Fire ResQube

A Social Application for Firefighter Training, Fitness, and Knowledge Sharing

Nathan Johnson
Dawson Jordan

Table Of Contents:

Nathan Johnson
Dawson Jordan

Abstract

Firefighters play a crucial role in community safety, consistently risking their lives and embodying qualities such as integrity, physical fitness, and teamwork. Despite these commendable attributes, a distinct need arises for a comprehensive tool to facilitate knowledge-sharing among firefighters across the United States. In response to these challenges, Fire ResQube emerges as an innovative mobile application designed with the React Native framework, aiming to connect, engage, and enhance the skills of the firefighting community. Notable features include user profiles, questions, discussions, search functionality, daily refreshers, and a secure login system. Our project emphasizes the meticulous development of the Weekly Workout Challenge. Leveraging an AWS Amplify backend and an interactive user interface, we enable users to upload personal statistics for weekly AI-generated workouts, access national leaderboards, and create custom groups for friendly competition and mutual accountability. By offering easily accessible workout history, we highlight individual improvements over time while simultaneously motivating users to maintain consistent workout streaks. Through the implementation of the Weekly Workout Challenge, our objective is to cultivate an engaging user experience that promotes fitness, camaraderie, and healthy competition among firefighters. This mission aligns seamlessly with Fire ResQube's broader goals of enhancing public safety, fostering skill development, and promoting community engagement.

Back to top

Qualifications

DAWSON J. JORDAN

(509) 939-1148 • dawson9jordan@gmail.com

EDUCATION

Montana State University – Bozeman, MT **August 2021 – Present**

BS in Computer Science (Expected May 2024) – 3.93 GPA

- Relevant coursework:
  - Joy and Beauty of Data (Python)
  - Data Structures and Algorithms (Java)
  - Web Development (HTML / CSS / JS)
  - Software Engineering
  - Databases (SQL)
  - Networks

Entrepreneurship and Small Business Minor

- Relevant coursework:
  - Integrated Online Marketing
  - Management and Organization
  - Entrepreneurial Experience
  - Business Communication

Walla Walla Community College – Walla Walla, WA **September 2019 – June 2020**

Associates in Science Transfer Degree – 4.0 GPA

- Walla Walla Community College President's List
- Phi Theta Kappa Honor Society
- Baseball Team Member

Washington State University – Pullman, WA **August 2018 – May 2019**

- Washington State University Honor Society

EXPERIENCE

Last Best Builders LLC – Bozeman, MT **August 2020 – Present**

Carpenter

- Work with a team to frame and side custom homes
- Learned to read architectural plans, perform calculations, and solve problems

Spokane Public Schools – Spokane, WA **June 2018 – August 2020**

Summer Programs Intern

- Assisted the Director of Afterschool Programs to prepare for Summer STEM Camps
- Gained experience coordinating with funders, answering phone calls to address inquiries, and developing attendance systems and camp protocols for child safety

STEM Summer Camp Leader

- Taught 3D printing and computer design to a diverse group of Elementary and Middle School students
- Prepared and distributed STEM kits and meals to over 2000 students in low-income communities during the coronavirus pandemic
- Learned the importance of adapting my communication methods to accommodate students of varying ages and backgrounds
- Developed a compassion for children in the devastating reality of underfunded communities through interactions with students experiencing homelessness, kids who lost parents to prison and addiction, and young immigrants who lived through war

Back to top

Nathan Johnson
(208) 912-5219 | nathan.johnson@drycreek.me | Boise, ID, 83714

Education

Montana State University — Bozeman, MT
B.S. Computer Science | Minors: Data Science, Statistics — **August 2020 - May 2024**
Cumulative GPA: 3.67/4.00

Employment

SECURA Insurance Companies — Neenah, WI
Software Developer Intern — **May 2023 - August 2023**

- Responsible for writing and testing new software requirements while utilizing IntelliJ IDEA, GitHub, and JIRA softwares.
- Write and review code in Java, XML, SQL, and JavaScript programming languages.
- Participate in an agile scrum team with daily scrum ceremonies, weekly backlog refinement meetings, and bi-weekly sprint review, spring planning, and retrospective meetings.

Montana State University | University Student Housing — Bozeman, MT
Assistant Community Director — **July 2022 - May 2023**

- Advise the hall student government and assume building responsibilities in the absence of the Community Director.
- Oversee staff team development and growth as employees and resident advisors.
- Coordinate three building-wide events per month, ranging from movie nights to campus-wide casino night with ~$2000 in prizes.

Resident Advisor — **August 2021 - May 2023**

- Responsible for maintaining a healthy environment that is conducive to student's academic, physical, and mental wellbeing.
- Create a lively and supportive community for residents to learn and grow as students, friends, and young adults.
- Organize various recreational, social, and educational activities on a monthly basis.
- Earned 2021-2022 Montana State University Resident Advisor of the Year Award (3 selected out of ~125).

Involvement

Montana State University — Bozeman, MT
Undergraduate Lab Assistant — **August 2023 - Present**

- Support students in a structured lab environment, enhancing their programming ability and opportunity to grow as a computer scientist.
- Assist undergraduate students in labs relating to introductory data structures and algorithms in Java.
- Aid students in designing problem solutions, implementing those designs, and debugging their implementations.

Technology and Artificial Intelligence Learning Seminar — **January 2023 - May 2023**

- Weekly seminar on signals, dynamics, statistics, and machine learning as a part of the NSF AI Institute in Dynamic Systems.
- Basic to advanced data science study including time series, linear regressions, and artificial intelligence software packages.

Relevant Coursework

Data Structures and Algorithms — **August 2020 - Present**

- Proficient with arrays, linked lists, graphs, trees, stacks, and queues.
- Able to use different data structures to implement searching and sorting algorithms as needed.

Data Wrangling/Clustering — **August 2022 - Present**

- Strong skills in R and Python to manipulate data for effective and efficient analysis.
- Ability to use SAS and R Shiny to make visually pleasing and interactive data reports.

Technical Skills

Github	Microsoft SSMS	TeamCity	Python	R	SQL
JIRA	SAS	OctopusDeploy	Java	C/C++	UML
Agile	JetBrains IDE	Microsoft Excel	SDLC	OOP	Databases

Back to top

Introduction

Fire ResQube serves as the only all-in-one platform specific to firefighter knowledge sharing. As a place where firefighters from all over the country can share their knowledge, ask questions, find mentors, stay up to date with the latest tactics, and challenge each other to stay physically fit, they can grow faster in their jobs and ultimately improve public safety.

As our computer science capstone project, we intend to build the foundation of Fire ResQube and implement the Weekly Workout Challenge feature within the app. This feature provides a new, trackable, workout to users each week and allows them to upload workout statistics which will be broadcasted on leaderboards nationally and within custom groups.

Our motivation to develop Fire ResQube stems from a newfound understanding of the multifaceted challenges faced by firefighters on a day-to-day basis. The career requires extensive preparation both mentally and physically in order to execute in a wide range of emergencies at any given time. Currently, no firefighter specific platform exists for national communication about experiences amongst one another. In addition, weekly fitness check-ins promote maintenance of the fitness levels necessary to carry out emergency responses. Fire ResQube can help fill these voids by offering an environment where firefighters can not only share expertise, seek guidance, and stay up to date on cutting-edge tactics but also prioritize their physical fitness in a supportive community. By introducing the Weekly Workout Challenge, our intent is to instill a sense of camaraderie, encouraging firefighters to challenge and support each other in staying physically fit.

Background

In the demanding profession of firefighting, it is imperative for firefighters to maintain peak physical conditioning and strength. The tasks inherent to their roles, such as carrying heavy equipment, navigating challenging terrain, and executing precise rescues, demand not only agility but sustained endurance and strength. These physical attributes are not just advantageous but are critical to the effectiveness and safety of firefighting operations.

The consequences of a firefighter being unable to perform optimally due to a lack of fitness extend beyond individual risk. In emergency scenarios, where time is of the essence, a compromised physical state places not only the individual firefighter at heightened danger but also the entire crew and the public (Kowal, 2018). The collaborative and dynamic nature of firefighting demands that each team member is not only proficient in their individual tasks but is physically prepared to contribute effectively to the overall success of the mission.

Recognizing these challenges, the Weekly Workout Challenge emerges as a proactive and innovative solution to support the fitness of firefighters throughout their careers. Beyond the conventional approach of rigid fitness programs, this unique initiative injects an element of fun into the process of maintaining peak physical condition. By introducing an engaging and dynamic platform that offers new, community based, workouts on a weekly basis, firefighters are motivated to consistently assess and enhance their fitness levels.

The competitive aspect of the Weekly Workout Challenge serves a dual purpose. Firstly, it transforms the often rigorous and demanding routine of physical training into an enjoyable and shared experience, fostering camaraderie among firefighters. Secondly, it provides a means for individual firefighters to benchmark their progress not just against personal goals but against the achievements of their peers nationwide. By infusing an element of enjoyment and competition into the pursuit of physical fitness, this initiative strives not only to address the immediate need for optimal performance but also to cultivate a sustained and resilient firefighting force dedicated to the safety and welfare of the public they serve.

Comparable Businesses

- **Strava:** An app that “connects runners, cyclists, hikers, walkers and other active people through the sports they love” using fitness tracking technology (Strava Inc., 2023).
- **Fire Rescue Fitness**: “The home of the world's most effective fitness programs and resources for first responders.” Fitness programs can be accessed through an app after a paid membership (Zamzow, 2022).
- **Yumuuv:** “Team wellness challenge is a step to increase employee happiness and engagement.” The app allows an employer to set up a fitness challenge for employees to compete in (YuMuuv, 2023).

How Fire ResQube is Different
These existing platforms cater to specific niches, such as Strava for general fitness enthusiasts, Fire Rescue Fitness for paid fitness programs for first responders, and Yumuuv for team wellness challenges. Fire ResQube emerges as an all-encompassing solution for firefighters. While these competitors excel in their respective focuses, they lack firefighter specification and integration of knowledge sharing and holistic fitness within a single platform. Fire ResQube distinguishes itself by being a social hub that not only facilitates the exchange of skills and knowledge among firefighters but also prioritizes their physical well-being.

Unlike other fitness apps, Fire ResQube's Weekly Workout Challenge feature is not a structured fitness program for firefighters to follow daily. Instead, it presents a dynamic and enjoyable way to test and showcase fitness once a week. This unique approach transforms fitness into a communal experience, allowing firefighters to follow their own programs, then compare their results with peers nationwide. Tailored specifically for firefighters across the nation, this free and community-driven service promotes a standardized level of fitness while emphasizing the camaraderie of shared goals within the firefighting community. Fire ResQube stands out by addressing the multifaceted needs of firefighters and providing a distinctive space for their professional and physical development.

Proposal Statement

In this section, we present the design details of the Weekly Workout Challenge feature within the Fire ResQube platform. We include requirements, architectural design diagrams, and our design decisions to document the thought process and plan behind our implementation.

*Italicized elements highlight requirements that were unable to be completed in the scope of the semester

Functional Requirements

- The system should generate a new workout challenge every week, including details such as exercise type, duration, reps, and any specific instructions.
- The system should provide a user-friendly interface for logging workout statistics, including exercise statistics and any notes the user wants to include.
- The system should only request workout details specific to the workout.
- The system must maintain national leaderboards that display the top performers based on performance metrics.
- The system must track and display users' progress over time, allowing them to view historical workout data and improvements.
- Users should earn achievements for completing challenges consistently or reaching milestones.
- *The system should send notifications to users to remind them of the new weekly workout challenge*
- *Users should have the ability to create and join custom groups (such as fire stations or local regions) to participate in group leaderboards.*
- *Users should be able to send messages within their custom groups.*
- *Users should be able to easily rate and review each week's workout challenge.*

Non-functional Requirements

- The system should use artificial intelligence to generate the weekly workout challenge.
- The system databases should be stored and accessed from a cloud service provider.
- Code should be well-documented to support future updates and maintenance.
- The system must be scalable to handle a growing user base, ensuring that performance remains consistent.
- User data should be encrypted during transmission and storage.
- Authentication mechanisms should be used to secure user accounts and prevent unauthorized access to fitness information.
- *The system should autonomously display and notify users about the new challenge each week.*
- *The system should be accessible from different devices and browsers.*

Performance Requirements

- The system should respond to user interactions, such as logging workouts or accessing leaderboards, within an average response time of 2 seconds.
- When users upload workout statistics, the system should handle the upload process efficiently, ensuring that data is stored in the database within 3 seconds.
- Leaderboards should be updated at least every 60 seconds to reflect the latest leaders.
- User data should be synced across devices within 5 seconds.

