PhoneAsId Release 3 Summary

Team members

Name and Student id	GitHub id	Number of story points that member was an **author** on.
Michał Wozniak 21941097	mv740	8(#72) + 8(#92) + 8(#26) + 5(#96) + 5(#97) + 8(#81) **= 42**
Francis Côté-Tremblay 26615287	francisct	3(#54) + 8(#56) + 13(#57) + 8(#67) + 13(#68) + 8(#69) + 3(#70) + 2(#71) + 13(#76) + 3(#79) + 13(#89) + 8(#93) + 5(#96) + 5(#97) + 13(#101) + 5(#123) + 2(#117) = **125**
Ahmed Dorias 26649874	ConfusedGiant	20(#73) + 5(#74) + 8(#85) + 8(#87) + 8(#88) + 8(#108) = **57**
Harrison Ianatchkov 26607403	zzharryzz	13(#55) + 3(#66) + 8(#95) + 5(#75) + 3(#102) + 3(#121) = **35**
Simon Monière Abes 26648568	simonma1	8(#49) + 8(#92) + 13(#101) = **29**
Sebastian Rafique Proctor-Shah 29649727	EXPSPACE	8(#72) + 5(#111) + 3(#111) = **16**

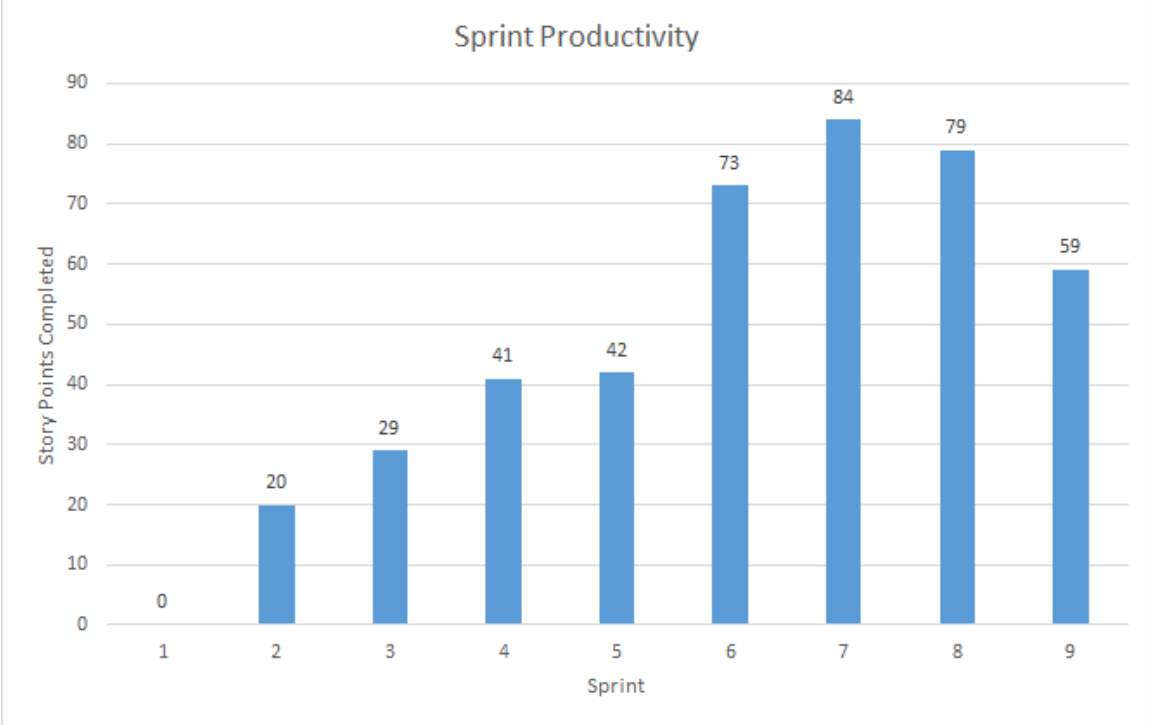
Stakeholders

Name	Github id
Jean Francois Bourgault	jfbourgault
Timothy Ni	timni

Project summary

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. It is then validated in the backend by a person on a web application. Some future features may include geo-detection with the use of iBeacons, purchasing printer credits, signing up and wirelessly authenticating membership to Concordia's LeGym, as well as displaying shuttle bus schedules.

