# Cover Page

T&T Health and Wellness Service – Indy-3

Project Leader: Troy Cope – Focused on Web and Backend Development

Team Leader: Turner Eison – Focused on Mobile iOS and Backend Development

Project Owners: Turner Eison and Troy Cope

Website: <https://capstone.personalwellnessservice.com/>

WE ARE LISTED AS THE PROJECT OWNERS BECAUSE WE INTEND TO RETAIN 100% OF THE IP FROM THIS PROJECT.

## Abstract / Overview / Executive Summary

The T&T Wellness Service is a capstone project dedicated to the betterment of individual health. It puts the power of monitoring and testing your health into your hands. The service provides an easy way to view and interpret this data from anywhere and log the data in real-time on your phone. The T&T Team has designed the application to make it simplistic and user-friendly.

Table of Contents

[Cover Page 1](#_Toc121129500)

[Abstract / Overview / Executive Summary 2](#_Toc121129501)

[Background information 2](#_Toc121129502)

[Requirements / Analysis / Results / Development 3](#_Toc121129503)

[iOS App 4](#_Toc121129504)

[Capstone Website 16](#_Toc121129505)

[Web Frontend 18](#_Toc121129506)

[Tech platform 20](#_Toc121129507)

[Development of software project 20](#_Toc121129508)

[Results 26](#_Toc121129509)

[Project Planning and Management 27](#_Toc121129510)

[Version Control 29](#_Toc121129511)

[Test Plan and Test Report 30](#_Toc121129512)

[Summary / Conclusion 30](#_Toc121129513)

[APPENDIX 31](#_Toc121129514)

# Background information

The T&T Wellness Service was founded on the principle of empowering individual health. The intent of this project was to extrapolate general health examinations that can go anywhere with you and allow you to view the trends of decline, improvement or stagnation. Since the average of each of the offered examinations is widely available, it is relatively easy to compare an individual to the masses to get a solid benchmark. Therefore, by giving individuals access to this service it helps empower improvements in their individual health while also maintaining their personal in a safe and secure manner.

# Requirements / Analysis / Results / Development

Tech platform review:

Initially, we were inclined towards a standard MERN (MongoDB, Express.js, React, Node) stack on the frontend and Swift and SwiftUI for the iOS application. We needed something to serve as the data management, which is where the MongoDB and the Node backend came in.

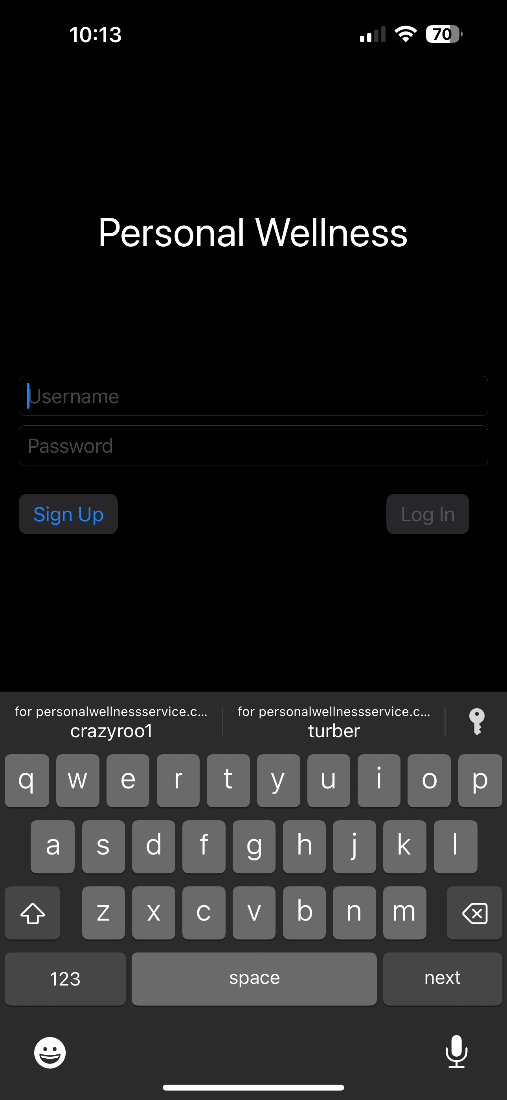
The iOS application was going to be written in Swift and SwiftUI. Since Turner has extensive experience in iOS development using both tools, it was an obvious choice. There was an option of using React Native or Android Studio for Android development, but for the sake of keeping the project concise these were not chosen. There is also a painful process of trying to launch apps on tools which are not specifically built for them, or tools that have large areas of specialization such as React Native. For this reason, it would have been very complicated to not only learn it but also implement it on an app we were creating for the first time.

There were other considerations for the database, such as Firebase, Oracle, MySQL, Microsoft’s SQL Server and a variety of NoSQL databases. MongoDB was decided on because we wanted a data structure that was simplistic and easy to work with, so having a database which could treat every entry like a JSON file was a must. For this reason, we considered Firebase very seriously as well, but decided that since we had experience in MongoDB that it would be the optimal choice.

The frontend was an obvious choice for us. Since we wanted to develop rapidly in a way where we had plenty of documentation and other items to reference, React was a prime contender. There are many other tools, such as Angular, that would have achieved rapid development on the Web end, but Troy has experience in React and React is known to play nicely with Node / Express. Therefore, it would go very well into the tech stack we were already piecing together.

