

# Software Design Specification

# MyConcordiald

---

## **Team Ewok**

Michal Wozniak

Francis Cote-Tremblay

Ahmed Dorias

Harrison Ianatchkov

Sebastian Rafique Proctor-Shah

Simon Moniere Abes

6th March, 2017



## Table of contents

<b>Introduction</b>	<b>2</b>
Purpose	2
System overview	2
<b>Architecture</b>	<b>3</b>
<b>Database Schema</b>	<b>5</b>
<b>Design Decisions</b>	<b>6</b>
<b>Security - OAUTH2</b>	<b>8</b>



## Introduction

### Purpose

This project is meant to virtualize the student ID system at Concordia. A mobile application will be used to allow students to readily have their student IDs with them. It will also allow them to take their own pictures to be used as their picture ID. The picture goes through an initial layer of validation by the mobile application and then validated in the backend by a person on the web application. Other features include an event creation and attendee tracking system as well as a marshalling card information page available to graduating students through the mobile application.

### System overview

The system is comprised of a mobile application, web client, backend api server and database.

#### **Mobile application**

The mobile client is a cordova ionic application that functions on both android and ios devices. The app serves as a virtual id card and whereby users authenticate through google OAuth. The app communicates through a REST api with the backend server and contains other features such as the ability to upload photos, view and scan for events and view marshalling card information.

#### **Web Client**

The web client is built using Angular JS and uses google OAuth for authentication. The purpose of the web client is for back staff to validate incoming photos uploaded from the mobile application. Like mobile the web client communicates through a REST api to the backend server. Other features available on the web client are the ability to create events, add attendees and modify the picture update period.