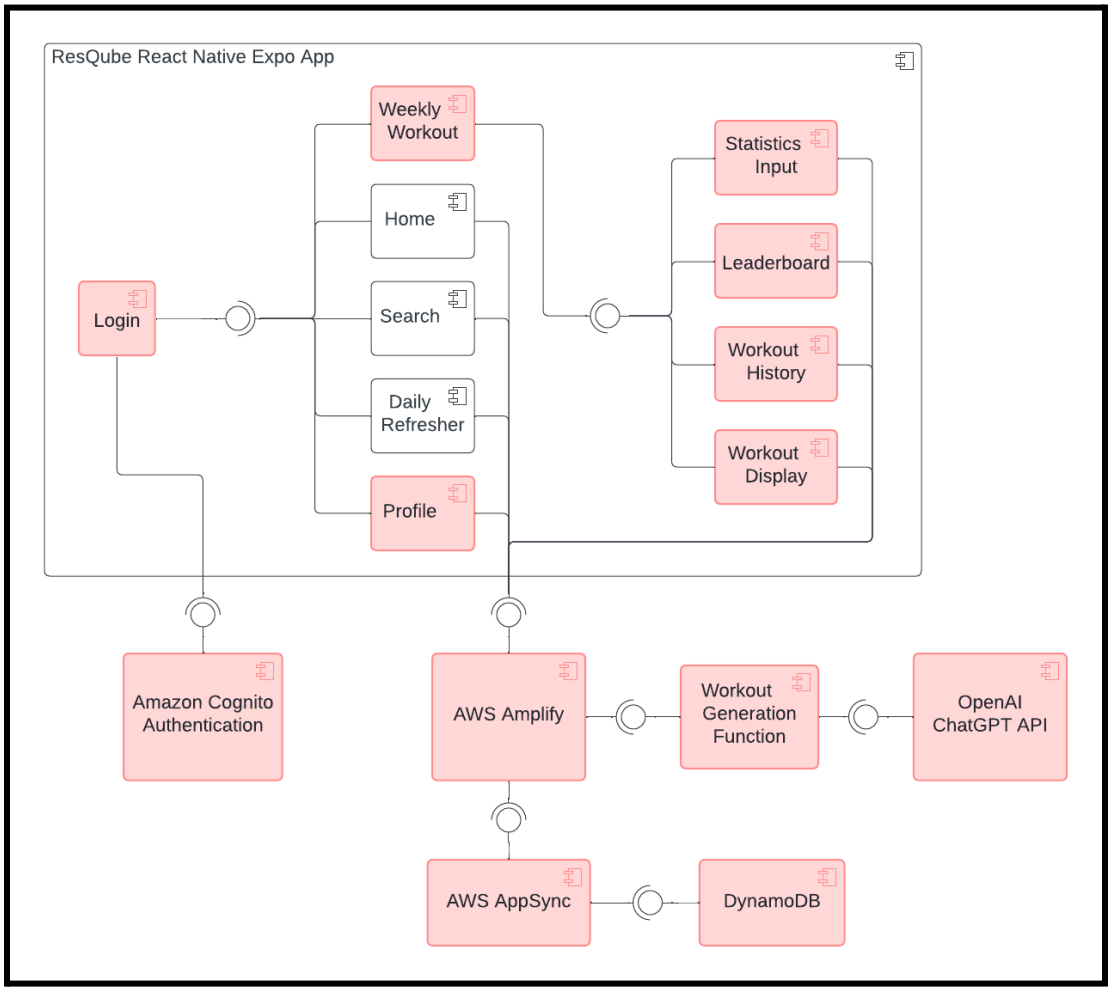
- User authentication should be completed within 1 second to minimize login delays.
- *Notifications reminding users of the Weekly Workout Challenge should be delivered to devices within 1 second of being triggered.*
- *The system should support a minimum of 10,000 concurrent users without significant performance issues.*

Interface Requirements

- The Weekly Workout Challenge interface should be visually appealing and easy to navigate.
- A calendar display should allow users to view personal and national results from previous weeks.
- A "workout streak" value should be displayed on the users' profile.
- Leaderboards should display national and local rankings in an ordered format.
- A "log" button should open a page that prompts the user for each required statistic of the weekly workout and allow them to submit it.
- *Custom groups should be created through a "create group" icon where users can search and add others to a group.*
- *Within a group, a leaderboard and group chat should be displayed.*
- *After logging a workout, the user should see a "rate and review" page containing a 5 star selection and a comments section.*

Fire ResQube Component Diagram

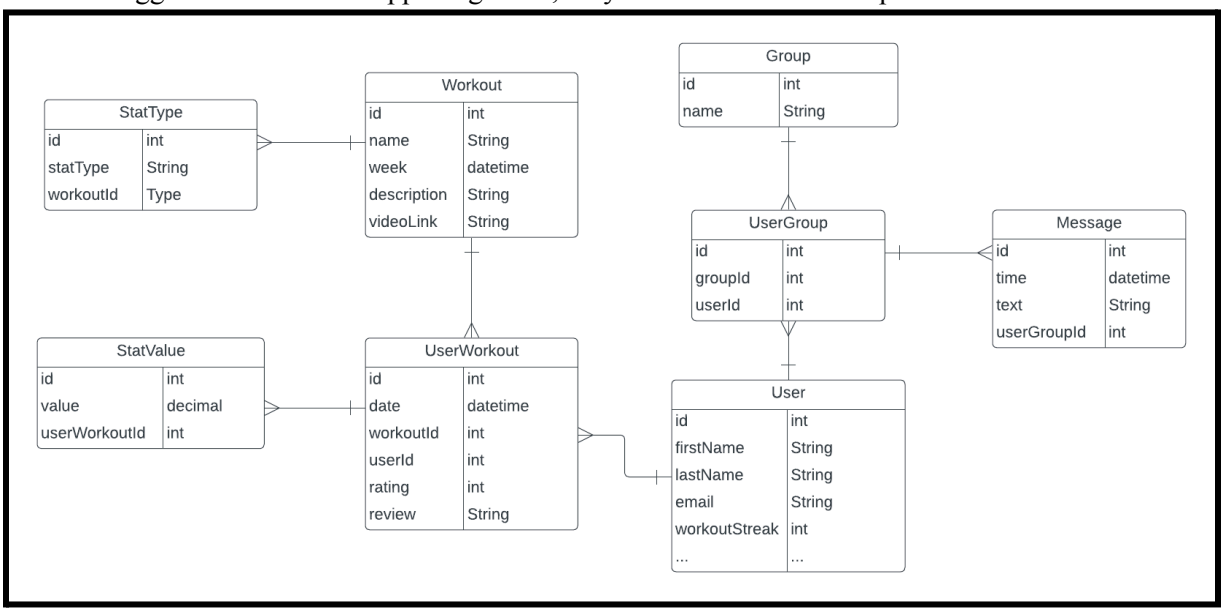
*Red components represent the focus of our capstone project



Back to top

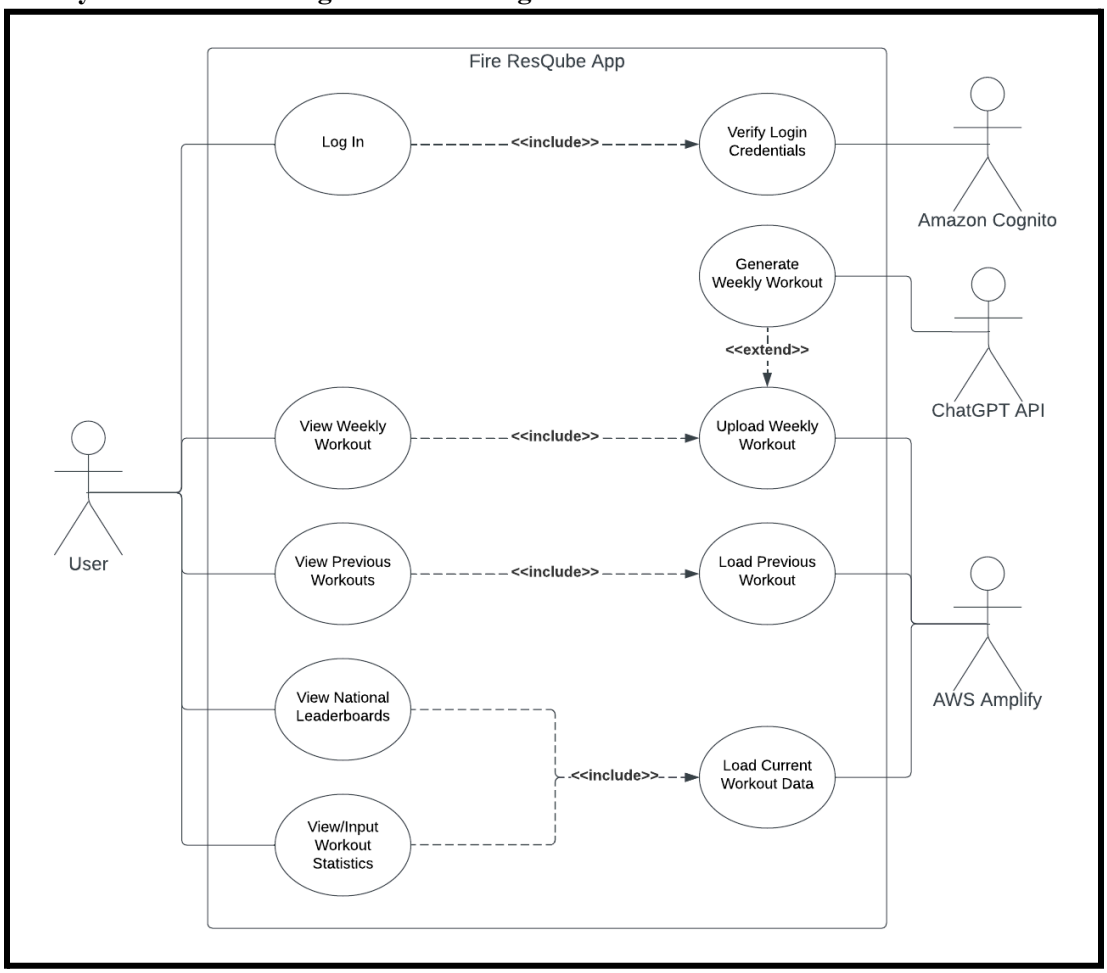
Weekly Workout Challenge ER Diagram

*Given struggles with database-app integration, only the User table was implemented



Back to top

Weekly Workout Challenge Use Case Diagram



Design Pattern
In order to integrate a workout generation function using the ChatGPT API with our React Native project, we will use the adapter pattern. React Native Expo is not directly compatible with Python, so we will incorporate AWS Lambda as an adapter by adding our Python algorithm as a 'function' within Amplify and accessing the generated workout from the cloud. This will allow us to communicate with the ChatGPT API from within Fire ResQube by using AWS Lambda as the translator for the incompatible components.

Development Standards
- **Naming:** Files, functions, and variables use descriptive names and camel case format for consistency and ease of readability.
- **Error Handling:** Use of try catch blocks to assist in handling errors with exceptions.
- **Version Control:** Descriptive commit messages to help determine work completed in each stage of the project.
- **Security:** Avoiding hard coded sensitive information in the database and using secure network communication protocols.
- **Code Optimization:** Avoidance of duplicated code fragments and unnecessary database connections for reusability and speed.
- **File Organization:** Logical grouping of files and directories in the project for ease of navigation.

Tools Used
- **Lucidchart:** To prepare and update architectural design documents.
- **GitHub:** To store code and collaborate remotely.
- **AWS Amplify:** The cloud computing provider to help run Fire ResQube and its databases.
- **Amazon Cognito:** Connects with AWS Amplify to support authentication and user management.
- **AWS AppSync:** Serves as the serverless API manager for the ResQube database.
- **DynamoDB:** A NoSQL database service to interact with tables and table values.
- **AWS Lambda:** A serverless Amazon service to run code outside of a project.
- **React Native Expo:** Serves as the framework of the application, allowing easy deployment to Android and IOS.

Work Schedule

Phase	Due Date
Planning Phase Part 1 - Gather project requirements - Completed By: Dawson	10/26/2023
Planning Phase Part 2 - Establish proposal requirements - Completed By: Nathan and Dawson	11/02/2023
Planning Phase Part 3 - Complete and submit proposal - Completed By: Nathan and Dawson	11/30/2023
Research Phase Part 1 - Declare software and technologies - Database and Cloud Software Completed By: Nathan - App UI and AI Algorithm Completed By: Dawson	12/14/2023
Research Phase Part 2 - Confirm proper code design - Completed By: Dawson	01/26/2024
Implementation Phase Part 1 - Complete proper App UI - Completed By: Dawson	02/26/2024
Implementation Phase Part 2 - Database, AI, Cloud components - Database and Cloud Completed By: Nathan - AI Completed By: Dawson	03/22/2024
Implementation Phase Part 3 - Testing - Completed By: Nathan and Dawson	04/05/2024
Final Implementation - Montana State University Research Celebration - Final Project 4'x3' Poster Completed By: Nathan - Final Project Implementation Completed By: Dawson	04/25/2024
Final Demonstration - Interdisciplinary Capstone Presentation - Completed By: Nathan and Dawson	04/30/2024

Planning Phase
The planning phase consists of two main parts, followed by the completion and submission of our project proposal. The CSCI 482 project proposal serves as a reference for our project, containing a plan and summary of each project component.

The first planning phase consists of gathering the overall project requirements. This is needed for us to determine what we want our project to be and the steps to get there. We will work with Dr. Izurieta to ensure that our project satisfies all requirements as needed for the course. This phase will give us a

general idea of our project to enable us to create a schedule, along with the requirements to complete along the way.

The second planning phase consists of establishing the overall proposal requirements. The proposal is an integral part of our project and serves as a summary and plan of our project. Although the proposal requirements are laid out in the course syllabus, this phase consists of taking those requirements and configuring them to Fire ResQube.

Research Phase
The research phase consists of two main parts to make sure we use the proper software to complete our project requirements and to ensure that our code will work seamlessly to make one complete program.

Nathan is in charge of researching software to use for our database and cloud computing requirements. The database softwares to be considered are Microsoft SSMS, IBM DB2, and Oracle RDBMS. Other database management systems may be considered if the aforementioned systems don't integrate with our program in the way that is necessary. Cloud software research will also be completed on Amazon AWS and Microsoft Azure, with Google Cloud to be considered if needed for specific project requirements.

Dawson is in charge of researching software to use for the app's framework, user interface, and an artificial intelligence algorithm. The main app framework to be considered is React Native, given its cross-platform compatibility. We will focus on coding in Javascript based on our React framework, but will consider the use of Swift for iOS development and Java for database management and artificial intelligence algorithms.

Implementation Phase
The implementation phase consists of two main parts, followed by thorough testing to ensure that the program works in the way that we intend it to. This is where we begin programming the project, with our main focus being on the App UI, cloud, database, and artificial intelligence components. When those are complete, we will perform a testing phase to ensure proper function.

To begin, Dawson is in charge of setting up the App UI using the programming language(s) determined during the research phase. The App UI will consist of a simple interface and a professional look to properly portray Fire ResQube. We will ensure the proper integration for the weekly workout challenge and individual account capability and security. When the App UI is complete, Dawson will implement the artificial intelligence integration determined during the research phase.

At the same time, Nathan is in charge of setting up the database and cloud components of the program. The cloud provider will be used to manage the database component holding workout and account information. We will ensure the proper integration of both of these softwares for optimized functionality, reliability, and security.

Following the implementation of the main components of the project outlined above, we will work together to conduct a comprehensive testing phase to ensure the proper functionality of Fire ResQube. With the App UI, artificial intelligence, database, and cloud components integrated into the app, our test

phase will cover each aspect. The main focus will be functionality and security of the app in preparation for the demo.

Final Implementation

After our testing phase is complete, the Fire ResQube app will be ready to present at the project demo on April 30, 2024.We will prepare the Fire ResQube App for presentation and create the 4’x 3’ poster required for the research celebration on April 25, 2024. The poster will include a summary and presentation of Fire ResQube, as well as the softwares used.

