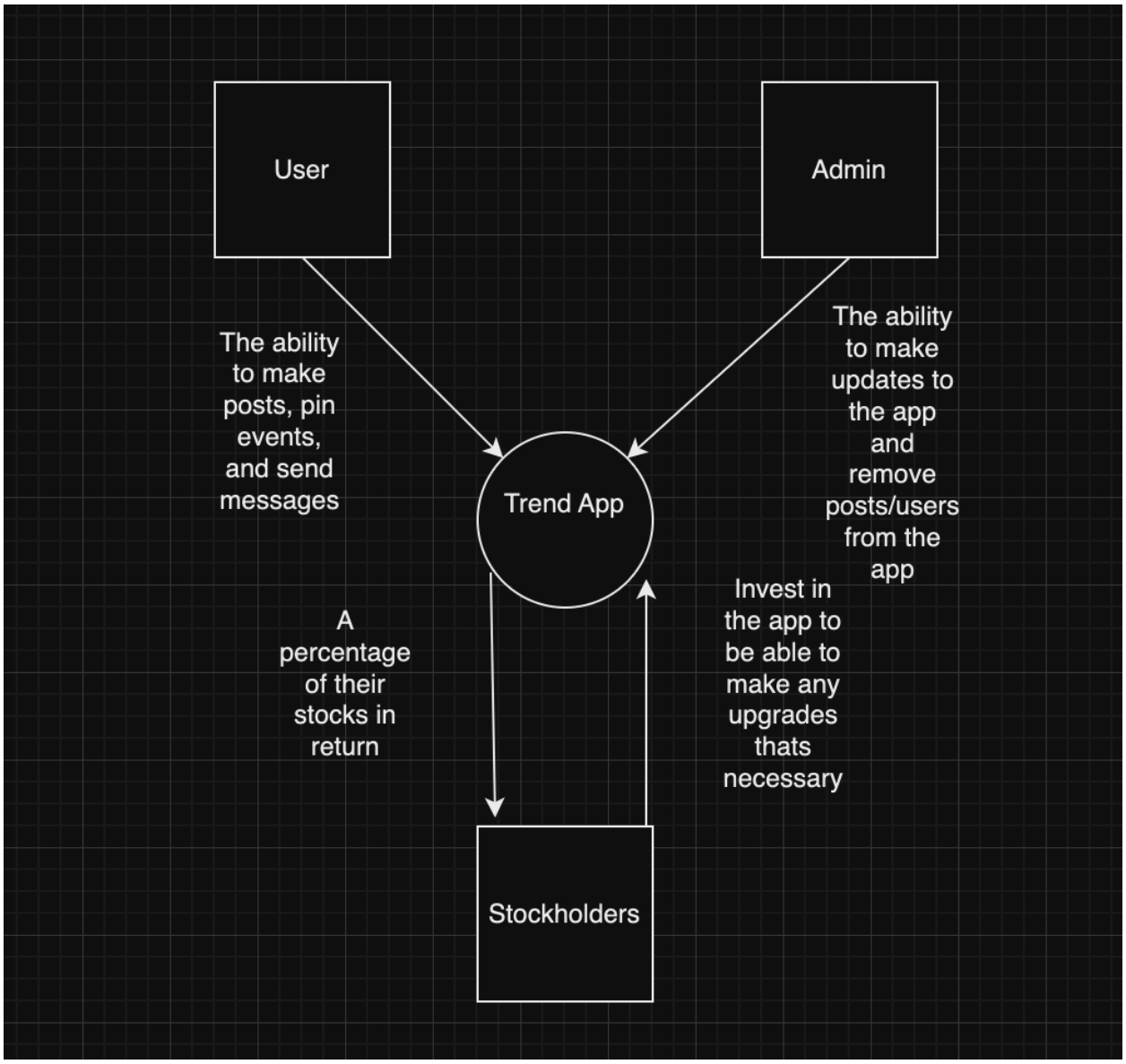
# ITCS 4155
# $SD^2$: Software Design Document

## 1. Project Overview

**Problem:** The problem that our project targets is the problem that college students are simply, not in the loop. Many college students across the country do not have a way of knowing what's going on at their campus. We plan to make a way for college students and professors to spread information about any events or news to get people more aware  of what's happening on campus.

