CFood

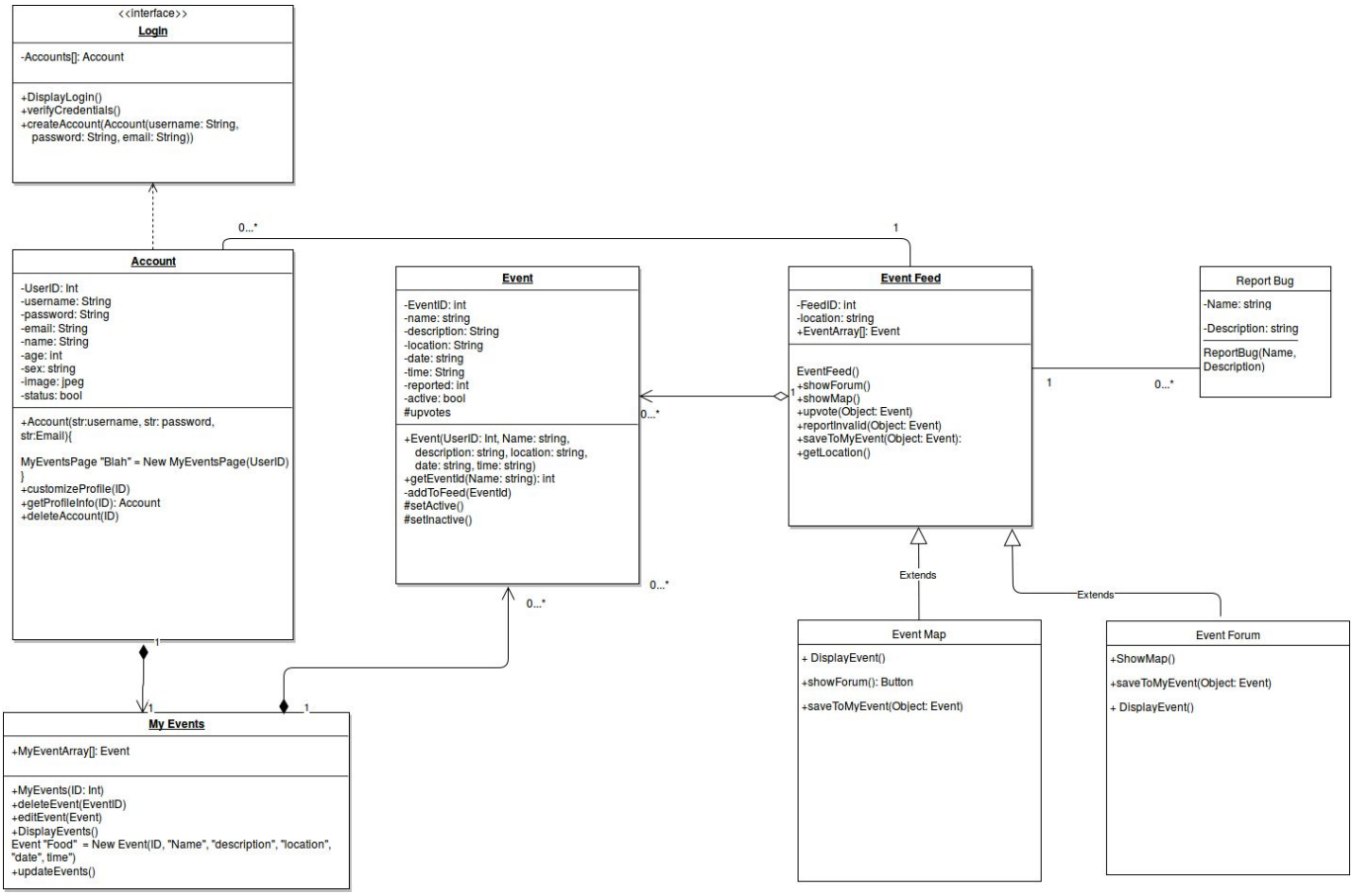
Andrew Candelaresi
Dylan Shepard
Leif Walder
Rachel Lewis
Austin Holler

1. *Features that were implemented:*
    - Login with Oauth2 using Google Authentication
    - User profile's are created at login
    - Forum displays all events currently stored in the database
    - Store/Retrieve event into SQLite Database
    - Store/Retrieve User in SQLite Database using Oauth2 token