Lifecycle Approach

As a lifecycle approach we plan to utilize agile methodology. Agile methodology is a project management style where the larger project is broken down into small, incremental steps. The timeframe of these steps are commonly known as “sprints” and we plan for our sprints to be two weeks each. At the beginning of each sprint, we will have a “sprint plan” meeting where we determine what to complete during the timeframe of each sprint. Note that each sprint will not consist of a singular, large overall task to complete, but rather a series of small tasks that when added together will help complete the large overall task.

We plan to utilize agile methodology as our lifecycle approach as it provides some specific benefits that are not provided by other lifecycle approaches such as the waterfall method. The main benefit of agile is that it provides us as programmers the ability to see if and how our code works at the end of the sprint. This allows the ability to correct issues during the next sprint if new issues arise throughout the implementation of the project. Intrinsically, this also enables the ability to correct and change requirements or client needs as the product is being developed, rather than having to go back and make changes after development is complete.

With previous internship and class experience, we have an introductory level of agile knowledge. We expect to get more comfortable as the semester progresses and find some unique tricks to maximize our productivity.

Results

After a semester of development, we successfully implemented crucial features of Fire ResQube. Setting up the framework and integrating the backend in the cloud presented more challenges than initially anticipated. However, once the setup was complete, we focused on developing the main components outlined in our proposal such as the login system, profile page, and weekly workout feature.

We were able to create a well laid out user interface for the app to serve as a great prototype. The authentication through Amazon Cognito is our most deployment ready feature as it provides secure two-factor authentication for account creation and login functionality. The integration with AWS Amplify proved to be our biggest struggle. We were able to link our React Native project with Amplify and implement our database schema through the amplify website, but struggled with the use of the Amplify API to allow us to access the database within our project and deploy CRUD operations. Along with the database issues, we had trouble leveraging AWS Lambda's 'functions' tool which would allow us to deploy our ChatGPT configured workout-generation component written in Python.

The use of agile sprints were extremely helpful when encountering these issues. We decided to begin meeting every week as opposed to our planned two week sprints in order to break the project up into smaller tasks and discuss any trouble we were having. Our problems ultimately led to us falling off course from the original schedule, but we were able to communicate effectively and continue making progress on the project thanks to the flexibility of the agile process.

Given all the struggles we came across, from React Native and IOS Simulator troubleshooting to AWS Amplify integration, we are proud of our results and all the learning that came with the project. Reflecting on our attempt to develop an app using JavaScript with a cloud-based backend, all of which we had little to no experience in, we've realized that the inability to complete all we set out to do was inevitable. We would have liked to finish the database integration in order to make the app more functional, but ran out of time at the end of the semester. If we could go back, we would have started development on Fire ResQube in the fall, and looked into more database options such as Supabase in order to make the backend integration a little simpler.

Overall, the capstone experience has provided us with new insights working with cloud providers and developing apps. Also, we have become much more proficient in troubleshooting unknown errors via the command line. Ultimately, all the things we have learned through the development of Fire ResQube will help us in our software careers.

References

Amazon. (2024). *AWS Amplify Documentation*. https://docs.amplify.aws/

Amazon. (2024). *Amazon Cognito Documentation*. https://docs.aws.amazon.com/cognito/

Expo. (2024) *Create amazing apps that run everywhere*. Expo Documentation. (n.d.). https://docs.expo.dev/

Kowal, E. (2018). *Nature of firefighting requires top physical condition*. www.army.mil. https://www.army.mil/article/212807/nature_of_firefighting_requires_top_physical_condition

OpenAI. (2022). ChatGPT. https://openai.com/chatgpt

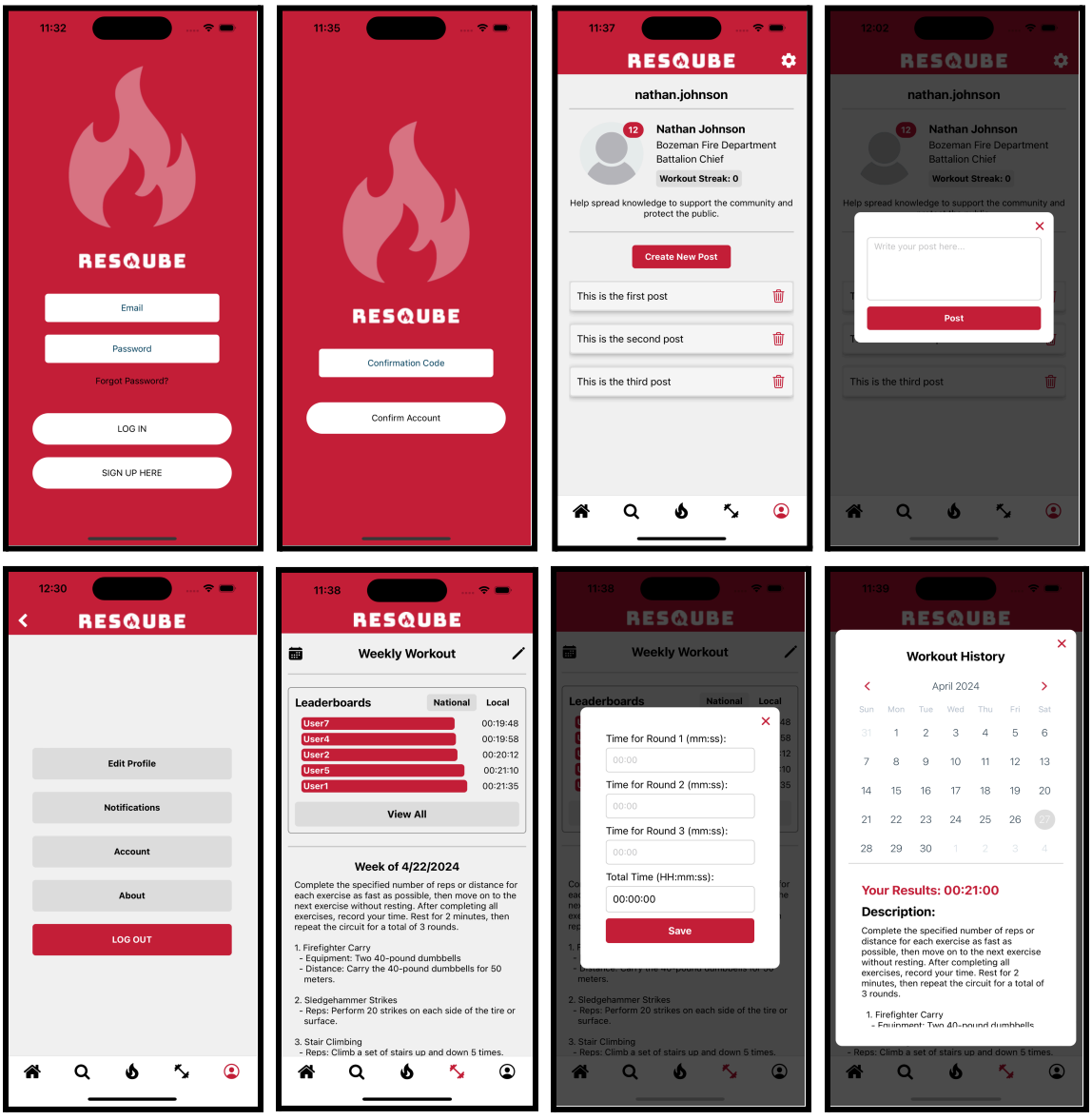
Strava Inc. (2023). *About Us*. Strava. https://www.strava.com/about

YuMuuv. (2023). *Team wellness*. https://yumuuv.com/

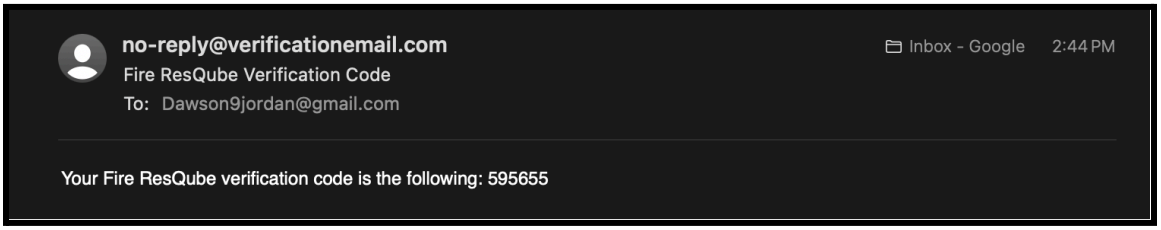
Zamzow, A. (2022). *Aaron Zamzow – firefighter & EMT: Owner at fire rescue fitness*. Fire Rescue Fitness. https://firerescuefitness.com/about-aaron/

Appendix

UI Design

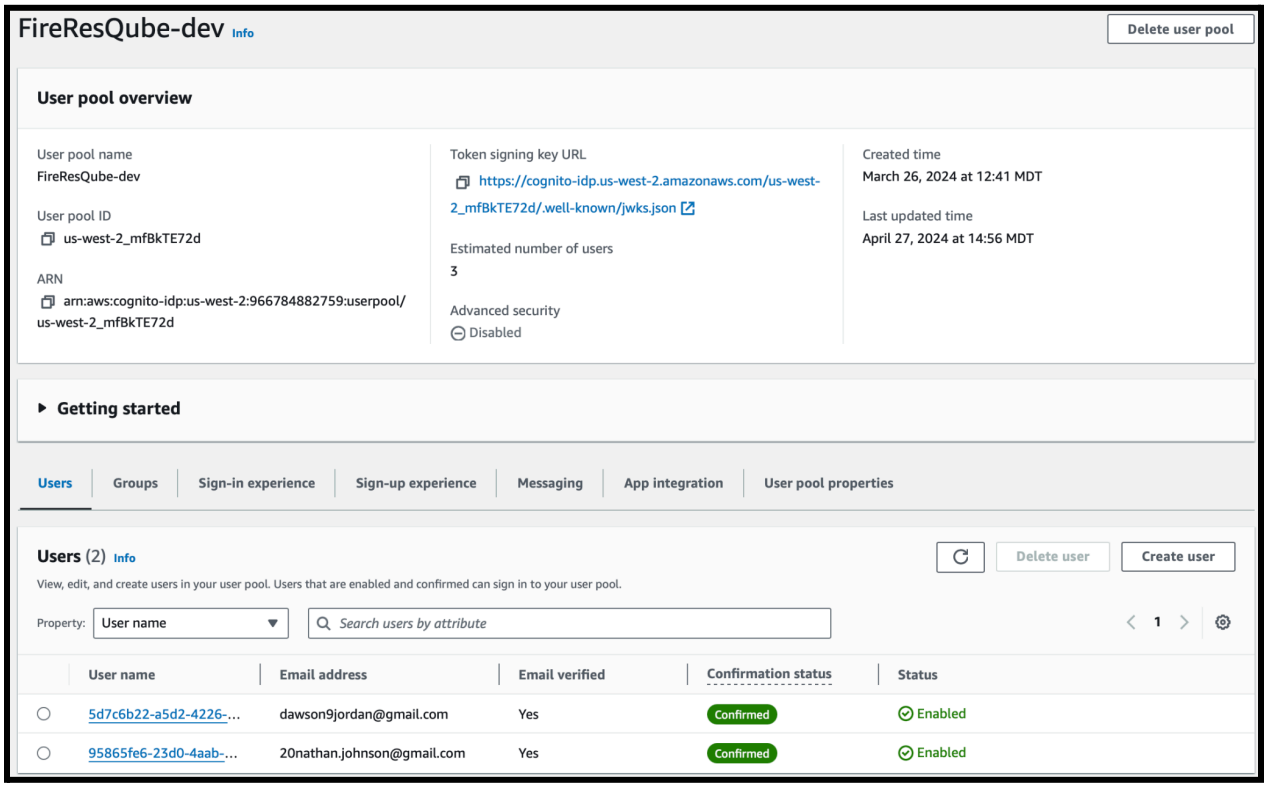


Verification Email



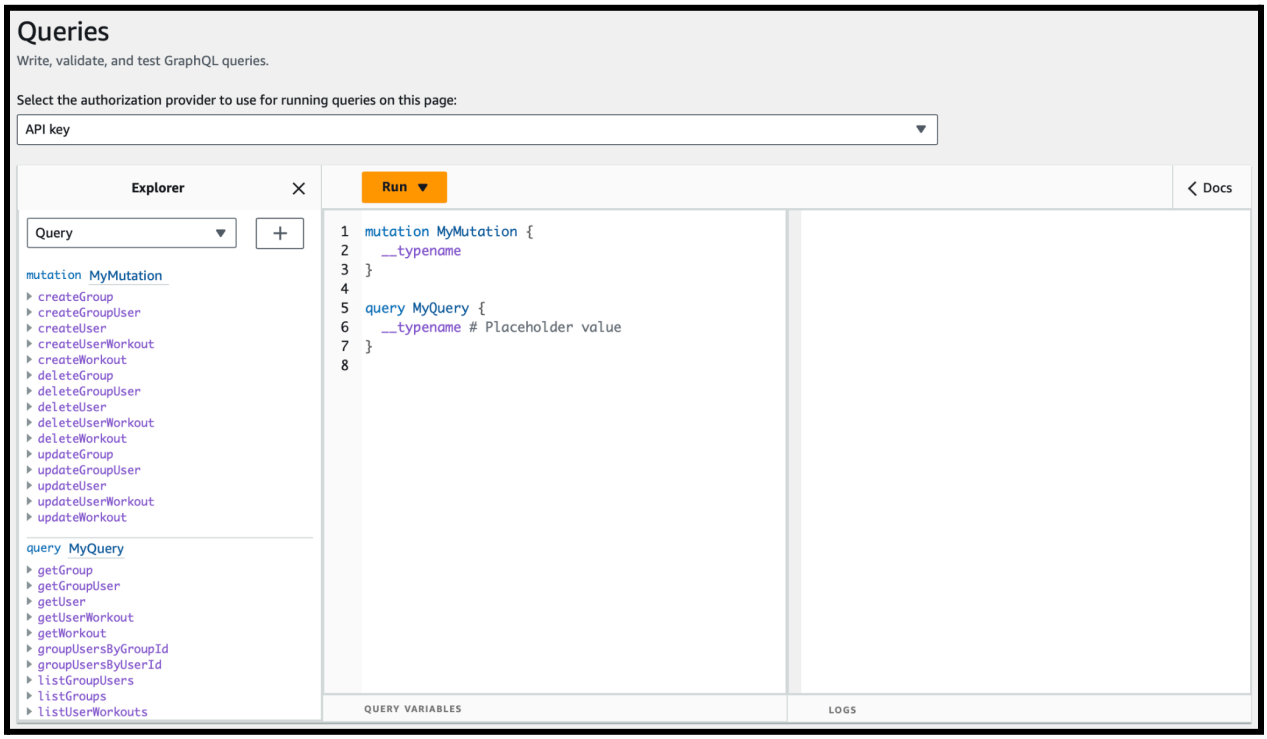
Back to top

Amazon Cognito (User Management)

