**5Software Requirements and Design**

**Document**

**For**

**Group <10>**

Version 1.0

Authors:

Austin Miller

Cole McGuire

Adam Sanderson

Aaron Dunn

Mike Cernkovic

1. **Overview (5 points)** Give a general overview of the system in 1-2 paragraphs (similar to the one in the project proposal).

*This application will basically be a fitness tracking/workout mobile application. This application will help you track weight, track macros, what food you eat, workouts, when you go to the gym. Can have a create your own workout option. While also helping you track workout weights. In its current state we are nowhere close to where we need to be but we are making progress.*

*We plan on using C# and Xamarin to build out the actual working mobile application itself, while also using firebase to build a more cohesive backend to the mobile application. Currently we have firebase set up to connect a Real Time database to our current code. This will help us to store the information for users, workouts, etc.. in the future.*

2. **Functional Requirements (10 points)**List the functional requirements in sentences identified by numbers and for each requirement state if it is of high, medium, or low priority. Each functional requirement is something that the system shall do. Include all the details required such that there can be no misinterpretations of the requirements when read. Be very specific about what the system needs to do (not how, just what). You may provide a brief design rationale for any requirement which you feel requires explanation for how and/or why the requirement was derived.

*This application will be responsible for maintaining a vast amount of data related to personal fitness. With that being said, the main goal for this iteration is to get the database and flow of data set up. The functional requirements for this are:*

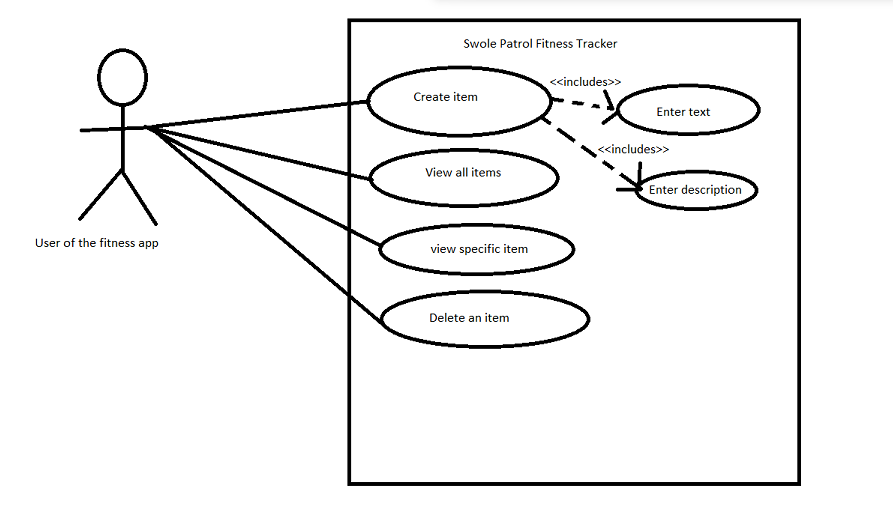
1. *Implement a skeleton code for a cross platform mobile fitness application capable of displaying users of the app with their information. A user has an id, username, password, name, birthday, gender, height, weight, and email. (high)*
2. *Set up a Firestore RealTime database and establish a connection of the database with the code. (high)*
3. *Implement the functionality to view all items in the browse tab. (medium)*
4. *Implement the functionality to view a specific item’s details. (medium)*
5. *Implement the functionality to add an item to the database. (medium)*
6. *Implement the functionality to update the information of an existing item and update the user interface with the new data. (medium)*
7. *Implement the functionality to delete an item from the database and update the user interface with the new data.*

3. **Non-functional Requirements (10 points)**List the non-functional requirements of the system (any requirement referring to a property of the system, such as security, safety, software quality, performance, reliability, etc.) You may provide a brief rationale for any requirement which you feel requires explanation as to how and/or why the requirement was derived.

1. *Safety*
2. *Asthetics*
3. *Peformance*

4. **Use Case Diagram (10 points)**This section presents the use case diagram and the textual descriptions of the use cases for the system under development. The use case diagram should contain all the use cases and relationships between them needed to describe the functionality to be developed. If you discover new use cases between two increments, update the diagram for your future increments. Textual descriptions of use cases: For the first increment, the textual descriptions for the use cases are not required. However, the textual descriptions for all use cases discovered for your system are required for the second and third iterations.