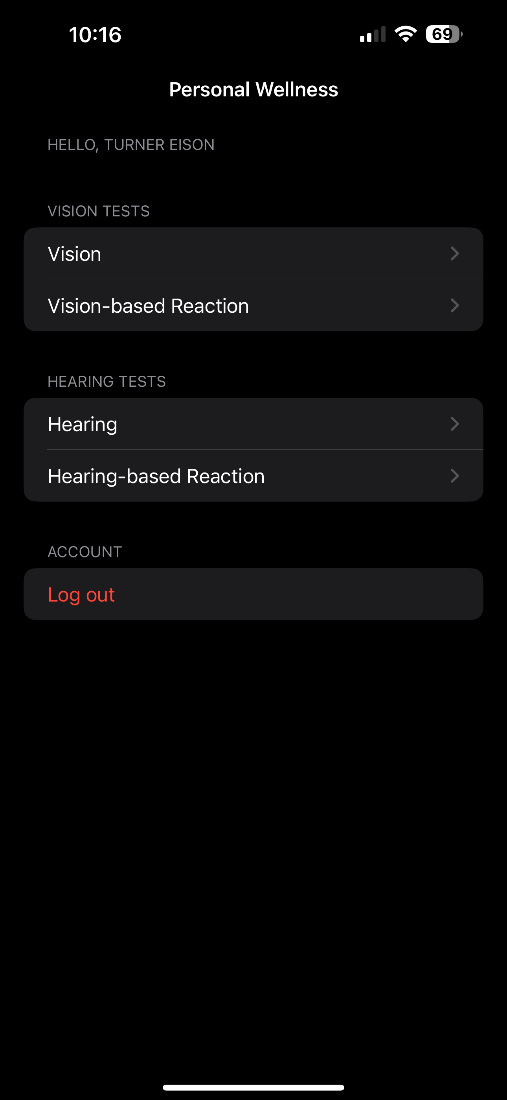
At this point, the backend was already sorted out. Both Troy and Turner had confidence with the rest of the tech stack, so the final item was what would bind the database to the iOS application and the Web application. Since the rest of the tech stack was already assigned, the most likely choice as something that compliments each aspect and would be easy to make an API out of. For this, we chose to use Node and Express. They are both very flexible for the API and can easily be integrated directly into React. They both had plenty of tutorials using MongoDB, and therefore it was a perfect addition to the tech stack.

# iOS App



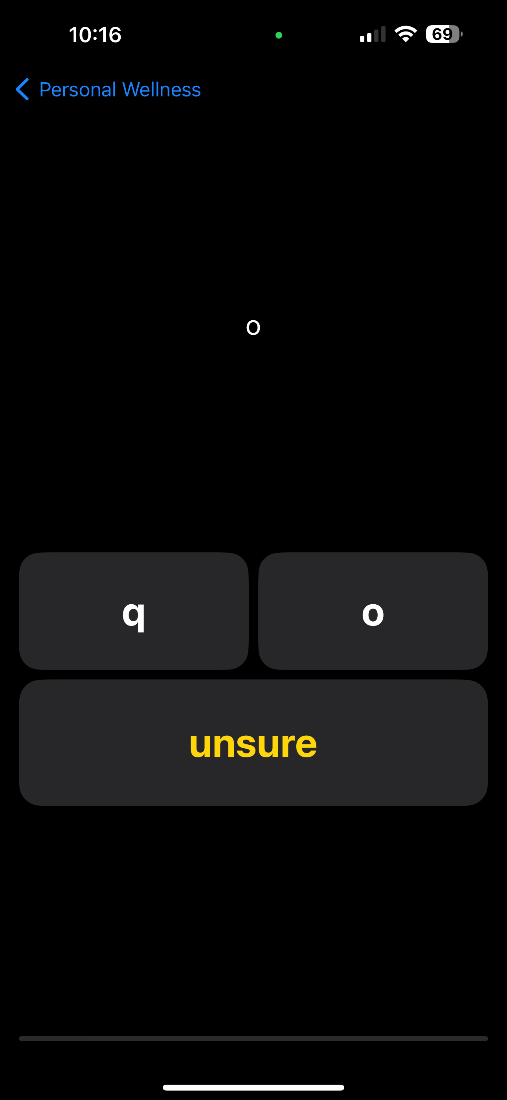
Login Page

Above is the login page which allows a sign-in and has a sign-up option as well. They are also automatically completed, which helps with efficiency.



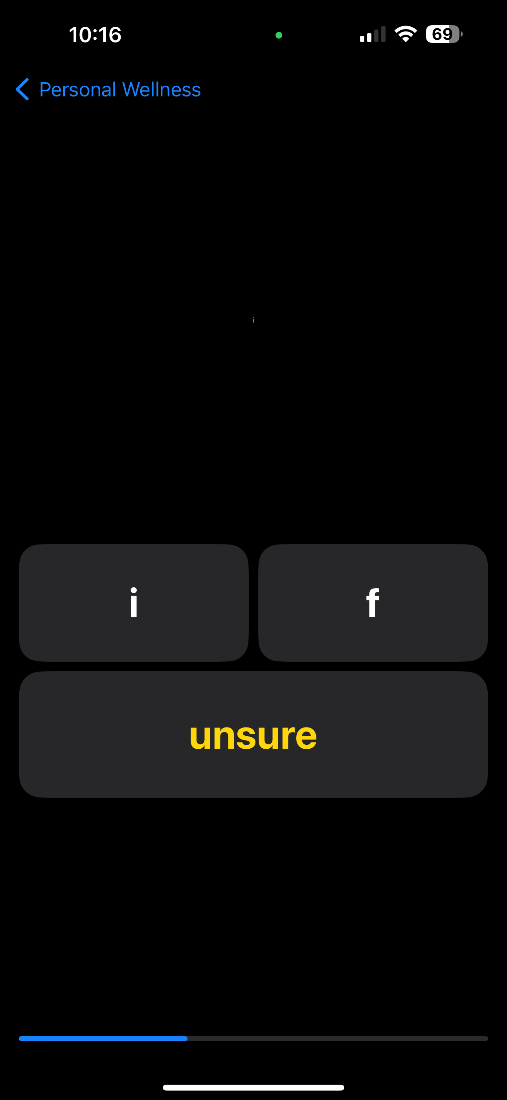
Home Page

The hope page of the app is shown above, which gives various options for the user to try and record data for. The logout function works intuitively and as expected.