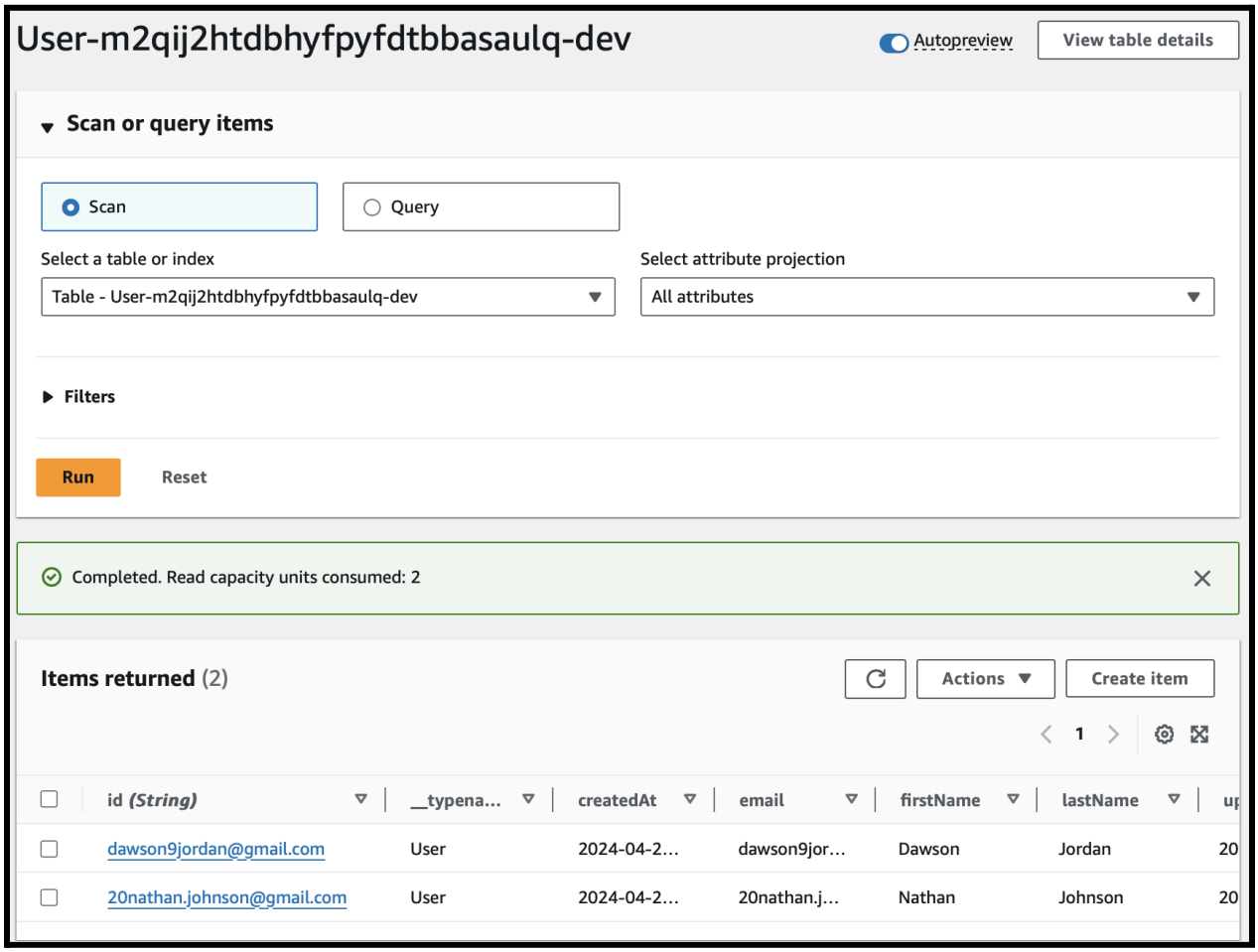


Back to top

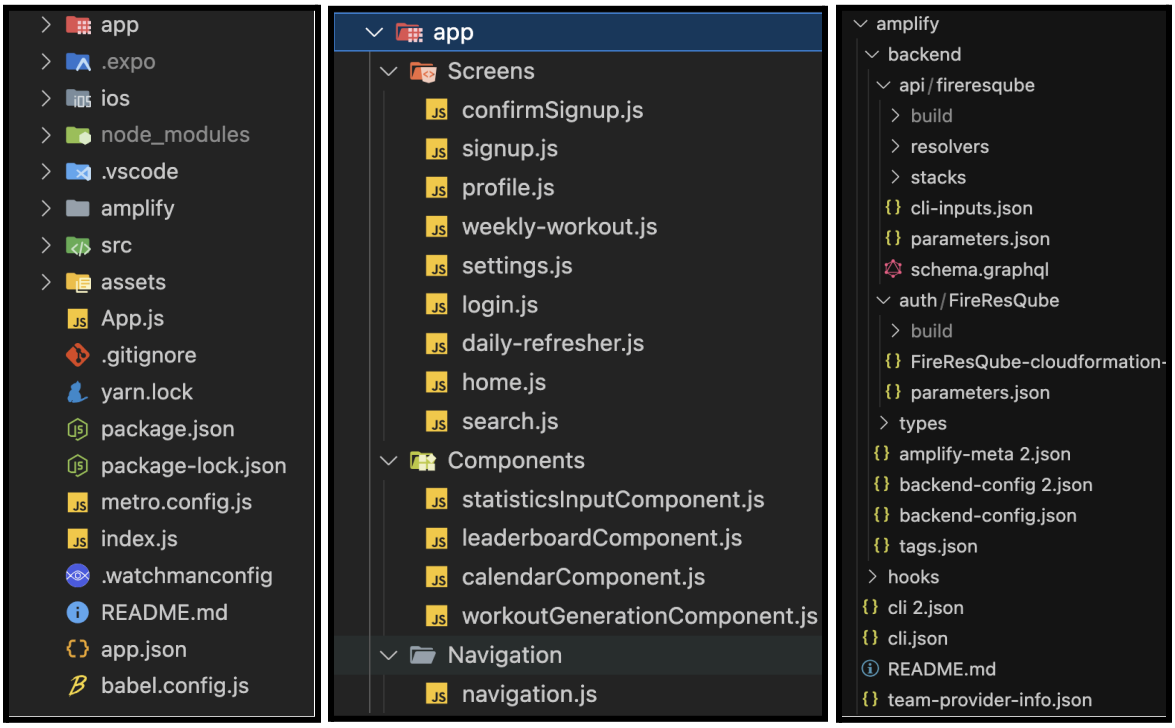
AWS AppSync screenshot



DynamoDB


User-m2qij2htdbhyfpyfdtbbasaulq-dev
Autopreview
View table details
Scan or query items
Scan
Query
Select a table or index
Table - User-m2qij2htdbhyfpyfdtbbasaulq-dev
Select attribute projection
All attributes
Filters
Run
Reset
Completed. Read capacity units consumed: 2
Items returned (2)
Actions
Create item
1
id (String)
__typena...
createdAt
email
firstName
lastName
dawson9jordan@gmail.com
User
2024-04-2...
dawson9jor...
Dawson
Jordan
20nathan.johnson@gmail.com
User
2024-04-2...
20nathan.j...
Nathan
Johnson

File Structure



Code

*ChatGPT was used to aid in JavaScript UI coding

App.js

```
import * as nav from "./app/Navigation/navigation"
import { Amplify } from "aws-amplify";
import amplifyconfig from './src/amplifyconfiguration.json';
Amplify.configure(amplifyconfig);

function App() {
 return (
   nav.AppNavigation()
 );
}

export default App;
```

navigation.js

```
import * as React from 'react';
import { Image, TouchableOpacity } from 'react-native';
import { FontAwesome, MaterialIcons, Ionicons } from '@expo/vector-icons';
import { NavigationContainer, useNavigation } from '@react-navigation/native';
import { createBottomTabNavigator } from '@react-navigation/bottom-tabs';
import { createStackNavigator } from '@react-navigation/stack';

import * as home from "../Screens/home";
import * as profile from "../Screens/profile";
import * as dailyRefresher from "../Screens/daily-refresher";
import * as weeklyWorkout from "../Screens/weekly-workout";
import * as search from "../Screens/search";
import * as settings from "../Screens/settings";
import * as login from "../Screens/login";
import * as signUp from "../Screens/signup";
import * as confirmSignUp from "../Screens/confirmSignup";

const Tab = createBottomTabNavigator();
const AppStack = createStackNavigator();
const ProfileStack = createStackNavigator();

const ProfileStackFun = () => {
 const navigation = useNavigation();
 return (
   <ProfileStack.Navigator
     screenOptions={{
       headerTitle: () => (<Image style={{ objectFit: 'contain' }}
source={require('../../assets/images/ResQube.png')} />),
       headerStyle: { backgroundColor: "#C41E3A"},
       headerLeft: ()=>null
     }}>
     <ProfileStack.Screen
       name="Profile"
       component={profile.ProfileScreen}
       options={{
         headerRight: () => (
           <TouchableOpacity
             onPress={() => navigation.navigate('Settings')}
             style={{ marginRight: 15 }}>
             <Ionicons name="settings-sharp" size={26} color="white" />
           </TouchableOpacity>
```

```
        ),
      }}
    />
    <ProfileStack.Screen
      name="Settings"
      component={settings.SettingsScreen}
      options={{
        headerLeft: () => (
          <TouchableOpacity
            onPress={() => navigation.navigate('Profile')}
            style={{ marginLeft: 15 }}>
            <FontAwesome name="chevron-left" size={24} color="white" />
          </TouchableOpacity>),
      }}
    />
  </ProfileStack.Navigator>
);
};

const NavigationBar = () => {
 return (
    <Tab.Navigator
      screenOptions={{
        showIcon: true,
        tabBarShowLabel: false,
        headerTitle: () => (<Image style={{ objectFit: 'contain'}}
source={require('../../assets/images/ResQube.png')} />),
        headerStyle: { backgroundColor: "#C41E3A"},
      }}>
      <Tab.Screen
        name="Home Tab"
        component={home.HomeScreen}
        options={{ tabBarIcon: (tabInfo) => (<FontAwesome name="home" size={30}
color={tabInfo.focused ? "#C41E3A" : "black"} />)}}
      />
      <Tab.Screen
        name="Search Tab"
        component={search.SearchScreen}
        options={{ tabBarIcon: (tabInfo) => (<FontAwesome name="search" size={26}
color={tabInfo.focused ? "#C41E3A" : "black"} />)}}
      />
      <Tab.Screen
```

```
          name="Daily Refresher Tab"
          component={dailyRefresher.DailyRefresherScreen}
          options={{ tabBarIcon: (tabInfo) => (<MaterialIcons
name="local-fire-department" size={30} color={tabInfo.focused ? "#C41E3A" : "black"}
/>)}}
        />
        <Tab.Screen
          name="Weekly Workout Tab"
          component={weeklyWorkout.WeeklyWorkoutScreen}
          options={{ tabBarIcon: (tabInfo) => (<MaterialIcons name="fitness-center"
size={28} color={tabInfo.focused ? "#C41E3A" : "black"} />)}}
        />
        <Tab.Screen
          name="Profile Tab"
          component={ProfileStackFun}
          options={{
            tabBarIcon: (tabInfo) => (<Ionicons name="person-circle-outline" size={30}
color={tabInfo.focused ? "#C41E3A" : "black"} />),
            headerShown: false
          }}
        />
      </Tab.Navigator>
  );
}

export function AppNavigation() {
  return (
    <NavigationContainer>
      <AppStack.Navigator
        screenOptions={{ headerShown: false,}}
        initialRouteName="Login"
      >
        <AppStack.Screen
          name="Login"
          component={login.LoginScreen}
        />
        <AppStack.Screen
          name="Signup"
          component={signUp.SignupScreen}
        />
        <AppStack.Screen
          name="NavigationBar"
```

```
          component={NavigationBar}
        />
        <AppStack.Screen
          name="ConfirmSignUp"
          component={confirmSignUp.ConfirmSignUpScreen}
        />
      </AppStack.Navigator>
    </NavigationContainer>
  );
}
```

login.js

```
// initial login code from
https://code.tutsplus.com/common-react-native-app-layouts-login-page--cms-27639t

import { StatusBar } from "expo-status-bar";
import { useNavigation } from '@react-navigation/native';
import React, { useState } from "react";
import { signIn, signOut } from 'aws-amplify/auth';
import {
 StyleSheet,
 Text,
 View,
 Image,
 TextInput,
 TouchableOpacity
} from "react-native";

export function LoginScreen() {
 const [email, setEmail] = useState("");
 const [password, setPassword] = useState("");
 const [error, setError] = useState("");
 const navigation = useNavigation();

 const handleLoginPress = async () => {
   setError('');
   try {
     await signOut();
   } catch (error) {
     console.log('error signing out: ', error);
   }

   try {
     const { isSignedIn } = await signIn({
       username: email,
       password,
     });
     console.log(isSignedIn);
     if (isSignedIn) {
       navigation.navigate('NavigationBar');
     }
   } catch (e) {
     setError(e.message);
```

```
  }
};
const handleSignupPress1 = () => {
  navigation.navigate('Signup')
}

return (
  <View style={styles.container}>
    <Image style={styles.image} source={require("../../assets/images/flame.png")} />
    <Image style={styles.image} source={require("../../assets/images/ResQube.png")}
/>
    <StatusBar style="auto" />
    <View style={styles.inputView}>
      <TextInput
        style={styles.TextInput}
        placeholder="Email"
        placeholderTextColor="#003f5c"
        autoCapitalize="none"
        onChangeText={(email) => setEmail(email)}
      />
    </View>
    <View style={styles.inputView}>
      <TextInput
        style={styles.TextInput}
        placeholder="Password"
        placeholderTextColor="#003f5c"
        autoCapitalize="none"
        //secureTextEntry={true}
        onChangeText={(password) => setPassword(password)}
      />
    </View>
    <TouchableOpacity>
      <Text style={styles.forgot_button}>Forgot Password?</Text>
    </TouchableOpacity>
    <TouchableOpacity
      onPress={handleLoginPress}
      style={styles.loginBtn}
    >
      <Text>LOG IN</Text>
    </TouchableOpacity>
    <TouchableOpacity
      onPress={handleSignupPress1}
```

```
        style={styles.loginBtn} // Same style as login button
      >
        <Text>SIGN UP HERE</Text>
      </TouchableOpacity>
      {error && <Text>{error}</Text>}
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: "#C41E3A",
    alignItems: "center",
    justifyContent: "center",
  },
  image: {
    marginBottom: 40,
  },
  inputView: {
    backgroundColor: "#FFF",
    borderRadius: 5,
    width: "70%",
    height: 45,
    marginBottom: 20,
    alignItems: "center",
  },
  TextInput: {
    height: 50,
    flex: 1,
    padding: 10,
  },
  forgot_button: {
    height: 30,
    marginBottom: 10,
  },
  loginBtn: {
    width: "80%",
    borderRadius: 25,
    height: 50,
    alignItems: "center",
    justifyContent: "center",
```

```
    marginTop: 20,
    backgroundColor: "#FFF",
  },
  signOutText: {
    textDecorationLine: 'underline',
    color: '#FFF', // Set color to white
    marginTop: 10,
  },
});
```

