to work out which information is associated with which habit you just had to cross to the habit name. The font size change also made the information on the habits less intimidating.



3rd UX Design Decision:

The third UX decision I am going to cover was regarding the sequence of input fields when adding information for the user. This was seen through out the application but is best conveyed through the Habit Creation Manager page. In this page there are three required inputs for adding a new habit - habit name, date to add initial day for the habit, did it? Indicating whether or not the user has done the habit.

The first approach made the user input initially whether or not they had done the habit and then asked for the day and then the habit name. This was confusing as the first thing the user saw when navigating to the page was an input saying did it? Which created confusion as the user may think they are adding a day. This design didn't navigate the user naturally through the input.

The second approach was more successful as now the habit name was at the top of the input of the input fields. Indicating right away to the user that they are inputting a new habit by getting them to input the new name of the habit right away. This provided some natural navigation for the user but the user was then again faced with the did it? question again without knowing what whether or not they did the habit was referring to.

The third and final approach I used was the most self explanatory and navigated the user through the input naturally. This was achieved by having the user first input the name of the habit indicating they are adding a new habit and then asking for day to add for the habit. Finally asking whether or not they did the habit at the habit the end. This ensured that by the time the user got to the did It? input, they knew that they were answering whether or not the did the named habit at the selected day.

← Habit Creation Manager	
Did It?	Did the habit? ▾
MM DD YY	Select Date
Habit Name	
ADD HABBIT	
Select Habit	Select habit to remove ▾
REMOVE HABBIT ↵	

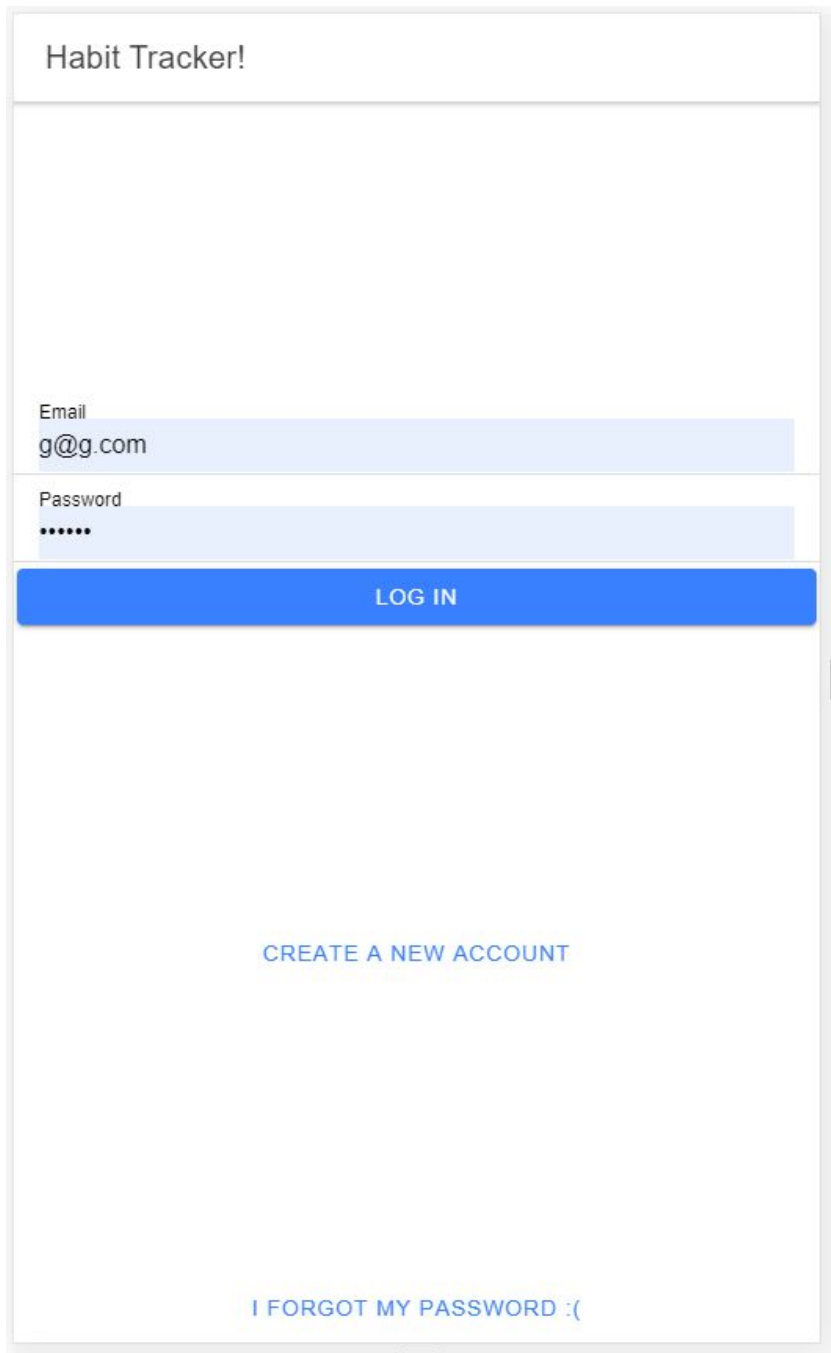
Habit Creation Manager	
Habit Name	
Did It?	Did the habit? ▾
MM DD YY	Select Date
ADD HABBIT	
Select Habit	Select habit to remove ▾
REMOVE HABBIT ↵	