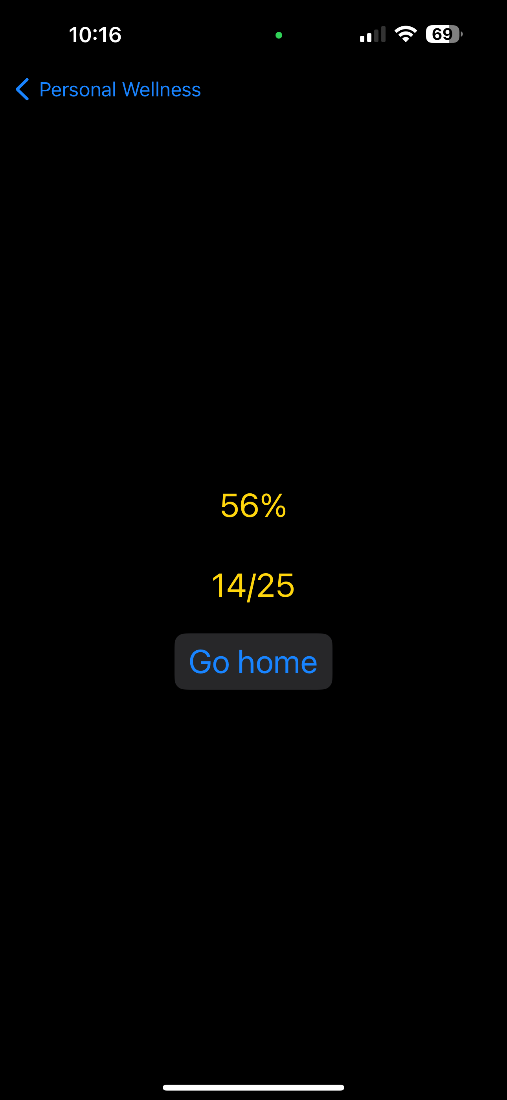
Vision Test

Clicking the buttons above will adjust the size of the font in order to give a good understanding of what is a comfortable size for the user to read.



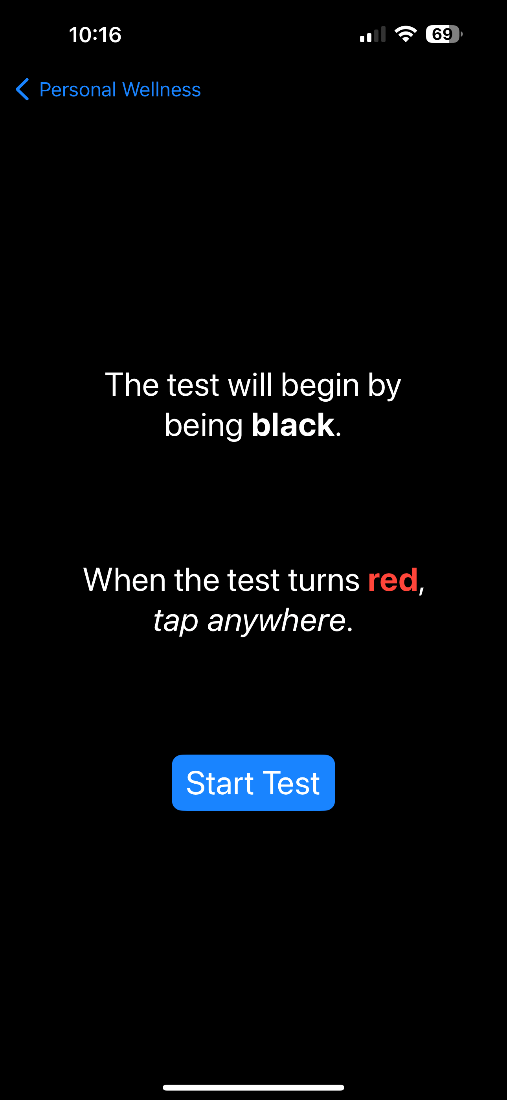
Vision Test contd.

Above shows the smallest the font gets, and honestly helps the user identify where their vision is measured at.



Vision Test Results

Shown above is the vision test result, the final screen of the vision test.



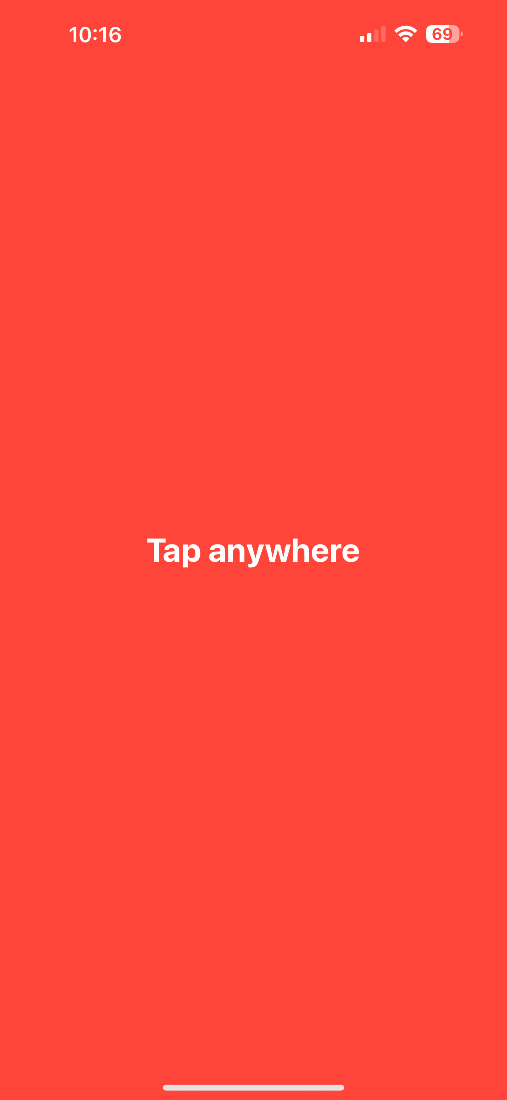
Vision Reaction Test Instructions

Shown above is the visual reaction test instruction which are shown to the user when they open the test.



