

Software Engineering (14:332:452)

Group 3

**Programming Project: Blockchain-Based Safe Sharing of Population
Descriptors**

Submission Date: December 9, 2019

**Team Members: Sasan Hakimzadeh, Pradyumna Rao, Nithyasree Natarajan,
Jack Dulin, Shruthi Sureshkrishnan, Nithya Kandappan, Breanna Higgins,
Sean Kearns**

Table of Contents

Responsibility Matrix	4
Section 1: Customer Problem Statement	8
Problem Statement	8
Glossary of Terms	11
Section 2: System Requirements	12
Enumerated Functional Requirements	12
Enumerated Nonfunctional Requirements	13
User Interface Requirements	14
Section 3: Functional Requirements Specification	15
Stakeholders	15
Actors & Goals	15
Use Cases	16
Detailed Use Case Descriptions	21
System Sequence Diagrams	23
Section 4: User Interface Specification	26
Preliminary Design	26
User Effort Estimation	32
Section 5: Effort Estimation Using Use Case Points	38
Section 6: Domain Analysis	39
Attribute definitions	43
Domain Model	44
System Operation Contracts	46
Section 7: Project Size Estimation	48
Section 8: Interaction Diagrams	51
Section 9: Class Diagram and Interface Specification	54
Class Diagram	54
Data Types and Operation Signatures	55
Traceability Matrix	59
Design Patterns	60
Contracts	60

Section 10: System Architecture and System Design	61
Architectural Styles	61
Identifying Subsystems	62
Mapping Subsystems to Hardware	64
Persistent Data Storage	64
Network Protocol	64
Global Control Flow	65
Hardware Requirement	65
Section 11: Data Structures	66
Section 12: User Interface Design and Implementation	67
Section 13: Design of Tests	74
Unit Testing	74
Test Coverage	74
Integration Testing	74
Other Testing	75
Section 14: History of Work	76
Section 15: References	78

Responsibility Matrix

[illegible]

[illegible]

Data Structures	12.5%	12.5%	12.5%	12.5%	12.5%	12.5%	12.5%	12.5%
User Interface design and Implementation	12.5%	12.5%	12.5%	12.5%	12.5%	12.5%	12.5%	12.5%
Unit Testing	12.5%	12.5%	12.5%	12.5%	12.5%	12.5%	12.5%	12.5%
Test Coverage	12.5%	12.5%	12.5%	12.5%	12.5%	12.5%	12.5%	12.5%
Integration Testing	12.5%	12.5%	12.5%	12.5%	12.5%	12.5%	12.5%	12.5%
Other Testing	12.5%	12.5%	12.5%	12.5%	12.5%	12.5%	12.5%	12.5%
Design Patterns	12.5%	12.5%	12.5%	12.5%	12.5%	12.5%	12.5%	12.5%
Contracts	12.5%	12.5%	12.5%	12.5%	12.5%	12.5%	12.5%	12.5%
History of Work	12.5%	12.5%	12.5%	12.5%	12.5%	12.5%	12.5%	12.5%
References	12.5%	12.5%	12.5%	12.5%	12.5%	12.5%	12.5%	12.5%

All team members contributed equally

Section 1: Customer Problem Statement

a. Problem Statement

The customers find that it is difficult to stay in shape and lead healthy lives. With everyone living busy lives focused on education, work, and families, one's health is not given a priority. As a result, people's well-being and fitness levels deteriorate over time. Therefore, the customers want a web-based application that provides its users with a consolidated, one-stop platform to input all of their health-related data. Such data would include users' hourly heart rate, amount of hours slept, their daily activities, their daily activities, etc.

There are many aspects that go into making sure an individual is healthy both physically and mentally, especially since the two go hand in hand. Although customers would like to be healthy, they find it overwhelming to keep track of all the different aspects of good health. For instance, it is difficult to remember everything one ate throughout the day, and then also recall it the next day or even later in the week. Therefore, users would want the application to track several different features, including heart rate, sleep, food, weight, activity, and exercise. This way, people can use the application to review the various aspects for the current week as well as look at how much they have progressed throughout the weeks.

Being able to look at data from previous weeks and months is very important to customers because they want to be able to compare the data gathered from various weeks. This will also allow users to see the improvements in their fitness. Many times it is difficult to see progress in one's health just by looking in the mirror. This results in being discouraged from working out and eating healthy. However, having the physical data from each week will support the developments being made and provide the encouragement needed to continue with one's regimen or make the necessary changes to gain the results desired.

Users want to be able to create a profile that will have information about them, such as date of birth, sex, height, and weight. They can then input information for each of the different aspects, every day. For the heart rate aspect, customers want to upload their heart rate and check if it is at a healthy level, based on age and other information. For the sleep aspect, users want to put in the time they went to bed and the time they got up to see how many hours of sleep they got per night. The application would then give feedback to the person, telling him/her to get more or less sleep, based on the person's age.

For the nutrition aspect, users want to input everything they ate and drank for the day for breakfast, lunch, and dinner as well as any snacks eaten throughout the day. This will allow customers to keep track of what they ate, if it kept them full, if they liked it, and if they would eat it again. For example, if they reviewed what they ate for the day and felt that they were still hungry throughout the day, they know to increase the food intake. However, if they felt too full throughout the day, then they know to eat less food. Users would also like to have a calorie option, so that when they input the food, the amount of calories shows up, or a person can manually input the amount of calories for that food. This way, those who calorie count can know they are on track for their daily amount of calories.

For the weight aspect, customers want to input their daily weight so that they can check if they are staying on track to reach their goal of losing weight, gaining weight, or staying the same weight. For the activity aspect, they can for instance, upload their daily step count. For the exercise aspect, users want to input each of the exercises that were done that day, how long each exercise was done, and how many times each exercise was done. The application should then be able to show how many calories were lost from doing these exercises. After inputting data every

day, users also want to see how they stand when compared to the average population within a similar demographic, such as age, gender, race, etc. This way, users can track their relative standing on a secure platform. Many times, the statistics provided by the government are outdated and too general, so having reliable information in real time is important for users.

Customers want to use the application to inspire themselves to set personal goals such as reaching a higher step count the following week or drinking more water tomorrow. As a result, users need pop-up messages to remind them of their weekly tasks and to notify them if they are close to reaching their goal. Users also want a message to congratulate them once they have reached their goal, so they continue to stay motivated. Users also need a daily reminder system to remind the user to input their data for the day and should be customizable to when and how many times a day users get reminders.

Users want an application that is free and easy to use. A web-based application enables the user to input data from any device they own. The customers also want this to be accessible for people of all ages. They want this application to be simple to use so that the elderly can also use it to track their health. Applications on phones are sometimes more confusing to use for older people, so a web-based application, which has bigger words and icons, will allow people of all ages to watch over their well-being. Users will also be able to use the pop-up messages to remind them to take their daily medication or do their other daily health related tasks. Customers would also like to use this application to make sure that they are healthy in general. If there is a potential abnormality based on the data uploaded by the user, then the users want a pop-up message that urges them to go visit a doctor.

b. Glossary of Terms

Encryption: The method by which plaintext or any other type of data is converted from a readable form to an encoded version that can only be decoded by another entity if they have access to a decryption key

Blockchain: A mathematical structure for storing digital transactions (or data) in an immutable, peer-to-peer ledger that is incredibly difficult to fake and yet remains accessible to everyone.

Primary Key - A specific choice of a minimal set of attributes that uniquely specify a row in a database. In our case, the username would be the primary key, as it needs to be unique per user so as to not duplicate accounts.

Health Profile - A holistic set of physiological measurements and features such as weight, height, BMI, caloric intake, calories burned, heart rate, sleep, etc. These set comprised of these features make up a single user's health profile and can be used as a data point.

Tracking Device - A wearable health tracker that may be part of a "smart" watch, such as a FitBit, Apple Watch, Samsung Gear. It should be able to track all the basic features that can be tracked automatically under the Health Profile. It should also have support to manually input the features that cannot be automatically tracked, such as caloric intake, weight, and height.

Health Rank - A numerical ranking statistic that serves to show a user's parameters in comparison to a given population. There may be multiple health ranks for any given user based on the population against whom he/she is querying. I.E. A user may have a different rank against only other male users, only other female users, only local users, etc

Section 2: System Requirements

a. Enumerated Functional Requirements

Label	Priority Weight (1-10)	Description
REQ-1	10	The system must ensure that the communal health data must be encrypted and secured.
REQ-2	5	The system must ensure that there is a daily reminder for the user to put in their data.
REQ-3	8	The system must ensure that users must not be able to access other user data other than the general trends from the population
REQ-4	9	The system must ensure that users should be able to log into their account using a username as a primary key, and a secure password which adheres to certain security measures such as length and special character inclusion.
REQ-5	7	The system must ensure that users must be able to compare themselves against others using real-time data that is constantly updated.
REQ-6	6	Users must be able to see a page where they can analyze visualizations (graphs & charts) which show their data points compared to their demographic.
REQ-7	5	There must be a max threshold for certain parameters so as to avoid fraudulent input of data which would skew calculations/trends.

b. Enumerated Nonfunctional Requirements

Label	Priority Weight (1-10)	Description
REQ-8	5	Functionality: Users should be able to set goals for themselves to be able to achieve (number of steps taken, miles walked, etc.)
REQ-9	3	Functionality: Users can “friend” others on the platform and share their results with their network, securely and privately.
REQ-10	3	Functionality: Users can personalize their profiles by uploading images of themselves as their thumbnail.
REQ-11	4	Functionality: Users should be able to post their progress to their social media accounts (Facebook, Twitter, etc.)
REQ-12	8	Performance: The application shall be responsive to commands/clicks with a reasonable amount of loading time, based on the information being refreshed or loaded.
REQ-13	8	Performance: The application shall optimize data storage space to improve data retrieval speeds.
REQ-14	8	Usability: The application shall be human-centric and usability should be prioritized, with an emphasis on making the front-end application easy to use and navigate.
REQ-15	7	Supportability: The application shall be easy to maintain and update, with code written clearly with comments to ensure other developers can read and understand the functionality for future updates, bug fixes, etc.
REQ-16	7	Supportability: The application shall be reliable with little to no failure or crashing. If the application does crash, disaster recovery should be set in place to ensure all data remains and is consistent. The distributed and non-centralized storage of the data helps to ensure that a breach or malfunction in one location will not affect data stored in another location.
REQ-17	5	Scalability: Although the application will be on a relatively small scale for the purposes of this project, the system shall be able to be scaled up to support thousands of users on it at the same time.

c. User Interface Requirements

Label	Priority Weight (1-10)	Description
REQ-19	7	The user interface should allow for users to input data and access their profiles from both mobile devices and desktops.
REQ-20	6	The graphs and charts displayed to the user as visual aid should be interactive and show smaller breakdown information when clicked.
REQ-21	8	There should be a friendly, welcoming login page that should have two (2) fields for username and password, as well as a forgot password/reset password link.
REQ-22	4	There should be a friendly reminder text message sent to the user's cell phone number at 10:00 P.M. reminding them to input their daily data if they have not already. Pop-up notifications are also an alternative.
REQ-23	9	There should be a menu that includes link to all the parts of the app, such as home, user profile, account and security, daily data input, health ranking and population querying, etc

Section 3: Functional Requirements Specification

a. Stakeholders

The following lists stakeholders who may have an interest in this application:

- General public: People who want to compare their fitness levels against others in the same demographic or want to know if they're in the range of what is considered healthy
 - This could apply to people who are already healthy and want to stay healthy, or those who are working toward obtaining a healthier lifestyle and want to know what that is
- Organizations that promote health and fitness, such as gyms
 - They could use this system as motivation and/or targets for their clients. They could also potentially use it to see the general trend of their clients
- Doctors could use this system to check if their patients are staying healthy
 - Doctors could check if their patients are following the regimen prescribed to them and if anything needs to be altered in order to prescribe a better course of action
- Research-based organizations: They could use the data to find potential patterns in certain populations
 - Patterns can show common occurrences that happen among different groups based on sex, age, or race, for instance
 - These common occurrences can identify norms for these groups or find health issues that need to be resolved
- Companies that sell medical devices and health care equipment
 - Companies can be sponsors to advertise their products on the website so that users can view these advertisements while inputting their data

b. Actors & Goals

The following lists all participating actors in the application and their goals

- User - The goal of the user is to be able to have a seamless and intuitive experience on the application where they can track their fitness metrics, compare it to the general population, set goals for themselves, build a social network, and visualize charts that show where they stand amongst others within their demographic.
- Database/Blockchain/Backend - The goal of the backend database & incorporated blockchain technology is to be able to store the user information as it changes in the app, and to ensure that all data is as secure as possible.
- UI - The goal of the UI is to offer the user a seamless experience on the application, without the need to question where to go in order to complete a task.

- Wearable Devices - The goal of the devices is to feed the data that we are tracking for the entire population. The data metrics we record for trends in various population demographics will be based on what is provided historically through these devices.

c. Use Cases

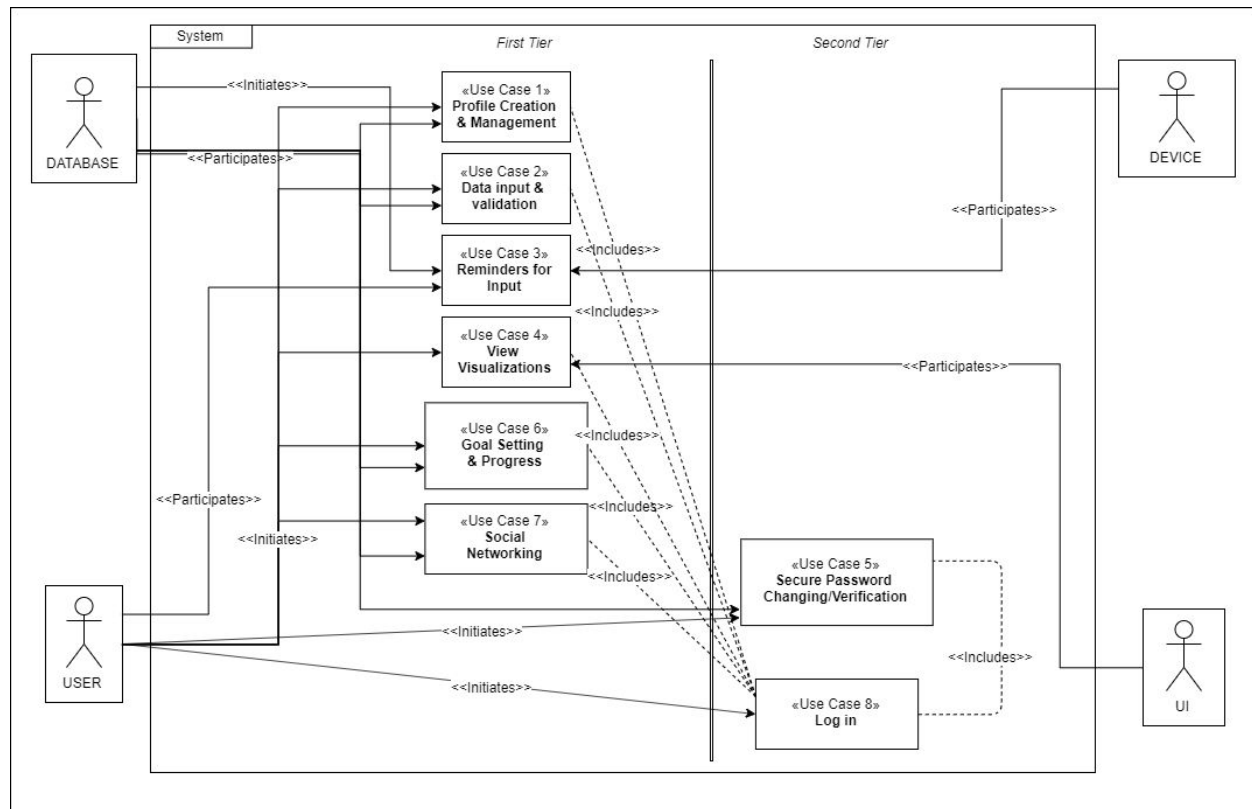
i. Casual Description

Actors and Goals	Use Case	Relevant Requirements	UC Number
Actor: User (Initiating), DBMS (Participating) Goal: Users can create & update their profiles as desired, and the DBMS stores these updates as they occur	Profile creation & management	REQ-10, REQ-19, REQ-13, REQ-16	1
Actor: for Data Input: User (Initiating), DBMS (Participating) Goal: User is able to input daily health metrics for analysis Actor: for Data range Validation: Database Management System (Initiating Actor) Goal: All of the data entered into the database is accurate/not skewed to ensure accurate visualizations and comparisons.	Data input & validation	REQ-7, REQ-1, REQ-12, REQ-13, REQ-14, REQ-15, REQ-18	2

<p>Actor:</p> <p>Database (Initiating)</p> <p>User (Participating)</p> <p>Device (Participating)</p> <p>Goal:</p> <p>Ensure users are engaged with the platform</p> <p>Try to obtain as full of a data set as possible to avoid sparse/empty rows that would skew visualizations/calculations.</p>	Reminders for input	REQ-2,REQ-22	3
<p>Actor:</p> <p>User (Initiating)</p> <p>UI(Participating)</p> <p>Goal:</p> <p>To provide a friendly user interface.</p> <p>View statistical data in user-friendly graphical representation.</p>	View visualizations	REQ-6, REQ-20, REQ-5, REQ-12, REQ-14, REQ-18	4
<p>Actor: User (Initiating), DBMS (Participating)</p> <p>Goal: Users should be able to securely change their password (either through email link or verification questions) and the database should be able to reflect that change in the users' profile once the change has been made.</p>	Secure Password Changing/Verification	REQ-21	5

<p>Actor:User (Initiating), DBMS (Participating)</p> <p>Goal:Motivate the user to keep putting in data so that there is an abundance of it to ensure statistical accuracy due to increased sampling. Meanwhile, the DBMS should be able to store these goals as they are created by the user.</p>	Goal Setting and Progress	REQ-8, REQ-14, REQ-16, REQ-18	6
<p>Actor: User (Initiating), DBMS (Participating)</p> <p>Goal: Users are able to curate their own user experience by sending friend requests, blocking users, denying requests, and sharing progress on social media to enhance user engagement with the application as well as increase motivation. Meanwhile, the database should keep records of social networking interactions and social media pages that are linked to each user.</p>	<p>Social Networking</p> <ul style="list-style-type: none"> -Friending/Unfriending -Blocking -Denying Requests -Share progress on social media 	REQ-11, REQ-9, REQ-10, REQ-17	7
<p>Actor: User (Initiating)</p> <p>Goal: The user is able to securely log into his/her account with the credentials he/she set with the account.</p>	Log In	REQ-21, REQ-1, REQ-3, REQ-4, REQ-15	8

ii) Use Case Diagram



iii. Traceability Matrix

The following traceability matrix shows the ability of the identified use cases to achieve all requirements set forth for the application. With the matched priority weighting of each related requirement, the matrix determines the most essential use cases. The three largest use cases are highlighted as they are the most critical to the system.