Velocity



Our **velocity** is 424 points over 9 iterations = **47.11 user story points / iteration**

Total: 25 stories, 205 points over 13 weeks*
Iteration 1 (0 stories, 0 points)
Iteration 2 (4 stories, 20 points)
Iteration 3, Release 1 (3 stories, 29 points)
*We started our first sprint 3 days before it was due because we were solidifying our project with IITS.

Iteration 4 (5 stories, 41 points)

Iteration 5 (6 stories, 42 points)
Iteration 6, Release 2 (8 stories, 73 points)

Total: 12 stories, 156 points, over 7 weeks
Iteration 7 (10 stories, 84 points)
Iteration 8 (11 stories, 79 points)
Iteration 9, Release 3 (~~9~~ 11 stories, ~~71~~ 59 points)

Release 3 Summary

The focus of this release was on creating the event feature. The purpose of the feature is to allow administrator to create events and notify all the students. Instead of receiving multiple emails, we could be informed of all available events happening at concordia though the mobile application.

The application can also serve as a mean to tracking the attendance of each event. if you are administrator of an event, you will unlock the scanning capability which you will use to scan the virtual id of other users.

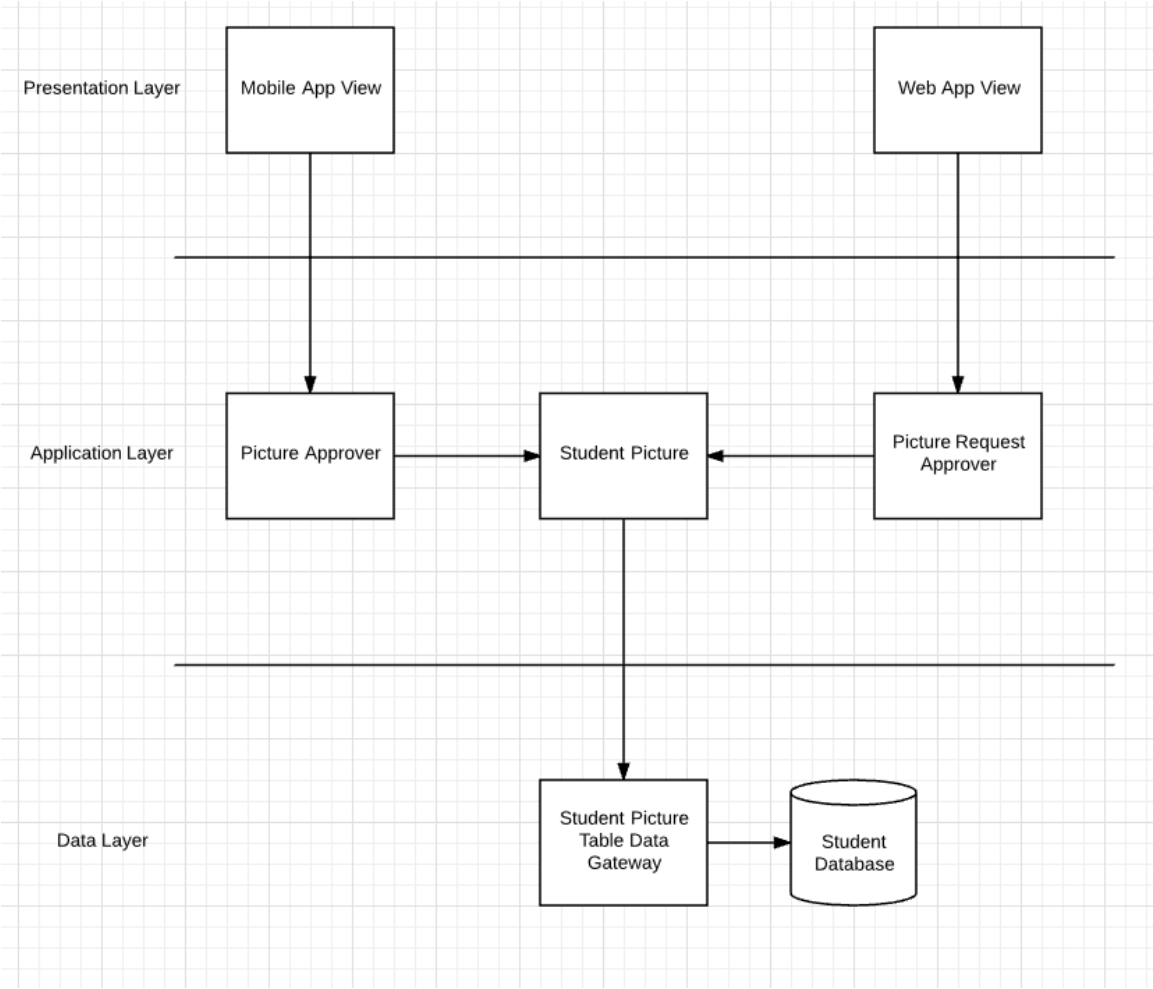
Plan up to next release

Iteration 10 (9 stories, 70 points)
Iteration 11 (6 stories , 42 points)

