# PhoneAsId Release 2 Summary

## Team members

|  |  |  |
| --- | --- | --- |
| Name and Student id | GitHub id | Number of story points that member was an **author** on. |
| Michał Wozniak 21941097 | mv740 | 20(#2) + 8(#15)+ 13(#28)+8(#29)+ 13(#33) 3(#38)+5(#48)+13(#50)+8(#46) =**91** |
| Francis Côté-Tremblay 26615287 | francisct | 8(#9) + 8(#15) + 8(#29) + 5(#48) + 13(#50) = **43** |
| Ahmed Dorias 26649874 | ConfusedGiant | 5(#16) + 5(#31) + 5(#30) + 13(#41) = 28 |
| Harrison Ianatchkov 26607403 | zzharryzz | 8(#45) + 13(#28) = **21** |
| Simon Monière Abes 26648568 | simonma1 | 20(#2) + 5(16) + 5(32) + 13(#27) = **43** |
| Sebastian Rafique Proctor-Shah 29649727 | EXPSPACE | 20(#2) + 5(#16) + 3(#35) +13(#33) + 8(#47) = **49** |

## Stakeholders

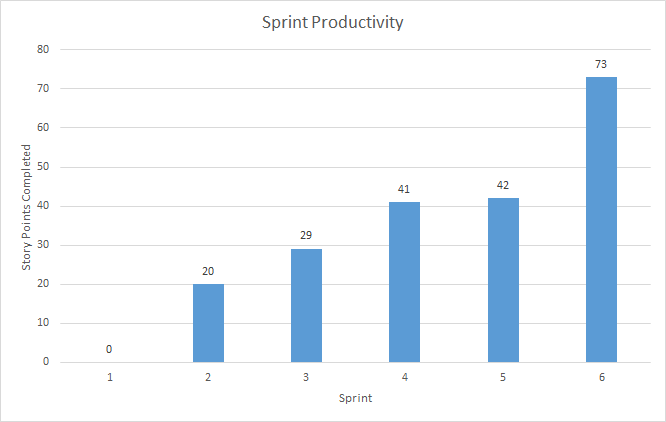
|  |  |
| --- | --- |
| Name | Github id |
| Jean Francois Bourgault | jfbourgault |
| Timothy Ni | timni |

## 

## 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 205 points over 6 iterations = **34 user story points / iteration**

Total: 25 stories, 205 points over 13 weeks\*

[Iteration 1](https://github.com/mv740/E-Wok-MyConcordia/milestone/1) (0 stories, 0 points)

[Iteration 2](https://github.com/mv740/E-Wok-MyConcordia/milestone/2) (4 stories, 20 points)

[Iteration 3](https://github.com/mv740/E-Wok-MyConcordia/milestone/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](https://github.com/mv740/E-Wok-MyConcordia/milestone/4) (5 stories, 41 points)

[Iteration 5](https://github.com/mv740/E-Wok-MyConcordia/milestone/5) (5 stories, 42 points)

[Iteration 6](https://github.com/mv740/E-Wok-MyConcordia/milestone/6), Release 2 (~~2~~ 8 stories,~~18~~ 73 points)

## Plan up to next release

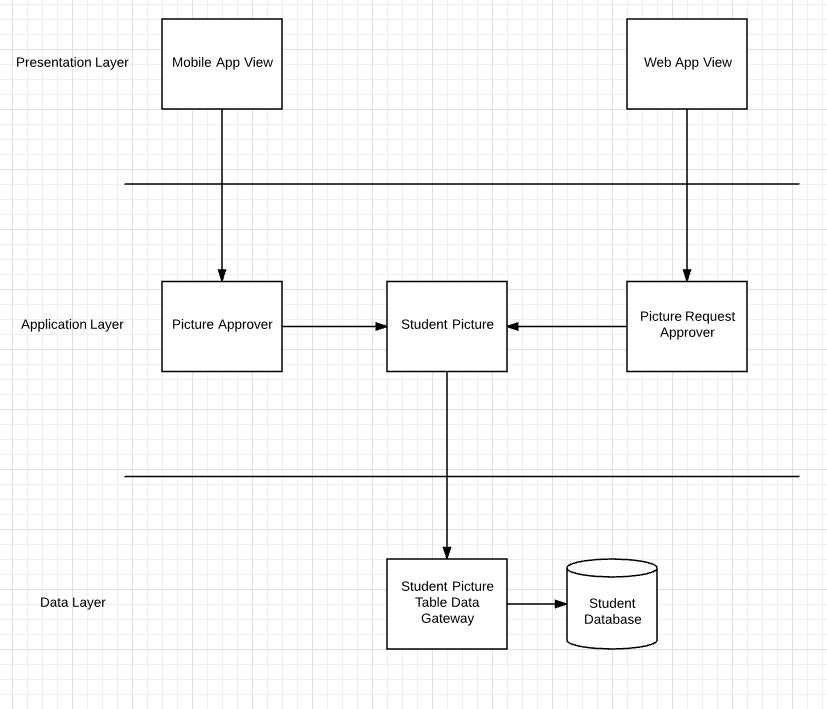
Total: 12 stories, 156 points, over 7 weeks