**Stakeholders:** The stakeholders in this project are the students and staff who will be using this application. We would like to cater to their needs and take feedback from them about what would work best for them or what features would be most helpful for them in this application.

**Context Diagram:**



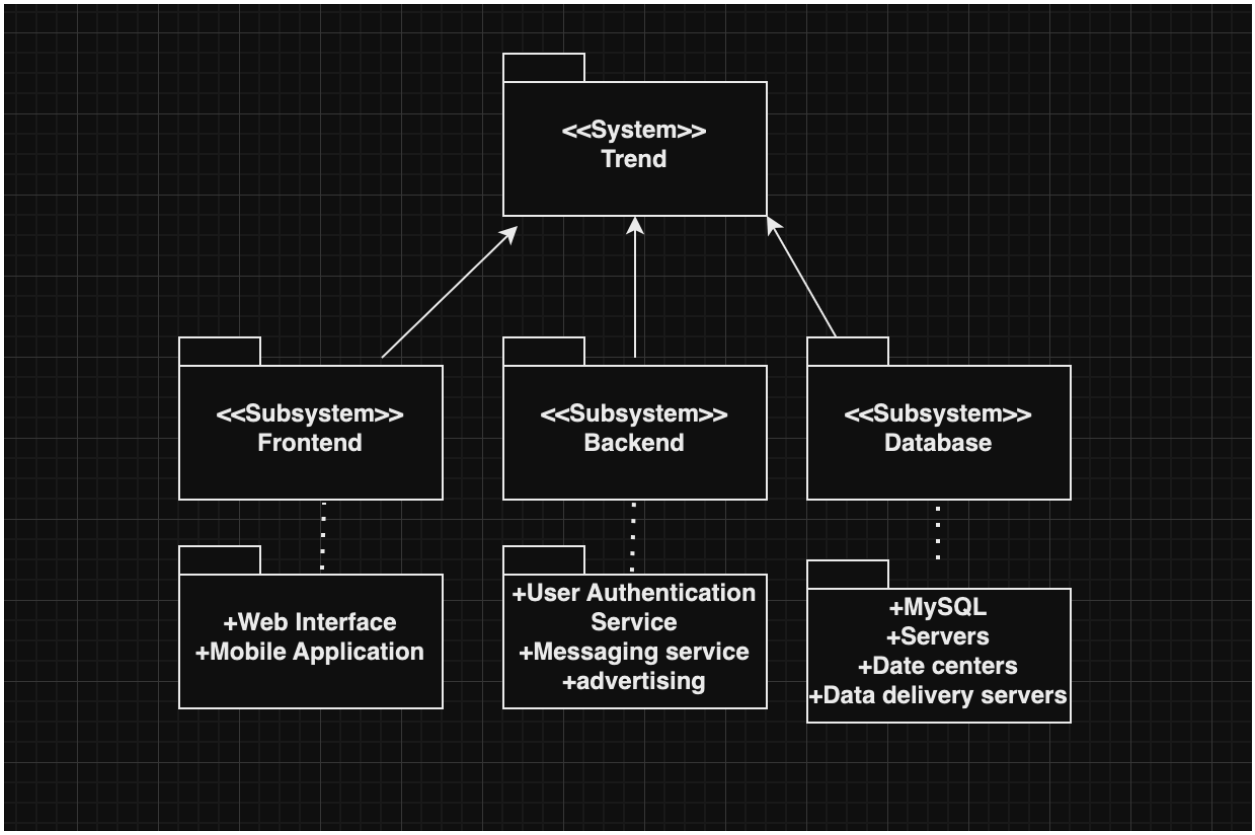

**User Stories:** The user stories are the major attributes to our application that makes it function correctly. Some of our highlighted features include creating new users for the app, being able to create and post events, adding other people as friends, being able to message people, and signing up for events that the user may go to. We are finishing up the user stories with only a few left to finish.