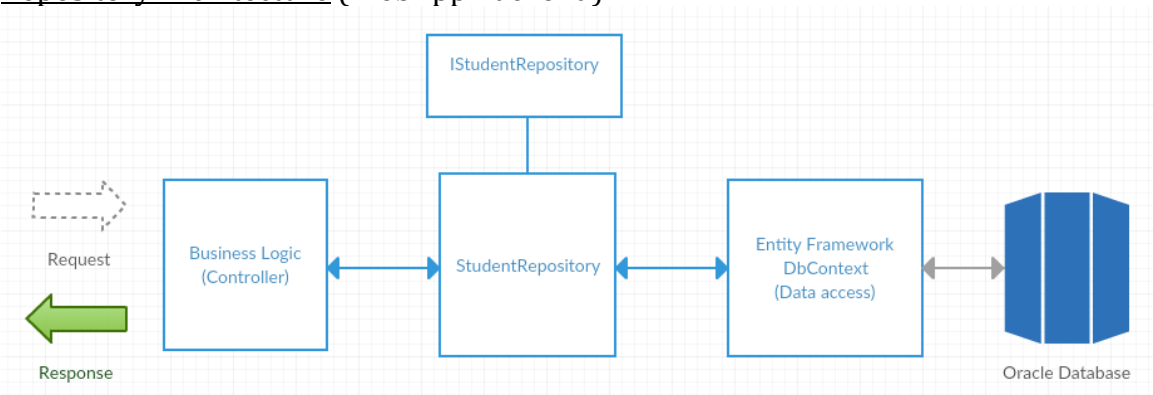
We will mostly be in a maintenance phase, where we will be fixing all the bugs and finishing the remaining small features. IIS will take over the project, therefore we will be producing a guide on how to use our system and a specification document which will help the next development team.

During spring 10, we will be presenting our project to IIS management and their development team.