[Iteration 7](https://github.com/mv740/E-Wok-MyConcordia/milestone/7) (~~4~~ 5 stories, ~~38~~ 41 points)

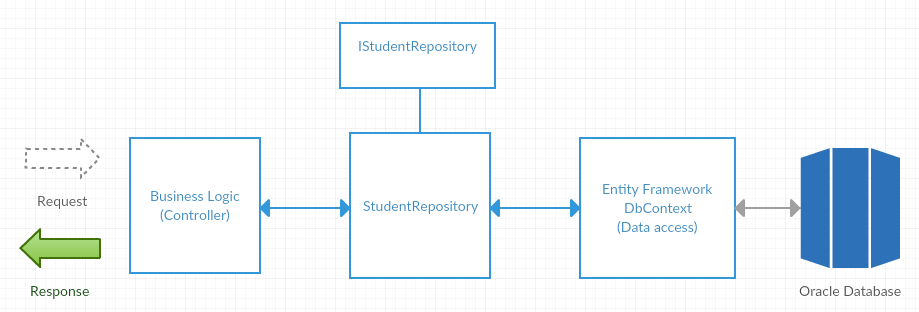
[Iteration 8](https://github.com/mv740/E-Wok-MyConcordia/milestone/8) ( 5 stories, 37 points)

[Iteration 9, Release 3](https://github.com/mv740/E-Wok-MyConcordia/milestone/9)  (0 stories, 0 points)