signup.js

```
// initial login code from
https://code.tutsplus.com/common-react-native-app-layouts-login-page--cms-27639t

import { StatusBar } from "expo-status-bar";
import { useNavigation } from '@react-navigation/native';
import React, { useState } from "react";
import { signUp } from 'aws-amplify/auth';
import { generateClient } from 'aws-amplify/api';
import * as mutations from '../../src/graphql/mutations';
import {
 StyleSheet,
 Text,
 View,
 Image,
 TextInput,
 TouchableOpacity,
} from "react-native";

export function SignupScreen() {
 const [email, setEmail] = useState("");
 const [password, setPassword] = useState("");
 const [error, setError] = useState("");
 const [first, setFirst] = useState("");
 const [last, setLast] = useState("");
 const navigation = useNavigation();
 const client = generateClient();

 async function createCurrentUser() {
   const userDetails = {
     id: email,
     email: email,
     firstName: first,
     lastName: last,
     workoutStreak: 0
   };
   try {
     await client.graphql({
       query: mutations.createUser,
       variables: { input: userDetails }
     });
   } catch (error) {
```

```
    console.error('Error creating user:', error);
  }
};

const handleSignupPress = async () => {
  setError('');
  try {
    await signUp({
      username: email,
      password: password
    });
    console.log("First Name: ", first);
    console.log("Last Name: ", last);
    console.log("Email: ", email);
  createCurrentUser();
  navigation.navigate('ConfirmSignUp', { email: email });
  } catch (e) {
    setError(e.message);
  }
};

return (
  <View style={styles.container}>
    <Image style={styles.image} source={require("../../assets/images/flame.png")} />
    <Image style={styles.image} source={require("../../assets/images/ResQube.png")}
/>
    <StatusBar style="auto" />
    <View style={styles.inputView}>
      <TextInput
        style={styles.TextInput}
        placeholder="Email"
        placeholderTextColor="#003f5c"
        autoCapitalize="none"
        onChangeText={(email) => setEmail(email)}
      />
    </View>
    <View style={styles.inputView}>
      <TextInput
        style={styles.TextInput}
        placeholder="Password"
        placeholderTextColor="#003f5c"
        autoCapitalize="none"
```

```
        // secureTextEntry={true}
        onChangeText={(password) => setPassword(password)}
      />
    </View>
    <View style={styles.inputView}>
      <TextInput
        style={styles.TextInput}
        placeholder="FirstName"
        placeholderTextColor="#003f5c"
        autoCapitalize="none"
        onChangeText={(first) => setFirst(first)}
      />
    </View>
    <View style={styles.inputView}>
      <TextInput
        style={styles.TextInput}
        placeholder="LastName"
        placeholderTextColor="#003f5c"
        autoCapitalize="none"
        onChangeText={(last) => setLast(last)}
      />
    </View>
    <TouchableOpacity
      onPress={handleSignupPress}
      style={styles.signupBtn}
    >
      <Text>SIGN UP</Text>
    </TouchableOpacity>
    {error && <Text>{error}</Text>}
  </View>
 );
}

const styles = StyleSheet.create({
 container: {
   flex: 1,
   backgroundColor: "#C41E3A",
   alignItems: "center",
   justifyContent: "center",
 },
 image: {
   marginBottom: 40,
```

```
  },
  inputView: {
    backgroundColor: "#FFF",
    borderRadius: 5,
    width: "70%",
    height: 45,
    marginBottom: 20,
    alignItems: "center",
  },
  TextInput: {
    height: 50,
    flex: 1,
    padding: 10,
  },
  forgot_button: {
    height: 30,
    marginBottom: 20,
  },
  signupBtn: {
    width: "80%",
    borderRadius: 25,
    height: 50,
    alignItems: "center",
    justifyContent: "center",
    marginTop: 20,
    backgroundColor: "#FFF",
  },
});
```

confirmSignup.js

```
import { StatusBar } from "expo-status-bar";
import { useNavigation, useRoute } from '@react-navigation/native';
import React, { useState } from "react";
import { confirmSignUp } from 'aws-amplify/auth';
import {
 StyleSheet,
 Text,
 View,
 Image,
 TextInput,
 TouchableOpacity,
} from "react-native";

export function ConfirmSignUpScreen() {
 const [confirmationCode, setConfirmationCode] = useState("");
 const [error, setError] = useState("");
 const navigation = useNavigation();
 const route = useRoute();
 const { email } = route.params;

 const handleConfirmationPress = async () => {
   setError('');
   try {
     const { isConfirmed } = await confirmSignUp({
       username: email,
       confirmationCode
     });
     navigation.navigate('NavigationBar', { email: email});
   } catch (e) {
     setError(e.message);
   }
 };

 return (
   <View style={styles.container}>
     <Image style={styles.image} source={require("../../assets/images/flame.png")} />
     <Image style={styles.image} source={require("../../assets/images/ResQube.png")}
/>
     <StatusBar style="auto" />
     <View style={styles.inputView}>
       <TextInput
```

```
        style={styles.TextInput}
        placeholder="Confirmation Code"
        placeholderTextColor="#003f5c"
        autoCapitalize="none"
        onChangeText={(confirmationCode) => setConfirmationCode(confirmationCode)}
      />
    </View>
    <TouchableOpacity
      onPress={handleConfirmationPress}
      style={styles.signupBtn}
    >
      <Text>Confirm Account</Text>
    </TouchableOpacity>
    {error && <Text>{error}</Text>}
  </View>
 );
}

const styles = StyleSheet.create({
 container: {
   flex: 1,
   backgroundColor: "#C41E3A",
   alignItems: "center",
   justifyContent: "center",
 },
 image: {
   marginBottom: 40,
 },
 inputView: {
   backgroundColor: "#FFF",
   borderRadius: 5,
   width: "70%",
   height: 45,
   marginBottom: 20,
   alignItems: "center",
 },
 TextInput: {
   height: 50,
   flex: 1,
   padding: 10,
 },
 forgot_button: {
```

```
    height: 30,
    marginBottom: 20,
  },
  signupBtn: {
    width: "80%",
    borderRadius: 25,
    height: 50,
    alignItems: "center",
    justifyContent: "center",
    marginTop: 20,
    backgroundColor: "#FFF",
  },
});
```

home.js

```
import React from 'react';
import { Text, View } from 'react-native';

export function HomeScreen() {
   return (
     <View style={{ flex: 1, justifyContent: 'center', alignItems: 'center' }}>
       <Text>Home!</Text>
     </View>
   );
};
```

search.js

```
import React from 'react';
import { Text, View } from 'react-native';

export function HomeScreen() {
   return (
     <View style={{ flex: 1, justifyContent: 'center', alignItems: 'center' }}>
       <Text>Home!</Text>
     </View>
   );
};
```

daily-refresher.js

```
import * as React from 'react';
import { Text, View } from 'react-native';

export function DailyRefresherScreen() {
   return (
     <View style={{ flex: 1, justifyContent: 'center', alignItems: 'center' }}>
       <Text>Daily Refresher!</Text>
     </View>
   );
 }
```