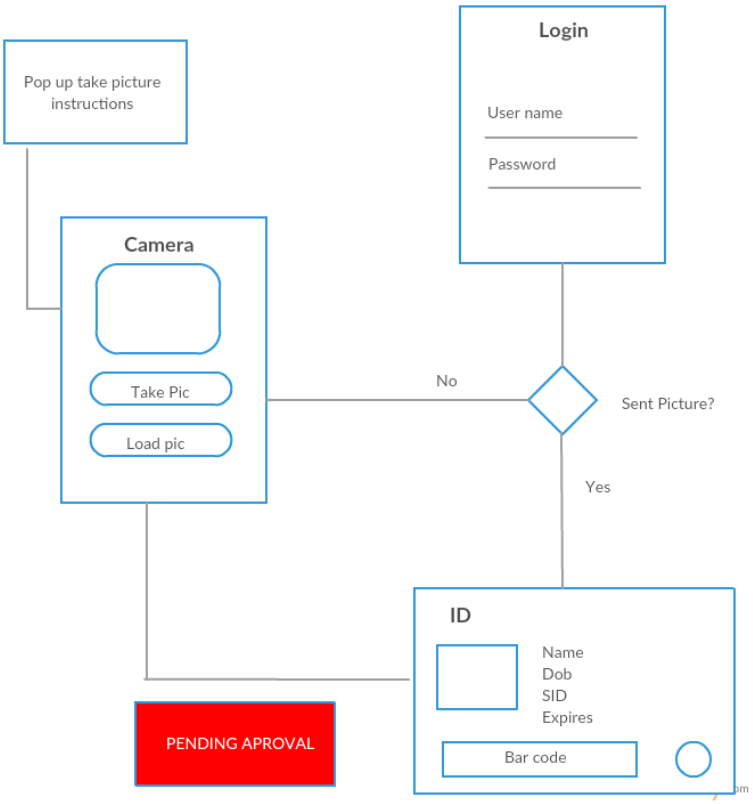
Overall Arch and Class diagram



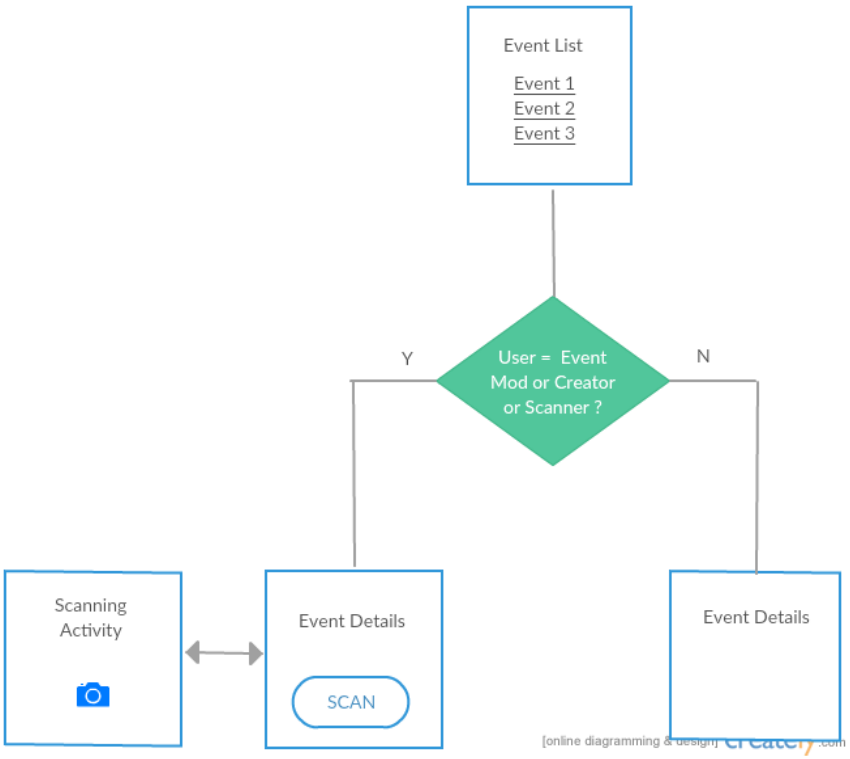
Repository Architecture (WebApp Backend)