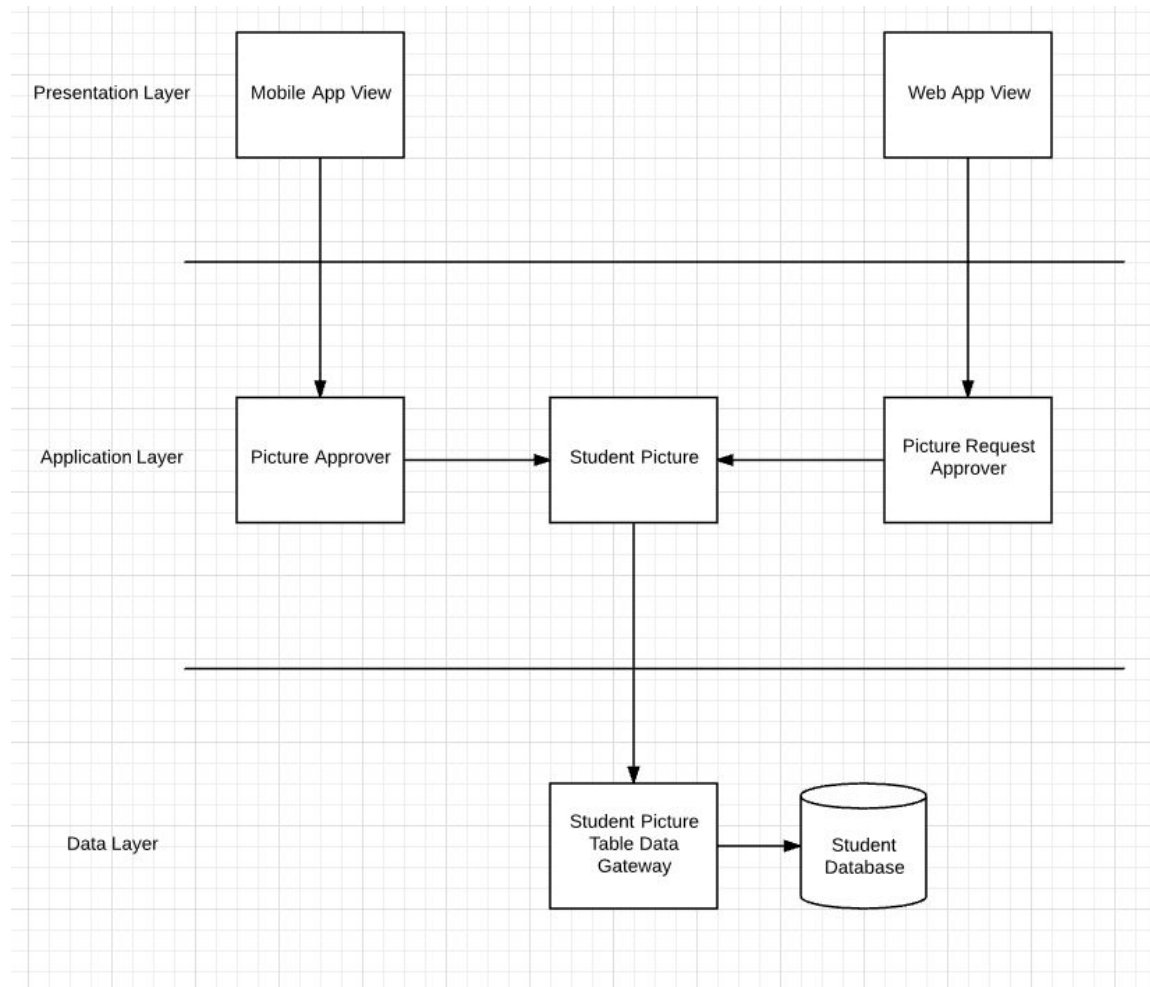
#### **Backend Server**

Our backend server is built using ASP.NET CORE and is the REST api endpoint for communication from web and mobile clients. Here different controllers handle different url paths for post, get, delete and put http requests. The backend communicates using the Entity framework with the database to resolve these requests.

## Database

We are using the relational database from Oracle to store all relevant information for web and mobile clients. For more details see the database schema section.

## Architecture



### Presentation Layer

The presentation layer consists of both the mobile app view and web client view. Both are single paged applications which leverage Angular Js for the web and ionic for mobile. Using both of these frameworks results in a thin server architectures where the majority of application logic is moved client side and the server is primarily used for data access through a REST api.

The web application is a single page application built in AngularJS. It's built using the Model View Controller (MVC) design pattern. The project is an index.html page that dynamically injects html partials as the user interacts with the client. These Views render the content from its associated angular Controller. This is done through data binding.

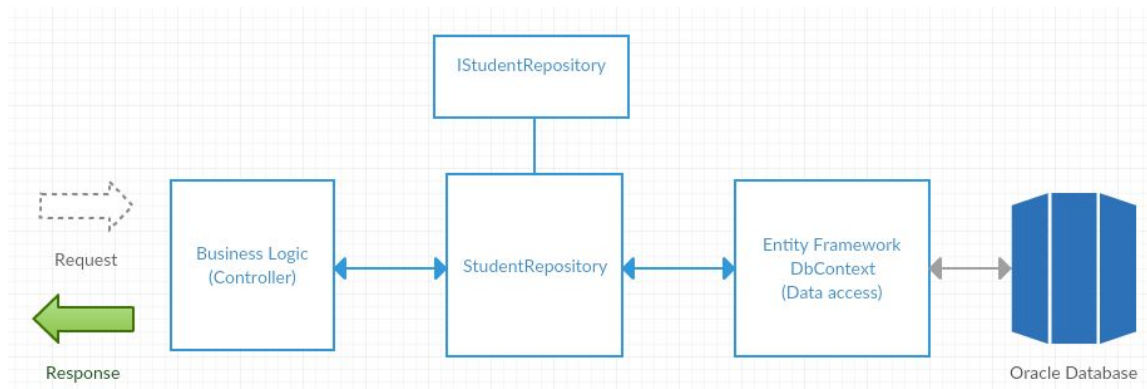
### **Application Layer**

The application layer consists of controllers and model classes using the asp.net framework. Different controllers are used to handle different api paths which resolve requests from the mobile and web clients. Controllers are group based on function such as ones for student and event activities or authentication and authorization. The model classes are used for returning and modifying information from different queries.

