FitDeck Update

Daphne Williams

Shane Trentham

At this point in the project, we have not deviated much, if at all, from the original idea; I believe we had a pretty clear idea of what we were going to develop at the beginning and are working on staying true to that. The main thing so far that has technically changed from the original was within each class, the data is private but because the setters and getters cannot be less private, I needed to create a separate public field that uses the setter to set the private field and the getter to get the private field. For example, for a private field name, I created a public field Name with a set and get that takes the value given and sets it to name and the getter returns name; no actual work is done to or with Name, it just acts as a go between. Although this doesn't technically deviate from the plan itself, this was something I completely forgot about and needed to alter the class diagram because of it so I felt it significant enough to mention. I decided to go with an ASP.NET web API as the middleman between the client application and the database. In the database 4 tables were added: PredeterminedRoutines_Workouts, PredetermineWorkouts_Exercises, UserCreatedRoutines_Workouts, and UserCreatedWorkouts_Exercises. These have only 2 columns both being a foreign keys the first to the Routine/Workout id from the respective routine/workout tables and the second being to the workout/exercise id from the respective workout/exercise tables. This solves the issue in the original ERD where we were trying to add lists into a row of the database tables.

As far as the implementation of the user interface, there is still a lot to be done for it but almost everything is at a standstill until we add the API for the exercises. I have been in contact with

someone at ExRx and we have set up a time to meet on zoom to discuss the API and for him to instruct us on its use this Tuesday. In the meantime, I have completed the pages before the profile screen, where the exercises will begin being utilized, and part of the profile screen itself is done as well. In further detail, I have finished with the initial screen the user sees when using the app for the first time; this screen just displays the buttons login and signup. When the user selects login, at this point, it does nothing and has nothing but the page does exist; the database completion and calls to the database still need to be implemented for this screen to then lead to the currently-partially-created profile screen. When signup is selected, it leads the user to the screen where they enter their initial information of username, email, and password. This screen makes sure the user enters the information in each box and makes sure they know the password by reentering the information. This screen still has a little bit of work done once the database is created as it will need to verify that the information is indeed unique; I also need to implement other features that reject the password if it doesn't meet specific credentials and reject the email if it isn't in the right format. This screen leads to the complete-the-profile screen where the user is to enter their name, birthday, height, and weight. The birthdays and height are entered through a series of dropdown boxes with predetermined entries that I have implemented in order to keep the dates and height within an appropriate range. The user will press a button once finished that leads to the profile screen. This screen so far only displays the user's name, username, email address, height and weight at the top of the screen; has a small space for the graphs (once implemented); and a list of cells which will house the workouts once added. After looking at this, I think I may add a portion of the top of the profile screen for the user to add an image for themselves if they would like and have a default image there if they decide not to add one. I have also added most of the classes that are included in the class diagram (Exercise class, Workout

class, etc.). I still need to push the most recent changes but I got caught up with making the profile look decent and was distracted with that for a while; the use of Visual Studio with iOS development is a bit tricky and coding the XAML by hand is tedious although not too difficult. So far I have set up all of the tables in the database and I have completed all the stored procedures needed for the User table. I have also completed the models for the tables in the web API. I have completed the Identity which is used in formatting the user data to use the ASP.NET login and signup standards. Then, the Services has been created which is used to create a Token for the User upon signup/login.

There are still plenty of features still needed to be added once I have access to the exercises in the API's database, including the predetermine workouts and displaying the workouts themselves. I can continue working on other features such as the graphs and the implementation of the buttons that go along with editing a workout as well as the screens that will display them while I wait for said access. To do this, I would just need to use the classes already created to complete the data for the graphs and figure out how to get that imagery to display and then just create a few buttons which is simple enough, especially after downloading the NuGet package for displaying graphical data. I need to finish the rest of the stored procedures as needed for when we need data from the database. I also need to finish the repositories in the API based on these added stored procedures. Then, I need to finish the controllers to add more endpoints to the API as needed for the rest of the data.

As of this moment, for the UI, the only issue is that is that calling to an API is beyond what I know how to do, as I have never needed to do it, and I have to wait on one of the developers working with this specific exercise API to help me understand what it is and how I can use it with our app. Additionally, I've never coded anything that created a graph using its data, that

will update based on new data and reset after a period of time; I know I can do it and actually just watched a tutorial on creating charts with Xamarin forms recently so it should not be too hard to figure out. The only issue on the server side so far is a System.Aggregate exception upon building the API. I'll just need to spend the time reading over my code to see where I made a mistake and once this is cleared up the User account registration and sign in functionalities will be working. Once these issues are resolved it will more than likely be smooth sailing to the completion of the project.