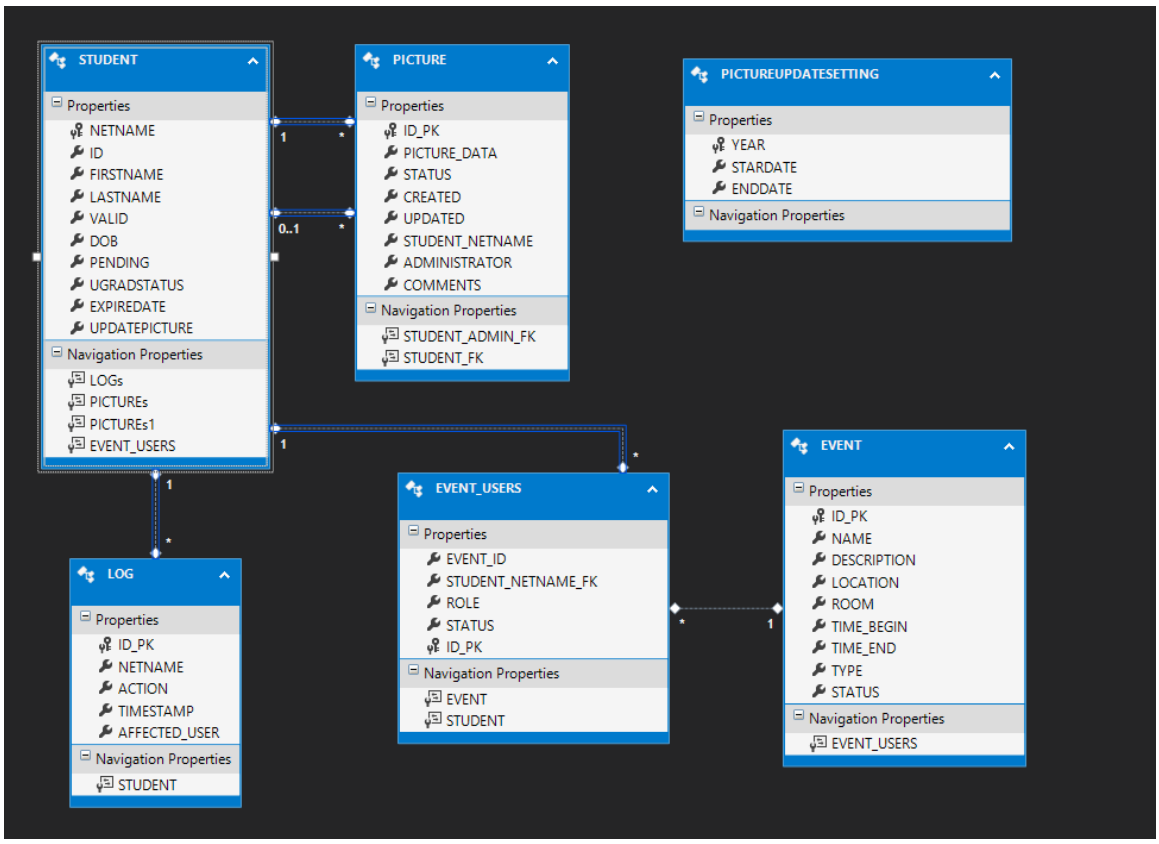
Control Flow Camera - Mobile App



Control Flow Events - Mobile App



Database Diagram



The database contain 6 tables :

- Student
- Picture
- PictureUpdatePeriodSetting
- Logs
- Event_users
- Event

The student tables are containing each pertinent information about a student. Each picture send by a student will be stored in the Picture table. The PictureUpdatePeriodSetting is used to store the valid period where everyone valid student will be able to update their profile picture. Log table is used to logs user's actions. We can therefore know which user did which action and which user was affected by it.

Event table is holding information about an event. Event_user is containing a list of

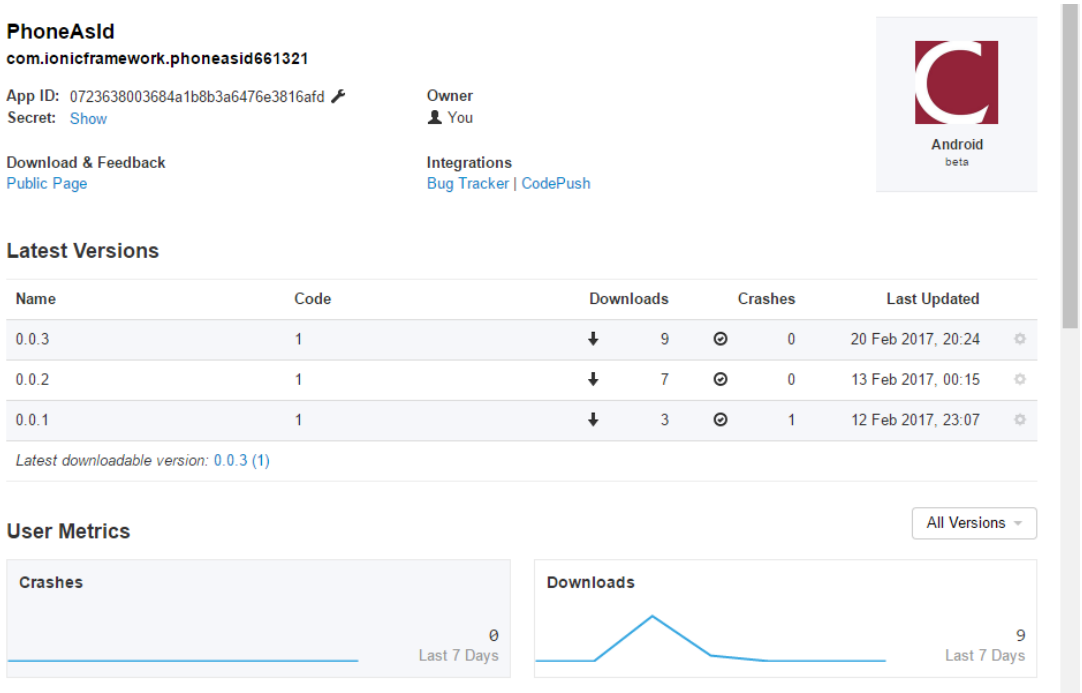
every users that are participating to an event.

