17-356/17-766: Software Engineering for Startups
Homework 3: Sprint 1: Prototypes

100 points

Due: February 13th, Midnight (two weeks)
This homework is to be done with your group.

In this homework, you will work conduct your first sprint with your team towards your MVP. The core goal in this assignment is to *prototype* the core elements of your product.

References: http://faculty.washington.edu/ajko/books/design-methods/

The learning goals of this assignment are that, once completing it, you will be able to:

- Develop data models for representing data and relationships.
- Use an iterative design process to develop rapid user interface prototypes.

Your task for the next two weeks is to prepare a number of different forms of prototypes:

Object Data Model

An Object Data Model (ODM) is a way of specifying the format of data your application will interact with. There are many different ways of developing an ODM; ranging from building object-oriented class structures to contain data (like you may have learned in 15-214), creating a database schema (like in SQL) or even as a formal UML diagram. Creating an ODM is crucial any time different components of an application are to interact (for example, a database and server-side application, or a MVC frontend and a server-side application).

In this assignment, we will be focused on creating a document-based object data model. In Javascript, objects can be represented in a format called JSON. For instance, a User might be represented as follows:

```
{
        "username" : "ricksanchez",
        "password" : "735ed12abec12bdcb42b",
        "full_name" : "Rick Sanchez",
        "followers" : [
                "msmith",
                "bperson",
                "kmichael",
                "mpbh"
        ]
}
```

While this is a good example of a user, it is not entirely explanatory of the datatypes and relationships expressed in the data. Likewise, we might write out a document that looks something like this to help guide development:

```
{
        “username” : String, // UNIQUE, URL-SAFE
        “password” : String, // Salted, Hashed Password.
        “full_name” : String,
        “twitter_id” : ?String, // Optional, hence the question mark
        “followers” : [String] // Reference to other USER objects.
}
```

Much like developing in an object-oriented fashion within a single language, developing an object model like this allows team members to coordinate development even across multiple projects (say, frontend and backend). As such, your team should:

- As a group, decide on a list of relevant objects (Drones, Users, etc).
- For each of these objects, create an example object (like the first example above). Note that you may need to have multiple example objects if an object can be in more than one state.
- For each of these objects, create a corresponding Object Data Model (like above).
- Put all of the above examples and objects into your repository wiki.

Object Validation

In an ideal world, all actors (developers on your project, users of your API, etc) would respect your object data model. However, this isn't always going to be the case; developers can make mistakes, API users can mis-use your API, and hackers can attempt to abuse your object model for nefarious purposes.

To combat this, your application should programmatically validate objects, as defined in your ODM. The way you should implement this depends on your choice of backend framework (from Homework 2):

LOOPBACK

Loopback includes built-in functionality for defining data models:

- Follow the loopback tutorial for creating object models.
- Write some simple mocha tests for Loopback models.

EXPRESS.JS

Unlike Loopback, Express.JS does not include built-in functionality for defining data models. Instead, you should install **mongoose,** a package for validating JSON objects for the **mongodb** database:

- Install and save the **mongoose** package to your project repository.
- Create **Schema** objects for all of the objects in your ODM. Follow this guide.
- Write some simple unit tests for your schema validation. Follow **part one only** of this guide.

Paper Prototype
To help you synthesize all your team's ideas into a single design, you will start by developing a series of paper prototypes. Each individual team member is responsible for generating a total of 6 different paper prototype sketches. Each person should create 3 sketches of the UI for the customer page, and 3 sketches of the UI for the employees. The UIs do not need to be "good", but they should be all different from each other. After generating these sketching individually, your team should meet to synthesize your design into a single design that incorporates the best of all the designs.

To turn these in, each team member should scan or take a picture of their sketches, and then upload them to a wiki. You should turn in a link to the location in your your github wiki.

Frontend Prototype

As a startup with limited funding, you should be focused on building a presentable prototype as quickly as possible. However, as you don't want to do a lot of re-work, you decide to develop your prototype using an MVC framework, a tool which makes it very easy to create dynamic client-side applications. To keep things simple, all groups will be using **vue.js** as our MVC framework. You can familiarize yourself with **vue.js** using the following resources:

- https://vuejs.org/v2/guide/
- https://scotch.io/tutorials/build-a-to-do-app-with-vue-js-2

Using **vue.js,** your team should develop a prototype:

- Develop a **view-only** prototype for the 1-3 largest user stories in your project. What you do not accomplish in this sprint, you will need to accomplish later- so budget your time appropriately. This should be a prototype, so while it should demonstrate how a user will interact with the system, it does not need to be "pixel perfect". Because it is view only, you should be able to navigate across pages, but you do not need to implement any part of the back end (data processing, validation, generation) at this point.

- Verify this prototype correctly deploys to Azure.

Deliverables:
The deliverables for this assignment are:
- One link per team to the teams Object Data Model in their group repository wiki
- A link to the paper prototype page of your teams repo. Each team member should upload their paper prototypes there, they can be organized on different pages, but they should all link from a single page.
- An implementation of the Object Data Model in either Loopback or Express.js, committed to your team's repository
- An HTML and javascript front end prototype implemented in vue.js, committed to your teams repository.