**Software Requirements and Design Document**

**For**

**Group 2**

Version 1.0

**Authors**:

Lucas Albano

Spencer Dennis

Chris Santos

Jason Santos

Nick Watts

# Overview

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

**We are developing a mobile application for Dance Marathon here at FSU. It will contain information for upcoming events, such as the time/place/location of each, the points associated with each event that users can add to their profile, and a “check in” feature that users can use to verify they have attended the event. It will also contain profile information including name, email, gathered points, position, and organization.**

**It will also contain an “about” page that lists general information about Dance Marathon. We will also include a page for “Classy”, which is an API that Dance Marathon uses to show users’ fundraising progress. Our page will allow users to hopefully see their fundraising progress directly from the application.**

# Functional Requirements

*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.*

**Sign in: Once users are already signed up, they have the ability to sign in again as their login information will be stored in our back end database.**

**Sign up: If a user has no already created an account with the application, they are given the option to create an account with our sign up feature.**

**Sign out: If a user has already used the sign up feature to create an account, and the sign in feature to login to the application, they will have the ability to sign out of their account within the application.**

**Check in: Users are encouraged to attend events associated with Dance Marathon to earn “points” to add to their account. Each event will have a “check in” feature that allows them to confirm and verify they have attended that event. On successful check in, the points associated with the event will be added to the user’s account.**

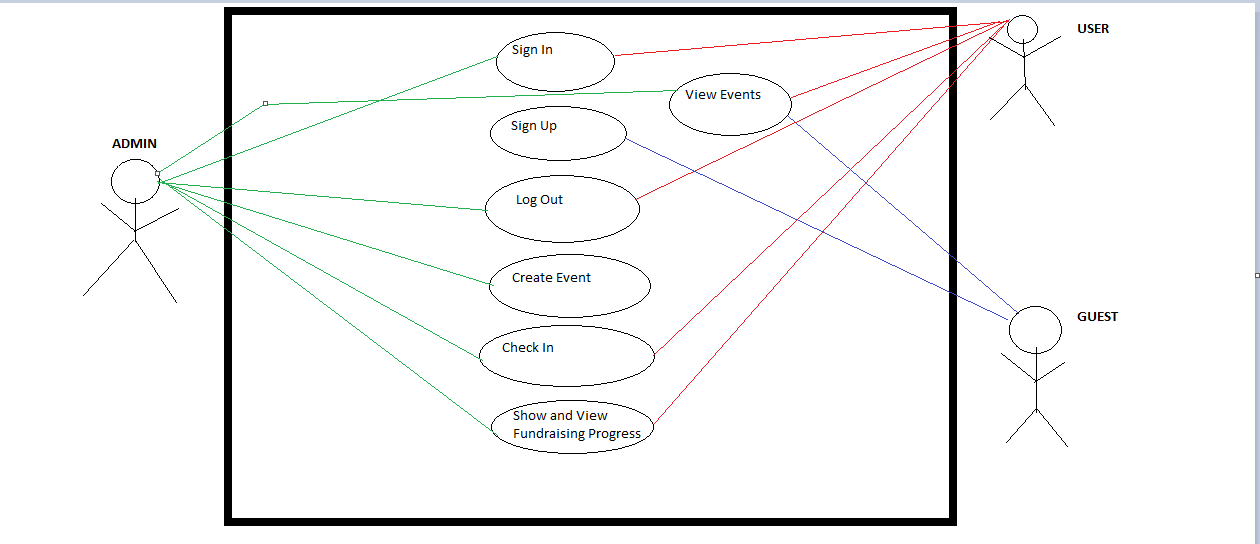
# Non-functional Requirements

*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.*

**In the case of security. We will require all API calls to be verified with a session key. Even guests who don’t have accounts will be assigned session keys that can verify they are actually a user of some sorts. The idea for this is so keep unauthorized individuals from accessing the API outside of the app with the intention of malicious wrong-doings.**

# Use Case Diagram

*This section presents the* ***use case diagram*** *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.*