[Link to Github]

2. **Architectural Overview:** The application is designed to be used on a desktop or laptops as a website with the hope that potentially down the line it can be a mobile application for the users convenience. While a mobile application might be more convenient, if we want to connect this application to ninernet as another application like schedule wizard or degree works, then a web application or website would be better served. We figured that with the creation of a website instead of an application there would be less room for user error when they would post an event they didn't mean to. We thought a website would be easier for

students and people creating events to use and would be much more efficient than a mobile application. It will use HTML, CSS, and Javascript.

**2.1 Subsystem Architecture** For the subsystem architecture, it is rather straightforward in how it was approached for each level. The front end will contain the web interface and have a mobile application side to the design to allow for the ease we think students will need. The backend needs to contain the user authentication service, have the messaging service, allow for advertising, and any other functionality that may come up in the future. The needed Database infrastructure will include servers and the data centers, content delivery networks, as well as the database management systems like MySQL and other caching systems in the process. As for the Dependencies, it will require third party services, external APIs, including web/mobile browsers or other dependencies.




**2.2 Deployment Architecture:**This software will run on a single processor

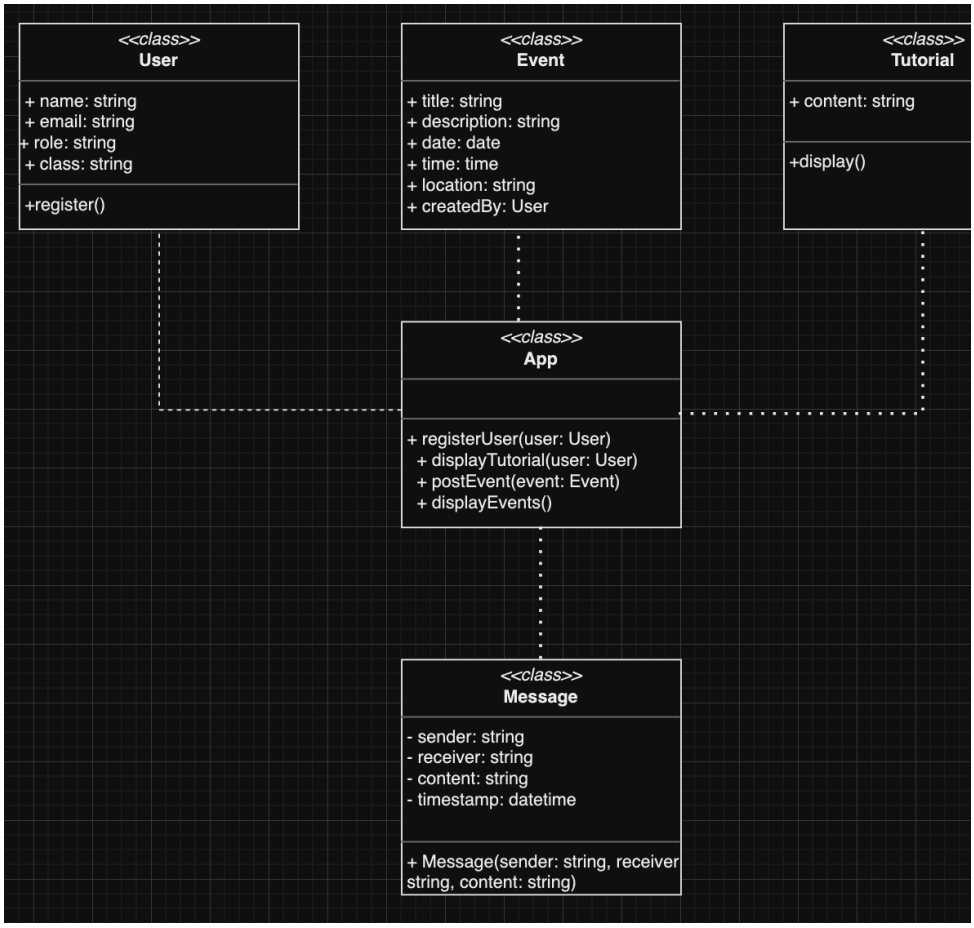
**2.3 Data Model:**Our application is going to use JSON files to store all of our user information in a library**.** This file will include user names, emails, passwords etc.