Habit Creation Manager	
Habit Name	
MM DD YY	Select Date
Did It?	Did the habit? ▾
ADD HABBIT	
Select Habit	Select habit to remove ▾
REMOVE HABBIT ↵	

Appendix:

Login screen:

This is the login screen for the application. It introduces the application through the title and has reference to the create account and forgot password pages. Its purpose is to be simple and indicate to the user to login into the application which is why the login is right in the middle of the screen to grab your attention.



The image shows a mobile app login screen titled "Habit Tracker!". It features a white background with a light gray border. At the top, the title "Habit Tracker!" is displayed in a dark gray font. Below the title, there is a large white rectangular area. In the lower-left portion of this area, there are two input fields. The first is labeled "Email" and contains the text "g@g.com". The second is labeled "Password" and contains six black dots. Below these fields is a prominent blue button with the text "LOG IN" in white, uppercase letters. Further down, centered on the screen, is the text "CREATE A NEW ACCOUNT" in a smaller, blue, uppercase font. At the very bottom, also centered, is the text "I FORGOT MY PASSWORD :(" in a small, blue, uppercase font. The overall design is clean and minimalist.

Habit Tracker!

Email
g@g.com

Password
.....

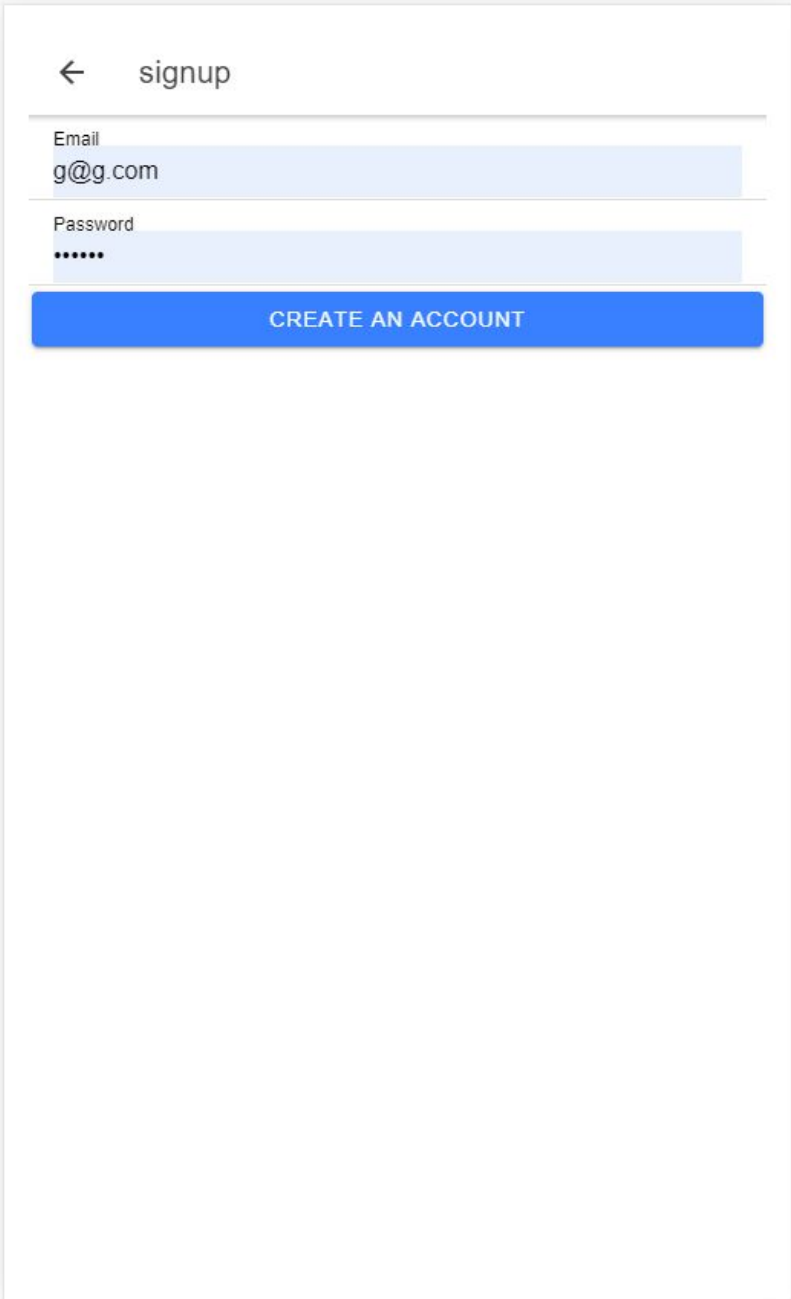
LOG IN

CREATE A NEW ACCOUNT

I FORGOT MY PASSWORD :(

Create new account:

The design of this page is very simple, yet effective. Its entire purpose is to give a place for a user to sign up if they don't already have an account. It indicates to input an email and password and then sign up. It also has a back button in case the user has ended up at this page and wants to go back to the login screen.



The image shows a mobile application interface for a signup page. At the top left, there is a back arrow icon and the text "signup". Below this, there are two input fields: "Email" with the placeholder text "g@g.com" and "Password" with masked characters "*****". A blue button with the text "CREATE AN ACCOUNT" is positioned below the password field. The entire form is enclosed in a light gray border, and there is a vertical double-line icon on the right side of the border.

Forgot password:

The purpose of this page is to give a place for the users to reset their password if they have forgotten it. It is simple and design and displays at the top to the user to input their email and then click the reset password button. Their also a back button in case the user remembers their password after navigating to this page.

[←](#) Password Reset




Email