**

# Class Diagram and/or Sequence Diagrams

*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.*

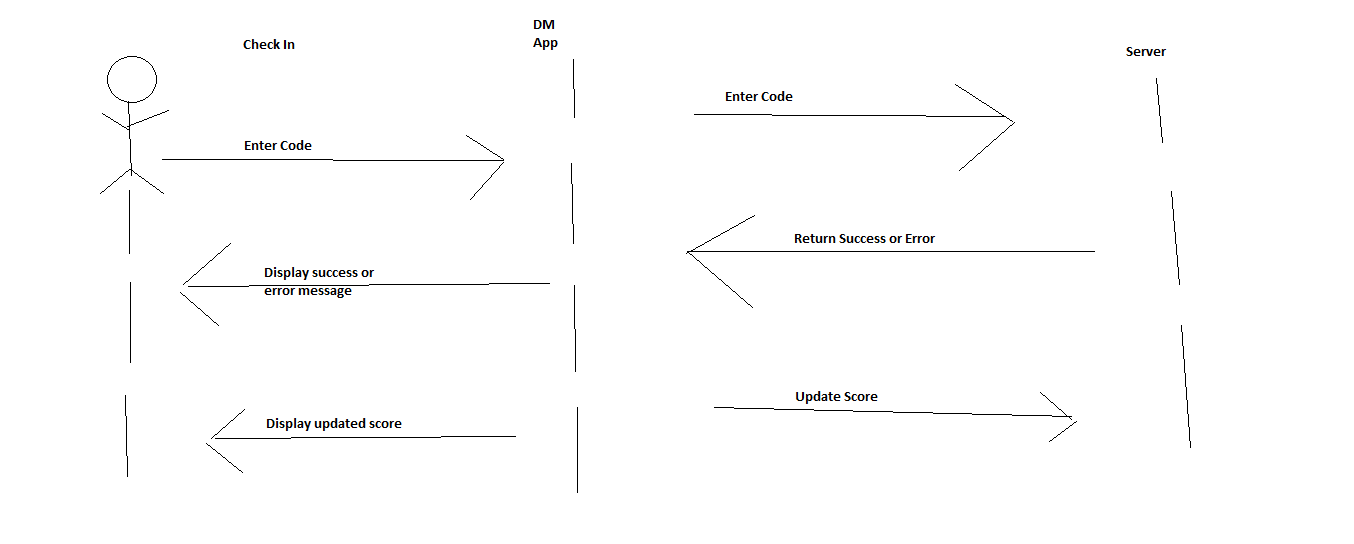
*A* ***Sequence Diagram*** *simply depicts* ***interaction******between objects*** *(or* ***functions -*** *in our case - for non-OOP systems) in a sequential order, i.e. the order in which these interactions take place. Sequence diagrams describe how and in what order the objects in a system function.*