profile.js

```
import React, { useState, useEffect } from 'react';
import { View, Text, Image, TouchableOpacity, StyleSheet, Modal, TextInput, ScrollView
} from 'react-native';
import { useNavigation } from '@react-navigation/native';
import { Ionicons } from '@expo/vector-icons';
import { generateClient } from 'aws-amplify/api';
import { getUser } from '../../src/graphql/queries';
import { getCurrentUser } from 'aws-amplify/auth';

async function currentAuthenticatedUser(setUserEmailCallback) {
 try {
   const user = await getCurrentUser();
   const { signInDetails } = user;
   if (signInDetails) {
     const { loginId } = signInDetails;
     setUserEmailCallback(loginId); // Set userEmail using the provided callback
   } else {
     console.log('SignInDetails not found.');
   }
 } catch (err) {
   console.log('Error fetching current user:', err);
 }
}

export function ProfileScreen() {
 const client = generateClient(); // client to interact with backend database
(DynamoDB)
 const [userEmail, setUserEmail] = useState(null); // State for userEmail

 useEffect(() => {
   currentAuthenticatedUser(setUserEmail);
 }, []);

 async function getCurrentUser() {
   const currentUser = await client.graphql({
     query: getUser,
     variables: { id: userEmail }
   });
   return currentUser.data.getUser; // returns table object of current user with above
ID
 }
```

```
const [firstName, setFirstName] = useState('');
const [lastName, setLastName] = useState('');
const [streak, setStreak] = useState('');

useEffect(() => {
  async function fetchUser() {
    const user = await getCurrentUser(); // assigns the current user object to "user"
    setFirstName(user.firstName);
    setLastName(user.lastName);
    setStreak(user.workoutStreak);
    setEmail(user.email);
  }
  fetchUser();
  }, [userEmail]);

const user = {
  username: userEmail, // calls email from the database
  name: firstName + " " + lastName, // calls first and last namea form the database
  bio: 'Help spread knowledge to support the community and protect the public.',
  department: 'Bozeman Fire Department',
  Title: 'Battalion Chief',
  experience: 12,
  postCount: 0,
  profilePicture: require('../../assets/images/profile-picture.jpg'),
  workoutStreak: streak, // calls streak from the database
};

const [modalVisible, setModalVisible] = useState(false);
const [newPostContent, setNewPostContent] = useState('');
const [posts, setPosts] = useState([ //CONNECT TO DATABASE
  { id: 1, content: 'This is the first post' },
  { id: 2, content: 'This is the second post' },
  { id: 3, content: 'This is the third post' },
]);

const navigation = useNavigation();
const navigateToWeeklyWorkout = () => {
  navigation.navigate('Weekly Workout Tab');
};

const handleNewPost = () => {
```

```
  // Check if not empty
  if (newPostContent.trim() !== '') {
    const newPostId = posts.length + 1;
    const newPost = { id: newPostId, content: newPostContent };
    setPosts(prevPosts => [...prevPosts, newPost]);
    setNewPostContent('');
    setModalVisible(false);
  }
};

const handleDeletePost = (id) => {
  setPosts(prevPosts => prevPosts.filter(post => post.id !== id));
};

return (
  <View style={styles.container}>
    {/* Username */}
    <Text style={styles.username}>{user.username}</Text>

    {/* Separator */}
    <View style={styles.separator} />

    {/* Profile Picture and User Information */}
    <View style={styles.profileContainer}>
      {/* Profile Picture */}
      <View style={styles.profilePictureContainer}>
        <Image style={styles.profilePicture} source={user.profilePicture} />
        {/* Badge */}
        <View style={styles.badge}>
          <Text style={styles.badgeText}>{user.experience}</Text>
        </View>
      </View>
      {/* User Information */}
      <View style={styles.userInfo}>
        {/* Name */}
        <Text style={styles.name}>{user.name}</Text>
        {/* Department */}
        <Text style={styles.userDetails}>{user.department}</Text>
        {/* Title */}
        <Text style={styles.userDetails}>{user.Title}</Text>
        {/* Workout Streak */}
        <TouchableOpacity onPress={navigateToWeeklyWorkout}>
```

```
          <View style={styles.workoutStreakBox}>
            <Text style={styles.workoutStreak}>{`Workout Streak:
${user.workoutStreak}`}</Text>
          </View>
        </TouchableOpacity>
      </View>
    </View>

    {/* Bio */}
    <Text style={styles.bio}>{user.bio}</Text>

    {/* Separator */}
    <View style={styles.separator} />

    {/* Button to create a new post */}
    <TouchableOpacity onPress={() => setModalVisible(true)}
style={styles.createPostButton}>
      <Text style={styles.createPostButtonText}>Create New Post</Text>
    </TouchableOpacity>

    {/* Modal for creating a new post */}
    <Modal
      animationType="slide"
      transparent={true}
      visible={modalVisible}
      onRequestClose={() => {
        setModalVisible(false);
      }}
    >
      <View style={styles.modalContainer}>
        <View style={styles.modalContent}>
          <TouchableOpacity onPress={() => setModalVisible(false)}
style={styles.closeButton}>
            <Ionicons name="close" size={24} color="#C41E3A" />
          </TouchableOpacity>
          <TextInput
            placeholder="Write your post here..."
            multiline={true}
            style={styles.postInput}
            value={newPostContent}
            onChangeText={text => setNewPostContent(text)}
          />
```

```
          <TouchableOpacity onPress={handleNewPost} style={styles.postButton}>
            <Text style={styles.postButtonText}>Post</Text>
          </TouchableOpacity>
        </View>
      </View>
    </Modal>

    {/* Posts */}
    <ScrollView style={styles.postsContainer}>
      {posts.map(post => (
        <View key={post.id} style={styles.postContainer}>
          <View style={styles.post}>
            <Text style={styles.postContent}>{post.content}</Text>
            <TouchableOpacity onPress={() => handleDeletePost(post.id)}
style={styles.deleteButton}>
              <Ionicons name="trash-outline" size={24} color="#C41E3A" />
            </TouchableOpacity>
          </View>
        </View>
      ))}
    </ScrollView>
  </View>
 );
}

const styles = StyleSheet.create({
 container: {
   flex: 1,
   alignItems: 'center',
   padding: 20,
 },
 username: {
   fontSize: 20,
   fontWeight: 'bold',
   marginBottom: 10,
 },
 profileContainer: {
   flexDirection: 'row',
   alignItems: 'center',
   marginBottom: 20,
 },
 profilePictureContainer: {
```

```
  position: 'relative',
  marginRight: 20,
},
profilePicture: {
  width: 100,
  height: 100,
  borderRadius: 50,
},
badge: {
  position: 'absolute',
  top: 0,
  right: 0,
  backgroundColor: '#C41E3A',
  borderRadius: 50,
  paddingHorizontal: 8,
  paddingVertical: 4,
},
badgeText: {
  color: 'white',
  fontWeight: 'bold',
},
userInfo: {
  width: 200,
  height: 100,
  alignItems: 'flex-start',
},
name: {
  fontSize: 18,
  fontWeight: 'bold',
  marginBottom: 5,
},
userDetails: {
  fontSize: 16,
  marginBottom: 3,
},
workoutStreakBox: {
  backgroundColor: '#dcdcdc',
  padding: 5,
  borderRadius: 5,
  marginTop: 5,
},
workoutStreak: {
```

```
  color: 'black',
  fontWeight: 'bold',
},
bio: {
  textAlign: 'center',
  marginBottom: 20,
},
separator: {
  height: 2,
  width: '100%',
  backgroundColor: '#ccc',
  marginBottom: 20,
},
createPostButton: {
  backgroundColor: '#C41E3A',
  paddingVertical: 10,
  paddingHorizontal: 20,
  borderRadius: 5,
  marginBottom: 20,
},
createPostButtonText: {
  color: 'white',
  fontWeight: 'bold',
},
modalContainer: {
  flex: 1,
  justifyContent: 'center',
  alignItems: 'center',
  backgroundColor: 'rgba(0, 0, 0, 0.7)',
},
modalContent: {
  backgroundColor: 'white',
  padding: 20,
  paddingTop: 40,
  borderRadius: 10,
  width: '80%',
},
postInput: {
  borderWidth: 1,
  borderColor: '#ccc',
  borderRadius: 5,
  padding: 10,
```

```
  marginBottom: 10,
  minHeight: 100,
},
postButton: {
  backgroundColor: '#C41E3A',
  paddingVertical: 10,
  borderRadius: 5,
  alignItems: 'center',
},
postButtonText: {
  color: 'white',
  fontWeight: 'bold',
},
closeButton: {
  position: 'absolute',
  top: 10,
  right: 10,
  zIndex: 1,
},
closeButtonText: {
  color: '#C41E3A',
  fontWeight: 'bold',
  textAlign: 'center',
},
postsContainer: {
  width: '100%',
},
postContainer: {
  marginBottom: 10,
},
post: {
  flexDirection: 'row',
  alignItems: 'center',
  backgroundColor: '#f5f5f5',
  padding: 10,
  borderRadius: 5,
  marginBottom: 10,
  borderWidth: 2,
  borderColor: '#dcdcdc',
  shadowColor: '#000',
  shadowOffset: { width: 0, height: 2 },
  shadowOpacity: 0.3,
```

```
    shadowRadius: 4,
    elevation: 3,
  },
  postContent: {
    flex: 1,
    fontSize: 16,
  },
  deleteButton: {
    marginLeft: 10,
  },
});
import React, { useState, useEffect } from 'react';
import { View, Text, Image, TouchableOpacity, StyleSheet, Modal, TextInput, ScrollView
} from 'react-native';
import { useNavigation } from '@react-navigation/native';
import { Ionicons } from '@expo/vector-icons';
import { generateClient } from 'aws-amplify/api';
import { getUser } from '../../src/graphql/queries';
import { getCurrentUser } from 'aws-amplify/auth';

async function currentAuthenticatedUser(setUserEmailCallback) {
  try {
    const user = await getCurrentUser();
    const { signInDetails } = user;
    if (signInDetails) {
      const { loginId } = signInDetails;
      setUserEmailCallback(loginId); // Set userEmail using the provided callback
    } else {
      console.log('SignInDetails not found.');
    }
  } catch (err) {
    console.log('Error fetching current user:', err);
  }
}

export function ProfileScreen() {
  const client = generateClient(); // client to interact with backend database
(DynamoDB)
  const [userEmail, setUserEmail] = useState(null); // State for userEmail

  useEffect(() => {
    currentAuthenticatedUser(setUserEmail);
```

```
}, []);

async function getCurrentUser() {
  const currentUser = await client.graphql({
    query: getUser,
    variables: { id: userEmail }
  });
  return currentUser.data.getUser; // returns table object of current user with above
ID
}

const [firstName, setFirstName] = useState('');
const [lastName, setLastName] = useState('');
const [streak, setStreak] = useState('');

useEffect(() => {
  async function fetchUser() {
    const user = await getCurrentUser(); // assigns the current user object to "user"
    setFirstName(user.firstName);
    setLastName(user.lastName);
    setStreak(user.workoutStreak);
    setEmail(user.email);
  }
  fetchUser();
  }, [userEmail]);

const user = {
  username: userEmail, // calls email from the database
  name: firstName + " " + lastName, // calls first and last namea form the database
  bio: 'Help spread knowledge to support the community and protect the public.',
  department: 'Bozeman Fire Department',
  Title: 'Battalion Chief',
  experience: 12,
  postCount: 0,
  profilePicture: require('../../assets/images/profile-picture.jpg'),
  workoutStreak: streak, // calls streak from the database
};

const [modalVisible, setModalVisible] = useState(false);
const [newPostContent, setNewPostContent] = useState('');
const [posts, setPosts] = useState([ //CONNECT TO DATABASE
  { id: 1, content: 'This is the first post' },
```

```
  { id: 2, content: 'This is the second post' },
  { id: 3, content: 'This is the third post' },
]);

const navigation = useNavigation();
const navigateToWeeklyWorkout = () => {
  navigation.navigate('Weekly Workout Tab');
};

const handleNewPost = () => {
  // Check if not empty
  if (newPostContent.trim() !== '') {
    const newPostId = posts.length + 1;
    const newPost = { id: newPostId, content: newPostContent };
    setPosts(prevPosts => [...prevPosts, newPost]);
    setNewPostContent('');
    setModalVisible(false);
  }
};

const handleDeletePost = (id) => {
  setPosts(prevPosts => prevPosts.filter(post => post.id !== id));
};

return (
  <View style={styles.container}>
    {/* Username */}
    <Text style={styles.username}>{user.username}</Text>

    {/* Separator */}
    <View style={styles.separator} />

    {/* Profile Picture and User Information */}
    <View style={styles.profileContainer}>
      {/* Profile Picture */}
      <View style={styles.profilePictureContainer}>
        <Image style={styles.profilePicture} source={user.profilePicture} />
        {/* Badge */}
        <View style={styles.badge}>
          <Text style={styles.badgeText}>{user.experience}</Text>
        </View>
      </View>
```

```
      {/* User Information */}
      <View style={styles.userInfo}>
        {/* Name */}
        <Text style={styles.name}>{user.name}</Text>
        {/* Department */}
        <Text style={styles.userDetails}>{user.department}</Text>
        {/* Title */}
        <Text style={styles.userDetails}>{user.Title}</Text>
        {/* Workout Streak */}
        <TouchableOpacity onPress={navigateToWeeklyWorkout}>
          <View style={styles.workoutStreakBox}>
            <Text style={styles.workoutStreak}>{`Workout Streak:
${user.workoutStreak}`}</Text>
          </View>
        </TouchableOpacity>
      </View>
    </View>

    {/* Bio */}
    <Text style={styles.bio}>{user.bio}</Text>

    {/* Separator */}
    <View style={styles.separator} />

    {/* Button to create a new post */}
    <TouchableOpacity onPress={() => setModalVisible(true)}
style={styles.createPostButton}>
      <Text style={styles.createPostButtonText}>Create New Post</Text>
    </TouchableOpacity>

    {/* Modal for creating a new post */}
    <Modal
      animationType="slide"
      transparent={true}
      visible={modalVisible}
      onRequestClose={() => {
        setModalVisible(false);
      }}
    >
      <View style={styles.modalContainer}>
        <View style={styles.modalContent}>
```

```
          <TouchableOpacity onPress={() => setModalVisible(false)}
style={styles.closeButton}>
            <Ionicons name="close" size={24} color="#C41E3A" />
          </TouchableOpacity>
          <TextInput
            placeholder="Write your post here..."
            multiline={true}
            style={styles.postInput}
            value={newPostContent}
            onChangeText={text => setNewPostContent(text)}
          />
          <TouchableOpacity onPress={handleNewPost} style={styles.postButton}>
            <Text style={styles.postButtonText}>Post</Text>
          </TouchableOpacity>
        </View>
      </View>
    </Modal>

    {/* Posts */}
    <ScrollView style={styles.postsContainer}>
      {posts.map(post => (
        <View key={post.id} style={styles.postContainer}>
          <View style={styles.post}>
            <Text style={styles.postContent}>{post.content}</Text>
            <TouchableOpacity onPress={() => handleDeletePost(post.id)}
style={styles.deleteButton}>
              <Ionicons name="trash-outline" size={24} color="#C41E3A" />
            </TouchableOpacity>
          </View>
        </View>
      ))}
    </ScrollView>
  </View>
 );
}

const styles = StyleSheet.create({
 container: {
  flex: 1,
  alignItems: 'center',
  padding: 20,
 },
```

```
username: {
  fontSize: 20,
  fontWeight: 'bold',
  marginBottom: 10,
},
profileContainer: {
  flexDirection: 'row',
  alignItems: 'center',
  marginBottom: 20,
},
profilePictureContainer: {
  position: 'relative',
  marginRight: 20,
},
profilePicture: {
  width: 100,
  height: 100,
  borderRadius: 50,
},
badge: {
  position: 'absolute',
  top: 0,
  right: 0,
  backgroundColor: '#C41E3A',
  borderRadius: 50,
  paddingHorizontal: 8,
  paddingVertical: 4,
},
badgeText: {
  color: 'white',
  fontWeight: 'bold',
},
userInfo: {
  width: 200,
  height: 100,
  alignItems: 'flex-start',
},
name: {
  fontSize: 18,
  fontWeight: 'bold',
  marginBottom: 5,
},
```

```
userDetails: {
  fontSize: 16,
  marginBottom: 3,
},
workoutStreakBox: {
  backgroundColor: '#dcdcdc',
  padding: 5,
  borderRadius: 5,
  marginTop: 5,
},
workoutStreak: {
  color: 'black',
  fontWeight: 'bold',
},
bio: {
  textAlign: 'center',
  marginBottom: 20,
},
separator: {
  height: 2,
  width: '100%',
  backgroundColor: '#ccc',
  marginBottom: 20,
},
createPostButton: {
  backgroundColor: '#C41E3A',
  paddingVertical: 10,
  paddingHorizontal: 20,
  borderRadius: 5,
  marginBottom: 20,
},
createPostButtonText: {
  color: 'white',
  fontWeight: 'bold',
},
modalContainer: {
  flex: 1,
  justifyContent: 'center',
  alignItems: 'center',
  backgroundColor: 'rgba(0, 0, 0, 0.7)',
},
modalContent: {
```

```
  backgroundColor: 'white',
  padding: 20,
  paddingTop: 40,
  borderRadius: 10,
  width: '80%',
},
postInput: {
  borderWidth: 1,
  borderColor: '#ccc',
  borderRadius: 5,
  padding: 10,
  marginBottom: 10,
  minHeight: 100,
},
postButton: {
  backgroundColor: '#C41E3A',
  paddingVertical: 10,
  borderRadius: 5,
  alignItems: 'center',
},
postButtonText: {
  color: 'white',
  fontWeight: 'bold',
},
closeButton: {
  position: 'absolute',
  top: 10,
  right: 10,
  zIndex: 1,
},
closeButtonText: {
  color: '#C41E3A',
  fontWeight: 'bold',
  textAlign: 'center',
},
postsContainer: {
  width: '100%',
},
postContainer: {
  marginBottom: 10,
},
post: {
```

```
    flexDirection: 'row',
    alignItems: 'center',
    backgroundColor: '#f5f5f5',
    padding: 10,
    borderRadius: 5,
    marginBottom: 10,
    borderWidth: 2,
    borderColor: '#dcdcdc',
    shadowColor: '#000',
    shadowOffset: { width: 0, height: 2 },
    shadowOpacity: 0.3,
    shadowRadius: 4,
    elevation: 3,
  },
  postContent: {
    flex: 1,
    fontSize: 16,
  },
  deleteButton: {
    marginLeft: 10,
  },
});
```

settings.js

```
import React from 'react';
import { View, Text, StyleSheet, TouchableOpacity} from 'react-native';
import { useNavigation } from '@react-navigation/native';
import { signOut } from 'aws-amplify/auth';

export function SettingsScreen() {
 const navigation = useNavigation();

 const handleLogout = async () => {
   try {
     await signOut();
     navigation.navigate('Login');
   } catch (error) {
     console.log('error signing out: ', error);
   }
 };

 return (
   <View style={styles.container}>
     <TouchableOpacity style={styles.btn}>
       <Text style={styles.btnText}>Edit Profile</Text>
     </TouchableOpacity>
     <TouchableOpacity style={styles.btn}>
       <Text style={styles.btnText}>Notifications</Text>
     </TouchableOpacity>
     <TouchableOpacity style={styles.btn}>
       <Text style={styles.btnText}>Account</Text>
     </TouchableOpacity>
     <TouchableOpacity style={styles.btn}>
       <Text style={styles.btnText}>About</Text>
     </TouchableOpacity>

     <TouchableOpacity
       onPress={handleLogout}
       style={styles.logoutBtn}
     >
       <Text style={styles.logoutText}>LOG OUT</Text>
     </TouchableOpacity>
   </View>
 );
}
```

```
const styles = StyleSheet.create({
 container: {
   flex: 1,
   alignItems: "center",
   justifyContent: "center",
 },
 btn: {
   width: "80%",
   borderRadius: 5,
   height: 50,
   alignItems: "center",
   justifyContent: "center",
   marginTop: 20,
   backgroundColor: "#dcdcdc",
 },
 btnText: {
   color: "black",
   fontWeight: "bold"
 },
 logoutBtn: {
   width: "80%",
   borderRadius: 5,
   height: 50,
   alignItems: "center",
   justifyContent: "center",
   marginTop: 20,
   backgroundColor: "#C41E3A",
 },
 logoutText: {
   color: "#FFF",
   fontWeight: "bold"
 },
});
```

