*Florida International University*

*School of Computing and Information Sciences*

Software Engineering Focus

Final Deliverable

Technology Driven Active Learning in Human Anatomy Laboratory Ver 1.0

**Team Members:**

Darian Mendez

Hector Cen

**Product Owner(s)**:

Lisa Brinn

**Mentor(s)**:

Mohsen Taheri

**Instructor**: Masoud Sadjadi

The MIT License (MIT)

Copyright (c) *2016 Florida International University*

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

***Abstract***

*The Human Anatomy Lab at Florida International University provides models of the human body to students with the purpose of testing their abilities, throughout recognition and examination of each feature in the human body. Students that take this lab can only see a particular part of the plastic human model one time during the lecture that covers it. Once the students leave the classroom; they do not have access to the same model again. For this reason, students use the lecture time to take pictures instead of recognizing and examining the models the way it should be. Also, if the students do not take the pictures, then they lack material to study for the tests; leaving students with an impossible choice of taking pictures or paying attention to the lecture. This behavior is affecting their grades negatively.*

*The solution to this problem is to provide students with a media-rich and highly interactive environment that contain all the lecture materials and necessary tools to study. By giving them unlimited access to pictures of the unique FIU plastic human models.  The Anatomy Lab Ver 1.0 is an iPad application that was developed to help the students improve their Human Anatomy Lab experience. The app will allow students to have access to informative videos of the models. Furthermore, the students will be able to test their knowledge by taking practice quizzes. Human Anatomy Laboratory app will eliminate the student's necessity of taking pictures during the lab time since it will provide unlimited access to the lab materials in or out school. The application will revolutionize Florida International University by pushing it forward through the adoption of a friendly mobile classroom approach.*

Table of contents

[**Introduction** 6](#_Toc468454380)

[**Current System** 6](#_Toc468454381)

[**Purpose of New System** 6](#_Toc468454382)

[**User Stories** 7](#_Toc468454383)

[**Implemented User Stories** 7](#_Toc468454384)

[**Pending User Stories** 16](#_Toc468454385)

[**Project Plan** 18](#_Toc468454386)

[**Hardware and Software Resources** 18](#_Toc468454387)

[***Hardware*** 18](#_Toc468454388)

[***Software*** 18](#_Toc468454389)

[**Deployment requirements** 19](#_Toc468454390)

[**Sprints Plan** 20](#_Toc468454391)

[***Sprint 1*** 20](#_Toc468454392)

[***Sprint 2*** 21](#_Toc468454393)

[***Sprint 3*** 23](#_Toc468454394)

[***Sprint 4*** 26](#_Toc468454395)

[***Sprint 5*** 29](#_Toc468454396)

[***Sprint 6*** 30](#_Toc468454397)

[**System Design** 33](#_Toc468454398)

[**Architectural Patterns** 33](#_Toc468454399)

[**System and Subsystem Decomposition** 33](#_Toc468454400)

[**Design Patterns** 36](#_Toc468454401)

[**System Validation** 37](#_Toc468454402)

[**Glossary** 59](#_Toc468454403)

[**Appendix** 60](#_Toc468454404)

[**Appendix A - UML Diagrams** 60](#_Toc468454405)

[**Appendix B - User Interface Design** 83](#_Toc468454406)

[**Appendix C - Sprint Review Reports** 96](#_Toc468454407)

[**Appendix D - User Manuals, Installation/Maintenance Document, Shortcomings/Wishlist Document and other documents** 99](#_Toc468454408)

[**Sprint Retrospective**: 99](#_Toc468454409)

[**Installation** 103](#_Toc468454410)

[***Development environment setup*** 103](#_Toc468454411)

[**Maintenance** 104](#_Toc468454412)

[***Updating third party libraries*** 104](#_Toc468454413)

[**Shortcomings / Wishlist** 104](#_Toc468454414)

[***Wishlist*** 104](#_Toc468454415)

[***Shortcomings*** 105](#_Toc468454416)

[**References** 106](#_Toc468454417)

**Introduction**

The Technology Driven Active Learning in Human Anatomy Laboratory Ver 1.0 is an iPad application, developed using version 2.3 of the *Swift* programming language that aims to mitigate and alleviate the problems that students of Human Anatomy Lab face at Florida International University. The application will provide the students the possibility of study and examine the labeled pictures of the laboratory’s plastic models. Students will also have access to the videos that Dr. Lisa Brinn has created, where she explains in detail everything that the student needs to know about the human body through the models and/or pictures. Another important feature of the app is the possibility of taking unlimited quizzes to practice the material. Students will be able to take a multiple choice quiz, fill in the blank or a combination of both. The quizzes settings can be changed to match the user preferences; students can modify the time, the type of questions, and select what content to test.