Priority Weight	Requirement	UC-1	UC-2	UC-3	UC-4	UC-5	UC-6	UC-7	UC-8
5	REQ-1		x						x
10	REQ-2			x					
8	REQ-3								x
9	REQ-4								x
7	REQ-5				x				
6	REQ-6				x				
5	REQ-7		x						
5	REQ-8						x		
3	REQ-9							x	
3	REQ-10	x						x	
4	REQ-11							x	
8	REQ-12		x		x				
8	REQ-13	x	x						
8	REQ-14		x		x		x		
7	REQ-15		x						x
7	REQ-16	x					x		
5	REQ-17							x	
6	REQ-18		x		x		x		
7	REQ-19	x							
6	REQ-20				x				
8	REQ-21					x			x
4	REQ-22			x					

Max PW	8	8	10	8	6	8	5	9
Max PW	25	47	14	41	6	26	15	37

iv. Detailed Use Case Descriptions

Based on the given traceability matrix, the following detailed use cases are described as they have the highest priority weighting.

UC-2	Data Input/Validation
Related Requirements	REQ-7, REQ-1, REQ-12, REQ-13, REQ-14, REQ-15, REQ-18
Initiating Actor	User, Database Management System (DBMS)
Participating Actors	Database Management System (DBMS)
Preconditions	The user is logged into the system.
Postconditions	There has been a successful or unsuccessful attempt of adding more data into the system.
Event Flow (Success)	<ol style="list-style-type: none"> 1. The user has tried to enter a reasonable value. 2. The data point is entered into the system.
Event Flow (Failure)	<ol style="list-style-type: none"> 1. The user has entered an unreasonable value (e.g. it is above a certain threshold). 2. The data point is rejected from the system.

UC-4	View Visualizations
Related Requirements	REQ-6, REQ-20, REQ-5, REQ-12, REQ-14, REQ-18
Initiating Actor	User
Participating Actors	Data Visualization Software, DBMS
Preconditions	The user is logged into the system.

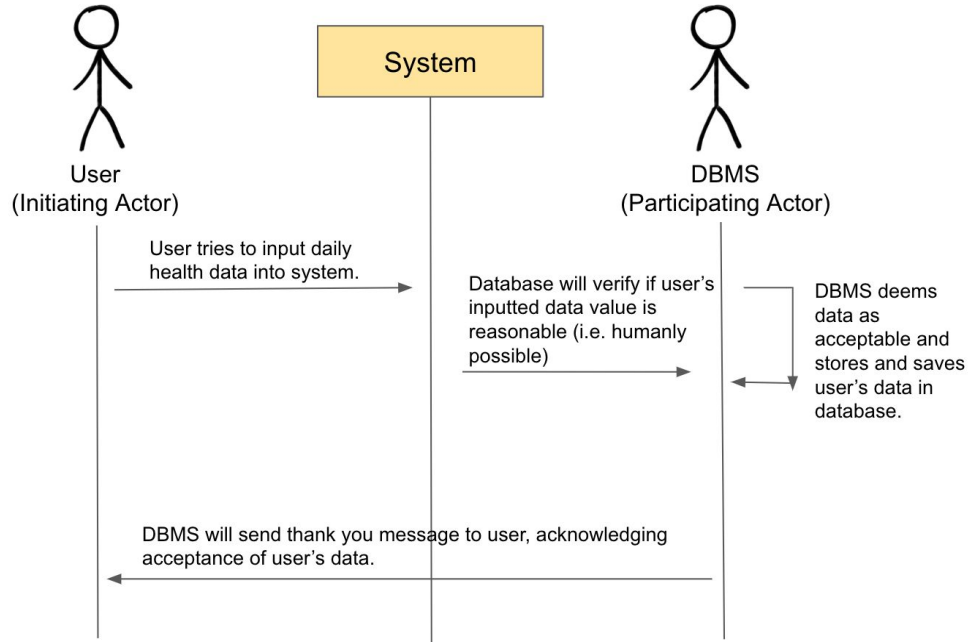
Postconditions	The user sees a collection of their data in the form of visuals.
Event Flow (Success)	<ol style="list-style-type: none"> 1. The user requests a visualization of their inputted data. 2. The data visualization software shows the user their data in a presentable way, whether through a pie chart or a bar graph.
Event Flow (Failure)	<ol style="list-style-type: none"> 1. The user requests a visual representation of their inputted data. 2. The data visualization software is unable to accomodate the user's request.

UC-8	Log In
Related Requirements	REQ-21, REQ-1, REQ-3, REQ-4, REQ-15
Initiating Actor	User
Participating Actors	Database Management System (DBMS)
Preconditions	The user is currently at the screen in which they are asked to enter his/her username and password.
Postconditions	There has been an attempt to get into the system through a correct or an incorrect username and password.
Event Flow (Success)	<ol style="list-style-type: none"> 1. The user types in the correct username and password. 2. The user is taken to the home page of our application.
Event Flow (Failure)	<ol style="list-style-type: none"> 1. The user types in the incorrect username and/or password. 2. They have another opportunity to type in the correct username and password. 3. They are asked if they have forgotten their username or password.

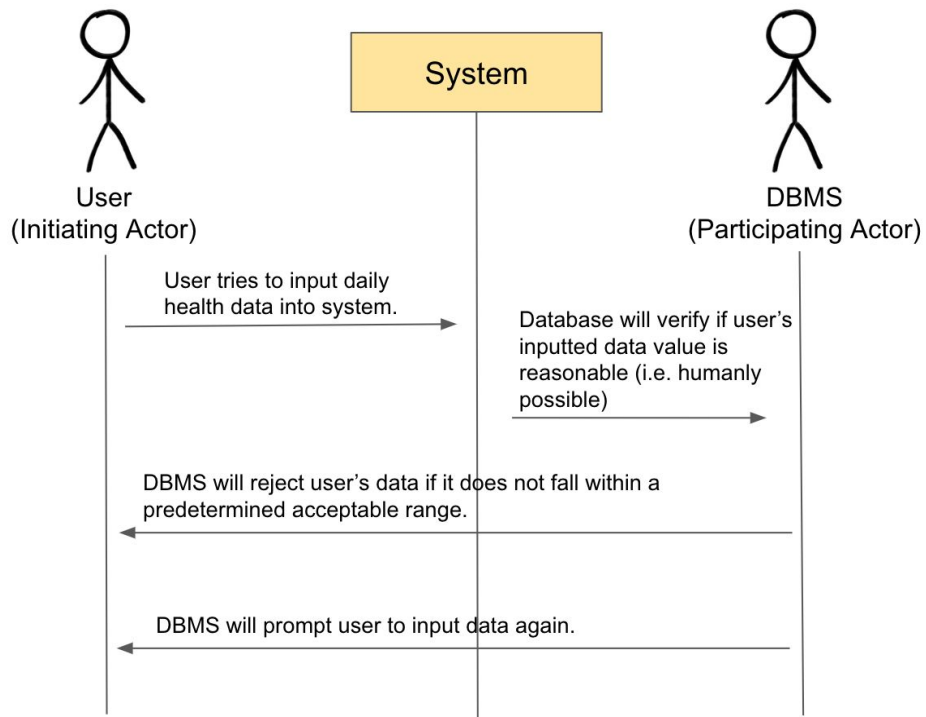
d. System Sequence Diagrams

The following diagrams model the event flows of the three fully-dressed use cases

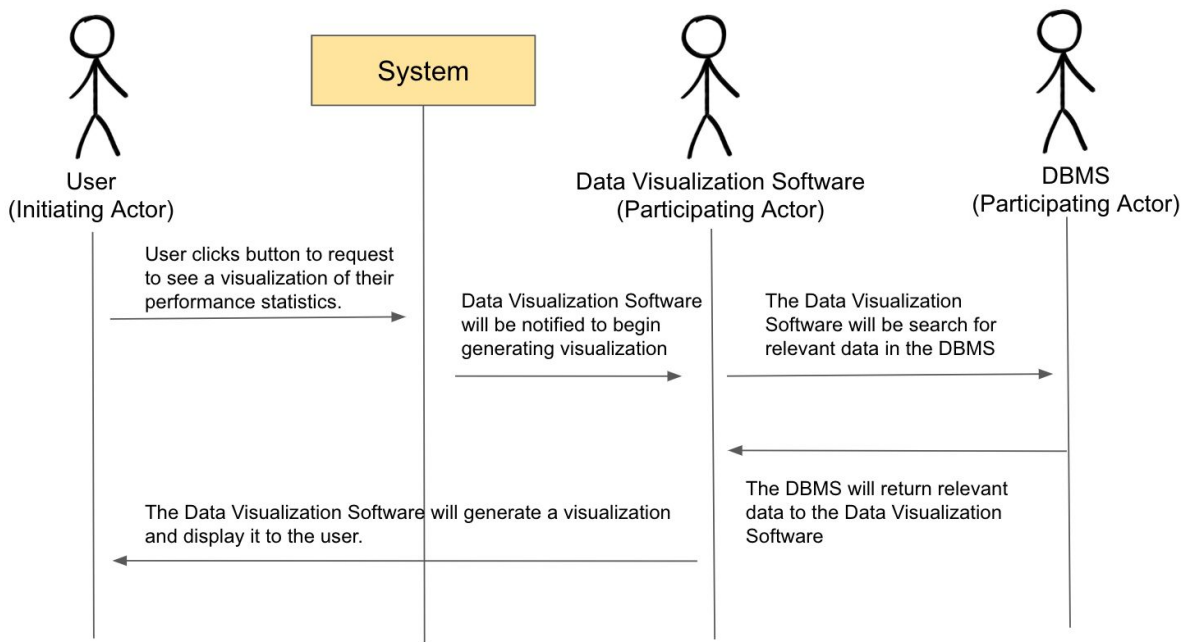
Use Case 2: Data Input / Validation (Success)



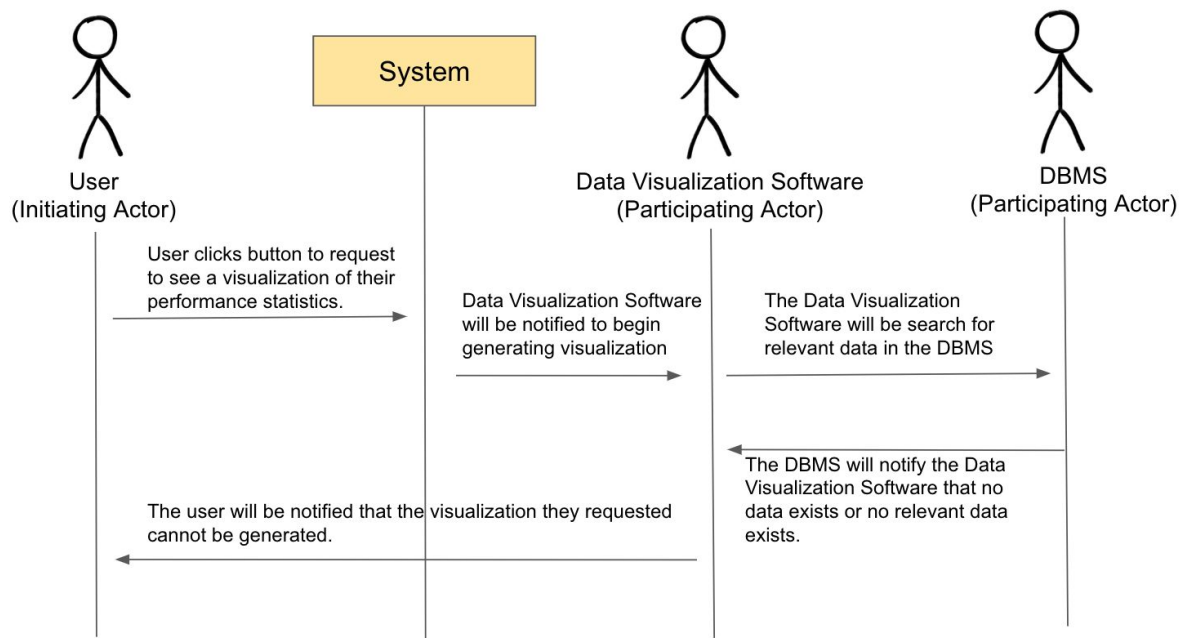
Use Case 2: Data Input / Validation (Failure)



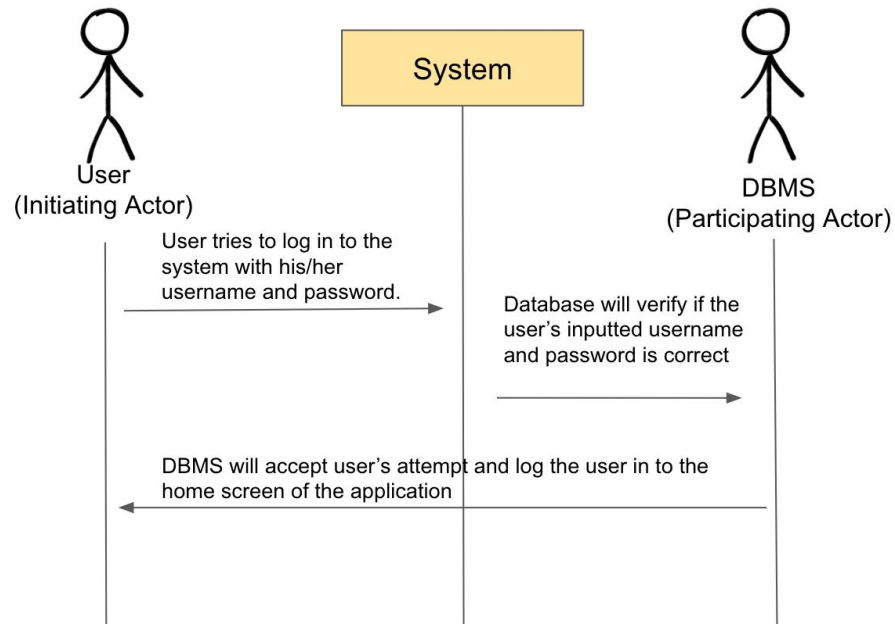
Use Case 4: View Visualizations (Success)



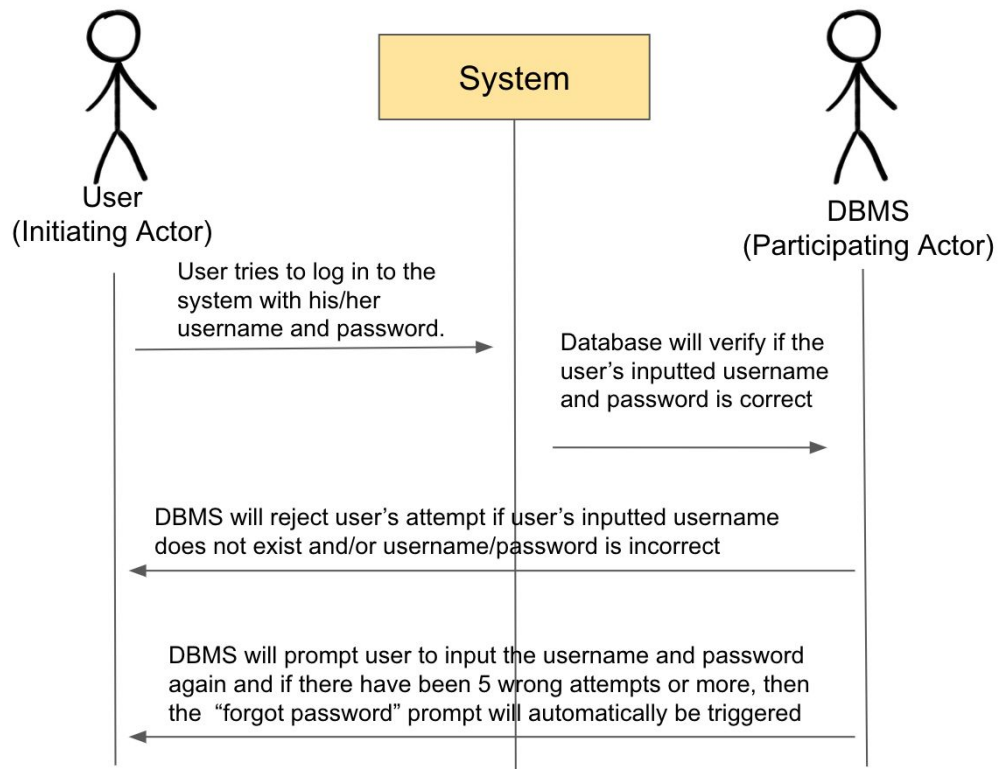
Use Case 4: View Visualizations (Failure)



Use Case 8: Log In (Success)



Use Case 8: Log In (Failure)



Section 4: User Interface Specification

a. Preliminary Design

i. Use Case 2

1. User is on the data entry page “Add Data.” He will enter any relevant information that he may wish to add for the day to continue tracking his progress. He will enter information in the appropriate fields and then submit it by clicking the “Add Data” button on the appropriate side of the screen.