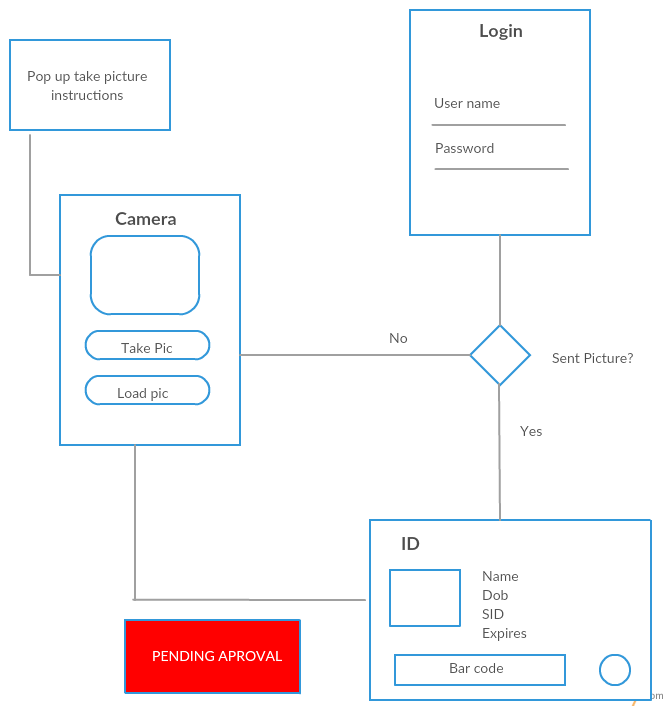
## Overall Arch and Class diagram



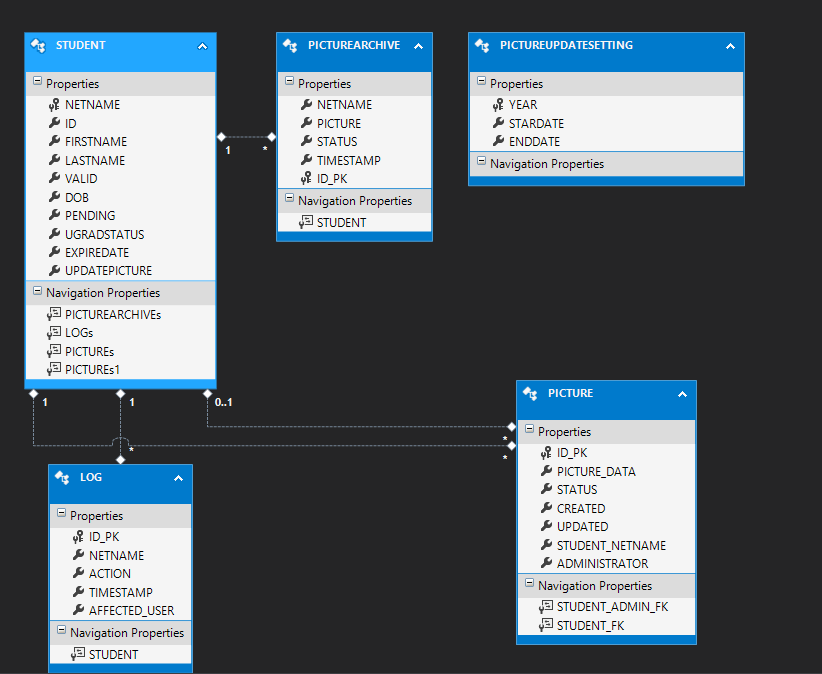
Repository Architecture (WebApp Backend)



Control Flow Mobile App



Database Diagram



We have added the log and picture tables to our database. We previously stored the pending picture and the profile picture in the student table. We realise it would be better to refactored this by extracting the pictures into it own table. . This made handling specific action on a picture more easy.   
  
We keep track of each action in the log table. The “picturearchive” table will be removed in the next version.

## Infrastructure

* [**Phonegap/ionic**](http://ionicframework.com/)
* [**ASP.NET**](https://www.asp.net/get-started)
* [**Entity Framework**](https://www.asp.net/entity-framework)
* [**Active Directory**](https://azure.microsoft.com/en-us/documentation/articles/active-directory-aadconnect/)
* [**Oracle Database (Oracle 12C)**](http://www.oracle.com/us/corporate/features/database-12c/index.html)
* [**Amazon Relational database service**](https://aws.amazon.com/rds/)
* [**Azure**](https://azure.microsoft.com/)
* [**Application Insights**](https://azure.microsoft.com/en-us/services/application-insights/)
* [**Google Vision**](https://cloud.google.com/vision/)
* [**netlify**](https://www.netlify.com)
* [**Jenkins**](http://173.176.41.65:1111/job/MyConcordiaID/)
* [**swagger**](http://swagger.io/)

**Infrastructure Changes**

We are now using continuous integration system to increase our productivity. We have split our web application into two system : web client , api server. The Api server is being build by jenkins. Every commit to our web-api-dev branch will notify jenkins, which will build the latest commit, run all the unitest and if everything pass successfully, it will publish the new version to azure. The web client is being monitored by netlify, which build the latest commit from the web-dev branch and host the latest version of our client application.

* [Jenkins url](http://173.176.41.65:1111/job/MyConcordiaID/)
* [netlify url](https://concordiaidclient.netlify.com/WebApp/app/login)

I am using the swagger framework to document the backend api. We can now use simple “java doc” to comment our api. During the generation of the application, we get a nice [visual dynamic documentation](https://myconcordiaid.azurewebsites.net/swagger).

**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 in order to continue working on the issue. Because of this, there was a lot of back and forth between our stakeholders in order 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 looked into in the upcoming release.

## Name Conventions

* [John Papa’s Angular style guide](https://github.com/johnpapa/angular-styleguide/blob/master/a1/README.md#controllers)
* [C# Coding style for asp.net core](https://github.com/dotnet/corefx/blob/master/Documentation/coding-guidelines/coding-style.md)