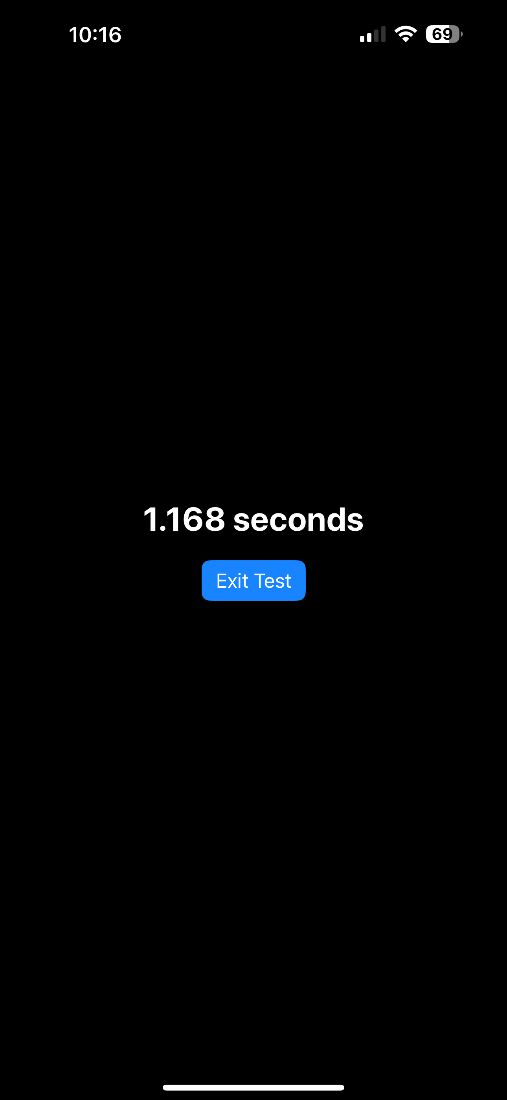
Vision Reaction Test

Shown above is the initial state of the reaction test, which will turn red and require user click in order to register how fast the user can note the change.



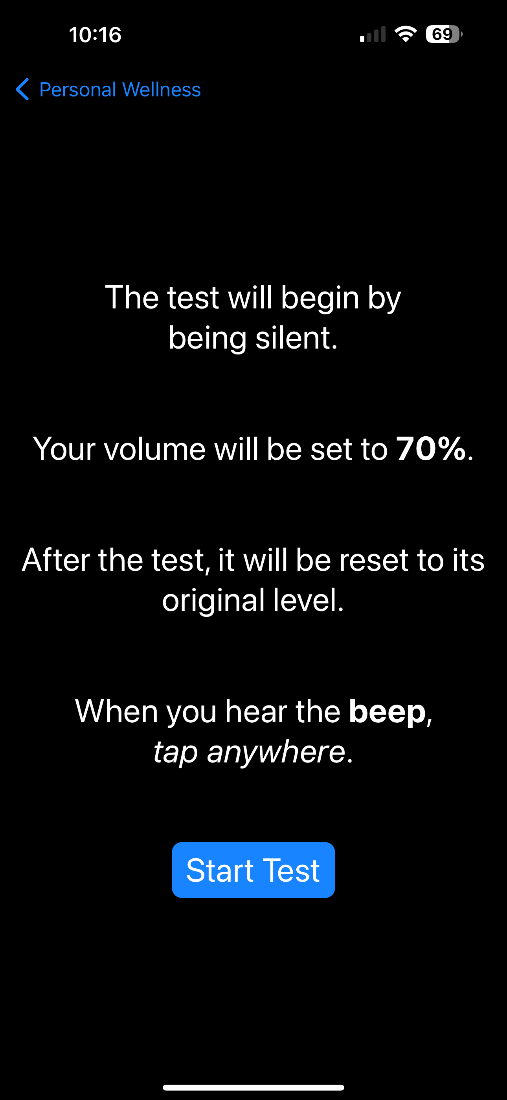
Vision Reaction Test contd.

The image above shows the intermediate stage of the vision reaction test when it is ready to be tapped by the user.



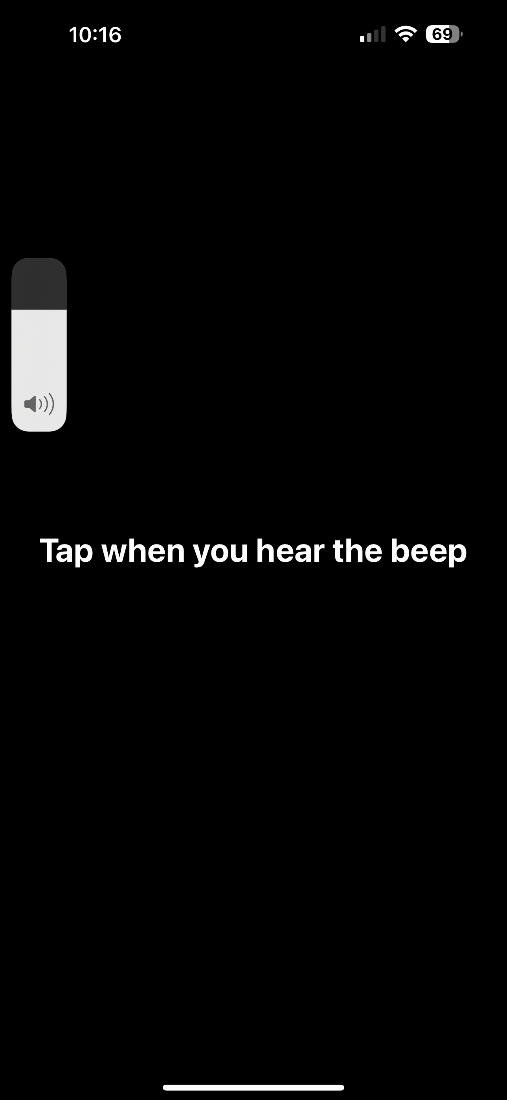
Vision Reaction Test Results

Shown above is the vision reaction results as well as the exit test button to confirm.



