Attendance Tracker

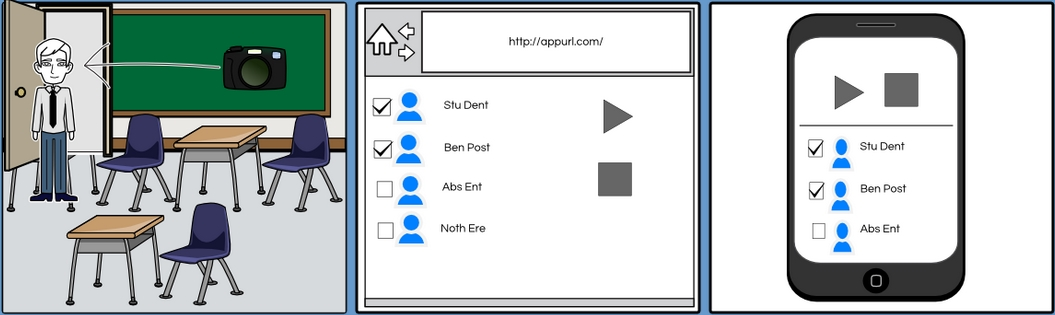
Jake Correa – 11 November 2014

# Overview

The Attendance Tracker promises to automate the task that teachers must perform at the beginning of their classes whereby they monotonously record which students show up to class that day. With the proposed system, a teacher could track attendance simply by loading the class roster in the web interface at the beginning of the semester and placing an Attendance Tracker device at the entrance to the classroom which records faces as they walk in to class and checks the faces against the roster, marking students attendance as they come in. Students can also connect to the Attendance Tracker website to ensure they are marked as having attended on a particular day and ensure the device recorded their entrance. The web interface will also have an admin panel (with login credentials) for editing the roster, and writing in custom exceptions for attendances if need be.

# Storyboard

In slide 1, the device is capturing the student’s face as they walk in and sends this data to the server, which marks the student as attended if they are recognized as being on the roster.



# API Usage

The application will make use of the SkyBiometriy (<http://www.skybiometry.com>) API which will allow me to easily make requests regarding facial detection and recognition of entrants. The API provides two methods, one for Facial Detection and one for Facial Recognition which will both be used to help determine when to check if a person is entering the room and to look up if the person is a student on the roster respectively.

The project will also utilize a Tessel (+ camera attachment) that will serve as the device which sits at the room entrance and scans faces as people enter the room. All scanned photos will be sent directly to the application server, which will handle the facial recognition/detection so that the device will have to do as little processing as possible.

# Routing Structure

The routing structure of the application server will be as follows:

|  |  |  |
| --- | --- | --- |
| Path | Method | Controller#Action |
| / | GET | home#index |
| /admin | GET | home#admin |
| /login | GET/POST | home#login |
| /users/:id | GET | users#show |
| /users/:id | POST | users#update |
| /users/:id | PUT | users#create |
| /users/:id | DELETE | users#destroy |
| /clear\_attendance | POST | users#clear\_attendance |

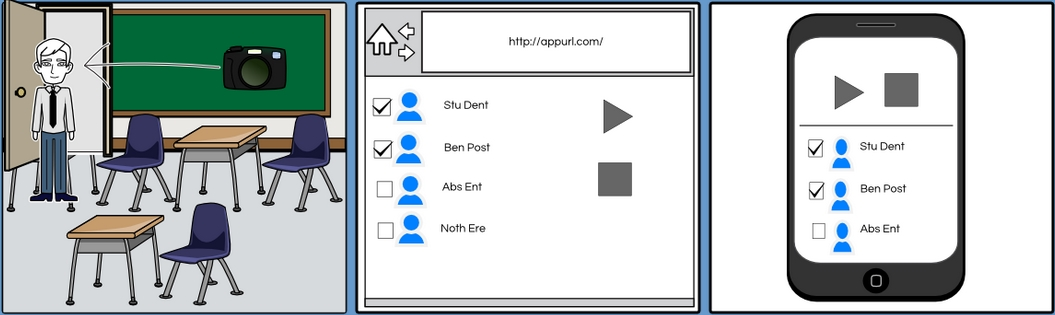
# Model Structure

The application logic is rather simple, containing only one model, Users, outlined below.

**Users**

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| Id (unique) | Integer | A unique ID to index Users on |
| First Name | String | The user’s first name |
| Last Name | String | The user’s last name |
| Photo | Image | A picture of the user |
| Attended | Boolean | Whether the user has attended |
| Role | String | Either “student” or “admin” |
| Date Created | Datetime |  |
| Date Updated | Datetime |  |

# Persistent Storage



All Users will be stored in a NoSQL database (MongoDB). I’m using MongoDB since it plays nicely with Node.js. If time permits, I will add a table to for Classes so it can be reused across multiple classes and keep previous class attendance info.