Data Entry Page

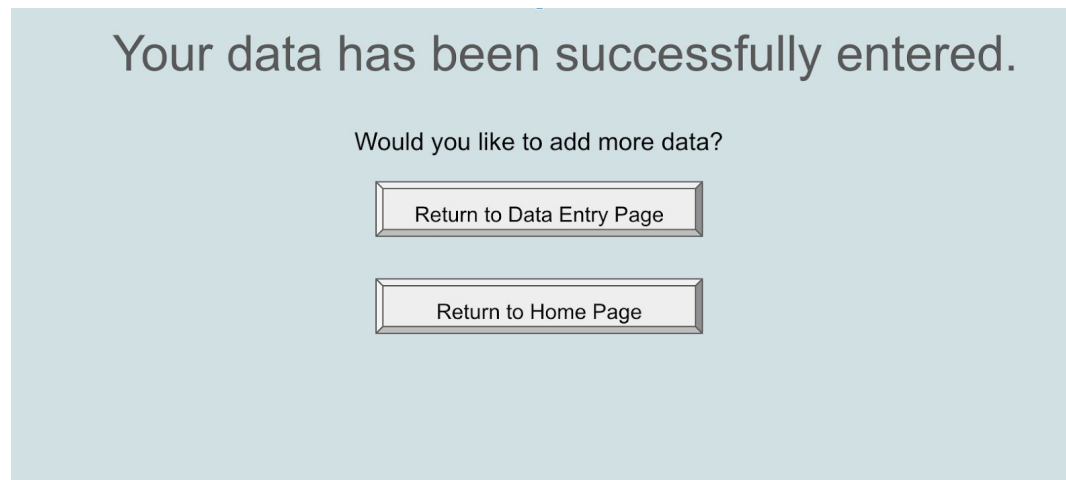
The screenshot shows a web interface titled "Add Data". On the left, there is a dark teal sidebar with a "Back" button at the top. Below it, the sidebar contains labels for "Weight:", "Blood Pressure:", and an "Add Data" button. The main content area is light blue and contains input fields for "Calories burned:" (with value 500), "Steps:" (with value 2000), "Distance:" (empty), and "Sleep:" (with values 0 and 00). There are also "Add Data" and "Upload from device" buttons in the main area.

2. The data will be processed for validity and make sure that it is simply not skewed, spam, or invalid data. If the data is recognized as invalid, it will be rejected, and the user will be notified with a pop-up notification in the application. The user will then be allowed to enter data again.

Data Entry Page

This screenshot shows the same "Add Data" page as the previous one, but with a pop-up notification. The notification is a blue box with a yellow warning triangle icon and the text "Data entered is not valid." with an "OK" button. The background shows the "Add Data" page with the "Calories burned:" field containing the value "-5".

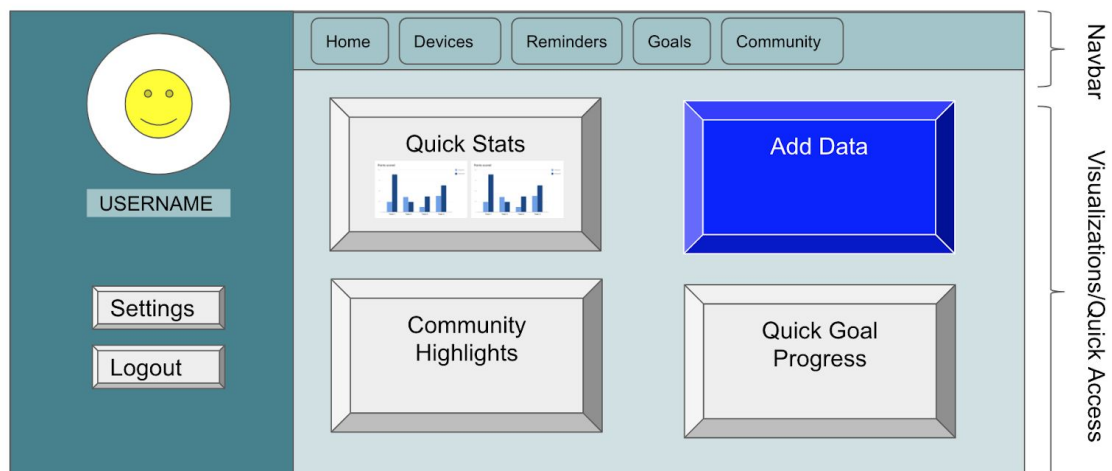
3. Once valid data is entered, the database will add it internally and associate it with the primary key, the user. A confirmation notification will appear for the user.



ii. Use Case 4

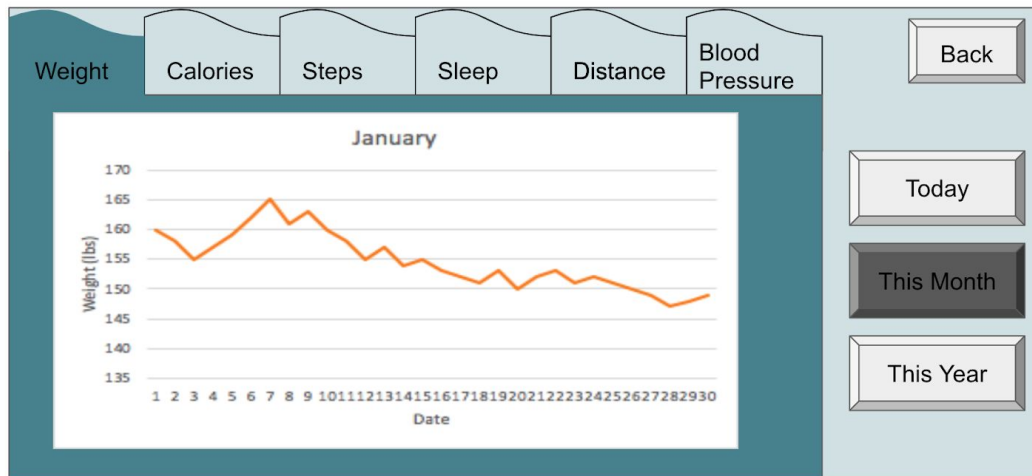
1. User will submit a request to visualize their performance and see their rankings and ratings in relation to other users or track their own progress over time. They will submit this request by pressing the “Quick Stats” button on the home page, and will not necessarily need to input numerical data. Rather, they are inputting a binary action (button press).

Home Page

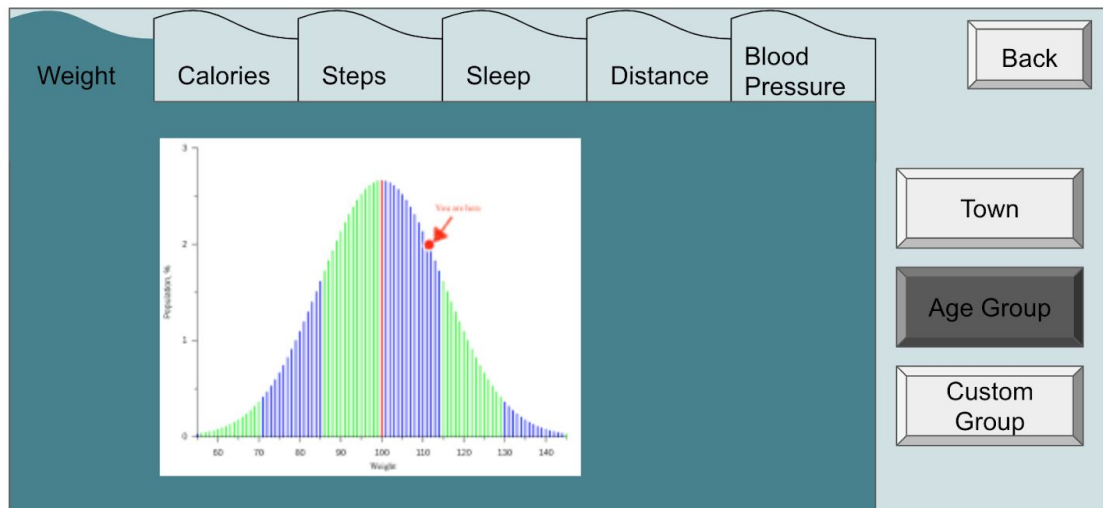


- The DB will pull all relevant information and the visualization module will format it accordingly based on what information was requested, to whom the data is in relation, how much time is in the scope of the query, etc.

User Statistics Page



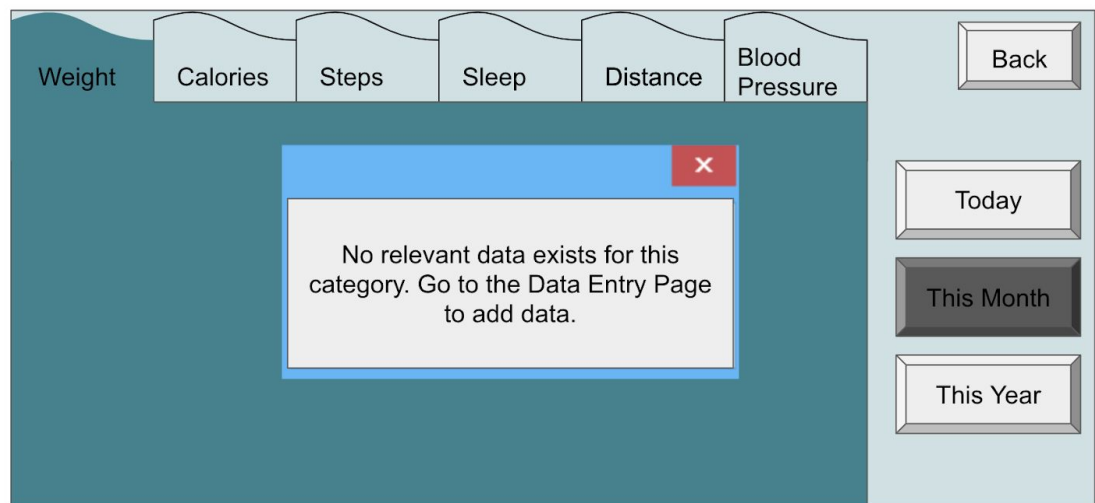
Population Statistics Page



3. The visualization will be displayed on the screen, and the user can freely look through the pictorial representations or return to the other pages of the application at their discretion. The user may also be informed no relevant data exists if that is the case.

For instance, if a user has never input data for their weight, then no relevant data would exist for that case.

User Statistics Page



iii. Use Case 8

1. User opens the application and enters his/her username in the “username” field.

Login Page

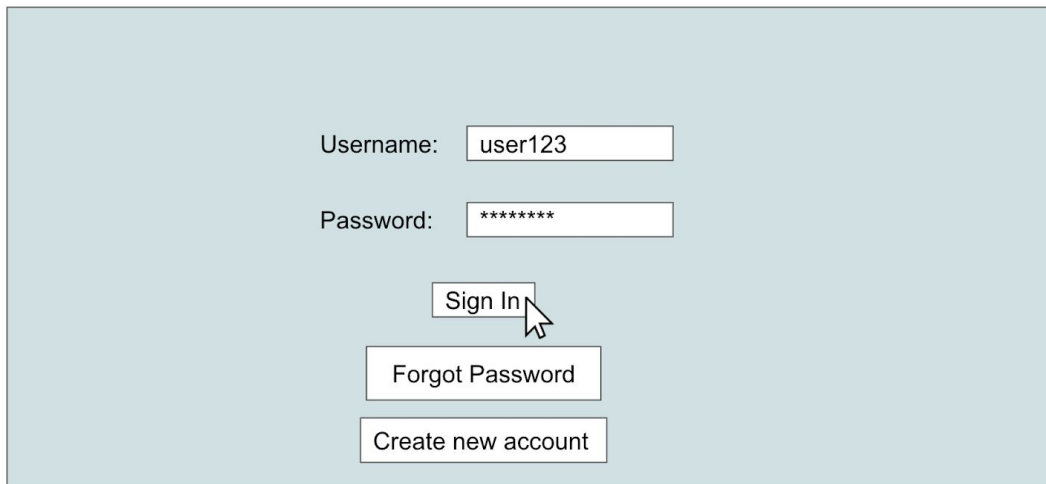


A screenshot of a login page with a light blue background. The page contains the following elements:

- Username:** A text input field containing the text "user123".
- Password:** An empty text input field.
- Sign In:** A button located below the password field.
- Forgot Password:** A button located below the "Sign In" button.
- Create new account:** A button located at the bottom of the form.

2. He then enters the password associated with that username in the “password” field. He then presses the “sign in” button.

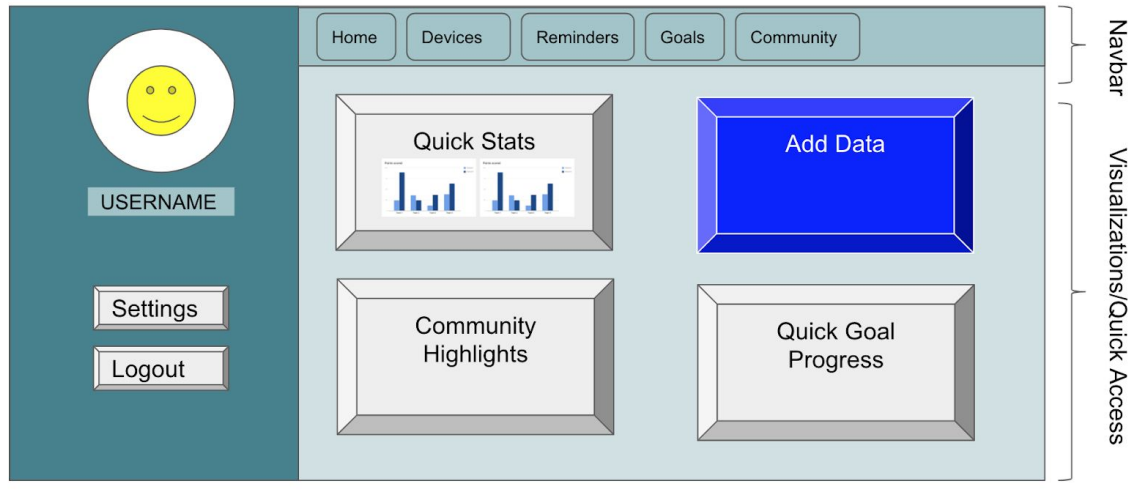
Login Page



A screenshot of the same login page, but now the password field is filled with seven asterisks "*****". A mouse cursor is pointing at the "Sign In" button, indicating it is about to be clicked. The other elements (Username field with "user123", "Forgot Password" button, and "Create new account" button) remain the same as in the previous screenshot.

3. If the username exists and the password entered is the correct password associated with that account, the user is taken to the home screen of his application.


Home Page



4. Otherwise, he is prompted that the entered username/password is incorrect. After 5 incorrect attempts, the forgot password is automatically prompted to the user.

Forgot Your Password?

Enter your email and we'll email you instructions on how to reset your password.