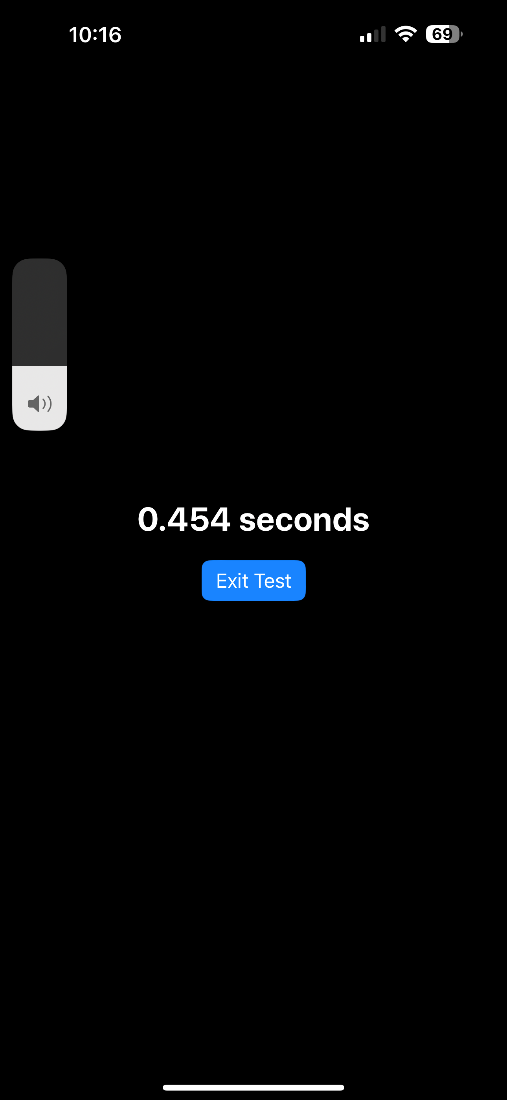
Hearing Reaction Test Instructions

Shown above is the hearing test instructions which require user input to continue.



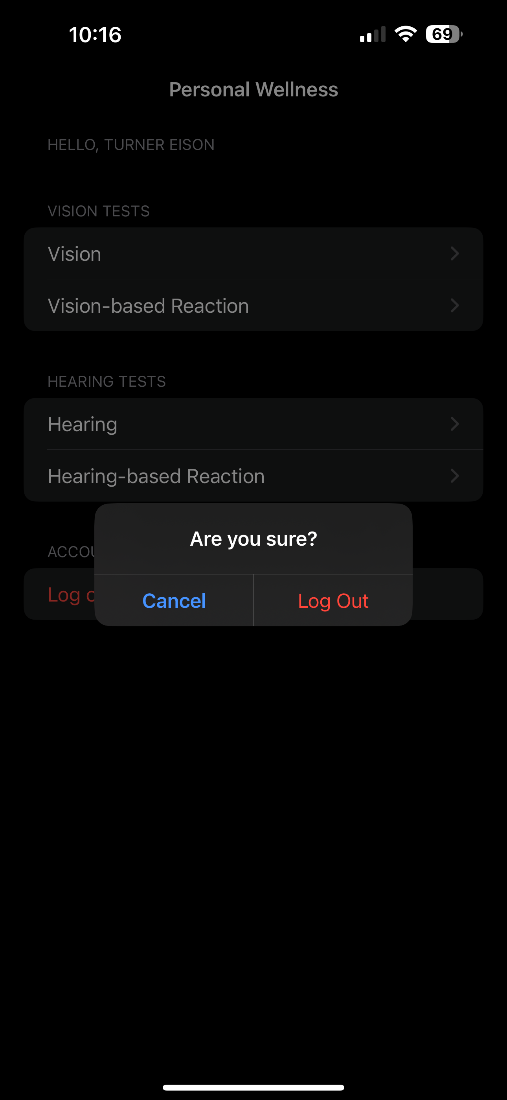
Hearing Reaction Test

The image above shows the intermediate stage of the hearing reaction test. It also shows the adjustment of the volume to coincide when the test begins.



Hearing Reaction Test Results

Shown above is a sample test result for the hearing test. This explains what the format and the test prompt look like when completed. The volume is then restored to its original state when the test finishes.



Logout Prompt from the IOS App

Shown above is the IOS App Logout function, giving a prompt to ensure that the user pressed the correct button. This will also adjust the token as necessary and keep the time stamp consistent.

# Capstone Website

Graphical user interface, text, application

Description automatically generated

Home Page

The home page gives the general synopsis of the website as well as what the app/web-end is used for.

Graphical user interface, text, application

Description automatically generated

About Us

The page above is the CCSE website linked about us page. It follows all their requirements, and the names are linked to each respective LinkedIn.

Graphical user interface, text, application, email

Description automatically generated

Technology

The images shown above and following as “technology contd.” are all parts of the technology page, which can be revealed when scrolling through the page.

Graphical user interface, text, application, email

Description automatically generated

Technology contd.