Infrastructure

- **Phonegap/ionic**
- **ASP.NET**
- **Entity Framework**
- **Active Directory**
- **Oracle Database (Oracle 12C)**
- **Amazon Relational database service**
- **Azure**
- **Application Insights**
- **Google Vision**
- **netlify**
- **Jenkins**
- **swagger**
- Jenkins url
- netlify url

Infrastructure Changes

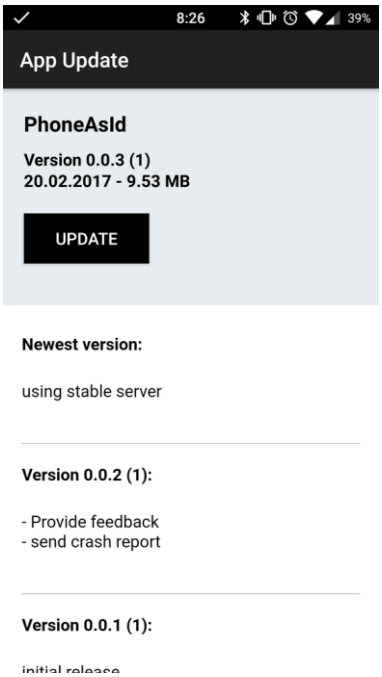
- **https://hockeyapp.net/**
- **android.myconcordiaid.me**



We are using hockeyapp for beta build distribution, crash reporting and getting user feedback. We automatically track specific event happening in our app. We will get a log of each successful login, attending to view the event page or id card.

User are forced to always be on the latest version.

.



Building Ios version

It's important to mention is that there were difficulties with regards to running the application on an iOS device. First off, building an iOS version of the application requires a MacBook and all the software related to it. Nobody on our team has a MacBook, and therefore we needed to acquire one to continue working on the issue. Because of this, there was a lot of back and forth between our stakeholders for them to finally lend one to us. Another issue was that they forgot to give us admin rights on the MacBook, and this delayed the process for building the application on the MacBook. Furthermore, a big challenge was getting accustomed to using a MacBook, and all the software needed to build and run our application on an iOS device. There were a lot of workarounds that needed to be done before a build would successfully compile. This includes needing an apple developer account when building the app, needing to gain familiarity with the software "XCode" in order to build and run the application on an iOS device, needing to use a 4k image in order to generate the iOS splash-screens, etc… That being said, the application has successfully compiled on an iOS device, but some of the features aren't functioning as planned, and some of the UI isn't displaying as intended. Those issues will be further considered in the upcoming release.

Name Conventions

- John Papa's Angular style guide
- C# Coding style for asp.net core

Code

File path with clickable GitHub link	Purpose (1 line description)
EventController	Event Related Apis
Event Repository	Store and Retrieve Event Information from the database
Event Controller (Mobile Application)	The event controller will get and display the list of all events that are available for the user. It will call a service that will return the list of events. That list will be displayed in the event.html page using ion-list.
EventController(Web application)	The event controller will get and display the list of all events that are available for the user. It will call a service that will return the list of events. That list will be displayed in the event.html page. In addition, it will handle the logic for displaying a list of a specific event's attendees.
studentModal.Controller.js	This controller is the biggest controller in the webapp and the most important. It handles most of the write interactions with the database as well as most of the reads. In addition to its designated job it is also an intermediary between some other components such as the imageModalController and the searchController. It follows the John papa's Angular style guide. Most of the logic is abstracted from the controller with the use of factories and services (Eg. StudentModalController uses StudentService)..
student.service.js (web app)	This service handles some the logic behind all the controllers. Every bit of logic that is related to student information passes through this service.

	It sorts of connect the backend and the frond end by handling all get and post requests. The services works as an abstraction layer between the backend and the frontend. All the services follows John papa's Angular style guide.
barcode.scanner.controller.js (mobile)	This controller will get the data for each specific event. If the user is registered for the event, the page will display the info for that event. If the user is a mod, an option will be available to scan users attending the event

Testing and Continuous Integration

Test File path with clickable GitHub link	What is it testing (1 line description)
GetEventById() Event Controller Unit Test Line 127	If we can retrieve a event information by using a Id.
GetEventByIdNotFound() Event Controller Unit Test Line 168	Test error handling if we can't find a event.
CancelEventNotFound() Event Api Unit Test Line 256	We send event id that doesn't exist, we are supposed to receive a 403 error since we can't cancel it.
UpdateEventNotFound() Event Api Unit Test Line 322	When we update an event we are required to send a correct event id which exist in the db or else we won't be able to find and will return a 403.

All unit tests made for the api server are runned each time by jenkins during the build process.

Finished SHORT Story summaries

Order your summaries by risk and priority. They should have the following form.