Email:

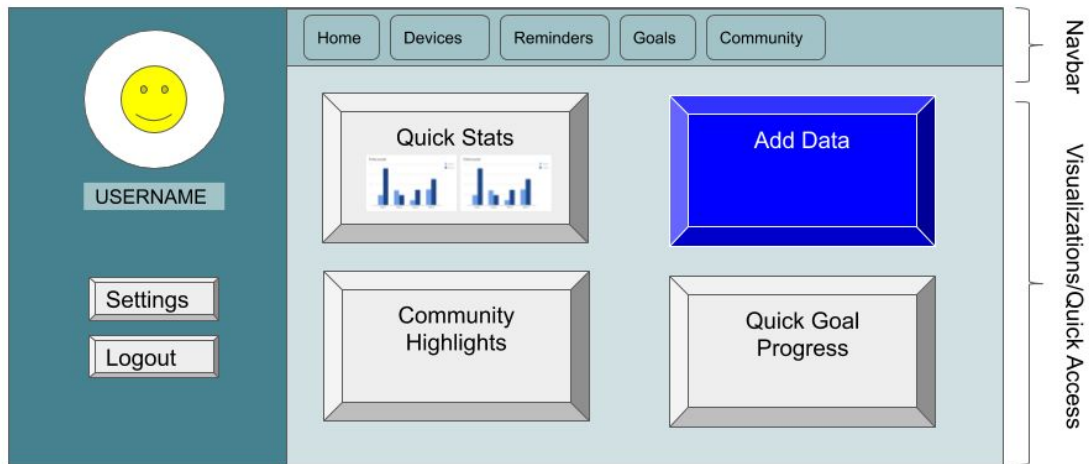
b. User Effort Estimation

Scenario 1: Inputting Daily Personal Data

Assume the application is open to the home page and that the user has logged in. What you need to do is (1) navigate to the Data Entry Page and (2) enter the data that corresponds to you

1. Navigation: 2 Total Mouse Clicks, as follows:
 - a. Click “Add Data” from the Home Screen-- after completing the data entry as referenced below --
 - b. Click “Add Data” on the Data Entry Page
2. Data Entry: 6 total mouse clicks and a variable number of keystrokes, depending on the length of the values you input, as follows:
 - a. Click cursor to the first data field, “calories burned”
 - b. Press the keys that correspond to the number of calories you burned that day
 - c. Press the “Tab” key to move to the next text field, “Steps”
 - d. Press the keys that correspond to the number of steps you took that day
 - e. Press the "Tab" key to move to the next text field, “Distance”
 - f. Press the keys that correspond to the number of steps you took that day
 - g. Click on the hours drop down menu for the number of hours you slept that night
 - h. Scroll on the drop down menu to select the number of hours
 - i. Click on the minutes drop down menu for the number of minutes you slept that night
 - j. Click “Add Data”

Home Page



↓Clicking “Add Data”↓

Data Entry Page

Back

Add Data

Weight:

Blood Pressure:

Add Data

Calories burned:

Steps:

Distance:

Sleep:

0

 :

00

Add Data

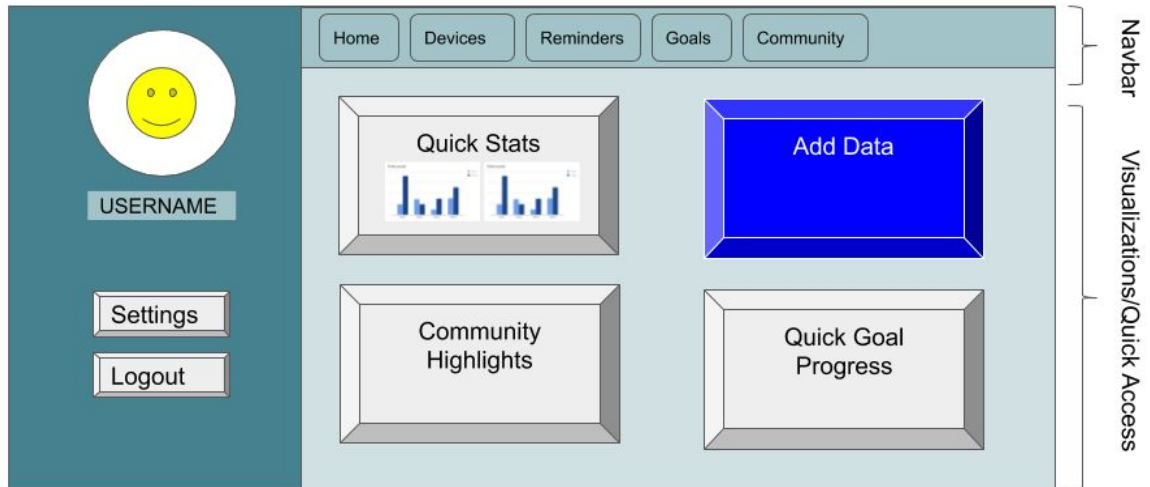
Upload from device

Scenario 2: Updating Goals

Assume the application is open to the home page and that the user has logged in. What you need to do is (1) navigate to the Goals Page and (2) enter the data that corresponds to your goals

1. Navigation: 2 Total Mouse Clicks, as follows:
 - a. Click “Quick Goal Progress” from the Home Screen-- after completing the data entry as referenced below --
 - b. Click “Update Goals” on the Goal Management Page
2. Data Entry: 2 total mouse clicks and a variable number of keystrokes, depending on the length of the values you input, as follows:
 - a. On the Goal Management Page, click into the first field, “Blood Pressure”
 - b. Press the keys that correspond to the updated goal for that field
 - c. Press the “Tab” key to move to the next text field, “Steps”
 - d. Press the keys that correspond to the updated goal for your steps
 - e. Press the "Tab" key to move to the next text field, “Weight”
 - f. Press the keys that correspond to the updated goal for your weight
 - g. Click “Update Goals”

Home Page



↓Clicking “Quick Goal Progress”↓

Goal Management Page

Update Goals

Blood Pressure:

Steps:

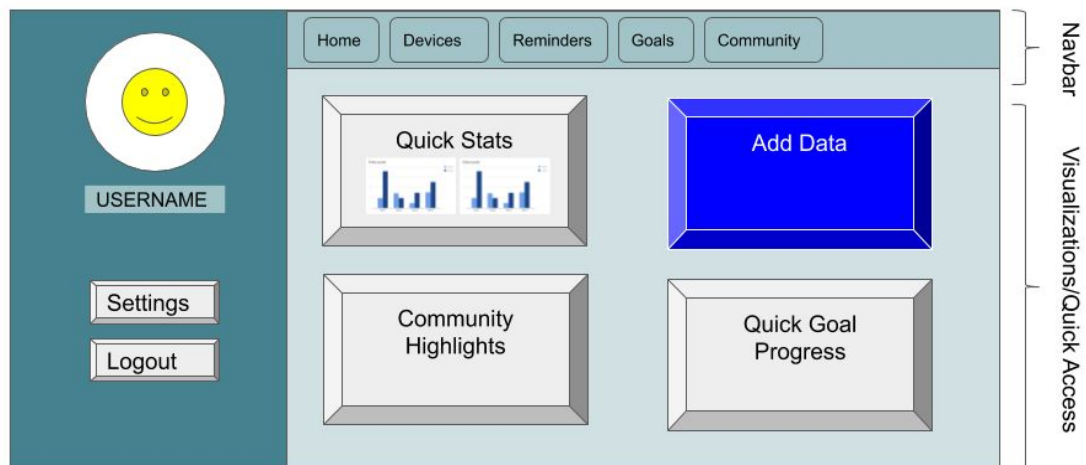
Weight:

Scenario 3: Viewing User Statistics

Assume the application is open to the home page and that the user has logged in. What you need to do is (1) navigate to the User Statistics Page and (2) View all metrics related to their health.

1. Navigation: 1 Total Mouse Click (to get to the User Statistics Page), as follows:
 - a. Click “Quick Stats” from the Home Screen
2. View all metrics: 5 total mouse clicks and 0 key strokes
 - a. Once you get to the User Statistics Page, you will automatically see the first chart, for weight.
 - b. Click on the next tab, Calories, to view your statistical data.
 - c. Click on the next tab, Steps, to view your statistical data.
 - d. Click on the next tab, Sleep, to view your statistical data.
 - e. Click on the next tab, Distance, to view your statistical data.
 - f. Click on the next tab, Blood Pressure, to view your statistical data.

Home Page



↓Clicking “Quick Stats”↓

User Statistics Page



Section 5: Effort Estimation Using Use Case Points

Duration = UCP * PF where UCP = UUCP*TCF* ECF and PF = 28

Given our eight (8) use cases, we know their relative complexities as

Use Case	Actor Weight	Use Case Weight
1	3	5
2	3	5
3	1	10
4	3	15
5	2	5
6	1	5
7	3	15
8	2	5

$UUCP = UAW + UUCW = 83$, $UAW = 2*1 + 2*2 + 4*3 = 18$, $UUCW = 5*5 + 1*10 + 2*15 = 65$

$TCF = 0.6 + 0.01*(2*3+1*3+1*3+1*2+1*5+0.5*1+0.5*4+2*2+1*5+1*5+1*4+1*0) = 0.995$

$ECF = 1.4-0.03*(1.5*2+0.5*2+1*4+0.5*3+1*4+2*5-1*0-1*3) = 0.785$

Therefore, based on these calculations, perceived impacts, and weights our duration estimate will be: $28*(65*0.995*0.785) = 1421.56$

We can also check this for relative accuracy. Assuming each of our team members puts in an average of 12 hours per week into our project with 8 members on the team, we will need to put in about $1421.56/(8*12) =$ roughly 14 weeks, which is about the length of our semester. This seems to check out, given that this is a semester long project.

Section 6: Domain Analysis

i. Concept Definitions

Responsibility	Type	Concept
Stores the user's profile picture, bio, list of posts, etc.	K	Profile
Allows the user to create, edit, or delete their profile.	D	ProfileChanger
Database that stores all user profiles and the information stored within them.	K	ProfileStorage
Attempts to place a user data point into the profile Database Management System (DBMS) if it's within a reasonable range.	D	DataEntry
Ensures that the data point is within a reasonable range. If so, the data point goes into the user's profile.	D	DataChecker
Database that stores all of the data points associated with each profile.	K	DataStorage
Checks the profile DBMS to see if the user put in any data at 10:00 p.m. every day.	D	TaskChecker
If the user hasn't put in their data in at 10:00 p.m., the system will remind the user to do so.	D	Reminder
Upon request, the image generating software creates an image for the user to see their data in their desired form (e.g. bar graphs, histograms, pie charts, etc.).	D	ImageGenerator
Displays the image on the screen after the image generating software creates it.	D	ImageDisplay
Sends an email to the user upon request of changing their password.	D	EmailNotifier
Changes the user's password after the user successfully confirms their email.	D	PasswordChanger
Allows the user to set goals and then store them into the goals DBMS (GoalStorage).	D	GoalSetter
Stores the list of goals associated with each user.	K	GoalStorage

Allows the user to add, block, or delete friends.	D	ModifyFriends
Takes the goals you have from the goals DBMS and then shares them with your friends.	D	ShareProgress
User attempts to enter a password to gain access to the system.	D	PasscodeEntry
Ensures that the user typed in the correct password.	D	PasscodeVerifier
Database that stores the password associated with each user.	K	PasscodeStorage
Check to make sure the user hasn't exceeded the maximum number of attempts to enter a password.	D	Controller

ii. Association definitions

Concept Pair	Association Description	Association Name
ProfileChanger <-> Profile	Changes made using ProfileChanger feature must be visually reflected on user's Profile.	update profile
DataEntry <-> DataChecker	Data user enters during DataEntry must be verified by DataChecker to ensure that the data is valid.	verify data
DataChecker <-> DataStorage	Data must be checked by DataChecker before it is stored in the Database Management System during DataStorage.	store data
TaskChecker <-> Reminder	Reminder will be sent out only if TaskChecker determines that the user has not inputted their daily data by 10:00 P.M.	verify message
ImageGenerator <-> ImageDisplay	ImageGenerator must generate visual before ImageDisplay displays it to the user.	prepare visualization
PasswordChanger <-> EmailNotifier	User must change password with PasswordChanger before EmailNotifier notifies them of their password change confirmation.	confirm password change
GoalSetter <-> GoalStorage	GoalStorage stores goals made by GoalSetter.	store goals
GoalStorage <-> ShareProgress	ShareProgress shares goals stored by GoalStorage.	share goals
PasscodeEntry <-> PasscodeVerifier	PasscodeVerifier verifies the validity of the passcode entered during	check passcode

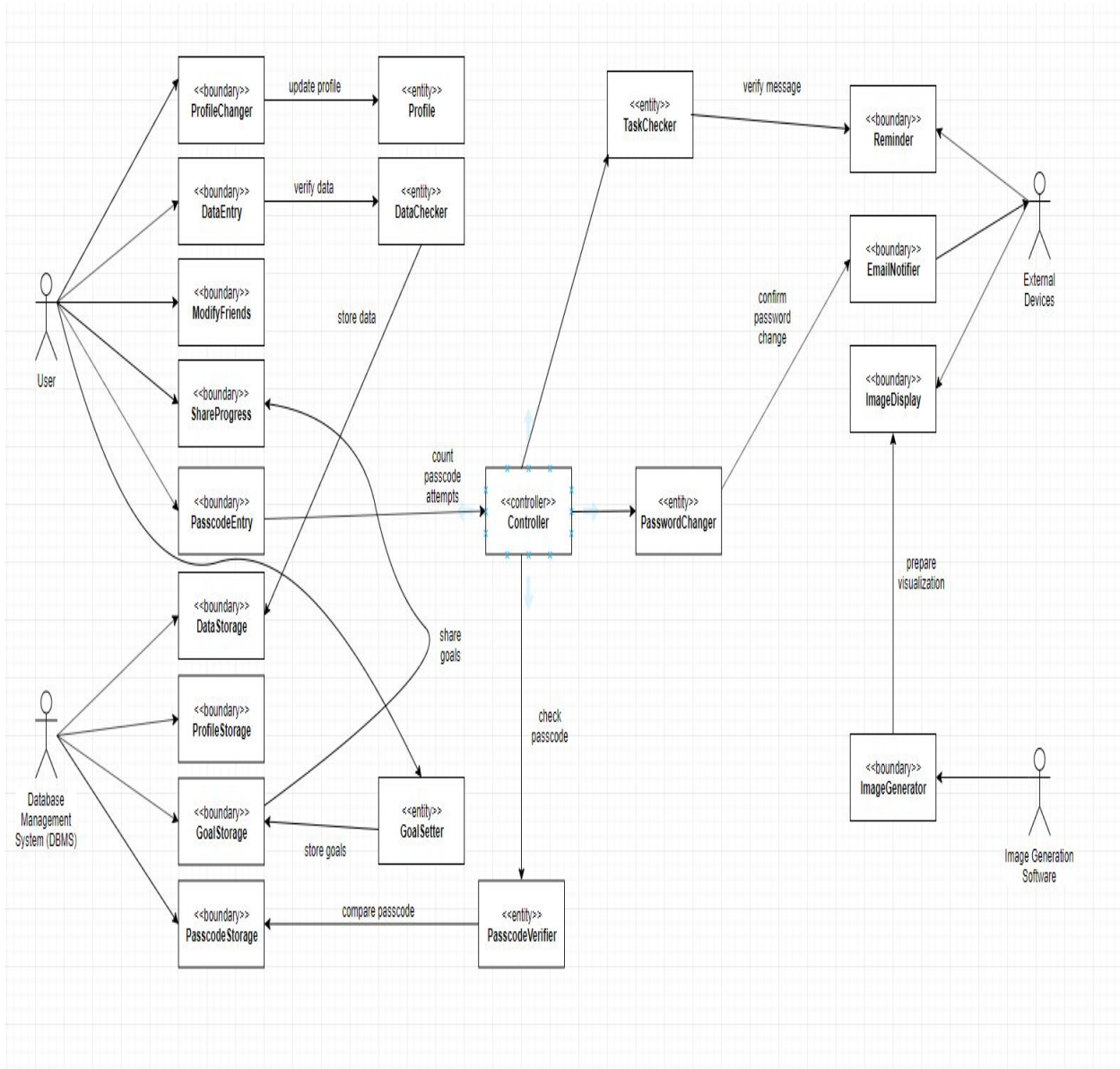
	PasscodeEntry.	
PasscodeVerifier <-> PasscodeStorage	PasscodeVerifier checks user-inputted password against correct password stored during PasscodeStorage.	compare passcode
PasscodeEntry <-> Controller	Controller verifies that count of user's passcode inputs during PasscodeEntry does not exceed a pre-set value.	count passcode attempts

iii. Attribute definitions

Concept	Attributes	Attribute Description
Profile	Cached user settings	Store a variety of user settings in one location which may affect operation characteristics of the UI. These diverse settings must share a common interface for later data persistence
ProfileStorage	Persisted user settings	Using the profile data entered into the Profile, store this data into the database for consistent UX between multiple login sessions
DataEntry	Data validation lookup table	Table of values that determines the acceptable range for data input
DataStorage	User fitness data	Data derived from a user's wearable device
TaskChecker	User activity log	Log within database to track activity in user sessions over time
Reminder	User activity log	This shares the activity log to generate reminder text for the expected user behavior
ImageGenerator	User fitness data	Access to this data required for generating graphics
ImageGenerator	Data visualization	Graphics representative of user data
EmailNotifier	User private information	Securely stored user private information used for access to user data outside of the main application. Mainly used to send emails.
PasswordChanger, PasswordVerifier, PasswordStorage	User credentials	Secure and encrypted user information
GoalStorage, ShareProgress	Goal	Common data structure in database extended by the user to create goals which affect the behavior of the UX for a number of login sessions for a

		single user
ModifyFriends, ShareProgress	Social Media Connector	API access to third party software for user to share information on other digital platforms

a. Domain Model



iv. Traceability Matrix

		Use Cases							
	Use Case	UC-1	UC-2	UC-3	UC-4	UC-5	UC-6	UC-7	UC-8
	Priority Weight	25	47	14	41	6	26	15	37
Domain Concepts	Profile	x							
	ProfileChanger	x							
	ProfileStorage	x							
	DataEntry		x						
	DataChecker		x						
	DataStorage		x						
	TaskChecker			x					
	Reminder			x					
	ImageGenerator				x				
	ImageDisplay				x				
	EmailNotifier					x			
	PasswordChanger					x			
	GoalSetter						x		
	GoalStorage						x		
	ModifyFriends							x	
	ShareProgress							x	
	PasscodeEntry								x
	PasscodeVerifier								x
	PasscodeStorage								x
	Controller								x

b. System Operation Contracts

Operation	UC-1 : User Profile Management
Preconditions	<ul style="list-style-type: none">• User has been authenticated
Postconditions	<ul style="list-style-type: none">• User profile data has been updated in the database• Updated user profile information is immediately updated in the UI for this session