Graphical user interface, text, application, email

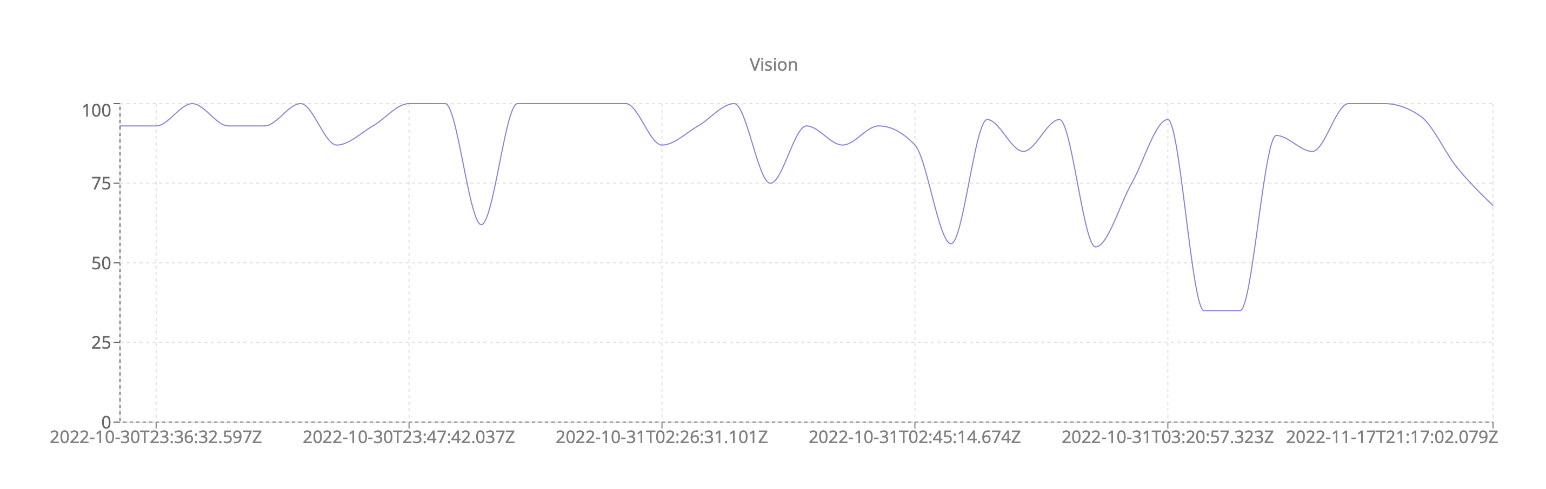
Description automatically generatedTechnology contd.

# Web Frontend

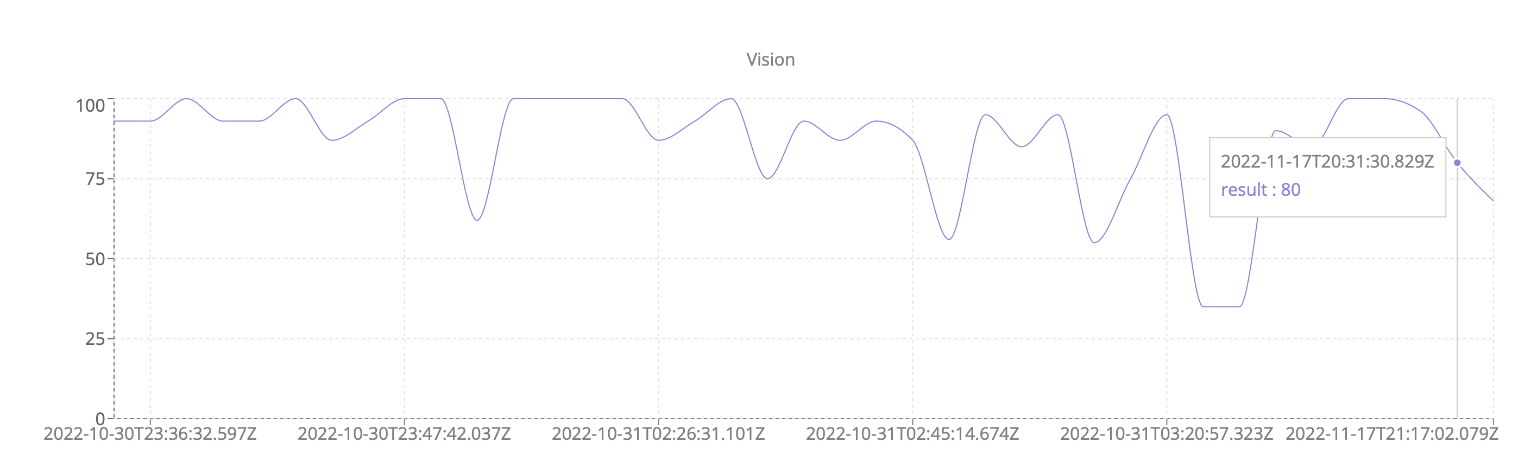
Graphical user interface, website

Description automatically generatedLogin Page

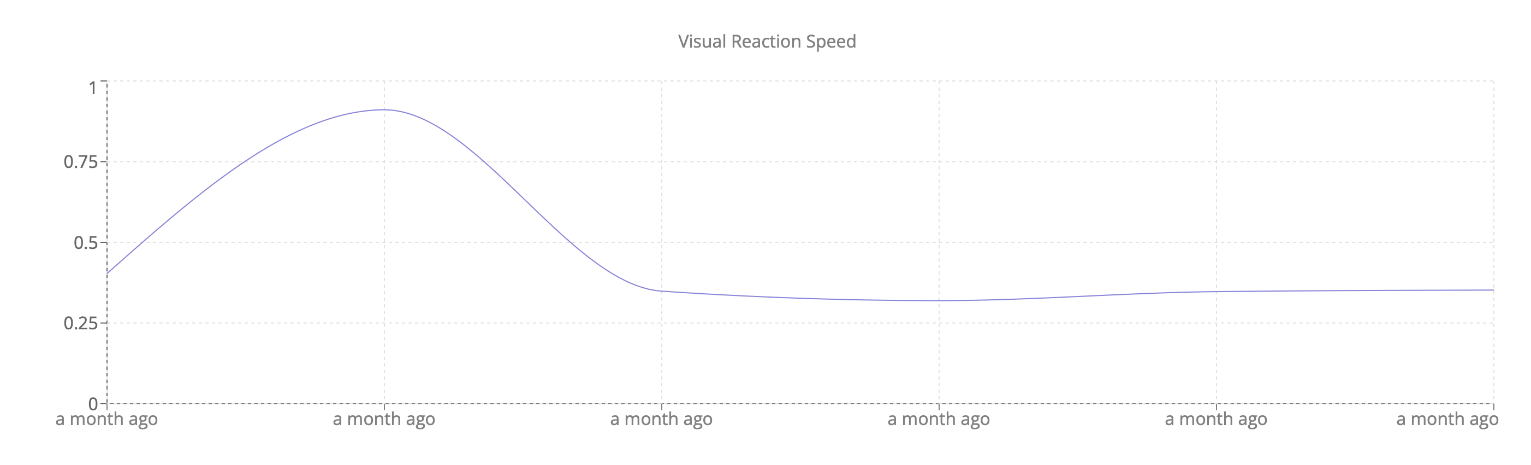
The login page is the entry page for the web-end dashboard. This is what verifies the user and pulls their data when attempting to open the dashboard. Users can also swap to the sign-in forum with the notes button above. It follows a nice animation and gives a sign-up form which will add the user to the backend database.