**2.4 Global Control Flow:**Our application is event-driven because every user will have the freedom to choose what they want to when they want to. There is a partial procedure involved which is the tutorial, users will go through the same required steps of setting up an account and then going through the tutorial of how to navigate through the application. With this being a tutorial we would

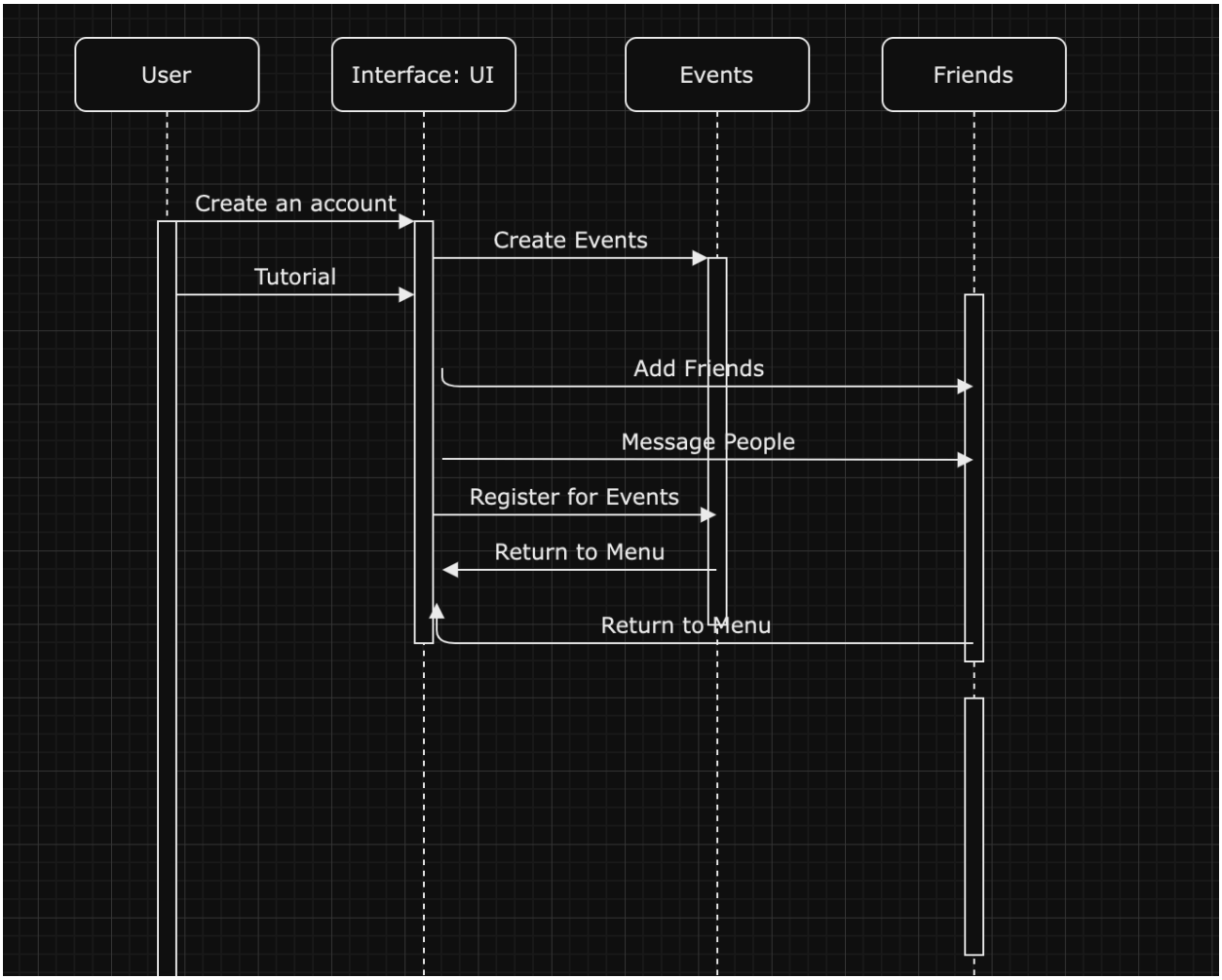
like to make sure we give the users enough time to read through the tutorials so our application is event responsive meaning that it will not continue with the procedures unless the users acknowledges that they are ready to move on. Lastly our application is controlled by a single thread.