Operation	UC-2 : Data Input and validation
Preconditions	<ul style="list-style-type: none">• User has at least one device registered to the system which collects health and fitness data• Data being entered is deemed valid where validity is determined by the data falling within an acceptable range
Postconditions	<ul style="list-style-type: none">• If the data is valid<ul style="list-style-type: none">◦ New data is persisted to the database◦ Blockchain network updates population to include this data• If the data is invalid<ul style="list-style-type: none">◦ The UI tells the user that the data is valid◦ The application permits the user to try to input data again

Operation	UC-3 : Reminders for input
Preconditions	<ul style="list-style-type: none">• The current time of day is equal to the reminder time in the user's profile and settings
Postconditions	<ul style="list-style-type: none">• <None>

Operation	UC-4 : View visualizations
Preconditions	<ul style="list-style-type: none">• The user has selected the type of visualizations to view
Postconditions	<ul style="list-style-type: none">• The backend provides the UI with the data pertinent to the user's selection

Operation	UC-5 : Secure Password Changing/Verification
Preconditions	<ul style="list-style-type: none">• User has been authenticated

Postconditions	<ul style="list-style-type: none"> Database updates the tables containing login data to reflect changes
----------------	--

Operation	UC-6 : Goal Setting and Progress
Preconditions	<ul style="list-style-type: none"> The user has initiated at least one goal
Postconditions	<ul style="list-style-type: none"> <None>

Operation	UC-7 : Social Networking
Preconditions	<ul style="list-style-type: none"> User has been authenticated The user has connected at least one social media account the application The user has at least one element of data or visualizations which is marked as “shareable”
Postconditions	<ul style="list-style-type: none"> The shareable content is sent over the API for the selected social media platform

Operation	UC-8 : Login
Preconditions	<ul style="list-style-type: none"> <None>
Postconditions	<ul style="list-style-type: none"> If login is successful: <ul style="list-style-type: none"> User has been authenticated UI provides user with login success This login session is persisted to the database with the user credentials, timestamp of the attempted login, and details on rather or not the login was successful

Section 7: Project Size Estimation

The following table highlights the 8 identified use cases in terms of the size in Use Case Weights (UCW). The categories are assigned based on the complexity of achieving their success criteria and their interconnectedness in the Domain Model.

Use Case	Description	Category	Weight
UC-1 Profile creation & management	GUI for profile management and data persistence for user settings	Average	10
UC-2 Data Input and Validation	Controller for receiving user input and interfaces for validating data. Connection to Blockchain layer	Complex	15
UC-3 Reminders for input	GUI element that connects to the User's computer or handle device to display text	Average	10
UC-4 View Visualizations	Receive user selection for generating graphics and use Blockchain layer to display population statistics	Complex	15
UC-5 Password Verification	Securely handle user credentials	Simple	5
UC-6 Goal Setting and progress	UI interaction for storing data related to user interactions over time	Average	10
UC-7 Social Networking	Select certain data from the database to be shared over APIs for common social	Average	10

	media platforms		
UC-8 Login	Secure user authentication and encryption of password storage	Simple	5

Based on the UCW shown above, the Unadjusted Use Case Weight is as follows: $2 * 5 + 4 * 10 + 2 * 15 = 80$.

The following table calculates the technical complexity factor (TCF) based on the community standard constant values determined by the relevance of the technology listed below

Factor	Description	Weight	Relevance (0-5)
T1	Distributed system	2.0	5
T2	Response time/performance objectives	1.0	2
T3	End-user efficiency	1.0	4
T4	Internal processing complexity	1.0	4
T5	Code reusability	1.0	4
T6	Easy to install	0.5	1
T7	Easy to use	0.5	4
T8	Portability to other platforms	2.0	3
T9	System maintenance	1.0	3
T10	Concurrent/parallel processing	1.0	0

T11	Security features	1.0	5
T12	Access for third parties	1.0	4
T13	End user training	1.0	0
Technical Factor (TF)			40.5

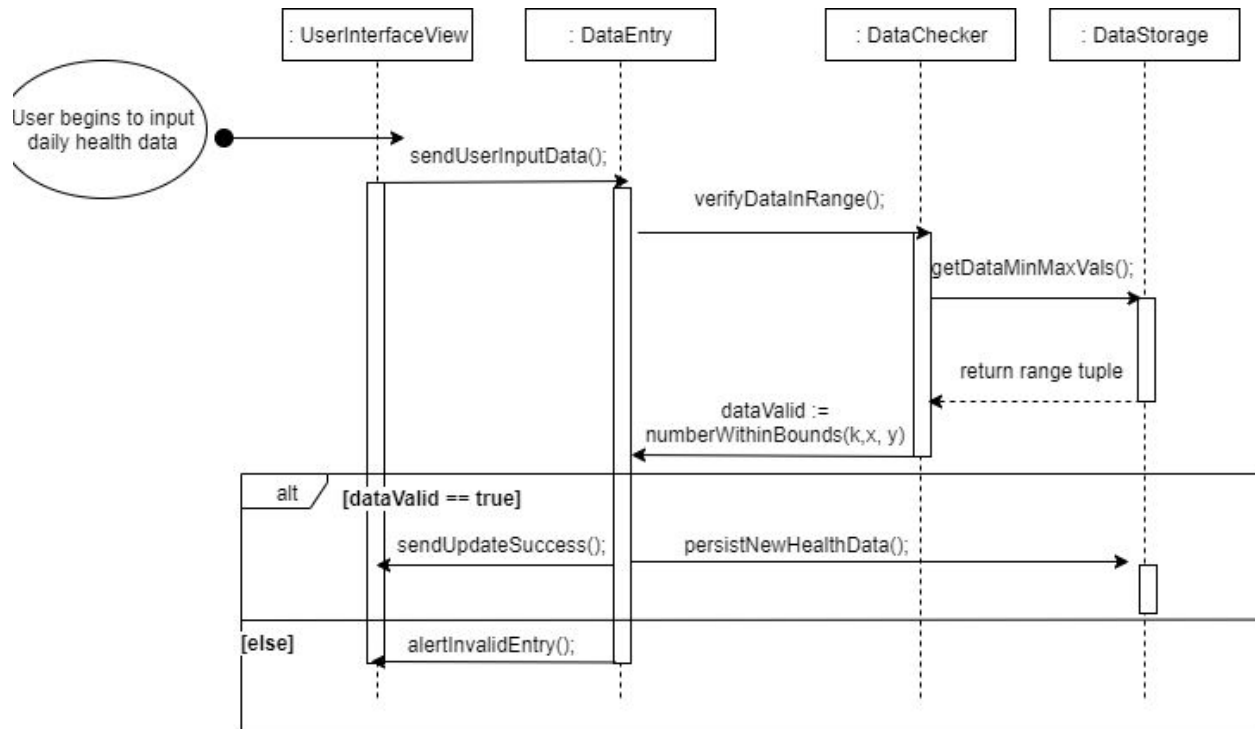
Based on the formula of $TCF = 0.6 + TF/100$, the calculated TCF is 1.005.

With an assumed ECF = 1, the total size of the project as determined by $UCP = UUCW * TCF$
 *ECF is 80.4 UCP

Section 8: Interaction Diagrams

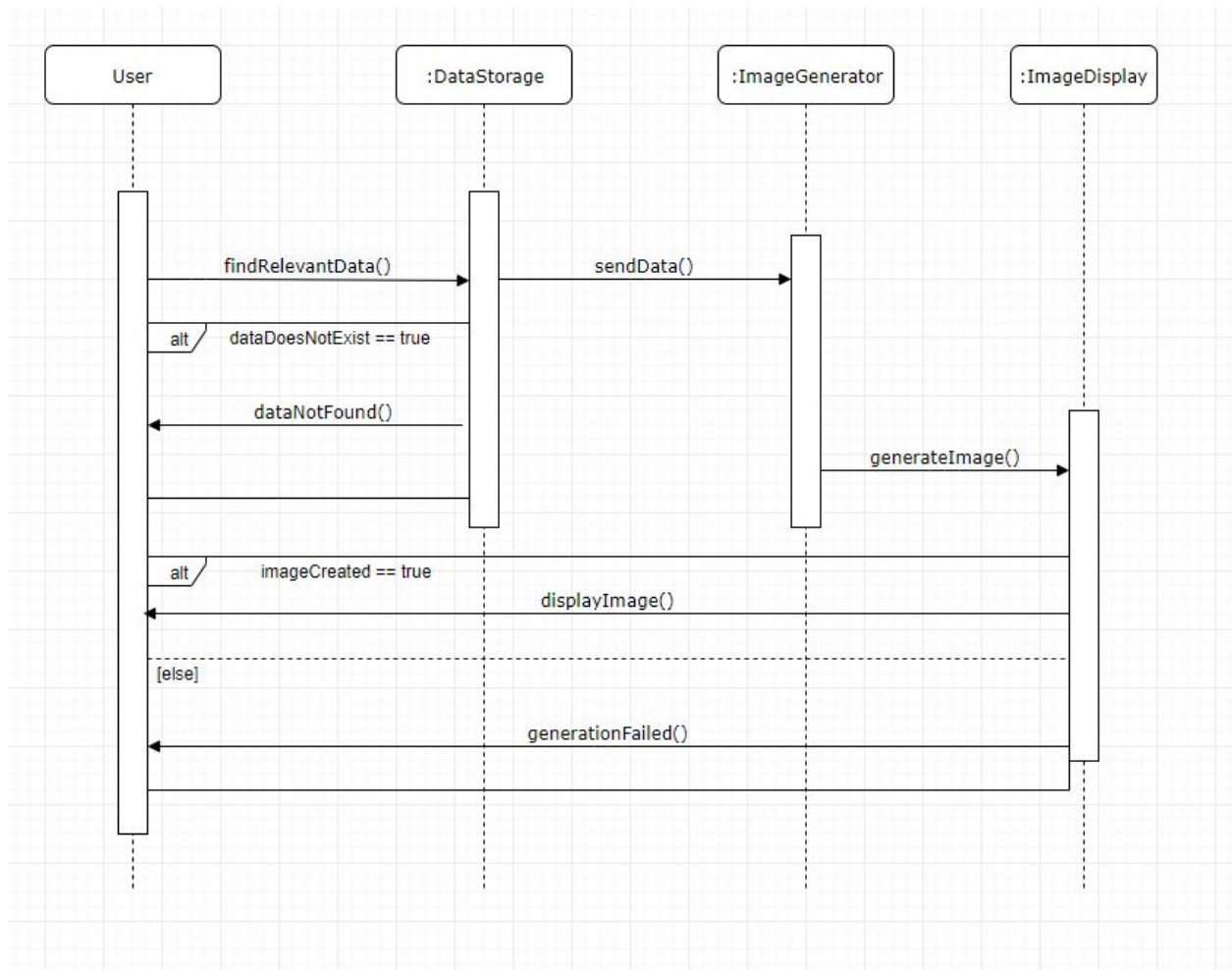
The following sections describe the main use cases of our application in terms of interaction diagrams.

Use Case 2:



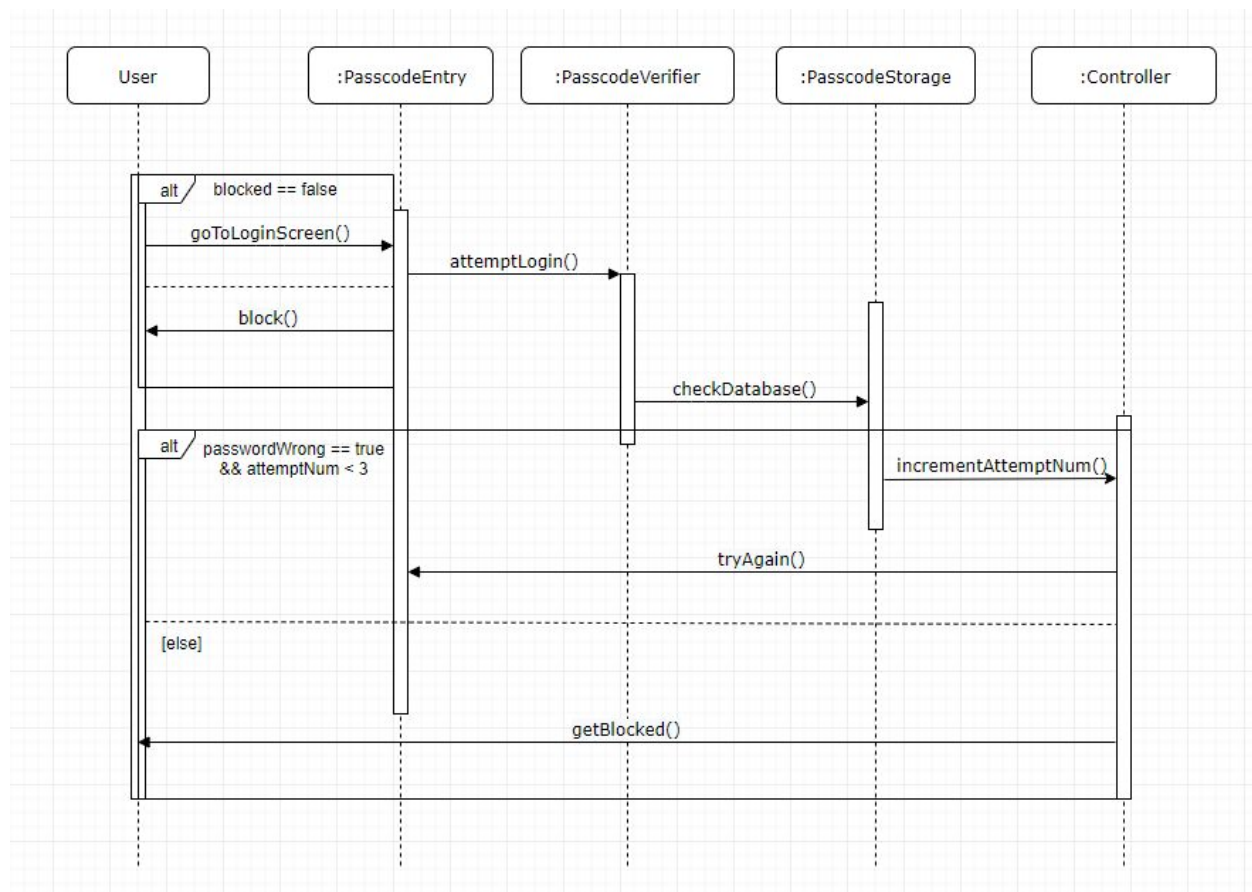
The above interaction diagram is derived from the system sequence diagrams to show the interaction between objects within the system. The design principle guiding this diagram was the main single functionality, low-coupling, for the responsibilities of each module.

Use Case 4:



The interaction diagram was motivated by the system sequence diagram. For each object, the design emphasized keeping the responsibilities (or number of modules) of a single object to a minimum to avoid too much dependency on one object in order to make future refactoring easier.