## Code

|  |  |
| --- | --- |
| File path with clickable GitHub link | Purpose (1 line description) |
| [StudentController](https://github.com/mv740/E-Wok-MyConcordia/blob/master/MyConcordiaID/src/MyConcordiaID/Controllers/StudentController.cs) | Student Related Apis |
| [Student Repository](https://github.com/mv740/E-Wok-MyConcordia/blob/master/MyConcordiaID/src/MyConcordiaID/Models/Student/StudentRepository.cs) | Store and Retrieve Student Information from the database |
| [Id Controller](https://github.com/mv740/E-Wok-MyConcordia/blob/master/PhoneAsId/www/js/id.controller.js) (Mobile Application) | Id controller is used to pass information related to the student to be displayed on the student card and on the marshalling card. The controller was implemented in the form of controlleras, following the [John Papa’s Angular style guide](https://github.com/johnpapa/angular-styleguide/blob/master/a1/README.md#controllers) standard. It is using a service to fetch the data to be used |
| [Barcode Directive](https://github.com/mv740/E-Wok-MyConcordia/blob/master/PhoneAsId/www/js/barcode.directive.js) (Mobile Application) | Directive in Angular are used to manipulate DOM objects. In this case this directive will take data from the id controller and generate a barcode from the value passed in the html, which will add an id. It will pass that id to the JsBarcode function, which is from a library to generate barcode. |
| [studentModal.Controller.js](https://github.com/mv740/E-Wok-MyConcordia/blob/master/WebApp/app/partials/review/modals/studentModal/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](https://github.com/johnpapa/angular-styleguide). Most of the logic is abstracted from the controller with the use of factories and services (Eg. StudentModalController uses StudentService).. |
| [student.service.js](https://github.com/mv740/E-Wok-MyConcordia/blob/master/WebApp/app/services/student.service.js) | 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](https://github.com/johnpapa/angular-styleguide). |

## Testing and Continuous Integration

|  |  |
| --- | --- |
| Test File path with clickable GitHub link | What is it testing (1 line description) |
| GetStudentById()  [Student Controller Unit Test Line 149](https://github.com/mv740/E-Wok-MyConcordia/blob/master/MyConcordiaID/src/UnitTestCore/StudentControllerTests.cs) | If we can retrieve a student information by using a Id. |
| GetStudentByIdNotFound()  [Student Controller Unit Test Line 168](https://github.com/mv740/E-Wok-MyConcordia/blob/master/MyConcordiaID/src/UnitTestCore/StudentControllerTests.cs) | Test error handling if we can’t find a user. |
| FindPendingPicture()  [Student Api Unit Test](https://github.com/mv740/E-Wok-MyConcordia/blob/master/MyConcordiaID/src/UnitTestCore/StudentUnitTest.cs) Line 256 | If we can retrieve a user pending picture by searching using an id. |
| ValidatePicture()  [Student Api Unit Test](https://github.com/mv740/E-Wok-MyConcordia/blob/master/MyConcordiaID/src/UnitTestCore/StudentUnitTest.cs) Line 322 | if an administrator deny a pending picture, when we search for this picture, it status is “denied” |
| Search function  [Webapp behavior test](https://github.com/mv740/E-Wok-MyConcordia/blob/master/WebApp/app/tests/search.test.js) Line 15 | When searching for a specific ID, it is expected to get one result. |
| StudentModal info fetch  [Webapp behavior test](https://github.com/mv740/E-Wok-MyConcordia/blob/master/WebApp/app/tests/search.test.js) Line 24 | When clicking on a result, it is expected to display the StudentModal with some information related to the student displayed in the result list. |

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 #: [2](https://github.com/mv740/E-Wok-MyConcordia/issues/2) Authentication

Iteration # [4](https://github.com/mv740/E-Wok-MyConcordia/milestone/4)

Users required authentication before being able to use the web and mobile clients. The method of authentication of choice was OAuth2. Very little documentation was available for configuring this authentication so the process was long and tedious, with a significant amount of trial and error. Because of this, the feature was pushed back several by sprints.

Points: 13, Priority: 1, Risk: High

Story # [50](https://github.com/mv740/E-Wok-MyConcordia/issues/50) Info on Enlarged Archived Pictures

Iteration # [6](https://github.com/mv740/E-Wok-MyConcordia/milestone/6)

Archived pictures now display information pertaining to that student and to the picture. In order to be able to display associated information we had to modify the database and the backend as well as the front end. Adapting the database was the biggest part and that is why the number of points assigned was 13. We managed to save some time by using validation modal as a template to display an enlarged archive picture.