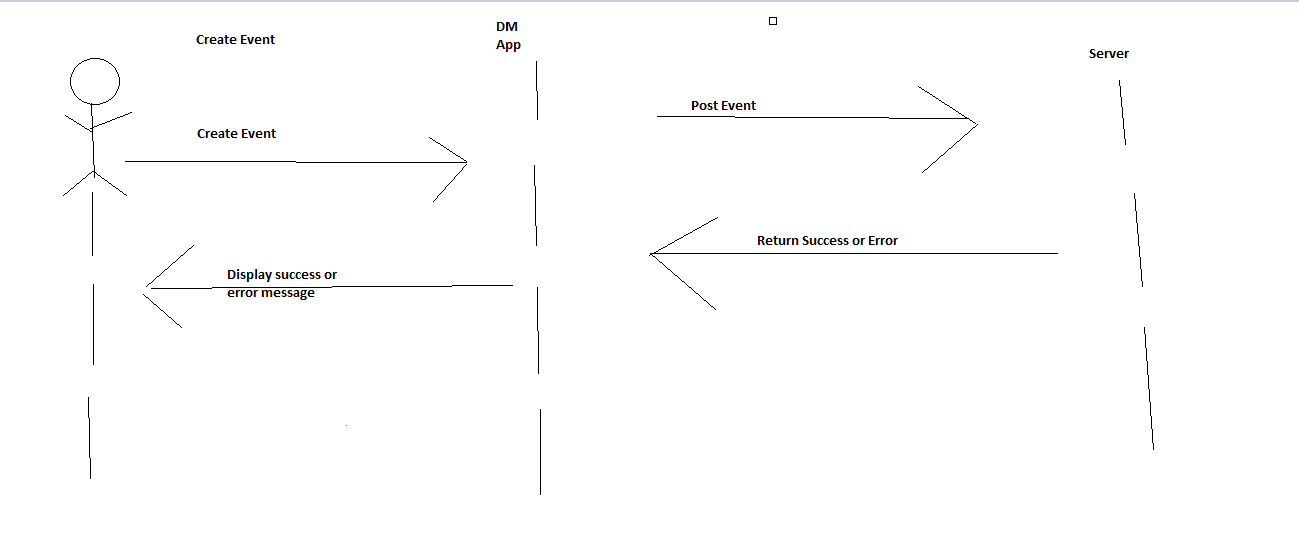
**Class Diagram:**

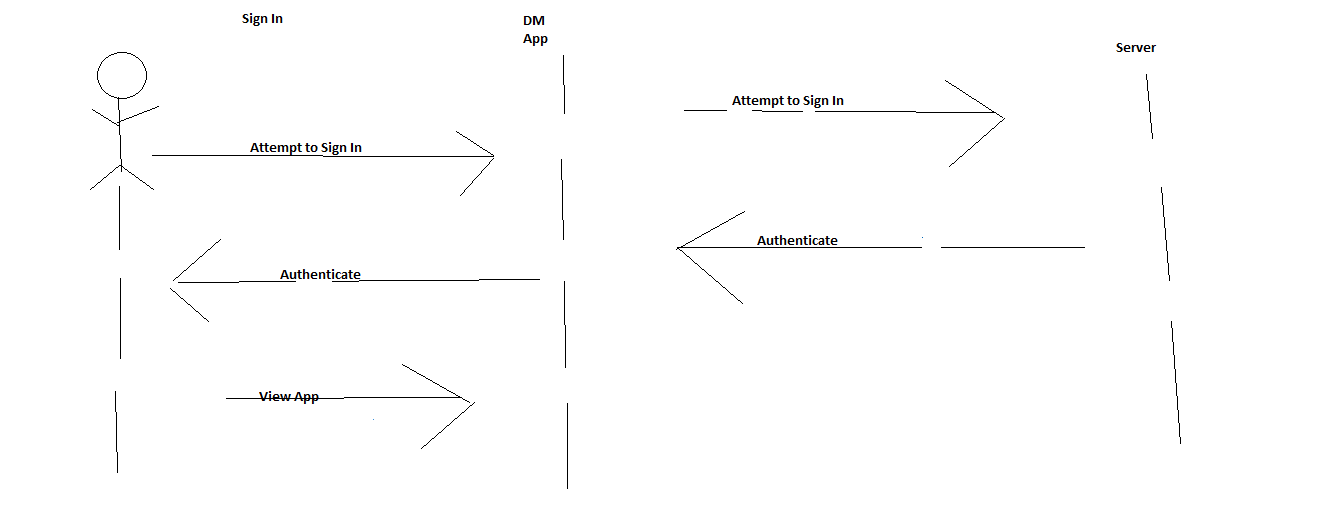
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Profile** | **Events** | **About** | **Classy** | **Login** | **Setting** |
| Email  Events  Organization  Points  Id  Signedin | Id  Name  Time  Place  Points  Code  User\_list  codeEntered | description | signedIn  Name |  |  |
| **Constructor()**  **componentDidMount()**  **showGuest()**  **decideScreen()**  **render()** | **Constructor()**  **componentDidMount()**  **lapsList()**  **openModal()**  **closeModal()**  **render()** | **showDescription()**  **render()** | **showProgress()**  **showLeaderboard()**  **render()** | **Render()**  **\_signInAsync()** | **Render()**  **changeName()**  **deleteAccount()**  **changePassword()** |

|  |  |
| --- | --- |
| Signing In | Signing Out |
|  |  |
| Render()  ComponentDidMount() | Render()  ComponentDidMount()  \_signOut() |

**Sequence Diagram:**

****





# Operating Environment

*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 operating environment is limited to mobile. It will only have a mobile application implementation. However, we decided not to limit our application to just iOS or just Android, we are using a framework (React Native) that allows our application to be cross-platform. Additionally, other software components that must coexist with our application is Classy API which will be used to show user’s fundraising progress.**

# Assumptions and Dependencies

*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.*

**We are assuming Classy API will be functional and be well documented so we are able to implement easily in our application. We are assuming our client will not change any requirements. We are also assuming async storage does not become fully deprecated (it is currently deprecated but still usable, there is a community version and a non-community version. Expo does not support the community version which is why we are currently using the deprecated non community version).**

**Our dependencies include React Native, React Navigation, Expo, Native Base, Classy API, Django, REST framework.**