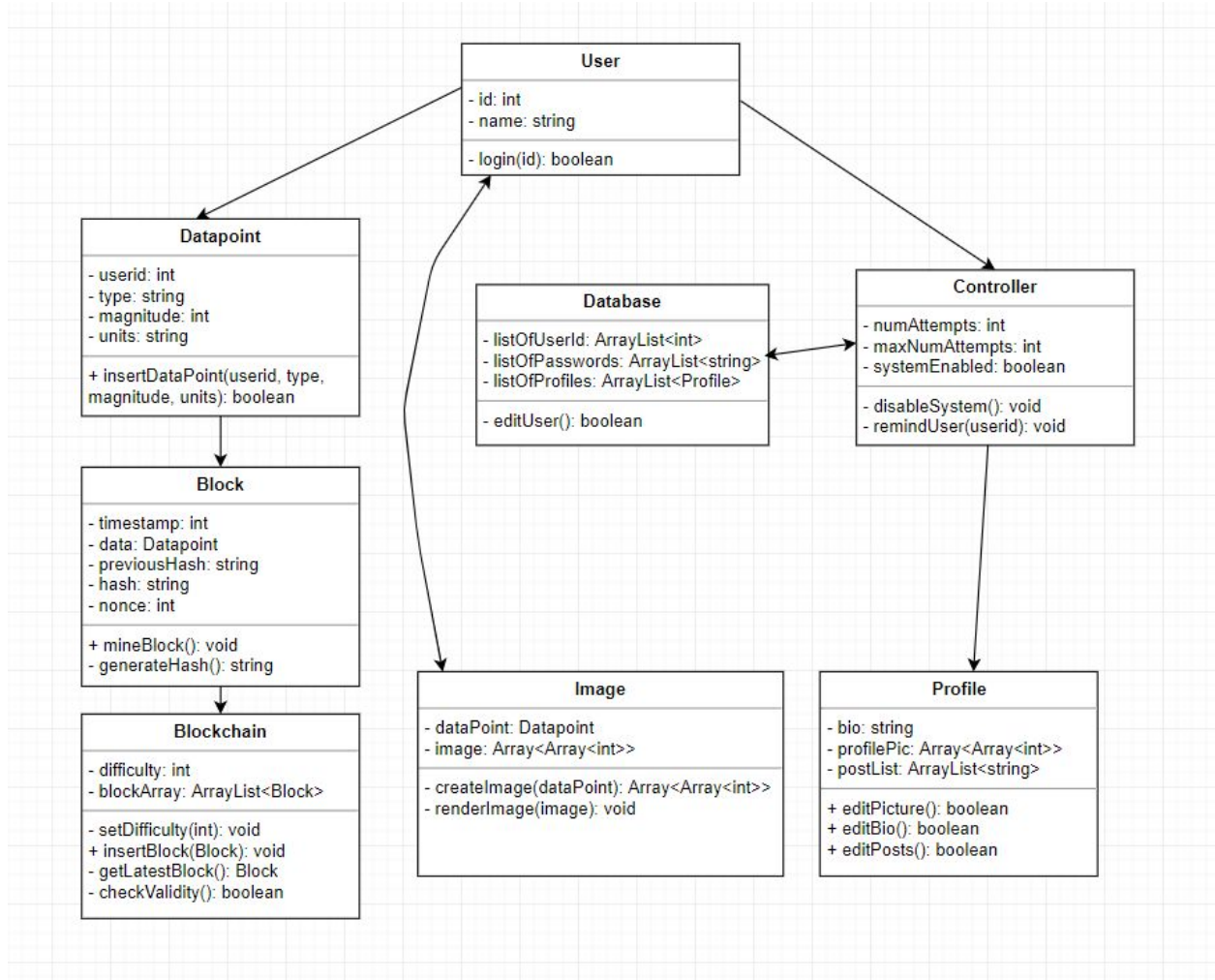
Use Case 8:



Like the other interaction diagrams, this tool was developed utilizing the system sequence diagrams from Report #1. There is a limited amount of functionality for each object, which results in a low amount of coupling. While this might take more effort upfront and require more modules, this will save effort in the long term because refactoring will be streamlined due to fewer dependencies on other objects.

Section 9: Class Diagram and Interface Specification

a. Class Diagram



b. Data Types and Operation Signatures

Profile Class:

Attributes:

-bio: string

- This is a string variable that refers to the bio of the user.

-profilePic: Array<Array<int>>

- Since pictures are stored using 2D arrays of integers that represent rgb values, the profile picture of the user is stored in an array of array of integers.

-postList: ArrayList<string>

- Since each post consists of a string, the complete list of posts that a user has is represented by an ArrayList of strings. It's stored in an ArrayList instead of an array so that the list of posts is mutable, making it easier to alter the list of posts that a user has.

Operations:

+editPicture(): boolean

- This is a boolean function that allows the user to upload or delete a profile picture. Upon failure, this function returns false. This could happen when the user uploads something that is not a picture.

+editBio(): boolean

- This is a boolean function that allows a user to upload a new bio, edit a pre-existing bio, or delete their bio. This function fails and returns false when the user tries to upload a bio that exceeds the character limit.

+editPosts(): boolean

- This is a boolean function that allows a user to upload a new post, edit a pre-existing post, or delete any post of their choosing. This function fails and returns false when the user tries to upload a post that exceeds the character limit.

Datapoint Class:

Attributes:

-userid: int

- This int parameter keeps track of the userid that this particular data point is linked to.

-type: string

- This string parameter tells you what type of parameter has been inserted (e.g. hours of sleep, miles walked, push-ups performed, etc.).

-magnitude: int

- This int parameter tells you how much of a certain action was performed (e.g. you slept for 8 hours last night).

-units: string

- This string parameter gives you the units of the particular data point you just entered (e.g. miles, hours).

Operations:

+ insertDataPoint(userid, type, magnitude, units): boolean

- This boolean method inserts the data point into the blockchain and returns true if the data point consists of a valid userid, the type is within a predetermined set of values, the magnitude is within a reasonable range, and the units are acceptable. Upon failure of any of these conditions, it returns false.

Database Class:

Attributes:

-listOfUserId: ArrayList<int>

- This attribute stores the list of userid's in the system.

-listOfPasswords: ArrayList<string>

- This attribute stores the list of passwords in the system.

-listOfProfiles: ArrayList<Profile>

- This attribute stores the list of profiles in the system.

Operations:

-editUser(userid): boolean

- This boolean method alters the profile data linked to a particular userid, whether it is adding a new user or deleting a pre-existing user from the database and it returns true upon success. Otherwise, it will fail and return false.

Controller Class:

Attributes:

-numAttempts: int

- This int attribute gives the current number of failed consecutive attempts to get into the system.

-maxNumAttempts: int

- This int attribute gives the maximum amount of consecutive failed attempts that are allowed before the system is no longer enabled and the user has to change their password.

-systemEnabled: boolean

- This boolean attribute tells you whether the system is enabled or not.

Operations:

disableSystem(): void

- This method disables the system after an incorrect password has been entered three consecutive times in a row.

remindUser(userid): void

- If the user hasn't put in any data that day, this method will send a message to the user reminding them to input their data for that day.

Image Class:

Attributes:

-dataPoint: Datapoint

- This attribute gives the image class the information it needs to generate an image based on the data point that has been entered by the user.

-image: Array<Array<int>>

- This attribute stores the image of the user's progress that will be shown to the user.

Operations:

+createImage(dataPoint): Array<Array<int>>

- This method creates an image (a 2D array of integers that represent rgb values) using the data point that has been given as a parameter.

+renderImage(image): void

- This method shows the image stored in the image parameter to the user.

User class:

Attributes:

-id: int

- This attribute stored the userid associated with this user.

-name: string

- This attribute stores the name of the user.

Operations:

+login(userid): boolean

- This method logs the user into the system. This will return false if the user entered an incorrect password.

Block class:

Attributes:

-timestamp: int

- This int attribute tells you how many milliseconds have passed since January 1, 1970 up until the creation of this block.

-data: Datapoint

- This attribute is an instance of the Datapoint class that details the type of data that is within this particular block.

-previousHash: string

- This string attribute tells you the hash of the previous block that points to this current block. This is used to ensure that the blockchain retains its validity.

-hash: string

- This string attribute is a calculated value that depends on the value of every other attribute. If any of the attributes get tampered even slightly, there will be a dramatic change in this value and it will render the blockchain invalid.

-nonce: int

- This int attribute is used as a noise value that aids in the process of mining a block until it contains a hash that is prepended by a certain number of zeros. This number of required zeros is determined by the difficulty attribute in the Blockchain class.

Operations:

+mineBlock(): void

- This void method tries to find a hashcode that has a sufficient number of zeros prepended to it. Once it does that, the new block can now become a part of the blockchain.

-generateHash(): string

- Using all of the other data found within the block, this method generates a hashcode for this current block.

Blockchain class:

Attributes:

-difficulty: int

- This determines how many zeros should be at the beginning of a hash in order for it to be accepted as secure enough for secure use.

-blockArray: ArrayList<Block>

- This attribute is the current list of the blocks that are currently in the blockchain.

Operations:

-setDifficulty(int): void

- This method takes an int parameter that determines how many zeros a hash should have in the beginning in order for it to be deemed as secure enough for general use.

+insertBlock(Block): void

- This method inserts the block into the blockchain. There is no need for this method to be a boolean because the verification of the data occurred before the data enters the blockchain.

-getLatestBlock(): Block

- This method returns the block that has been placed into the blockchain the most recently.

-checkValidity(): boolean

- This boolean method tells the caller whether the blockchain is still valid or not. If a rogue agent has attempted to modify the blockchain, this method will return false.

c. Traceability Matrix

		Classes							
	Class	Profile	Datapoint	Database	System	Image	User	Block	Blockchain
Domain Concepts	Profile	x							
	ProfileChanger	x					x		
	ProfileStorage			x					
	DataEntry		x				x	x	x
	DataChecker		x						
	DataStorage		x				x	x	x
	TaskChecker				x				
	Reminder				x				
	ImageGenerator					x			
	ImageDisplay					x			
	EmailNotifier				x				
	PasswordChanger			x			x		
	GoalSetter			x			x		
	GoalStorage			x			x		
	ModifyFriends			x			x		
	ShareProgress						x		
	PasscodeEntry			x	x		x		
	PasscodeVerifier			x	x				
	PasscodeStorage			x					
	Controller				x				

d. Design Patterns

Our use of the specific design patterns in the particular interaction diagrams improves the design. Design patterns help anticipate software change, and change is needed to keep up with the reality.

We used a **Publisher-Subscriber Pattern**, which improved the design by decreasing coupling, increasing cohesion, and disassociating unrelated responsibilities. It helps simplify and remove complex conditional logic. This improved our design because we focused on detecting and dispatching events.

The focus is on the Publisher object and the environment that it is observing for the events.

For example: The user wants to login

- Entered login is valid
- Entered login is invalid

Publisher:

Controller (receives login info)

LoginChecker (checks Login Validity: classifies as valid/invalid)

Subscriber:

UI Control

Database Control

- These then do the work based on the received event.

Another example is when Checking the Validity of daily data input.

Either:

- Entered data is valid OR
- Entered Data is invalid

Publisher:

Database (Receives Data)

ValidityChecker (checks Data Validity)

Subscriber:

UI Control

Database Control

- These will then respond accordingly to the received event.

The **Command Pattern** was also used and also has benefits such as the fact that execution is usually called with other business logic, so it is decoupled from preparation. With this, different codes can evolve independently by different developers. These commands can also be reversed/undone which was beneficial.

e. Contracts

A contract is used in software design for various reasons. Such reasons include avoiding misunderstandings and bugs, supporting clear documentation of all modules, and helping clients understand the purpose of the code. All contracts require a precondition and a postcondition.

For Use Case 2 in which the User must input their daily health data into the database:

context User::login()

pre: status = USERSTATUS::loggedin

post: status = USERSTATUS::loggedin and Inputting

For Use Case 8 in which the application must generate user-requested visualizations:

context Data::inputted()

pre: status = DAILYDATA::inputted

post: status = DAILYDATA::inputted and isGeneratingImage

Other Constraints that exist in the project are that:

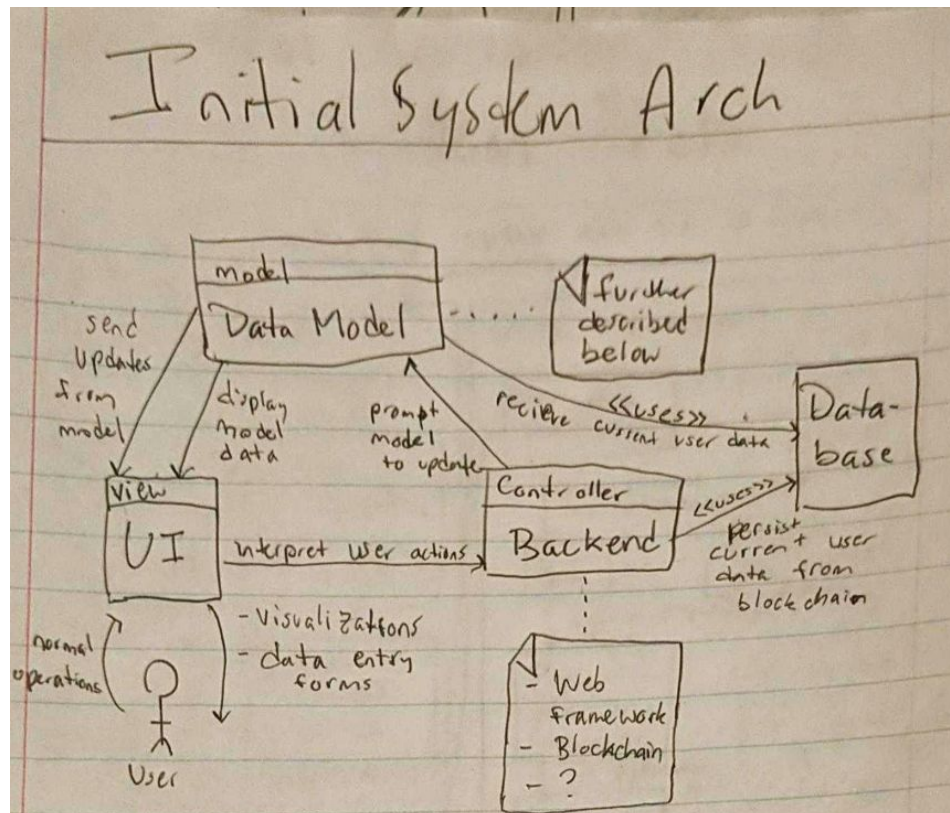
- context Reminder inv:
self.HourOfDayWhenMessagesSent = 22
- context PasscodeEntry inv:
self.NumberOfIncorrectEntries <= 5
- context TaskChecker inv:
self.NumberOfTasks <= 10
- context PasswordChanger inv:
self.DaysBetweenPreviousPasswordChange <= 7
- context EmailNotifier inv:
self.TimeAfterPasswordChangeRequestedInSeconds <= 30

Section 10: System Architecture and System Design

a. Architectural Styles

The architectural style of our application is mainly classified as a Multitier architecture in which the presentation of our application is separated from the actual processing component as well as the data management component. The presentation layer, or the UI/view layer is what the user will see and deal with. When users log in, view their personal data, view general population metrics, update their profiles, etc. it will all be done on this layer. This layer is largely built using Javascript/HTML to allow easy navigation throughout the application. The processing component of our application is the blockchain which securely stores user information/data metrics in blocks. Each user entry is a block which is associated to the respective user, and others will not be permitted to view anyone else's personal information or data. This layer is also built using javascript. The data management component is where all user profiles and data points are stored. For example, user credentials and personal information such as date of birth, gender, weight, etc. will be stored here. Our database management system is based on a SQL server which will be performing this storage.