weekly-workout.js

```
import React, { useState, useEffect } from 'react';
import { View, ScrollView, StyleSheet, Modal, TouchableOpacity, Text} from
'react-native';
import { Ionicons } from '@expo/vector-icons';
import { getCurrentUser } from 'aws-amplify/auth';
import CalendarComponent from '../Components/calendarComponent';
import StatisticsInputComponent from '../Components/statisticsInputComponent';
import LeaderboardComponent from '../Components/leaderboardComponent';
import WorkoutGenerationComponent from '../Components/workoutGenerationComponent';

async function currentAuthenticatedUser(setUserEmailCallback) {
 try {
   const user = await getCurrentUser();
   const { signInDetails } = user;
   if (signInDetails) {
     const { loginId } = signInDetails;
     setUserEmailCallback(loginId); // Set userEmail using the provided callback
   } else {
     console.log('SignInDetails not found.');
   }
 } catch (err) {
   console.log('Error fetching current user:', err);
 }
}

export function WeeklyWorkoutScreen() {

 const [userEmail, setUserEmail] = useState(null); // State for userEmail

 useEffect(() => {
   currentAuthenticatedUser(setUserEmail);
 }, []);

 const [nationalData, setNationalData] = useState ([ // INTEGRATE DATABASE
   { id: 1, username: 'User1', time: '00:21:35', date: '2024-04-25' },
   { id: 2, username: 'User2', time: '00:20:12', date: '2024-04-25' },
   { id: 3, username: 'User3', time: '00:22:45', date: '2024-04-25' },
   { id: 4, username: 'User4', time: '00:19:58', date: '2024-04-25' },
   { id: 5, username: 'User5', time: '00:21:10', date: '2024-04-25' },
   { id: 6, username: 'User6', time: '00:24:03', date: '2024-04-25' },
   { id: 7, username: 'User7', time: '00:19:48', date: '2024-04-25' },
```

```
]);

const [localData, setLocalData] = useState ([
  { id: 6, username: 'User6', time: '00:24:03', date: '2024-04-25' },
  { id: 2, username: 'User2', time: '00:20:12', date: '2024-04-25' },
  { id: 1, username: 'User1', time: '00:21:35', date: '2024-04-25' },
]);

// Control modal visibility
const [isCalendarModalVisible, setCalendarModalVisible] = React.useState(false);
const [isStatisticsModalVisible, setStatisticsModalVisible] = React.useState(false);

const saveTotalTime = (totalTime) => {

  setStatisticsModalVisible(false)

  const newNationalEntry = {
    id: nationalData.length + 1,
    username: userEmail, // Replace with actual username - DATABASE
    time: totalTime,
    date: new Date().toISOString().split('T')[0],
  };
  setNationalData([...nationalData, newNationalEntry]);

  const newLocalEntry = {
    id: localData.length + 1,
    username: userEmail, // Replace with actual username - DATABASE
    time: totalTime,
    date: new Date().toISOString().split('T')[0],
  };
  setLocalData([...localData, newLocalEntry]);
};

return (
  <ScrollView contentContainerStyle={styles.container}>
    {/* Header */}
    <View style={styles.header}>
      <TouchableOpacity onPress={() => setCalendarModalVisible(true)}>
        <Ionicons name="calendar" size={24} color="black" />
      </TouchableOpacity>
      <Text style={styles.headerTitle}>Weekly Workout</Text>
      <TouchableOpacity onPress={() => setStatisticsModalVisible(true)}>
```

```
          <Ionicons name="pencil" size={24} color="black" />
        </TouchableOpacity>
      </View>

      {/* Separator */}
      <View style={styles.separator} />

      {/* Leaderboards Section */}
      <View style={styles.section}>
        <LeaderboardComponent nationalData={nationalData} localData={localData}/>
      </View>

      {/* Separator */}
      <View style={styles.separator} />

      {/* Workout Generation Component */}
      <View style={styles.section}>
        <WorkoutGenerationComponent />
        <TouchableOpacity onPress={() => setStatisticsModalVisible(true)}
style={styles.button}>
          <Text style={styles.buttonText}>Upload Workout</Text>
        </TouchableOpacity>
      </View>

      {/* Calendar Modal */}
      <Modal visible={isCalendarModalVisible} animationType="slide" transparent={true}>
        <CalendarComponent onClose={() => setCalendarModalVisible(false)}
localData={localData} />
      </Modal>

      {/* Statistics Input Modal */}
      <Modal visible={isStatisticsModalVisible} animationType="slide"
transparent={true}>
        <StatisticsInputComponent onClose={() => setStatisticsModalVisible(false)}
onSave={saveTotalTime}/>
      </Modal>
    </ScrollView>
  );
}

const styles = StyleSheet.create({
  container: {
```

```
    flexGrow: 1,
    paddingVertical: 20,
    paddingHorizontal: 10,
  },
  header: {
    flexDirection: 'row',
    justifyContent: 'space-between',
    alignItems: 'center',
    marginBottom: 20,
  },
  headerTitle: {
    fontSize: 20,
    fontWeight: 'bold',
  },
  section: {
    marginBottom: 20,
  },
  separator: {
    height: 2,
    width: '100%',
    backgroundColor: '#ccc',
    marginBottom: 20,
  },
  button: {
    backgroundColor: '#C41E3A',
    paddingVertical: 10,
    paddingHorizontal: 20,
    borderRadius: 5,
    marginBottom: 20,
    alignSelf: 'stretch',
  },
  buttonText: {
    color: 'white',
    fontWeight: 'bold',
    textAlign: 'center',
  },
});
```

leaderboardComponent.js

```
import React, { useState, useEffect } from 'react';
import { Ionicons } from '@expo/vector-icons';
import { View, Text, StyleSheet, TouchableOpacity, Modal, ScrollView } from
'react-native';

const LeaderboardComponent = ({ nationalData, localData }) => {

 const [showNationalLeaderboard, setShowNationalLeaderboard] = useState(true);
 const [modalVisible, setModalVisible] = useState(false);
 const [leaderboardData, setLeaderboardData] = useState([]);

 useEffect(() => {
   setLeaderboardData(nationalData);
 }, []);

 // toggle between national and local leaderboards
 const toggleLeaderboard = () => {
   setShowNationalLeaderboard(!showNationalLeaderboard);
   setLeaderboardData(showNationalLeaderboard ? localData : nationalData);
 };

 const handleSeeAll = () => {
   setModalVisible(true);
 };

 // convert time strings to seconds
 const convertTimeToSeconds = (timeStr) => {
   const [hours, minutes, seconds] = timeStr.split(':').map(Number);
   return hours * 3600 + minutes * 60 + seconds;
 };

 // Sort leaderboard data by time
 const sortedLeaderboard = leaderboardData.sort((a, b) => {
   const timeA = convertTimeToSeconds(a.time);
   const timeB = convertTimeToSeconds(b.time);
   return timeA - timeB;
 }).slice(0, 5); // display the top 5 scores

 return (
   <View style={styles.container}>
     <View style={styles.headingContainer}>
```

```
      <Text style={styles.heading}>Leaderboards</Text>
      <View style={styles.toggleContainer}>
        <TouchableOpacity
          style={[styles.toggleButton, showNationalLeaderboard &&
styles.activeToggleButton]}
          onPress={toggleLeaderboard}
        >
          <Text style={styles.toggleButtonText}>National</Text>
        </TouchableOpacity>
        <TouchableOpacity
          style={[styles.toggleButton, !showNationalLeaderboard &&
styles.activeToggleButton]}
          onPress={toggleLeaderboard}
        >
          <Text style={styles.toggleButtonText}>Local</Text>
        </TouchableOpacity>
      </View>
    </View>
    {sortedLeaderboard.map((user, index) => (
      <View key={index} style={styles.rowContainer}>
        <View style={styles.barContainer}>
          <View
            style={[styles.bar, { width: (convertTimeToSeconds(user.time) /
convertTimeToSeconds(sortedLeaderboard[0].time)) * 220 + 20 }]}
          />
        </View>
        <Text style={styles.username}>{user.username}</Text>
        <Text style={styles.time}>{user.time}</Text>
      </View>
    ))}
    <TouchableOpacity style={styles.seeAllButton} onPress={handleSeeAll}>
      <Text style={styles.seeAllButtonText}>View All</Text>
    </TouchableOpacity>
    <Modal visible={modalVisible} animationType="slide">
      <View style={styles.modalContainer}>
        <View style={styles.modalContent}>
          <TouchableOpacity style={styles.closeModalButton} onPress={() =>
setModalVisible(false)}>
            <Ionicons name="close" size={24} color="#C41E3A" />
          </TouchableOpacity>
          <Text style={styles.modalHeading}>Leaderboard</Text>
          <ScrollView style={styles.scrollView}>
```

```
            {leaderboardData.map((user, index) => (
              <View key={index} style={styles.modalRowContainer}>
                <View style={styles.modalBarContainer}>
                  <View
                    style={[styles.modalBar, { width:
(convertTimeToSeconds(user.time) / convertTimeToSeconds(sortedLeaderboard[0].time)) *
150 }]}
                  />
                </View>
                <Text style={styles.modalUsername}>{user.username}</Text>
                <Text style={styles.modalTime}>{user.time}</Text>
              </View>
            ))}
          </ScrollView>
        </View>
      </View>
    </Modal>
  </View>
 );
};

const styles = StyleSheet.create({
 container: {
   borderWidth: 1,
   borderColor: 'gray',
   padding: 10,
   borderRadius: 5,
 },
 headingContainer: {
   flexDirection: 'row',
   justifyContent: 'space-between',
   alignItems: 'center',
   marginBottom: 10,
 },
 heading: {
   fontSize: 18,
   fontWeight: 'bold',
 },
 toggleContainer: {
   flexDirection: 'row',
 },
 toggleButton: {
```

```
  paddingHorizontal: 10,
  paddingVertical: 5,
  borderRadius: 5,
  backgroundColor: '#f2f2f2',
  marginRight: 5,
},
toggleButtonText: {
  fontSize: 14,
  fontWeight: 'bold',
  color: 'black',
},
activeToggleButton: {
  backgroundColor: '#dcdcdc',
},
rowContainer: {
  flexDirection: 'row',
  alignItems: 'center',
  marginBottom: 5,
},
username: {
  color: 'white',
  position: 'absolute',
  left: 5,
  zIndex: 1,
  paddingHorizontal: 8,
  fontWeight: 'bold',
},
barContainer: {
  flex: 1,
  height: 20,
  backgroundColor: '#f2f2f2',
  marginLeft: 10,
  borderRadius: 5,
  position: 'relative',
},
bar: {
  height: '100%',
  backgroundColor: '#C41E3A',
  borderRadius: 5,
},
time: {
  marginLeft: 10,
```

```
},
seeAllButton: {
  marginTop: 10,
  backgroundColor: '#dcdcdc',
  paddingVertical: 10,
  borderRadius: 5,
  alignItems: 'center',
},
seeAllButtonText: {
  color: 'black',
  fontSize: 16,
  fontWeight: 'bold',
},
modalContainer: {
  flex: 1,
  justifyContent: 'center',
  alignItems: 'center',
  backgroundColor: 'rgba(0, 0, 0, 0.7)',
},
modalContent: {
  backgroundColor: '#fff',
  borderRadius: 10,
  padding: 20,
  width: '80%',
},
modalHeading: {
  fontSize: 18,
  fontWeight: 'bold',
  marginBottom: 10,
},
modalRowContainer: {
  flexDirection: 'row',
  alignItems: 'center',
  marginBottom: 5,
},
modalBarContainer: {
  flex: 1,
  height: 20,
  backgroundColor: '#f2f2f2',
  marginLeft: 10,
  borderRadius: 5,
  position: 'relative',
```

```
  },
  modalBar: {
    height: '100%',
    backgroundColor: '#C41E3A',
    borderRadius: 5,
  },
  modalUsername: {
    color: 'white',
    position: 'absolute',
    left: 5,
    zIndex: 1,
    paddingHorizontal: 8,
    fontWeight: 'bold',
  },
  modalTime: {
    marginLeft: 10,
  },
  closeModalButton: {
    position: 'absolute',
    top: 10,
    right: 10,
    zIndex: 1,
  },
  closeModalButtonText: {
    color: 'black',
    fontSize: 16,
    fontWeight: 'bold',
  },
  scrollView: {
    maxHeight: '80%',
  },
});

export default LeaderboardComponent;
```

workoutGenerationComponent.js

```
import React from 'react';
import { View, Text, StyleSheet  } from 'react-native';

const WorkoutGenerationComponent = () => { // WE WOULD QUERY WORKOUT DATABASE FOR DATE OF MOST RECENT MONDAY (IF NONE EXIST, GENERATE NEW WORKOUT)
  // Sample workout description and required statistic
 const workoutDescription = `Complete the specified number of reps or distance for each exercise as fast as possible, then move on to the next exercise without resting. After completing all exercises, record your time. Rest for 2 minutes, then repeat the circuit for a total of 3 rounds.

1. Firefighter Carry
 - Equipment: Two 40-pound dumbbells
 - Distance: Carry the 40-pound dumbbells for 50
   meters.

2. Sledgehammer Strikes
 - Reps: Perform 20 strikes on each side of the tire or
   surface.

3. Stair Climbing
 - Reps: Climb a set of stairs up and down 5 times.

4. Battling Ropes
 - Reps: Perform 20 waves followed by 20 slams.

5. Dummy Drag
 - Distance: Drag a dummy or heavy object attached
   to a sled for 50 meters.

6. Sprint
 - Distance: Sprint 100 meters at maximum effort.`;

 const getMostRecentMonday = () => {
   const today = new Date();
   const dayOfWeek = today.getDay();
   const diff = today.getDate() - dayOfWeek + (dayOfWeek === 0 ? -6 : 1);
   const monday = new Date(today.setDate(diff));
   return monday.toLocaleDateString();
 };
```

```
 return (
   <View>
     <View>
       <Text style={styles.heading}>Week of {getMostRecentMonday()}</Text>
     </View>

     <View style={styles.container}>
       <Text style={styles.description}>{workoutDescription}</Text>
     </View>
   </View>
 );
};

const styles = StyleSheet.create({
 container: {
   padding: 10,
   marginBottom: 20,
 },
 heading: {
   fontSize: 18,
   fontWeight: 'bold',
   textAlign: 'center',
 },
 description: {
   marginBottom: 10,
 },
});

export default WorkoutGenerationComponent;
```