Points: 8, Priority: 1, Risk: High

Story # [15](https://github.com/mv740/E-Wok-MyConcordia/issues/15) Student Search

Iteration # [4](https://github.com/mv740/E-Wok-MyConcordia/milestone/4)

Being able to search for a specific student account using different search parameter : id, first name, last name, birth date. Required to compose a query in multiple steps to chain optional search parameter. By using entity framework, we could use their queries to build one in pieces, during execution it would consolidate all the part into one single query. Lot of manual testing was required to test all different search possibility.

Points: 13, Priority: 1, Risk: Medium

Story # [41](https://github.com/mv740/E-Wok-MyConcordia/issues/41) iOS Application Access

Iteration # [6](https://github.com/mv740/E-Wok-MyConcordia/milestone/6)

Since the application is built using PhoneGap, most of the code created for android can be reused for the iOS build. That being said, there are requirements needed to build for iOS, such as: building and running on a MacBook, having an apple developer account (and being signed in), installing XCode, and running the application on an iPhone. This has been successfully tested on an iPhone 5c, although some UI and features aren’t working as expected

Points: 5, Priority: 1, Risk: Low

Story #: [16](https://github.com/mv740/E-Wok-MyConcordia/issues/16) General purpose student ID

Iteration # [4](https://github.com/mv740/E-Wok-MyConcordia/milestone/4)

This is the view that represents the student’s Id card. It contains all the related information such as name, id number, barcode, program, etc. This data is gotten from a GET request to the server, and is then displayed and styled using CSS

Points: 13, Priority: 2, Risk: High

Story # [27](https://github.com/mv740/E-Wok-MyConcordia/issues/27) Student Id Generated Barcode

Iteration # [6](https://github.com/mv740/E-Wok-MyConcordia/milestone/6)

Concordia Id cards contain a barcode which is generated from the student id number and that are to be displayed on the student card. This barcode will be generated from a javascript library called JsBarcode, which will take a value and generate a barcode from it, following the desired barcode standard, which is code 39 in our case. This was implemented using a directive which will output the generated barcode directly in the DOM.

Points: 5, Priority: 2, Risk: High

Story # [30](https://github.com/mv740/E-Wok-MyConcordia/issues/30) Generated Barcode for Marshalling Card

Iteration # [6](https://github.com/mv740/E-Wok-MyConcordia/milestone/6)

Each marshalling card contains a barcode which is generated from the unique marshalling code linked to the student. This barcode is generated from JsBarcode, which is a javascript library that takes a value and generates a barcode from it. It can generate the barcode using different barcode standards, including “code 39” which is the one concordia uses. A directive is used to out the generated barcode directly in the DOM, making it visible on the marshalling card.

Points: 13, Priority: 2, Risk: Medium

Story # [33](https://github.com/mv740/E-Wok-MyConcordia/issues/33) Marshalling Card Data

Iteration # [5](https://github.com/mv740/E-Wok-MyConcordia/milestone/5)

We had to create a fake service which would return a user’s marshalling card. This service would usually call a concordia web service which would pass us the marshalling card information. The marshalling card data format was given to us by the stakeholders.

Points: 5, Priority: 2, Risk: Medium

Story # [48](https://github.com/mv740/E-Wok-MyConcordia/issues/48) Reapprove declined pictures

Iteration # [6](https://github.com/mv740/E-Wok-MyConcordia/milestone/6)

The administrator can now approve a previously denied picture. He can also deny a current valid profile picture forcing the student to send a new one. This mean the status of each picture can be changed. It required a rigorous backend logic to handle different cases : profile picture get denied, denied picture became profile picture , denied picture is approved but student didn’t have any profile picture …

Points: 8, Priority: 2, Risk: Low

Story # [29](https://github.com/mv740/E-Wok-MyConcordia/issues/29) Admin can view logs

Iteration # [5](https://github.com/mv740/E-Wok-MyConcordia/milestone/5)

The logs are basically the text format of the archived pictures (when and what). In order to display the logs a whole new UI was needed. Another story was created to account for the points of designing a new UI. Displaying the log was a matter of a few lines in the html file. However some javascript and some backend code was required to make it work.

Points: 13, Priority: 3, Risk: Low

Story # [28](https://github.com/mv740/E-Wok-MyConcordia/issues/28) Update Period

Iteration # [5](https://github.com/mv740/E-Wok-MyConcordia/milestone/5)

The work involved finding a library for a calendar popup, setting up the HTML layout of the web page, then adding validation for each input field. The validation included the 720Kb library tooltips that appear indicating which fields are required when the user hovers over it with a mouse. In addition, borders change colour as the user enters data properly, allowing him/her to submit a new Update Period.