b. Identifying Subsystems



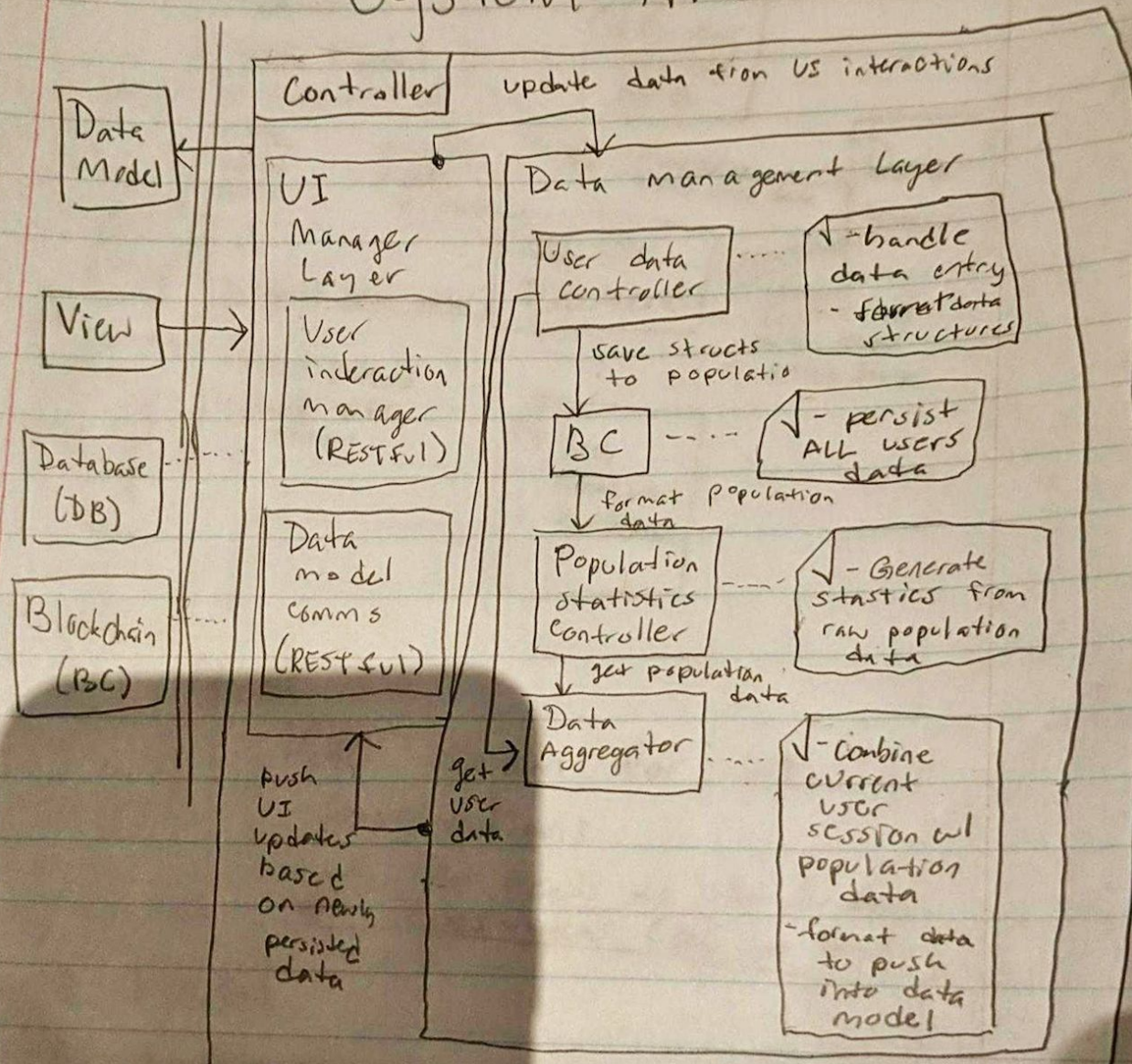
Notes:

- MVC system for UI interactions
- Controller needs to be broken into subsystems (consider layered arch)

Data Model

- User preferences
 - eg: active devices, reminder schedule
- User device data
 - eg: steps walked, weight, blood pressure
- ~~Generated~~ Generated Population statistics

Initial Controller System Arch



c. Mapping Subsystems to Hardware

Because our application is based on blockchain and the whole motivation behind it is for multi user functionality, it is correct that our system should run on multiple computers. One user who submits his/her metrics for that day should be taken into account on another user's computer in regards to the public population metric. For example, if a 30 year old female enters 2.6 miles walked that day, then this value should be taken into account for the "30-35 Year Old Female" bucket in regards to miles walked. This value should be added and averaged with all the other values in this field. Thus, when another user (let's say a 31 year old woman) accesses her data and would like to see population metrics, the value from the first user should have an effect on the average and thus display a new value which takes into account her value. Our client is currently a web browser which the user can access, and it is being hosted on a WAMP server. The web browser can be accessed from any computer, while the WAMP server is hosted independently and externally.

d. Persistent Data Storage

The application has two types of storage requirements which drive the design of the two persistent data storage methods implemented. The program needs to store the fitness data of each user, which must persist when the user closes the application. This data storage will be done through a blockchain system so that the data is secure. The program must also store common app data and user session information. This is performed in a common Relational Database System (RDBS). Through this dual-implement system, the application accesses the secure blockchain for sensitive health data but also maintains well-organized and easily queryable databases for user session data.

e. Network Protocol

The app is to be hosted on a website capable of performing user interactions client-side and handling data storage needs server-side. For client-to-server communications, HTTP is used to serve the site files to the user's web browser from the central server. For client-side behavior, multiple javascript files must be handled within the system. The communication between these multiple files is performed in a Single Page Application Router which uses REST APIs to load HTML and js data from multiple pages on a single page.

f. Global Control Flow

For interactions within the application, the execution orderliness of the app is event-based. All actions within the app, such as adding data, viewing statistical visualizations, modifying settings, or managing goals, must be initiated by clicking the button that brings that view to the main window. Retrieval of data to display a page is performed by the initiation brought forth by the button click. Further actions such as data validation or data persistence is once again, triggered by user interaction.

With regards to time dependency, there is a timer in the system that would alert the app user to input their data at the user's preferred time selected every night if they have not already done so.

g. Hardware Requirement

The application was designed to be cross-platform. As such, any device capable of accessing a web browser should be able to run the app. There are no resolution requirements as the bootstrap-framework-based UI is built for responsive design. In order to run the app, the browser used must be able to run JavaScript.

Section 11: Data Structures

Arrays are being used to store the blockchain structure, and the main reasons an array was chosen was for simplicity, difficulty of removal, and ease of access of each element. Using arrays instead of linked lists makes it so that an element can be added to the chain by simply using the `push()` operation in JavaScript instead of manipulating pointers and having to possibly traverse the entire list in the case of linked lists. Also, a blockchain should be designed such that elements are difficult to remove after placing them into the blockchain. An array would be a better choice than a linked list because in a linked list, removal is constant time when you have access to the element you want to remove. However, arrays have $O(N)$ removal time if there are N elements in the array (or in this case, N blocks in the chain). Additionally, you would potentially have to shift all of the elements in the array instead of manipulating a few pointers like you would after removing a node from a linked list. Therefore, it would be more computationally expensive to remove a block in an array-based blockchain than it would be in a blockchain stored in a linked list, which is the desired behavior of a blockchain structure. Lastly, ease of access for certain elements is required in some instances. For example, we will need to access the last block in the chain to obtain the latest block that has been added to our blockchain. In a linked list, we could potentially use a dummy node to keep track of where the last element is in a linked list, but that would take up additional memory that wouldn't be necessary in an array. We will also need to sequentially access elements in our blockchain during the validation phase, and using an array facilitates this process.

Section 12: User Interface Design and Implementation

Use Case 2: Data Input/ Validation

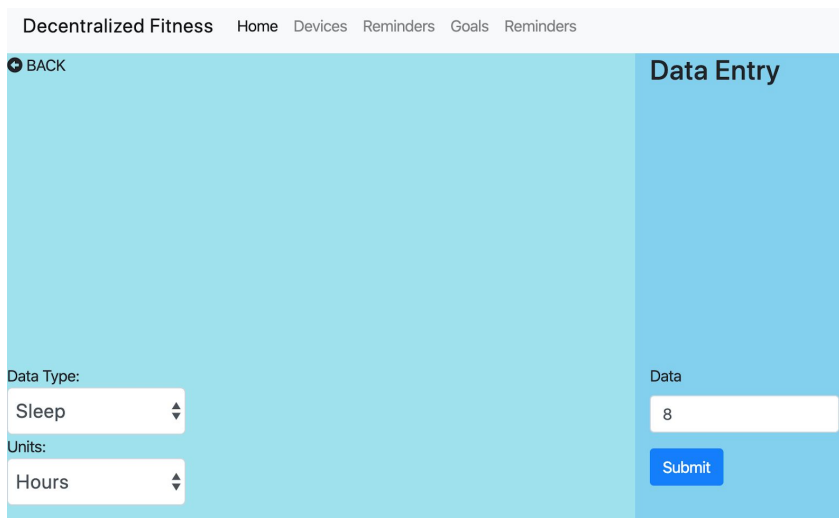
Initial Mockup

Data Entry Page



The initial mockup of the Data Entry page is divided into two main sections. The left section, with a dark teal background, contains input fields for 'Weight' and 'Blood Pressure', and an 'Add Data' button. The right section, with a light blue background, contains input fields for 'Calories burned' (500), 'Steps' (2000), 'Distance', and 'Sleep' (0:00), along with an 'Add Data' button and an 'Upload from device' button. A 'Back' button is located in the top left corner.

Current Implementation



The current implementation of the Data Entry page features a navigation bar at the top with links: 'Decentralized Fitness', 'Home', 'Devices', 'Reminders', 'Goals', and 'Reminders'. Below the navigation bar, there is a 'BACK' button. The main content area is split into two columns. The left column has a 'Data Type' dropdown menu set to 'Sleep' and a 'Units' dropdown menu set to 'Hours'. The right column has a 'Data' input field with the value '8' and a blue 'Submit' button.

We changed the screen mockups to make it easier to use for our customers. In the initial mockup, we split the “Data Entry” page into two sides, where one side was used to add the user’s weight and blood pressure and the other side of the page was used to add the number of calories burned, the amount of steps taken, the distance walked, and the hours slept. However, this was confusing for the user because there were two “Add Data” buttons, one on each side of the screen. If we kept the design the same, then it is possible for users to click on the wrong “Add Data” button, which will result in the user’s information not being saved. Therefore, to make it easier to use and less effort overall, we decided to have one “Submit” button, rather than two “Add Data” button.

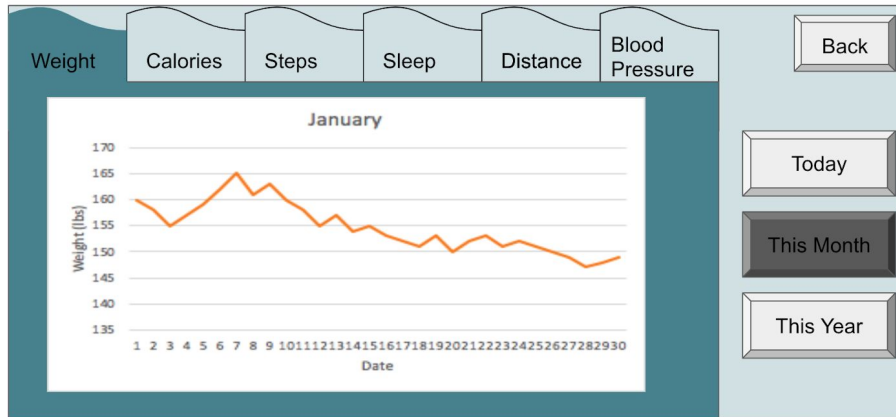
The image displays three panels of a user interface mockup, each featuring a light blue background and a white input area. The first panel on the left has a 'Data Type:' label above a dropdown menu showing 'Walking distance' with up and down arrows. Below it, a 'Units:' label is above a dropdown menu showing 'Kilometers' with up and down arrows. The middle panel shows a 'Data Type:' dropdown menu with a list of options: 'Sleep', 'Food', 'Walking distance' (which has a checkmark), and 'Heart Rate'. Below this, a 'Units:' dropdown menu shows 'Kilometers' with up and down arrows. The third panel on the right has a 'Data Type:' dropdown menu with a list of options: 'Hours', 'Calories', 'Steps', 'Kilometers' (which has a checkmark), 'Miles', and 'BPM'.

We also noticed that our mockup did not take into account units for each of the categories. Therefore, if a user put in the value “2” for distance, it could be kilometers or miles or any other unit used for measuring distance, but the application would not know. Having customers put in the units themselves requires more effort from the users’ end. It can also lead to issues if the unit is spelled incorrectly, or if an abbreviation is used, such as km for kilometers. As a result, we changed the way the data would be inputted by the user. Instead of putting in all the data at once, which can be overwhelming and confusing for users, we created a drop down menu called “Data Type”, to allow users to select the data category they want add information for. Below that is another drop down menu called “Units”, which is used to select the appropriate unit for the data type chosen. Having a drop down menu is easier to operate because users will already know what units the application accepts, which results in less issues when adding their information. Users will also be able to save time by not having to type out this information themselves.

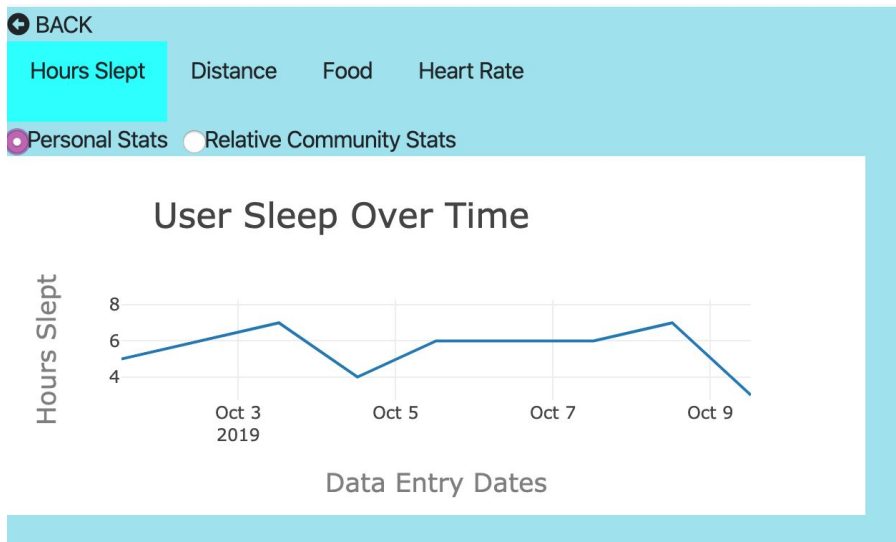
Use Case 4: Viewing Visualizations

Initial Mockup

User Statistics Page



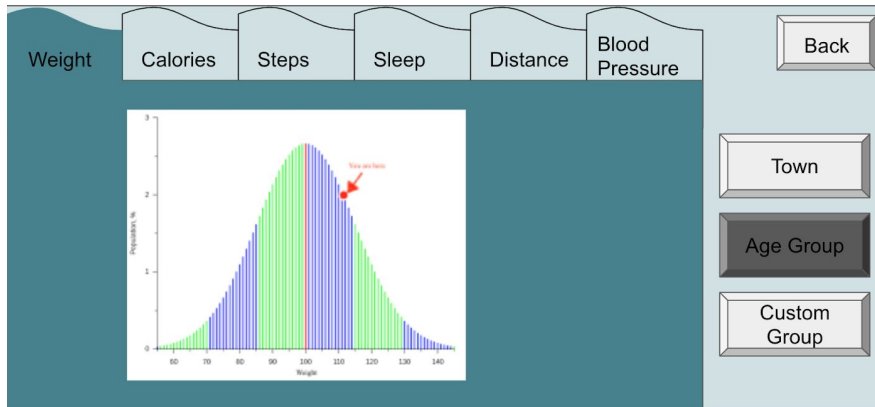
Current Implementation



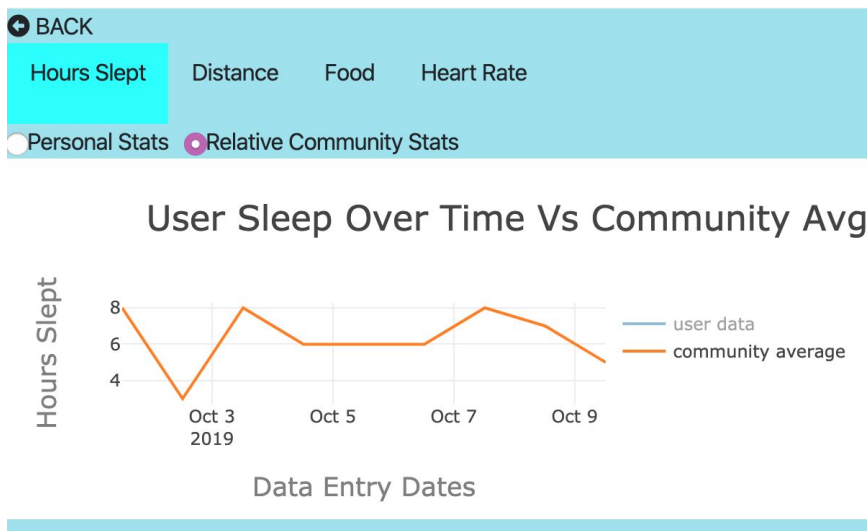
The visuals for the user's data are very similar to the original mockup. Although the implemented page may look a little different visually, the functionality remains the same. The user is able to view his/her personal information as a graph, which allows the user to see his/her data in a meaningful way.

Initial Mockup

Population Statistics Page



Current Implementation



The visuals for the community data are similar to the original mockup. Although the implemented page may look a little different visually, the functionality remains the same. The user is able to view community statistics from a specific population as a graph, which allows the user to see where the general population lies for a specific category.