Points: 20, Priority: 1, Risk: High
Story #: 73 Porting android features and UI on iOS
Iteration #7
Even though ionic is made to build cross-platform apps, we still can gets some problems porting from one platform to another. Some UI needed to be modified

since they weren't scaling properly, as well as the app's features needed to be tested to make sure they work.

Points: 8, Priority: 1, Risk: High
Story #: 85 Back-port Sprint 7 to iOS
Iteration # 8
This user story was used to migrate the relevant changes made during the last port into the Sprint 7 version of the application. It was also used to tweak a couple of features of the iOS application.

Points: 8, Priority: 1, Risk: High
Story #: 87 iOS: Pages of the app are scaled down
Iteration # 8
In sprint 7 of the android app, there have been a lot of improvements that have been done to the UI. These improvements have fixed a lot of the issues that have been occurring when it comes to the page scaling on iOS. There were some issues that were still occurring, and required a fix, which have been worked on.

Points: 8, Priority: 1, Risk: High
Story #: 88 Back-port Sprint 8 to iOS
Iteration # 9
In the previous 2 sprints, backporting consisted of creating a new iOS branch from the previous version of the android app, merging previous iOS changes, and building on top of it. This time, the iOS branch was merged into the main android "dev" branch, which means there will no longer be any need to backport for the iOS.

Points: 8, Priority: 1, Risk: High
Story #: 108 Fix Scaling/UI issues
Iteration # 9
Building on top of the Scaling/UI changes previously made, this time around, the "Update Picture" page required some UI fixing, as well as the alerts generated throughout the iOS application.

Points: 8, Priority: 1, Risk: Medium
Story #: 95 Event descriptions
Iteration # 8
This user story involved mostly interfacing the web application and the backend API. The service calls had to be made in the front end to fetch various forms of information pertaining to an event such as an event's name, location, room description, date, and time. The backend had to be structured to be able to fetch data from the database and make it available to the front end. No issues encountered.

Points: 8, Priority: 1, Risk: Medium
Story #: 93 Create event page on webapp

Iteration # 8
We used fullpage.js to build the event UI. Fullpage.js does not work out of the box with angular so we had to modify a directive we found. There was a lot a of css involved to make fullpage.js work properly with scrolling container within some sections of fullpage. Fullpage.js is made to be used as a singleton component so we had to modify some code to make it work in the admin page and the event page.. We made a step by step process that is intuitive to use.

Points: 8, Priority: 1, Risk: Medium
Story #: 102 Admin academic year reformat
Iteration # 9
In order to follow the format used by Concordia for academic years (e.g. 2016-2017), we changed the format of the academic years selectable by the user when selecting the academic year.

Points: 5, Priority: 1, Risk: Low
Story #: 96 Modify event
Iteration # 8
Creating and event and modifying an event have the same action flow. The ui was reused.

Points: 5, Priority: 1, Risk: Low
Story #: 97 View event
Iteration # 8
We added a button to view details. Clicking on this button opens up a modal with the event information.

Points: 13, Priority: 2, Risk: Medium
Story #: 55 Admin Settings Aesthetics
Iteration # 7
Changes were made to the admin page esthetics. Much of the work consisted of placing the calendars side by side on the admin page. However, it was decided that using the fullpage.js library was the theme we wanted to go with. So, we decided to instead have the user be taken by a step by step process to change the update period, especially since this task is to be performed only once or twice a year.

Points: 13, Priority: 2, Risk: medium
Story #: 56 Student modal User interface
Iteration #7
Whenever you want to access a student account, you will be presented with a beautiful interface. We based our design on the tinder application, which uses a popular approve/deny process.

Points: 8, Priority: 2, Risk: medium
Story #: 72 Scan barcode for event

Iteration # 8
Certain users on the mobile device are designated with the ability to scan attendees. These roles are creator, mod and scanner. On the event details page a button will display giving these users the ability to track attendance.

Points: 8, Priority: 2, Risk: Medium
Story #: 92 See List of Events
Iteration # 8
For the event feature, every user needed to be able to see the list of events available to them. That list would only contain the events for which that user is either registered for, a scanner for, as well as open events. The information related to the event have to be displayed on the list.

Points: 5, Priority: 2, Risk: Medium
Story #: 75 Set Event Modification Privileges
Iteration # 9
This user story was underestimated in both number of points, and risk. First, the attendee infrastructure was not done prior to starting this feature. This included the html for the attendee section, the javascript for its controller, the necessary switch from using angular modals to angular material dialogs, as well as the addition of the mdDialog's html and javascript controller.