Points: 8, Priority: 3, Risk: Low

Story # [47](https://github.com/mv740/E-Wok-MyConcordia/issues/47) Picture View UI

Iteration # [6](https://github.com/mv740/E-Wok-MyConcordia/milestone/6)

On mobile, you have to take a profile picture or load one from your phone. We have update the interface to follow concordia’s color with a simple design showing the available options.

Points: 8, Priority: 3, Risk: Low

Story # [45](https://github.com/mv740/E-Wok-MyConcordia/issues/45) User Feedback For Empty Results

Iteration # [6](https://github.com/mv740/E-Wok-MyConcordia/milestone/6)

This story involved handling specific errors returned by the server upon making requests that return null, or empty results. Empty results or error messages were handled by displaying pertinent messages, or re-routing users to different pages.

Points: 8, Priority: 3, Risk: Low

Story # [46](https://github.com/mv740/E-Wok-MyConcordia/issues/46) ID Card View UI

Iteration # [6](https://github.com/mv740/E-Wok-MyConcordia/milestone/6)

Display the student card information in a clean interface where it is easy to understand each part. Difficult to design a simple interface when we are restricted by the size of the picture.

Points: 5, Priority: 3, Risk: Low

Story # [31](https://github.com/mv740/E-Wok-MyConcordia/issues/31) Pending Approval Card

Iteration # [4](https://github.com/mv740/E-Wok-MyConcordia/milestone/4)

A pending approval card was created in order to let the student know if the desired ID picture is sent and waiting to be approved. If the picture has been successfully sent, then a pending approval card will be displayed in place of the photo ID. Once the photo has been approved, then the pending approval card will be removed, and the students’ actual picture will be displayed instead.

Points: 5, Priority: 3, Risk: Low

Story # [32](https://github.com/mv740/E-Wok-MyConcordia/issues/32) Marshalling Card

Iteration # [5](https://github.com/mv740/E-Wok-MyConcordia/milestone/5)

When a student is about to graduate, they will gain access to a card that contains information regarding their graduation such as the date, time, location, as well as a barcode for them to be authenticated. A sample marshalling card was given with which the styling of the marshalling page was based on. This styling was done in css following the requirements

Points: 3, Priority: 3, Risk: Low

Story # [35](https://github.com/mv740/E-Wok-MyConcordia/issues/35) Login UI

Iteration # [4](https://github.com/mv740/E-Wok-MyConcordia/milestone/4)

This is the first page you see when you open the mobile application. It must look professional, therefore we have used a background image representing the EV building which most of the student and faculty will recognize. We used the official colors of concordia.

Points: 3, Priority: 3, Risk: Low

Story # [38](https://github.com/mv740/E-Wok-MyConcordia/issues/38) Login UI

Iteration # [5](https://github.com/mv740/E-Wok-MyConcordia/milestone/5)

When you initiate the login process which is using oauth 2, you will need to follow a couple of step to login successfully. To make this as easy as possible, each step have clear indication of what is require to be able to proceed to the next one.

'

## IP agreement

The type of IP agreement that is applicable for our project is an inequitable agreement where the stakeholder will keep all rights to the code and project. It is probably different from other teams in that we are cooperating with a learning establishment, instead of a large, for profit, company. Concordia University, our stakeholder, will keep ownership of the project, and will be free to release or improve upon the work that was done by our team. The service our application is offering will not provide any direct revenue to the school. It is a way for them to improve the service they offer to its student and keep up with today’s technology with which the students are familiar with and are using everyday. Since the idea for this app was still a proof of concept Concordia was open to the idea of letting some of its student work on this idea for them. Since collaboration with some member of the actual SIS team was necessary for the advancement of the project, Concordia had to make sure that all rights to the product would remain with them upon completion of our time working on the application. We were faced with the choice of dropping the project or agreeing to give full rights to the assignee.

Having this type of agreement is usually not profitable for the assignors, because it ensure that no benefit from the success of the project will go to them. We accepted to work on this task because of the nature of the capstone course and also because we would like to help future students appreciate the technological advances of the school. We also believed that working alongside representative of the school we were about to graduate from to be a great opportunity that presented itself to us and one that we could not decline. All that is asked in return is mention of the effort put in by all members of our team if the school was to ever release our product.