statisticsInputComponent.js

```
import React, { useState, useEffect } from 'react';
import { Ionicons } from '@expo/vector-icons';
import { View, Text, TextInput, TouchableOpacity, Modal, StyleSheet } from
'react-native';

const StatisticsInputComponent = ({ onSave, onClose }) => {
 const [statistics, setStatistics] = useState({
   round1: '',
   round2: '',
   round3: '',
 });

 const [totalTime, setTotalTime] = useState('');

 const handleSave = () => {
   onSave(totalTime);
   setStatistics({
     round1: '',
     round2: '',
     round3: '',
   });
 };

 const handleClose = () => {
   onClose();
 };

 useEffect(() => {
   // Calculate total time whenever inputs change
   const calculateTotalTime = () => {
     const round1Seconds = convertTimeToSeconds(statistics.round1);
     const round2Seconds = convertTimeToSeconds(statistics.round2);
     const round3Seconds = convertTimeToSeconds(statistics.round3);

     const totalSeconds = round1Seconds + round2Seconds + round3Seconds;
     setTotalTime(convertSecondsToTime(totalSeconds));
   };

   calculateTotalTime();
 }, [statistics]);
```

```
const convertTimeToSeconds = (timeStr) => {
  if (!timeStr) return 0;
  const [minutes, seconds] = timeStr.split(':').map(Number);
  return minutes * 60 + seconds;
};

const convertSecondsToTime = (totalSeconds) => {
  const hours = Math.floor(totalSeconds / 3600);
  const minutes = Math.floor((totalSeconds % 3600) / 60);
  const seconds = totalSeconds % 60;
  return `${hours.toString().padStart(2, '0')}:${minutes.toString().padStart(2,
'0')}:${seconds.toString().padStart(2, '0')}`;
};

return (
  <Modal visible={true} animationType="slide" transparent={true}>
    <View style={styles.modalContainer}>
      <View style={styles.container}>
        <TouchableOpacity style={styles.closeButton} onPress={handleClose}>
          <Ionicons name="close" size={24} color="#C41E3A" />
        </TouchableOpacity>
        <Text style={styles.label}>Time for Round 1 (mm:ss):</Text>
        <TextInput
          style={styles.input}
          value={statistics.round1}
          onChangeText={value => setStatistics({ ...statistics, round1: value })}
          placeholder="00:00"
          keyboardType="numeric"
          maxLength={5} // only allow 5 chars (mm:ss)
        />
        <Text style={styles.label}>Time for Round 2 (mm:ss):</Text>
        <TextInput
          style={styles.input}
          value={statistics.round2}
          onChangeText={value => setStatistics({ ...statistics, round2: value })}
          placeholder="00:00"
          keyboardType="numeric"
          maxLength={5}
        />
        <Text style={styles.label}>Time for Round 3 (mm:ss):</Text>
        <TextInput
          style={styles.input}
```

```
          value={statistics.round3}
          onChangeText={value => setStatistics({ ...statistics, round3: value })}
          placeholder="00:00"
          keyboardType="numeric"
          maxLength={5}
        />
        <Text style={styles.label}>Total Time (HH:mm:ss):</Text>
        <Text style={styles.totalTime}>{totalTime}</Text>
        <TouchableOpacity style={styles.saveButton} onPress={handleSave}>
          <Text style={styles.saveButtonText}>Save</Text>
        </TouchableOpacity>
      </View>
    </View>
  </Modal>
 );
};

const styles = StyleSheet.create({
 modalContainer: {
   flex: 1,
   justifyContent: 'center',
   alignItems: 'center',
   backgroundColor: 'rgba(0, 0, 0, 0.7)',
 },
 container: {
   width: '80%',
   backgroundColor: '#fff',
   padding: 40,
   borderRadius: 10,
 },
 label: {
   fontSize: 16,
   marginBottom: 5,
 },
 input: {
   borderWidth: 1,
   borderColor: '#ccc',
   borderRadius: 5,
   padding: 10,
   marginBottom: 10,
 },
 closeButton: {
```

```
    position: 'absolute',
    top: 10,
    right: 10,
    zIndex: 1,
  },
  closeButtonText: {
    color: '#C41E3A',
    fontSize: 16,
  },
  saveButton: {
    backgroundColor: '#C41E3A',
    borderRadius: 5,
    padding: 10,
    alignItems: 'center',
  },
  saveButtonText: {
    color: '#fff',
    fontSize: 16,
    fontWeight: 'bold',
  },
  totalTime: {
    borderWidth: 1,
    borderColor: '#ccc',
    borderRadius: 5,
    padding: 10,
    marginBottom: 10,
    fontSize: 16,
  },
});

export default StatisticsInputComponent;
```

calendarComponent.js

```
import React, { useState, useEffect } from 'react';
import { View, Text, StyleSheet, TouchableOpacity, ScrollView } from 'react-native';
import { Ionicons } from '@expo/vector-icons';
import { Calendar } from 'react-native-calendars';
import { getCurrentUser } from 'aws-amplify/auth';

const CalendarComponent = ({ onClose, localData}) => {
 const [currentMonth, setCurrentMonth] = useState(new
Date().toISOString().split('T')[0]);
 const [selectedDate, setSelectedDate] = useState(null);

 async function currentAuthenticatedUser(setUserEmailCallback) {
   try {
     const user = await getCurrentUser();
     const { signInDetails } = user;
     if (signInDetails) {
       const { loginId } = signInDetails;
       setUserEmailCallback(loginId); // Set userEmail using the provided callback
     } else {
       console.log('SignInDetails not found.');
     }
   } catch (err) {
     console.log('Error fetching current user:', err);
   }
 }

 const [userEmail, setUserEmail] = useState(null); // State for userEmail

 useEffect(() => {
   currentAuthenticatedUser(setUserEmail);
 }, []);
  const markedDates = localData
   .filter(entry => entry.username === userEmail)
   .reduce((acc, entry) => {
     acc[entry.date] = { marked: true, dotColor: '#C41E3A', textColor: '#C41E3A' };
     return acc;
   }, {});

   const workoutDetails = localData
   .filter(entry => entry.username === userEmail)
   .reduce((acc, entry) => {
```

```
    acc[entry.date] = {
      workoutCompleted: `Complete the specified number of reps or distance for each
exercise as fast as possible, then move on to the next exercise without resting. After
completing all exercises, record your time. Rest for 2 minutes, then repeat the
circuit for a total of 3 rounds.
  1. Firefighter Carry
   - Equipment: Two 40-pound dumbbells
   - Distance: Carry the 40-pound dumbbells for 50 meters.
  2. Sledgehammer Strikes
   - Reps: Perform 20 strikes on each side of the tire or surface.
  3. Stair Climbing
   - Reps: Climb a set of stairs up and down 5 times.
  4. Battling Ropes
   - Reps: Perform 20 waves followed by 20 slams.
  5. Dummy Drag
   - Distance: Drag a dummy or heavy object attached to a sled for 50 meters.
  6. Sprint
   - Distance: Sprint 100 meters at maximum effort.`,
      stats: {
        time: entry.time,
      },
    };
    return acc;
  }, {});

 const handleMonthChange = (newMonth) => {
   setCurrentMonth(newMonth.dateString ? new Date(newMonth.dateString) : new Date());
 };

 const handleDateSelect = (date) => {
   setSelectedDate(date);
 };

 return (
   <View style={styles.modalContainer}>
     <View style={styles.container}>
       <TouchableOpacity onPress={onClose} style={styles.closeButton}>
         <Ionicons name="close" size={24} color="#C41E3A" />
       </TouchableOpacity>
       <Text style={styles.title}>Workout History</Text>
       <View style={styles.calendarContainer}>
         <Calendar
```

```
          current={currentMonth}
          markedDates={{ ...markedDates, [selectedDate]: { selected: true } }}
          onMonthChange={handleMonthChange}
          onDayPress={(day) => handleDateSelect(day.dateString)}
          theme={{
            arrowColor: '#C41E3A',
            todayTextColor: '#C41E3A',
            selectedDayBackgroundColor: '#dcdcdc',
          }}
        />
      </View>
      {selectedDate && workoutDetails[selectedDate] && (
        <ScrollView style={styles.workoutContainer}>
          <Text style={styles.stats}>Your Results:
{workoutDetails[selectedDate].stats.time}</Text>
          <Text style={styles.workoutTitle}>Description:</Text>
          <Text
style={styles.workoutDetails}>{workoutDetails[selectedDate].workoutCompleted}</Text>
        </ScrollView>
      )}
    </View>
  </View>
 );
};

const styles = StyleSheet.create({
 modalContainer: {
  flex: 1,
  justifyContent: 'center',
  alignItems: 'center',
  backgroundColor: 'rgba(0, 0, 0, 0.7)',
 },
 container: {
  width: '95%',
  backgroundColor: '#fff',
  padding: 20,
  paddingTop: 30,
  borderRadius: 10,
  position: 'relative',
 },
 calendarContainer: {
  borderBottomWidth: 1,
```

```
    borderBottomColor: '#ccc',
  },
  workoutContainer: {
    maxHeight: 250, // Maximum height for scroll
    padding: 20,
    marginTop: 10,
    marginBottom:10,
  },
  title: {
    fontWeight: 'bold',
    marginBottom: 10,
    fontSize: 20,
    alignSelf: 'center',
  },
  workoutTitle: {
    fontWeight: 'bold',
    fontSize: 20,
    marginBottom: 10,
  },
  stats: {
    marginBottom: 10,
    fontSize: 20,
    fontWeight: 'bold',
    color: '#C41E3A',
  },
  closeButton: {
    position: 'absolute',
    top: 10,
    right: 10,
    zIndex: 1,
  },
  workoutDetails: {
    marginBottom:20,
  },
});

export default CalendarComponent;
```

api/schema.graphql

```
# This "input" configures a global authorization rule to enable public access to
# all models in this schema. Learn more about authorization rules here:
https://docs.amplify.aws/cli/graphql/authorization-rules
input AMPLIFY { globalAuthRule: AuthRule = { allow: public } } # FOR TESTING ONLY!

type Group @model @auth(rules: [{allow: public}]) {
 id: ID!
 groupName: String
 numMembers: Int
 Users: [User] @manyToMany(relationName: "GroupUser")
}

type Workout @model @auth(rules: [{allow: public}]) {
 id: ID!
 workoutName: String
 week: AWSDate
 workoutDescription: String
 videoLink: AWSURL
 UserWorkouts: [UserWorkout] @hasMany(indexName: "byWorkout", fields: ["id"])
}

type UserWorkout @model @auth(rules: [{allow: public}]) {
 id: ID!
 date: AWSTime
 workoutID: ID! @index(name: "byWorkout")
 userID: ID! @index(name: "byUser")
 rating: String
 review: String
}

type User @model @auth(rules: [{allow: public}]) {
 id: ID!
 firstName: String
 lastName: String
 email: String
 workoutStreak: Int
 UserWorkouts: [UserWorkout] @hasMany(indexName: "byUser", fields: ["id"])
 groups: [Group] @manyToMany(relationName: "GroupUser")
}
```

auth/parameters.json

```
{
 "identityPoolName": "testAuthIdentityPool",
 "allowUnauthenticatedIdentities": false,
 "resourceNameTruncated": "firerecfacb692",
 "userPoolName": "FireResQube",
 "autoVerifiedAttributes": [
   "email"
 ],
 "mfaConfiguration": "OFF",
 "mfaTypes": [
   "SMS Text Message"
 ],
 "smsAuthenticationMessage": "Your authentication code is {####}",
 "smsVerificationMessage": "Your verification code is {####}",
 "emailVerificationSubject": "Fire ResQube Verification Code",
 "emailVerificationMessage": "Your Fire ResQube verification code is the following:
{####}",
 "defaultPasswordPolicy": false,
 "passwordPolicyMinLength": 8,
 "passwordPolicyCharacters": [
   "Requires Lowercase",
   "Requires Numbers",
   "Requires Symbols",
   "Requires Uppercase"
 ],
 "requiredAttributes": [
   "email"
 ],
 "aliasAttributes": [],
 "userpoolClientGenerateSecret": false,
 "userpoolClientRefreshTokenValidity": 30,
 "userpoolClientWriteAttributes": [],
 "userpoolClientReadAttributes": [],
 "userpoolClientLambdaRole": "FireRecfacb692_userpoolclient_lambda_role",
 "userpoolClientSetAttributes": false,
 "sharedId": "cfacb692",
 "resourceName": "FireResQube",
 "authSelections": "identityPoolAndUserPool",
 "serviceName": "Cognito",
 "usernameAttributes": [
   "email"
```

```
  ],
  "useDefault": "manual",
  "userPoolGroups": false,
  "userPoolGroupList": [],
  "adminQueries": false,
  "thirdPartyAuth": false,
  "authProviders": [],
  "usernameCaseSensitive": false,
  "useEnabledMfas": true,
  "authRoleArn": {
    "Fn::GetAtt": [
      "AuthRole",
      "Arn"
    ]
  },
  "unauthRoleArn": {
    "Fn::GetAtt": [
      "UnauthRole",
      "Arn"
    ]
  },
  "breakCircularDependency": true,
  "dependsOn": [],
  "hostedUI": false
}
```

workoutGeneration.py

```
import openai

def generateWorkout():
   key = ""
   ai = openai.OpenAI(api_key=key)

   try:
       response = ai.completions.create(
           model="gpt-3.5-turbo-instruct",
           prompt = """Create a new workout with different exercises that is similar to
this format and can be timed:

               `Complete the specified number of reps or distance for each exercise as fast
as possible, then move on to the next exercise without resting. After completing all
exercises, record your time. Rest for 2 minutes, then repeat the circuit for a total of 3
rounds.

               1. Firefighter Carry
               - Equipment: Two 40-pound dumbbells
               - Distance: Carry the 40-pound dumbbells for 50
                   meters.

               2. Sledgehammer Strikes
               - Reps: Perform 20 strikes on each side of the tire or
                   surface.

               3. Stair Climbing
               - Reps: Climb a set of stairs up and down 5 times.

               4. Battling Ropes
               - Reps: Perform 20 waves followed by 20 slams.

               5. Dummy Drag
               - Distance: Drag a dummy or heavy object attached
                   to a sled for 50 meters.

               6. Sprint
               - Distance: Sprint 100 meters at maximum effort.'
               """,
           max_tokens=250
       )

       text = response.choices[0].text.strip()
       return text
```

```
    except Exception as e:
        return f"Error: {e}", None


if __name__ == "__main__":
    text = generateWorkout()
    print(text)
```