## 3 Detailed System Design

**3.1 Static view:** Below are our main classes that will be used in our application. These classes include the methods that will be used to activate certain things like the ability to message others and post events. Our five main classes are the User class, Event, Tutorial, App, and the Message class. Starting with the main class, the App, we can run all of our other classes. Next we have the User class. This class registers new users with their basic information like name and email. After a user is created then the tutorial will follow. The tutorial will guide the user through the app so that they can learn the basics of how to use it properly. Then we have the Event class which has the methods of how to create events. The events have the date and time of when it is held and the description of the event. Lastly we have the message class which allows users to add other users and message them privately.




**3.2 Dynamic view**




When a user first signs up for Trend they will be asked to insert their UNCC or other school email/ id to register. After registration the user will see a quick tutorial and will then be able to freely use the app. From there they will be able to make a post for an event, and or scroll through posts other users have made. If a user makes a post for a later date the app will store it in the database and post it on the later date given.

**Submission:**

Submit your design documents inside your team Github repo. Moreover, **submit a PDF document in Canvas along with the link to repo; we will not grade any other document type**. Name your electronic submission as follows: **Team<number>_D2.pdf.** Submit your team assignment via Canvas. Only one team member needs to submit the document.

**Tips:**

*Provide supplemental text to explain your diagrams through captions for them!* Make sure that you have described, somewhere in the document, the responsibilities of the elements (e.g., components/modules/classes) in your architecture/class/sequence diagrams. You should describe your design decisions that led you to this design, including a discussion of any alternative designs you considered but discarded. Providing these kinds of descriptions helps in understanding your design (as such, they can have a positive impact on your grade!). *Where to Stop/When to Stop Drawing Interaction Diagrams*? Generating a sequence diagram for the most important user stories is typically a good way to start. If you find that you have a bunch of sequence diagrams that essentially repeat the same interaction, then you are not creating useful models, just redundant ones. In this case, generalize the interaction and provide supplemental text that explains how and where the generalized interaction can be applied to capture multiple user stories/use cases. **Remember, you should model only what is needed and useful as discussed in the class.**

*Apply Design Principles!* You have learned about the importance of modules with high cohesion, a design with low coupling, and the benefits of relying on abstraction versus a concrete realization (e.g., application logic communicates with a hardware abstraction layer instead of directly with devices). Make sure that you apply these principles and clearly explain how your design achieves them.

*Proofread your documents!* Everyone on your team should proofread the document before it is submitted. It is important that we be able to understand your design to evaluate it, so you must take care in communicating your design. If you are not skilled in technical writing, plan in advance so that you can ask someone to proofread it for you. The university has a writing resource center that may be helpful to you.

*What UML tools should I use to draw the diagrams?* Any design tool may be used to draw your UML diagrams such as Draw.io and Lucidchart as we discussed in the class.

Be aware that while drawing programs may provide template tools for creating UML diagrams, those templates may not meet the diagramming guidelines we discussed in class. For instance, some versions of Microsoft Visio provided the wrong arrow for an “extends” relationship in the past for use cases. You should check to make sure your diagram meets the standards discussed in class and make manual edits in the drawing program if necessary to adhere to those standards.