Second, because we used fullpage.js sliders to slide sideways, they came with **clickable** arrows appearing in the background, which is not behaviour we wanted. Thankfully, it was easily solved using css. In addition, we wanted to switch to using angular-materials components, such as the md-select, but it conflicted with the angular-modal. Issues with the z-index. We therefore had to switch to md-dialogs, which involved quite a bit of risk. Thankfully it worked, but these risks should have been looked into before starting the story. This story should be worth at least 13 points, rather than 5, with High risk.

Points: 3, Priority: 2, Risk: Low
Story #: 54 Formatted dates
Iteration # 7
We first added our own service to parse dates but found out with some research that angular has some built in functionalities to parse dates.

Points: 3, Priority: 2, Risk: low
Story #: 112 Loading Indication on Login
Iteration # 9
We added feedback for users after clicking login so that they know they're request is being processed.

Points: 2, Priority: 2, Risk: Low
Story #: 117 Delete event
Iteration # 9
Feature eventFeature
Delete event consisted only of adding a button and making the call to the api to cancel an event.

Points: 3, Priority: 2, Risk: Low
Story #: 121 Add users to event
Iteration # 9
Feature eventFeature
This user story was made to allow the user to add attendees to an event. The work consisted of adding an input text field similar to the one found in the review page, keeping track of the clicked event, and ensuring that no duplicate attendees are added to the list of attendees. No issues encountered.

Points: 3, Priority: 2, Risk: Low
Story #: 101 Event Ui
Iteration # 9
Feature eventFeature
Display all the available events in a nice user interface.

Points: 8, Priority: 3, Risk: medium
Story #: 49 Read a barcode
Iteration #7
For the event feature of the application, it was required that persons with the authorization should be able to scan the barcode of students that are register for an events. This story involved being able to open the barcode scanner from the application, and storing that info.

Points: 8, Priority: 3, Risk: Medium
Story #: 67 picture validation comments
Iteration # 7
In the student modal, we added a functionality for the admin to leave a comment associated with a picture.

Points: 8, Priority: 3, Risk: medium
Story #: 81 Application logs
Iteration # 9
In a production environments, we will need to be able to keep a log of any errors. Since we are using a windows system, we will use their event log infrastructure. Each errors will be displayed on the window event viewer.

Points: 5, Priority: 3, Risk: Low
Story #: 74 iOS Concordia Splash-screen
Iteration # 8
iOS applications require a 4K image in order to generate the varying sizes of splash-screens tailored for different types of iPhones. Since the android splash-screen is not 4K resolution, another official Concordia splash-screen was acquired from our stakeholders and is now integrated into the iOS version of the application.

Points: 5, Priority: 3, Risk: low
Story #: 111 Fix login screen CSS delay in loading
Iteration # 9
There was a brief flash of white after the initial splash screen ended and before the initial page was showing giving a poor user experience. This is a due to the ionic platform taking more time to load. .

Points: 5, Priority: 3, Risk: Low
Story #: 123 Event search filter
Iteration # 9
Feature eventFeature
We added filters to make the event list more readable. You can filter the event list by their state (public, private, cancelled rescheduled..etc). The filter selection was made with md-chips which are part of angular-material library. They were not made for that purpose so some additional logic and css was written to make them behave properly.

Points: 3, Priority: 4, Risk: Low
Story #: 66 Keyboard Hotkeys
Iteration # 7
To customize the user's experience using the application, we are adding some hotkeys to the web application. For now, it is mainly for submitting the update period in the admin page, but we will be slowly adding some more to other parts of the website as needed. No issues encountered.
Points: 13, Priority: 4, Risk: low
Story #: 68 Nice side menu
Iteration #7
We moved from css to scss to have proper structure. Since the web application is already using scss, we will be able to easily create similar designs.

Points: 8, Priority: 4, Risk: low
Story #: 69 Id card UI improvement
Iteration #7
Since the id card is the most important part of our application, we have decided to update our previous generic template.

Points: 13, Priority: 4, Risk: Low
Story #: 89 Standardize loading and feedback
Iteration # 8
Loading and feedback were not standardized throughout our application. We made an http request wrapper that encapsulate md-toast which is part of angular-material. It means that one does not have to call explicitly md-toast.show() to give feedback, it does it automatically. During each call a loading notification appear and the user receives feedback upon task completion.