Dashboard – Vision

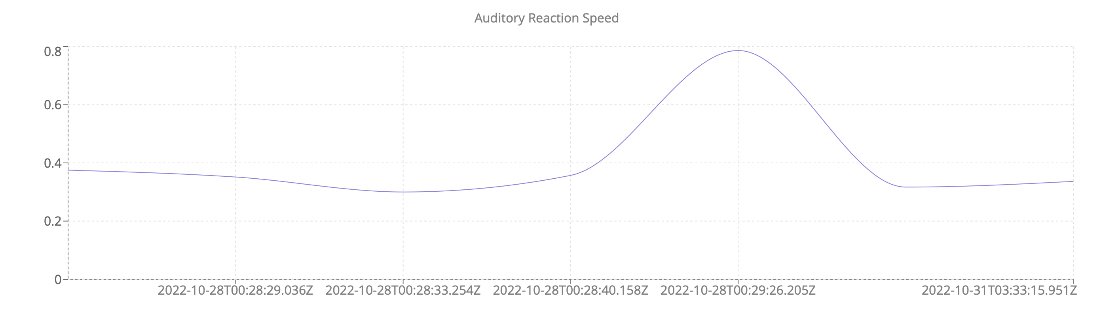
The dashboard shows the dates and has a popup for the results on each section. The overall trend can be observed above.

Dashboard – Vision with Tooltip

The dashboard shows the dates and has a popup for the results on each section. The overall trend can be observed above.

Dashboard – Visual Reaction Speed

The dashboard shows the dates and has a popup for the results on each section. The overall trend can be observed above.

Dashboard – Auditory Reaction Speed

The dashboard shows the dates and has a popup for the results on each section. The overall trend can be observed above.

## Tech platform

The T&T Wellness Service tech stack is comprised of a React and Node Frontend, an Express and MongoDB backend, and a Swift and SwiftUI mobile iOS application. The structure on the web is based on a dynamic standard MERN (Mongo-Express-React-Node) stack, and the mobile iOS app is based on MVC (Model-View-Controller) architecture.

## Development of software project

Text, letter

Description automatically generated

Our first diagram of white board architecture: (AUG 18)

This whiteboard drawing shows the first iterations of thought that the T&T project underwent. It shows the basic layout as well as the steps the project needs. There are some notes on the board that were discarded or are thoughts more than implementable concepts. An example of this is the “NLP Chatbot” and the “Py Script” ideas. The paperwork requirements are noted on the side as well as the most recent class.

Text, letter, whiteboard

Description automatically generated

Second whiteboard meeting for fleshing things out (AUG 23)

The second whiteboard meeting shows a better fleshing out of concepts. Here are the base principles as well as the ideal implementation languages and functions for each of our tech stack tools. The WTFAWD section shows the principles and connections that T&T wanted to go down.

Text, letter

Description automatically generated

Indy-3 In the creation process, (AUG 25)

This image is the Indy-3 in class writeup, which depicts the formal writeup of the project.

Text, letter

Description automatically generated

Quick writeup of TODO in post of class (AUG 25)

This was the to-do which was written with classmate input at the end of the formal writeup process. Indy-3 had officially taken place at this point and was noted on the board in the way shown above with the paper as reference.

Diagram, schematic

Description automatically generatedQuick website architecture writeup (OCT 25th)

The image above shows the rough website layout. Each of these boxes is the same, and they can all be written as cards in React. They were then written as cards later, which allowed them to be purposed for each individual usage. They were given a fitting description and mage, which can be seen in the website section.

Problems through the process:

React – Finding a good CSS or SCSS format

CSS and SCSS are very different in the way that they are written, for this reason we opted to do SCSS because we could easily write the animations in it with localized variables that could be varied if necessary. This also allowed us to write individual SCSS files for each component, compartmentalizing the already very well documented project.

React – Sourcing designs and coming up with animations

Any form of design and intuition for something to “look good” can be challenging especially when both developers value functionality over aesthetic. We eventually settled on the color themes and pictures shown in the final project and were very happy with the animations on the login/sign-up swap as well as the shadow effects on certain buttons.

React – Hooks

Hooks are exceptionally painful for new developers because React has a very explicit convention that they need to be called with. For this reason, arguably the hardest part of the React frontend was structuring it in such a way that hooks could be incorporated and used fluently. This was solved via a complete rewrite of some components, but due to the compartmentalized nature of React this was not too problematic.