*This use case diagram is for what we have create for this iteration and it will evolve over time.*



5. **Class Diagram and/or Sequence Diagrams (15 points)**This section presents a high-level overview of the anticipated system architecture using a class

diagram and/or sequence diagrams.

If the main paradigm used in your project is Object Oriented (i.e., you have classes or

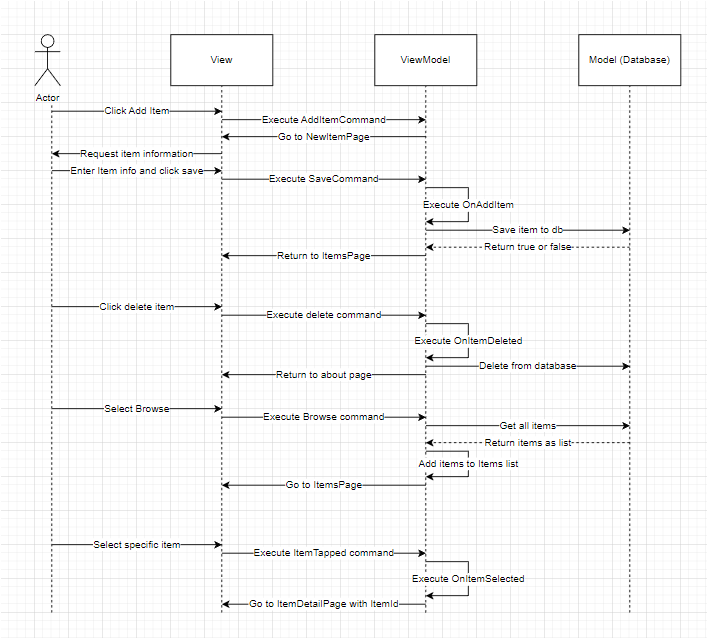
something that acts similar to classes in your system), then draw the Class Diagram of the

entire system and Sequence Diagrams for the three (3) most important use cases in your

system. If the main paradigm in your system is not Object Oriented (i.e., you do not have classes or anything similar to classes in your system) then only draw Sequence Diagrams, but for all the use cases of your system. In this case, we will use a modified version of Sequence Diagrams, where instead of objects, the lifelines will represent the functions in the system involved in the action sequence. Class Diagrams show the fundamental objects/classes that must be modeled with the system to satisfy its requirements and the relationships between them. Each class rectangle on the diagram must also include the attributes and the methods of the class (they can be refined between increments). All the relationships between classes and their multiplicity must be shown on the class diagram. A Sequence Diagram simply depicts interaction between objects (or functions - in our case - for non-OOP systems) in a sequential order, i.e. the order in which these interactions take place.

Sequence diagrams describe how and in what order the objects in a system function.

*This sequence diagram is for what we have created for this iteration and it will evolve over time.*

**

6. **Operating Environment (5 points)**Describe the environment in which the software will operate, including the hardware platform, operating system and versions, and any other software components or applications with which it must peacefully coexist.

*The environment that the system will be operating on is a mobile device, specifically a mobile phone. This is not limited to only an android or an apple device as we are using the Xamarin Platform to be able to develop on both platforms. The system should function on IOS and the Android operating system. Any operating system or system that can run .NET and Xamarin compiled code will be able to run this application.*

7. **Assumptions and Dependencies (5 points)**List any assumed factors (as opposed to known facts) that could affect the requirements stated in this document. These could include third-party or commercial components that you plan to use, issues around the development or operating environment, or constraints. The project could be affected if these assumptions are incorrect, are not shared, or change. Also identify any dependencies the project has on external factors, such as software components that you intend to reuse from another project.

*Assumed factors that could affect the requirements in this document are the following.*

1. *API’s or Connectors not compatible*
   1. *This means that something we plan on using for our app, specifically an outside source such as Firebase may not be fully compatible with our system, which will create errors.*
2. *UI Creation*
   1. *This issue is the limited availability of resources and factors that contribute to the UI system. There are currently not a lot of things in place for C# to be able to create a good looking UI platform.*
3. *Software & Hardware Components*
   1. *The usability of the code is entirely dependent upon hardware and software that the end user has to be able to run the code. Think of it as putting a nice paint job, or some good hardware, on a car with 450,000 miles. You can only do so much to make it run smoothly.*