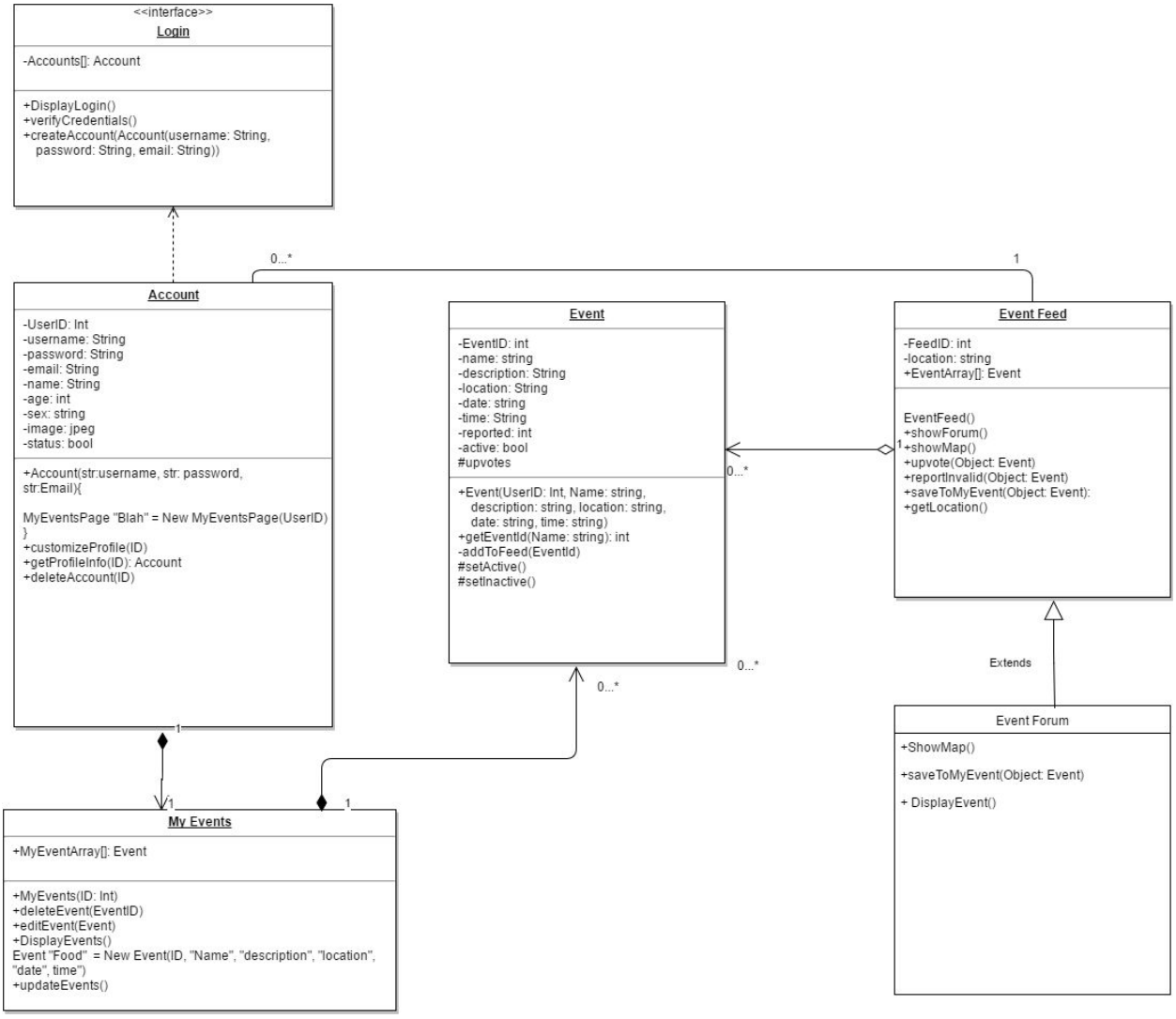
2. *Which features were not implemented from Part 2*
    - Attaching events to a map
    - Report bugs
    - Timing out created Events
    - Reporting/Removing invalid events
    - Displaying more verbose event information
    - Hosting a database on AWS, or some other platform to allow for better scalability
    - Creating Events

3. *“Show your Part 2 class diagram and your final class diagram."*

Class Diagram from Part 2



Final Class Diagram



"What changed?"
We eliminated the map class and Bug report class.

"Why?"
We realized that if no one was monitor our app as an administrator, bug reports would go unnoticed.
Linking our events to the map require more work than we could accomplish in the time frame of this class. We hope to continue to work on this app and implement this feature in the future.

"Discuss how doing the design up front helped in the development…"
Planning out our application through UML, helped us to understand all of the features that our application need. By creating requirements we were able to see what a user would want to be

able to do when they were using our app, what the system should be able to handle, and what we as administrators wanted to do. Knowing the needs of each of these roles allowed us to understand what objects, classes, and interfaces we would need to build our app. Once we actually began coding we had a clear plan of what we needed to implement what activities pages we wanted to create and how the data would flow from the user through the application. The sequence diagrams that we created help us to understand how objects would interact with each other and what kind of data structures would be needed to display our events. We saw that we would need to create user accounts and that we would need some way to authenticate user so people could log in. Also we could see from our design that we would need to have a database connected to our application so that we could store our users information and the applications events in perpetuity.

4. *"Did you make use of any design patterns in the implementation of your final prototype? If so, how? If not, where could you make use of design patterns in your system?"*

We made use of one design patterns in our application. We made use of the observer pattern when you create a button and add an "OnClickListener" it uses a callback from clicking the the button to execute the corresponding functions. For example: load a new activity page. The established listener is set to the new activity page and when the button is pushed the system implements the onclicklistener protocol.

Next, there were some design patterns we'd have used if we had time. We wanted to implement a proxy so that even if someone lost internet connection while they were viewing the forum, it would stay loaded. Finally, we wanted to use a singleton to create an app event dispatcher. We'd have one class that handles and dispatches events to all the Activities in our app. This is very similar to the React Native architecture. We used the Facade design pattern to create our events. A user navigates from the forum to the create event page, where they fill out all of the information that is necessary to create an event. They do not have to work with our constructor or any database code. They simply fill out the text boxes and click create and the background work is all executed.

5. *"What have you learned about the process of analysis and design now that you have stepped through the process to create, design and implement a system?"*

We had some problems with the Android Studio IDE, our group was not very familiar with android development so we spent quite a bit of time getting the basics of our application to work. We learned that for an easier transition from the UML design phase to the source code phase you should have a very solid background knowledge of your software framework. This way you can create more accurate diagrams and analysis. For instance we didn't know originally that we would need to create activities since we had not worked in android before. We had a basic idea from our knowledge of languages like C++, Java, of how objects interact and

what kind of relationships we would need to create, but the specifics of android studio make it necessary to make changes to what we initially imagined.
We have learned that through thoughtful analysis and planning that you can design a computer application without much coding work. Designing the classes and what their methods were going to be before actually coding allowed us to put together cohesive code when the time came. The diagrams that we used to plan our activities allowed us to know how the different classes were going to interact.