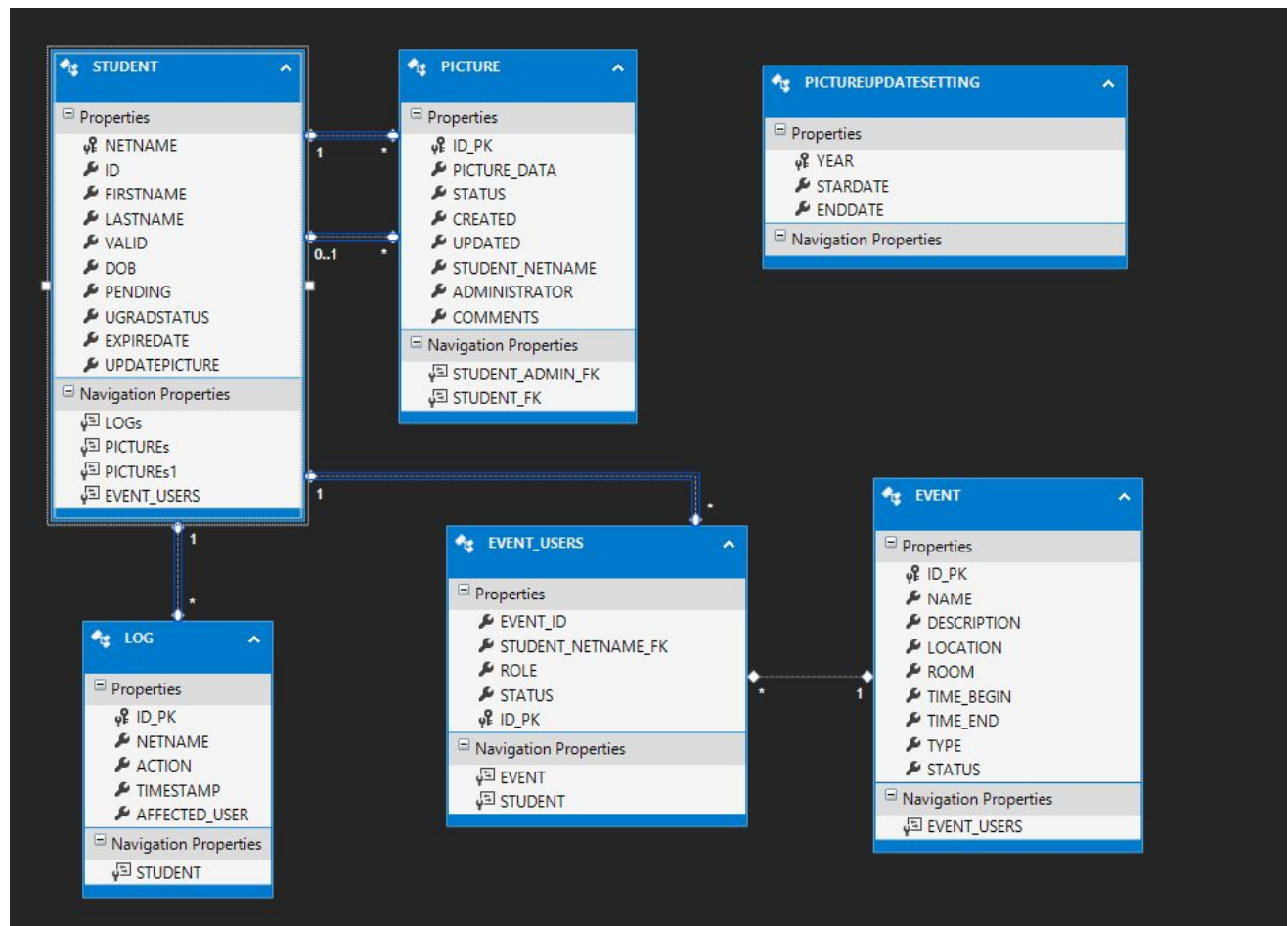
In the web application, the Controllers are the javascript controllers that are programmed leveraging AngularJs. The standard as suggested by [John Papa](#) is to have a one to one relationship between each html partial and angular controller. Within the controller, external libraries are injected whenever they are required. For example, angular-material's `<mdToast>` is injected in the `toastedHttpService.controller.js` file. The controllers handle any logic that has to do with manipulating data received from or data to be sent to the database. It acts as a bridge between the View and the Model. consists of the html partials that render the content from the associated angular controller. This is done through data binding.

### **Data Layer**

The repository pattern is used to create an abstraction between the application layer and the data access layer. Repository objects here perform CRUD methods to communicate with the database context and return domain objects to the application layer.



## Database Schema



The database contains the following 6 tables :

### **Student**

Table which is responsible for holding student identification information which is primarily used for display on the mobile virtual id page.

### **Picture**

Table which contains student's id photos as well as a history of all photos sent for approval or to be updated. Photos are used in the web client for validation as well as on the mobile virtual id page.

### **PictureUpdateSetting**

Table which contains the valid start and end dates for allowing updates of the id picture on the mobile client. These settings are modifiable by an administrator on the web client.

### **Log**

Table which records activities performed via the web or mobile clients. Activities include send/update picture (mobile) and approve/disapprove picture, change update period (web client).

### **Event Users**

Junction table which maps student and event tables together. This table is responsible for recording which students attend which event as well as their role in the event (attendee, scanner, moderator, creator). Roles which are not attendee have scanning permissions on the mobile client for the event. In case of closed events this table records who is eligible to attend.

### **Event**

Table which is responsible for holding event information and is primarily used for display on the mobile event details page.

In the web application, the aforementioned “data” that is manipulated is referred to as the Model. Again, it simply consists of the raw data you receive from the database.

## Design Decisions

### Database

- We were required to use oracle database to be able to easily integrate with concordia’s current architecture.
- Database First
  - Entity framework will create entities for you and after each modification, it will generate update these entities.
  - We can do manual changes to the database. We can then update the models directly from the database.