App – Hiding vision test if not in frame

The vision test needed to be hidden if the user came too close to their device. This required overlaying a view on top of the vision test as to not lose their progress. We wanted this overlay to appear with a quick fade animation so it wouldn’t be too jarring. Hiding the vision test required that we use the facial tracking functionality in ARKit, Apple’s augmented reality framework. We had to prompt the user to access this data and had to be mindful that it wasn’t running constantly – only during the vision test, so as not to drain their battery. We put runtime hooks in that started the face tracking whenever the vision test appears and disables it when it disappears.

App – Face tracking working on iOS

Face tracking was difficult to implement. There isn’t a built-in method call to get the user’s distance from their device. To get around this, we had to use ARKit to locate their eyes in real space, calculate the midpoint of their eyes, and calculate the distance of this midpoint from the camera.

App – Persisting Login and tokens

To persist the user’s login in the iOS app, we had to find a way to securely store their token. We couldn’t just save the raw token to the disk, as this can be easily compromised. Instead, we had to store their token in the device’s Keychain, which is an encrypted datastore. This took some work, as the Keychain APIs have some peculiarities.

## Results

1. Web App
2. Mobile App
3. Backend Server
4. Paperwork

By the time this project was finished, we had:

* A fully functional iOS App which included: sign-in, sign-up, authentication, time sensitive tokens, multiple types of personal wellness examinations, and information recording capabilities.
* A fully functional Web-end which included: sign-in, sign-up, authentication, time sensitive tokens, and displaying of individual user data recorded via multiple formats from the iOS App.
* A complete Node-MongoDB-backend which included: An API for pulling and sending data from the database, middleware for authentication, and stable data formatting which allowed for extensive use on both the iOS and the Web-end.
* A standard website which meets all C-Day requirements and links directly to the Web-end.

The entire project, along with all the important pieces, were created from scratch. Inspiration for formatting and multiple references to design each integral part were used. This project also gave us stronger leadership abilities as well as very concise and precise paperwork to accompany it. The documentation of the project was done via GitHub Issues, but due to the retention of ownership and it’s direct relationship with the code, this was omitted.

# Project Planning and Management

Collab tools:

The main methods of communication for this project will be in person, through Discord, iMessage, and on GitHub (using Github issues for management control). The code is written in multiple languages and has a large scope; therefore, Git is necessary for collaboration and version control.

The project leader Troy Cope will coordinate weekly and in-person meetings between group members. The team leader Turner Eison will also act as a go-between to facilitate communication between the team and the project owner(s), Turner Eison and Troy Cope.

Project meeting schedule:

The team will meet twice a week at predetermined times, based upon available schedule, in Discord or on campus to discuss milestone objectives. On top of these weekly occurrences, meetings were conducted on an “as necessary” basis were very informal. One of the advantages of a small team is that work can be very agile, which allowed meetings to be prompted within the same hour and be completed in under 10 minutes. Checkups like this were frequent as well as architecture meetings where both members would draw out their ideas on white boards. Any additional benchmarks or changes were then added to the Github issues page.

Gannt Charts:

The Gantt Chart below shows the work at the current time based on GitHub Issues. The purple shown sections which have not been completed yet, whereas the gray shows the section which are completed. Boxes and tasks can be shown as percentages which reflect in the gray and purple. The date is adjusted in order to show all the added tasks.

Phase 1 Gantt Chart:

Chart

Description automatically generated with medium confidence

Screenshot as of (8/28/2022)

Phase 1 Gantt Chart (some date here):

Chart

Description automatically generated

As of (10/3/2022)

Phase 1 Gantt Chart (some date here):

Chart

Description automatically generated

As of (11/18/2022)

Phase 2 Gantt Chart (some date here):

Chart

Description automatically generated

As of (11/28/2022)

## Version Control

The version control used for the project was Github and Github issues to contain all the tickets that are noted in the Gantt charts above. Screen shots for the Github Issues are not included because they are all transcribed in the Gantt charts. The only difference is that the tickets have labels, descriptions, and comments which specified what the tickets were for and their importance.

## Test Plan and Test Report

App

* Check login with existing account
* Check create new account
* Check login with new account
* Check that all tests are listed on home page
* Check each test is functional and works as expected
* Check logout is functional
* Check that auto login works

React

* Check login with existing account
* Check create new account
* Check login with new account
* Check browser saves and will autofill credentials
* Check that all tests with data show charts
* Check that charts have correct data
* Check logout is functional

Node Server

* Check server can access MongoDB
* Check login endpoint
* Check signup endpoint
* Check API endpoints
* Check data ingest
* Check data output

# Summary / Conclusion

In summary, the project was a very good reflection of the capabilities of two strong developers. The work the T&T Team did was objectively good, employing authentication and middleware from scratch to cover data as well as creating time sensitive tokens to keep users signed in. The data itself is very secure in the backend and pulling/pushing is standardized via an API which activates the middleware. The designs are sleek and simple, animations are crisp and tasteful. This team would have suffered to move from an agile approach to a stricter approach with more members, and therefore the ideal team was the two members selected. If it were a bigger project, there would be a lot more work allocations and problems with scheduling as well, which would have given negative influence to the project. Therefore, it is much better to keep a small team and work on compartmentalized sections. This makes the necessary code collaboration minimized, and therefore increases the efficiency of all members as they can work as it compliments their schedules.

# APPENDIX

React is a JavaScript library developed by Facebook which allows developers to create fast user interfaces with dynamic properties. It encapsulates CSS and HTML into components with the use of Babel.

Node is an asynchronous event-driven JavaScript runtime. It is built to be scalable and runs locally on a computer instead of in a browser.

Express is a web application framework which gives broad features. It is used for multipage, and hybrid web applications. It builds on top of Node and helps manage servers and routes.

MongoDB is an open-source NoSQL database which has no traditional relational databases and instead has a flexible and undefined structure. Specifically, it can be used for storing JSON-like objects.

Swift is an open-source programming language specifically for and by Apple. The SDKs are proprietary to Apple, and therefore require a Mac to develop on.

SwiftUI is a reactive, declarative, framework created exclusively for user interfaces in Apple platforms.