enter email address

RESET YOUR PASSWORD


Home page:

The most important part of the application is the home page. The design of this page was meant to be simple and not overcomplicate the tracking of the habits. All of the habits being tracked are displayed in the center of the screen so that if you are using this application, all you have to do is make it to the home page to see your progress. The page also has three buttons which are self explanatory with a person indicating the profile page, plus button indicating adding a habit, and a clock to indicate the add day to habit page.

Tracked Habits				
10PM sleep time	Days tracked: 1 Completed Days/Days: 0 / 1			
Gym	Days tracked: 1 Completed Days/Days: 1 / 1			


Habit Day Manager:

The purpose of this page was to allow the user to add and update their progress to the habits being tracked. It tells you this in the title and then is designed to indicate to the user to select the habit they want to add a day to then select the day and hit yes or no to whether not they did the application. A big blue button helps indicate to the user what they are doing on this page. A back button has been implemented in case the user didn't want to add a day to the habit.

 Habit Day Manager	
Select Habit	Select habit to add day... ▼
MM DD YY	Select Date to add
Did It?	Did the habit? ▼
<div>ADD DAY TO HABBITI</div>	

Habit day creation:

The purpose of this page was to provide the user a place to add or remove a habit. It was designed to indicate to the user where to implement a habit and has the Habit name highlighted blue so it is easy to see where to begin adding a habit. The buttons provide separation between input on whether the user is trying to add or remove a page. A back button has also been added incase the user no longer wants to add or remove a habit.

 Habit Creation Manager	
Habit Name	
MM DD YY	Select Date
Did It?	Did the habit? ▼
ADD HABBIT	
Select Habit	Select habit to remove ▼
REMOVE HABBIT :(

Profile page:

This page was a work in progress when I submitted the application. But it was designed with the intent for the user to be able to modify their information associated with application. It has a back button (back to home page) and logout button at the top of the page so the user can get back to the login screen if they want to change accounts. Down the bottom is a bin button which clears all of the habits being tracked currently on the application. This was placed down the bottom of the screen to prevent users from accidentally deleting their data.

← Profile Page ↻

Personal Information

Name Tap here to edit.

Date of Birth

Email Tap here to edit.

Password Tap here to edit.