## ASP.NET core

- Framework is open source.
- It is cross-platform : Runs on Windows, macOS and Linux.
- Currently, we must target the full 4.6 .Net framework because of the Oracle driver which wasn't update yet.
- WebApi is now part of the MVC.
- Data access using Entity Framework as ORM.


## Mobile

- The main requirement for the mobile application was that it needed to be made to work for both iOS and Android phone.
- PhoneGap/Cordova was recommended due to the Concordia team's knowledge of Javascript.
- Ionic is a framework built on top of Cordova, following the AngularJs syntax, which uses the MVC design pattern in order to separate the logic layer with the presentation layer.
- Beside the strength of Angular, Ionic is also very powerful when it comes to encapsulating Cordova plugin in order to make them fast and easy to use.
- Another of Ionic's strength is that it adapts the style of the application to match the native standard of each platform.
- The barcode generation is done using the [JsBarcode](#) library. Since it can be adapted to the users need, which in this case is generating code 39 barcode.

## Web Application

- MVC Model View Controller Pattern leveraging Angular Js
- John Papa angular style guide.
- The Review process to approve a student's picture was decided upon by considering that a student would have to present him or herself to birk's and confirm their identity before being able to obtain their printed student id. With this in mind, it made most sense to access each student's picture requests through a search bar rather than a queue.
- The dates are set in the UTC standard format.
- We're using the SCSS styling compiler to develop css. The purpose of this is to make it easy to build complex css with dynamic variables. It also increases cohesion in the styling. Our styling structure is to have base styles that we intend on using throughout the various css





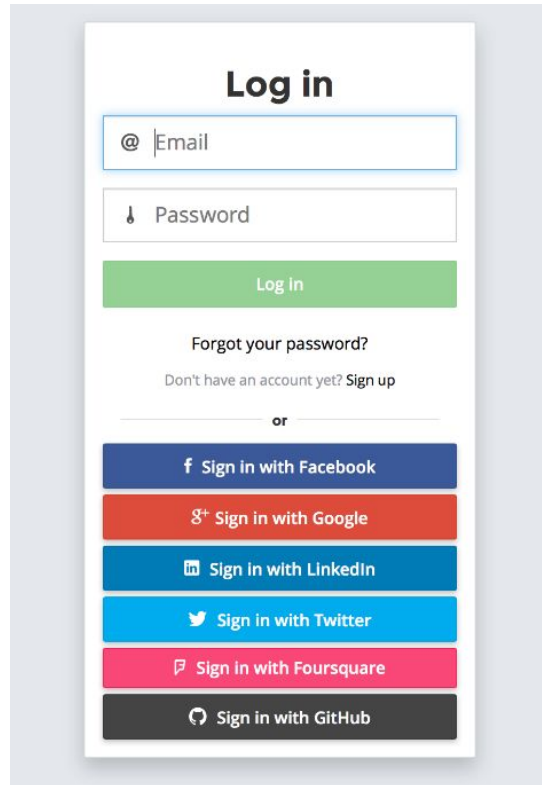
components such as margins, the components styling that modifies specific html components, and finally the views styling that only contain the imports of the various components or bases required by the specific html view.

- Currently we have one angular-modal being used in the review page, but we are moving towards using angular-material md-dialogs because we decided that they have better styling. An example of md-dialogs are in the events.controller.js, attendeeDialog.controller.js, and attendeeDialog.html. With this in mind, it is to be noted that md-materials is the preferred library to be used for html components such as select, or button, although we have our own custom 3d design of buttons.
- We are using fullpage.js to handle the scrolling up and down in the events and admin page sections, and the fullpage.js sliders to handle the sliding between the event and its attendees page. It is to be noted that we ran into significant issues with getting fullpage.js to work with angular, which is why we had to develop our own fullpage directive. Another thing to note that by default, fullpage sliders add side arrows to allow scrolling in between the slides. These were removed by overriding the display of the element to none.

## Security - OAUTH2

We have implemented oauth2 in our system where google is our oauth2 provider. Implementing this standard was complicated during development since we were required to set some specific urls and always use HTTPS.

Oauth is an open standard for authorization, user can authorize websites or application to access their information on other website without given them the password directly.



Using oauth2 we can login inside our custom application using different kind of third party applications.

### **Concordia Use Case**

It should be better to use an username with a password in our current case because we would be using an application inside concordia's network using concordia's information. It would be like if you would sign into your facebook account through oauth2 by contacting facebook itself. The whole point of oauth is providing a client a secure delegated access to server resources on behalf of a resource owner.

Oauth2 would be better used if we would want for example to sign in to outlook using our concordia's information. Outlook would receive our firstname, lastname, student status from concordia. Multiple websites that offer better prices for student could use this to automatically approve our account.