**Progress Report**

**- Increment 3 -**

**Group #2**

1. **Team Members**

Lucas Albano - la14j - LAMango

Jason Santos - jds17e - jds17e

Nick Watts - nsw12b - NickWattsCS

Chris Santos - ces17g - ces17g

Spencer Dennis - sd16g - sd16g

1. **Project Title and Description**

Dance Marathon App.

We are creating an application for the dancers of Dance Marathon to allow them to learn about and check into events, track points and donation amounts, and get notifications by staff.

1. **Accomplishments and overall project status during this increment**

BACKEND

We were finally able to integrate user login and registration in the API. We are inheriting Django’s built-in user model but added other attributes such as money\_raised and organization. This allowed us to use Djangos built-in authentication and user registration systems. He has custom serializers that allow us to ask specific things for successful user registration. We ask for a username, email, password1, password2, and organization.

Django takes care of comparing the passwords and has some password validations to only accept valid and good passwords. user login was also customized to only need email and password while Django’s default login requires username and password.

We also added another model, teams. Teams hold the individual Child that each team is raising money for. Each user has a relationship with a specific team and holds that relationship in the form of a Foreign Key.

The backend also began working on exporting data from the database in the form of an excel file. This now has support in the groups and users apps at the urls ‘/func/groups/csv-export’ and ‘/func/users/csv-export’. In the future, this will be applied to the graphs app in order to create a graph using NumPy.

FRONTEND:

On the frontend, we have successfully incorporated AysncStorage to store common user information such as username, email, id, points, and organization. This is accessible throughout the app so that each screen can display the information stored in the phone. When someone signs out, all information is cleared. This information is also stored when the app is closed so that the user is always logged in unless they choose to sign out. Along with this, geofencing has been added. The way geofencing works is when the user is prompted to check in, they are asked for a code. The first layer of authentication is the code must be correct. Additionally, a user cannot check into an event unless they are signed in before hand. Once points have been added, the button changes to black to notify that the points have been added. Additionally, navigation has been made simpler by adding more back buttons on IOS to prevent getting stuck. Lastly, we have added cards to the profile screen that display the users specific details such as money fundraised and points. The front end also managed to get geofencing to locate a user using Expo’s geofencing implementation. This was tested on a member’s phone and was successfully able to determine whether he was in a given area. This feature has been further extended to only permit users in a specific location to check in at an event. Now, if a user puts in the passphrase and they’re not in the specified region, the user will not be checked into an event, and the event’s details will not be reflected on their account. However, if the user is in the specific region and also puts in the correct passphrase, the user will be successfully checked in, and that event’s details will be applied to their account.

1. **Challenges, changes in the plan and scope of the project and things that went wrong during this increment**

BACKEND:

The largest challenge we faced was talking to our client and trying to get the API credentials for the donations applications they use. The application was called Classy and it tracks donations and money raised for whatever events on organization create. They also have an API that would allow outside apps to retrieve data from. Unfortunately, we were recently told that we would not be allowed access to the Classy API because the donation money is linked to hospital records and HIPPA prevents such things. Because of this, we found it difficult to find data to use for our charts, which made development of the charts also hard.

As a result, we are now planning on entering the data manually into the database. This can be done through Django’s command prompt instructions, or it can be done through Django’s backend site. Additionally, because of this new plan to input data manually, it can become difficult to input and process large amounts of data into the database. The process could be made more efficient by means of input files such as a .csv file.

Another challenge has been the amount of expertise that is required to properly manage a Django database. This is simply because of the monolithic amount of dependencies and internals that can create conflicts if any are misused. As an example, making a migration on the wrong folder or in the wrong order of dependencies can lead to a Django specific errors where the migration dependencies are stored incorrectly, which prevents you from updating the values in the database. Just to fix this issue, on multiple occasions the backend members had to delete all previous migrations and the database and start fresh. This solution was also difficult to find, as Django’s documentation is not written in a user-friendly manner, with documents being written like blog posts with occasionally irrelevant titles instead of like a manual.

FRONTEND:

For events, it was giving errors when checking the code for each event, so we had to rewrite the code that verified users were putting in the correct code. Additionally, we’ve been having to do a lot of research on async storage which is an absolute unit. Async storage in general was hard to deal with as the name suggest “async”, it is asynchronous so having things get to where they need to be is variable. It doesn’t really run line by line, it runs in the background so other code might be running while async is running in the background to completion. Additionally, geofencing had to have some research done to figure out how to make it work within our “React Navigation”.

1. **Team Member Contribution for this increment**
2. ***Progress Report***

**Lucas**: wrote part of 3 and 5

**Chris**: wrote (part of 3), (part of 4), part of 6

**Nick**: wrote part of 3 for the frontend, part of 3 and 4 for backend, and 5 where appropriate

**Jason**: helped with 3 and 4

**Spencer**: wrote part of front end 3, and 4

1. ***the requirements and design document, including the sections they wrote or contributed to***

**Lucas**: Created the new class diagram for backend models

**Chris**: Updated Non-functional requirements for backend API

**Nick**: Updated Functional Requirements for Sign In, Sign Up, Sign Out, Excel CSV

**Jason**: Updated the frontend class diagram

**Spencer**: Assumptions and Dependencies, helped with UML diagrams

* 1. ***the implementation and testing document, including the sections they wrote or contributed to***

**Lucas:** Updated 3 for backend

**Chris:** Updated 2 for back end

**Nick:** Updated 5 for backend

**Jason:** Updated parts 5 and 4

**Spencer:** Updated parts 4 and 5

* 1. ***the source code (be detailed about which parts of the system each team member contributed to and how)***

**Lucas**: Was able to merge the user registration with the main API. created custom serializers to allow login to only need email and password for login in instead of username and password. Also made custom serializer for the registration so that it includes the user’s organization when they sign up.

**Chris**: Wrote additional code to the groups module, adding a foreign key from groups to team. Added code to groups and team modules to export excel files via the admin page, as well as a view to achieve the same thing through a separate function.

**Nick**: Removed or changed parts of the source code to get migrations to run, created the teams app, including its models, serializers, views, and adding implementation to DanceMarathon/urls.py. Created page that implemented Chris’s view to allow a user to download a csv file from a specific url page.

**Jason**: Created sign up verification, made sure sign up information connects to back end, created log in feature and verification for credentials, incorporated AsyncStorage on sign in to store important user information to their phone. Also helped to increase ease of app navigation through added back buttons and the removal of the tab system on the landing page.

**Spencer**: Created events page, UI, getting the back end information of events to connect to front end, check in feature, about page. Additionally, worked on async storage for events/connecting back end information with front end through async for events. Finally, incorporated the location features / geofencing feature for front end events.

* 1. ***the video or presentation***

**Lucas**: Talked about user login and registration, backend demonstration

**Chris**: Created slide show, and demonstrated excel export feature

https://youtu.be/DCJz6UVwRGs

**Nick**: Presented the project up to the Demo, backend demonstration

**Jason**: Demonstration of profile page, helped make slide show

**Spencer**: Demonstration of events page and code/check in verification/about page