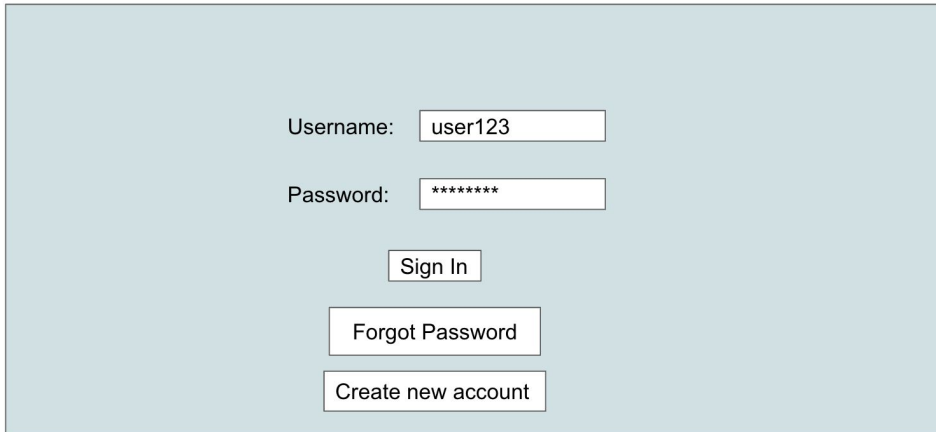


We also improved upon our mockup to make it easier for users to compare their data with the community data. This way, users will be able to see where they lie in relation to the general population. This will allow users to use these comparisons to make any necessary improvements or changes to their current lifestyle. Therefore, being able to look at the two graphs at once makes it easier and a more fluid transition between viewing only personal statistics and viewing performance in comparison to the general population.

Use Case 8: Log In

Initial Mockup

Login Page

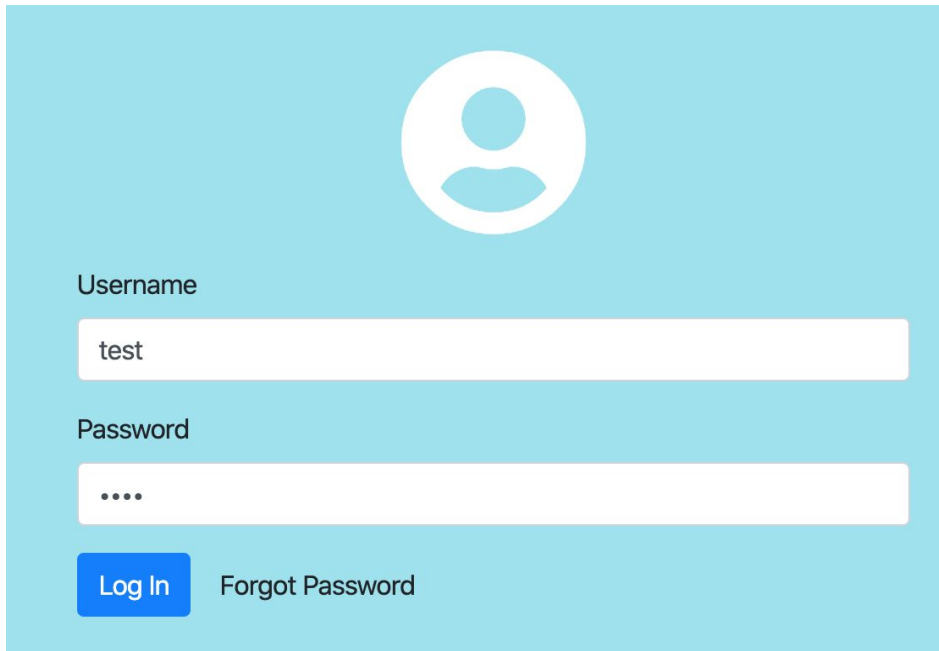


A mockup of a login page with a light blue background. It features a 'Username:' label followed by a text input containing 'user123'. Below it is a 'Password:' label followed by a password input showing seven asterisks. Three buttons are stacked vertically: 'Sign In', 'Forgot Password', and 'Create new account'.


Username:

Password:

Current Implementation



A screenshot of the current login page implementation. It has a light blue background and a white circular user icon at the top. The 'Username' label is above a text input containing 'test'. The 'Password' label is above a password input showing four dots. At the bottom, there is a blue 'Log In' button and a 'Forgot Password' link.



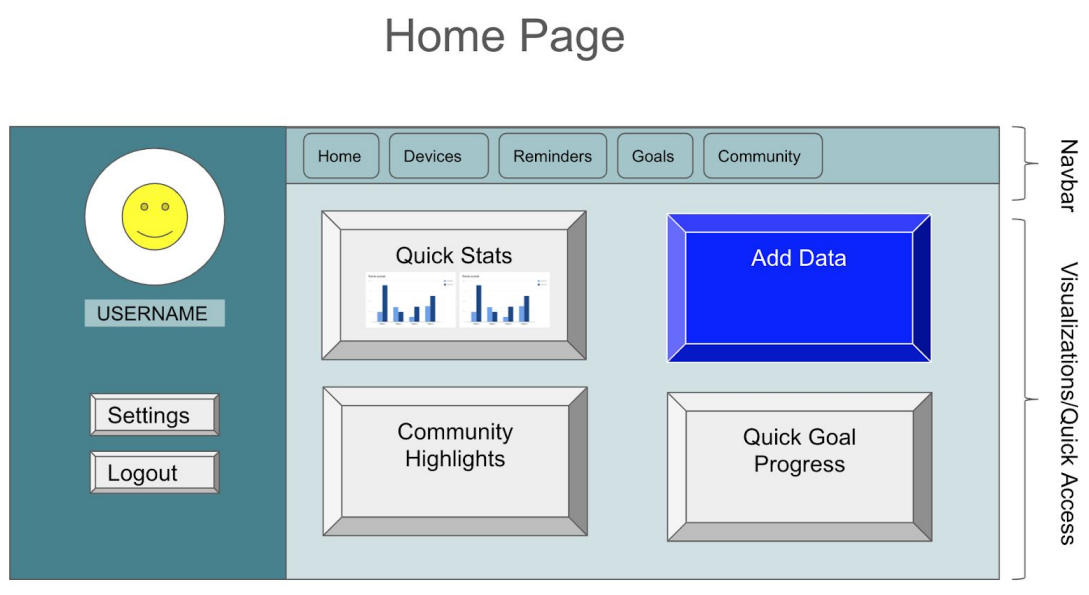
Username

Password

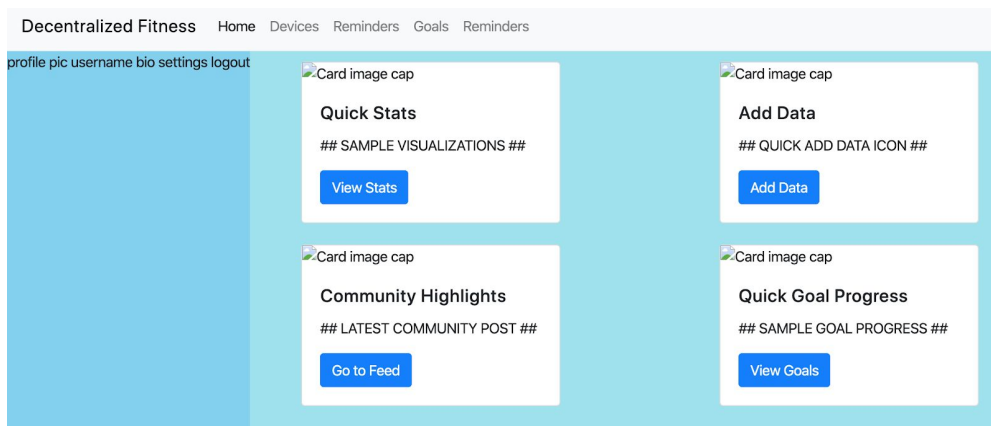
[Forgot Password](#)

The visuals for the login page are very similar to the original mockup. Although the implemented page may look a little different visually, the functionality remains the same. The user is able to log in with his/her correct username and password.

Initial Mockup



Current Implementation



The visuals for the home page are very similar to the original mockup. Although the implemented page may look a little different visually, the functionality remains the same. Once the user logs into his/her account, he/she will first see the home page and can access other pages from the home page.

Section 13: Design of Tests

a. Unit Testing

For the website portion of the application, the first step of testing is navigating to the following URL: <http://se-g3-decentralizedfitness.000webhostapp.com/>. Then, the next portion of testing involved navigating to the Data Entry section and then putting in data for various activities. Entering a data point should yield a pop-up message along with the data point saying that it has been successfully inputted.

With regards to verifying if the blockchain works correctly, two series of tests were run during the first demo. The first series of tests added two blocks of the blockchain, and this showed that the proof of work algorithm was effective because it took around 15 seconds for each block to be placed into the blockchain with a difficulty level of 5. (In reality, it should take around 10 minutes for a block to be placed into the blockchain, so the difficulty level should be set to a 6 or a 7 in reality. However, for demo purposes, the mining process was purposefully shortened.) It also verified that the previous hash attribute of one block actually matched up with the current hash of the previous block and that the data has been successfully inputted. The second battery of tests involved trying to alter the data in the blockchain and then checking the validity afterwards. During the demo, we checked the validity of the blockchain before altering the data. This yielded “true” (it was valid), as it should have done. However, we tried altering the data to another value and then we checked the validity again. This time, it returned “false,” which was the correct value.

b. Test Coverage

During the demo, we tested what would happen if you were to navigate to particular pages on the website and the response of the website whenever you inputted a new data point. We also ensured that the user was able to compare themselves to other users in the system using sample data. Checking for validity of the data points was not done during the first demo, but we will also test for invalid inputs in the next demo by making sure invalid data points don’t get placed into the blockchain and an error message pops up. In reference to the blockchain, we tested how long it would take for a block to be inserted into the chain given a certain difficulty and that the hashes were properly linked (i.e. the previous hash attribute of the current block lined up with the actual previous hash). We also ensured that the blockchain was no longer valid after an outside source tried to tamper with it, which will affirm that the user’s data is secure.

c. Integration Testing

In order to ensure that the blockchain implementation has been successfully integrated with the overall website, we will try inputting a certain data point in the Data Entry section of the website.

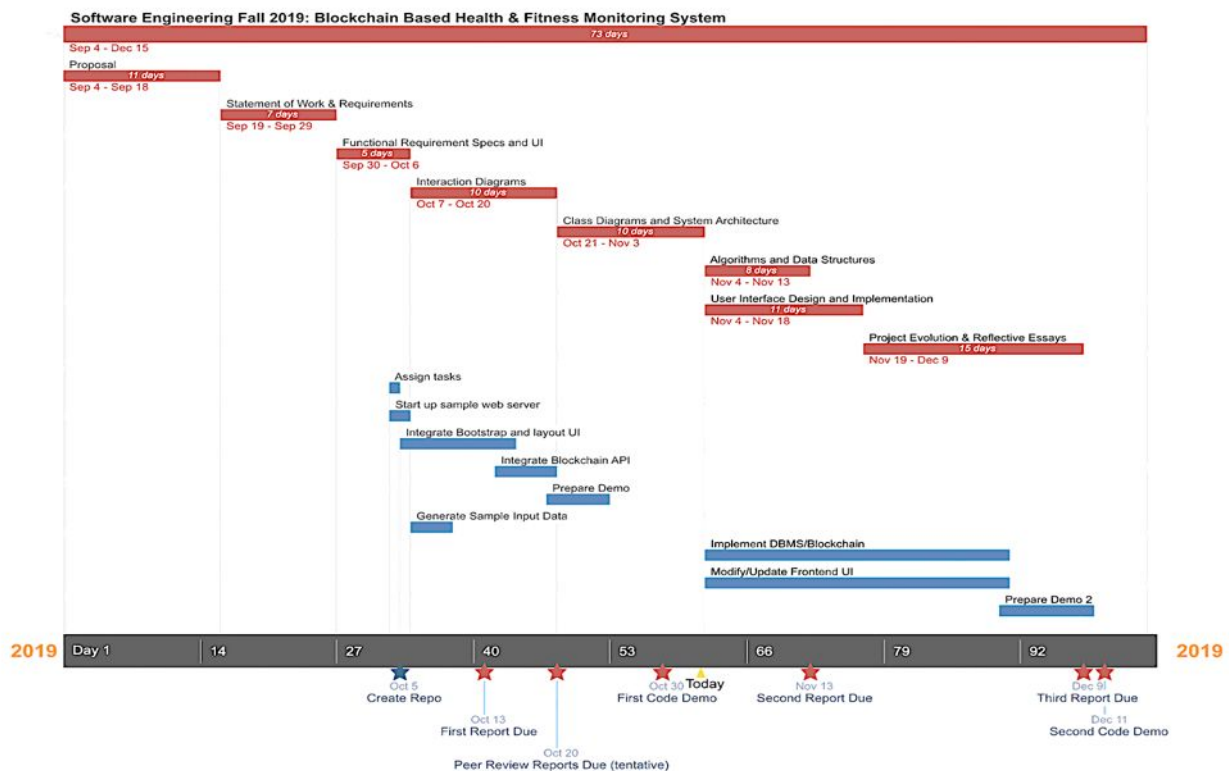
But before we do this, we will check the state of the blockchain. After entering data, we will check the state of the blockchain again. If the blockchain now contains one more data point than before (the data point we put into the website), then we can say that the two modules have been successfully integrated.

d. Other Testing

In the future, we will test what happens when multiple users are on the system simultaneously and then we will ensure that all of their data successfully goes into the system, even when multiple requests are being handled at the same time. We will also implement rollback in the blockchain, which saves the latest valid version of the blockchain in the case of a rogue agent trying to place falsified data points into the system. That way, all of the data won't be wiped out in the case of a catastrophe. This will be tested by trying to alter a data point like we did in the first demo, but after the blockchain detects an invalid blockchain, it will simply fetch the latest version that was valid and then we will make sure it fetches the previous version that was valid instead of wiping out the entire blockchain.

Section 14: History of Work

We used the Gantt chart to make and follow deadlines. For the most part, we followed the schedule that we made in our Gantt chart to keep us on track. We were not able to meet up as often as we liked because everyone had very different and busy schedules and we all lived far away from one another. Instead, we chose to communicate using group chats such as Slack and Facebook Messenger. This allowed us to communicate with one another efficiently to know the progress each of one us was making with specific tasks and if we needed any help with our designated task. As we completed milestones from our schedule, we were able to test if it worked and debugged as necessary. By tracking the multiple components of the software through an interactive spreadsheet, we are also able to equally distribute the workflow to all team members to ensure a smooth Agile development environment. Each team member contributed equally to the progress of our project.



The above Gantt chart is driving the team's deadlines and internal milestones.

Key Accomplishments

- Produced a software application that is user friendly and easy to use
- The application is eye catching and aesthetically pleasing
- Users are able to enter their specific data and the application is able to save it

- The application is able to show users' data in a meaningful way through graphs and charts
- Users can view and compare their data to data of other populations in real time

In the future, we would like to take this web based application and make it into a mobile application. As of now, users can access our website on their phone by using the internet to go to the website. It would be more convenient, however, if we could make an application that users can always access from their phone. This way they would not have to take the extra time to type in the website and log in each time. Since the ease of our users is very important to us, in the future, we would also like to be able to have data from smart watches be automatically entered into our application, instead of having to have users enter the data themselves.

Section 15: References

“Safe Sharing of Population Descriptors”

Marsic, Ivan.

https://content.sakai.rutgers.edu/access/content/group/fbfe7282-89ce-4a4d-a619-943dee895665/Programming%20Project/Programming%20Project_%20Safe%20Sharing%20of%20Population%20Descriptors.pdf

“Report #1 - User Effort Estimation”

Marsic, Ivan

<https://www.ece.rutgers.edu/~marsic/Teaching/SE1/report1-appA.html>

“What Is Blockchain Technology? A Step-by-step Guide For Beginners”

Ameer Rosic- Blockgeeks-Marko @marko-ceki-Alissa Brandemuhl-Tiffany

Mccullar-Mark Silen-Alex @sashabakht -

<https://blockgeeks.com/guides/what-is-blockchain-technology/>

“What Is Encryption? - Definition from Whatis.com”

Margaret Rouse-Kevin Ferguson-Margaret Rouse-Margaret Rouse -

<https://searchsecurity.techtarget.com/definition/encryption>

“Global Glossary Of Blockchain Terms 2.0 in 5 Languages”

<https://blockchaintrainingalliance.com/pages/glossary-of-blockchain-terms>

“Text Message Template Ideas To Increase Customer Engagement: Direct Sms”

<https://www.directsms.com.au/text-message-template-ideas-improve-customer-engagement/>

“The World Of Web and Mobile Applications”

Admin - <https://pctechieguy.com/the-world-of-web-and-mobile-applications.html>

Unified Modeling Language. (2019, August 25). Retrieved October 21, 2019, from

https://en.wikipedia.org/wiki/Unified_Modeling_Language#Interaction_diagrams

Marsic, I. Retrieved October 21, 2019, from

<https://www.ece.rutgers.edu/~marsic/books/SE/instructor/slides/>

Multitier Architecture. (2019, July 4). Retrieved November 2, 2019, from

https://en.wikipedia.org/wiki/Multitier_architecture#Three-tier_architecture

Software Architecture. (2019, October 13). Retrieved November 1, 2019, from
[https://en.wikipedia.org/wiki/Software_architecture#Examples_of_Architectural_Styles_.
2F_Patterns](https://en.wikipedia.org/wiki/Software_architecture#Examples_of_Architectural_Styles_.2F_Patterns)

Class Diagram. Retrieved November 3, 2019 from
<https://www.guru99.com/uml-class-diagram.html#10>