**Current System**

There is currently no system to help student at Florida International University to prepare for the tests of the Human Anatomy Lab class. Students have to use the laboratory time to take pictures of the models. To study for tests, students will use the photos to tag the term in the models. This approach undermines the student's efficient use of time during class hours, negatively affecting their learning process and ability to harness the knowledge needed to perform well in tests and evaluations.

**Purpose of New System**

The Technology Driven Active Learning in Human Anatomy Laboratory Ver 1.0 is an iPad application that will provide a starting point in solving the issues and challenges that students are facing in the Human Anatomy Lab. The application will provide student with access to all the course material, such as pictures of the models with the term tagged, and informative videos that explain the body parts in the plastic models in detail. It will also offer options to take practice quizzes. Quizzes can be of multiple choice, fill in the blank or a combination of both. Students will also be able to make their own study plan. Lab's stations can be marked in order to keep track of what needs to be reviewed, what is completed or if any station is a student's favorite. They will also be able to personalize their quiz experience by changing the time, the type of questions and material that they wish to be tested on. The main goal of the application is to provide a baseline upon which, much needed student feedback needs to be poured in and further improvement needs to be added, in order to achieve the intended learning environment. At the same time it will encourage other courses faculty to follow a similar path, and stimulate the addition of new tools and approaches to aid the teaching-learning process.

**User Stories**

The following section provides the detailed user stories that were implemented in this iteration of the Technology Driven Active Learning in Human Anatomy Laboratory Ver 1.0  project. These user stories served as the basis for the implementation of the project’s features. This section also shows the user stories that are to be considered for future development.

**Implemented User Stories**

**#118 Project template with TabBarController**

**Description:**  
As a developer I want the app to displays Labs information, takes Quizzes and contains Terminology information so that the students can study it in class.

**Acceptance Criteria:**  
The app displays the three tabs (Labs, Quizzes, Terminology.

**#119 Display the Labs list**

**Description:**

As a developer I want that when the student taps on Labs the app will display the Labs list in the left table so that my students can see a list of the labs that the class have.

**Acceptance Criteria:**

On tap on Labs tab, display the Labs list in the table

Set as default view

**#120 Display popup to select desired station**

**Description:**

As a developer I want that on tapping the selected lab in the menu, the app will display a popup with a list of lab stations so that the students can see the stations that the selected lab have.

**Acceptance Criteria:**

The app shows a popup with a list of stations for the selected lab

**#121 Load Lab 1 information**

**Description:**

As a developer I want to save the Lab 1 information in a static data structure inside the app bundle so that the data is saved in the application to be use in the view displays.

**Acceptance Criteria:**

The terminology will be divided by stations and each Lab will have several stations

**#123 Display Take Quiz button**

**Description:**

As a developer I want to display a button with a "Take Quiz" label so that the students will be able to click it and move to the take quiz view.

**Acceptance Criteria:**

A button with a "Take Quiz" label is shown.

**#124 Display Video button**

**Description:**

As a developer I want to display a button with a "Video" label so that when the students clicks on them they will be move to the station video view.

**Acceptance Criteria:**

A button with a "Video" label is shown.

**#125 Display station information**

**Description:**

As a developer I want that when the student taps on a station of a given Lab, the app will display information of that specific station in the Detail View Controller so that the students can select a station from the popup and see the data that the station have.

**Acceptance Criteria:**

The Detail View Controller will display the given station information for the given Lab.

**#135 Create a Welcome Screen**

**Description:**

As a teacher I would like my students to see a Welcome Screen from where they can Sign in using their PantherMail account so that only the university students can login to the application.

**Acceptance Criteria:**

The app displays a welcome screen when the student is not logged in.

The welcome screen display a title and a "Login with your PantherMail account".

**#129 Load and Play Station Videos**

**Description:**

As a teacher I will like my student to be able to click on the Video button of a selected station an see a list of available videos. Then the student may select a video, and that video will be display.

**Acceptance Criteria:**

The Video Button Shows a screen with a list of available videos for the selected station.

The student select a video, and the video plays.

**#130 Allow app login through PantherMail** **account**

**Description:**

As a teacher I would like my students to be able to login using their PantherMail credentials so they can access the information in the app.

**Acceptance Criteria:**

A student logs in successfully using the PantherMail account.

The app allows the student to access the Labs information.

The app won't allow access to users signing in with accounts with domains other than "fiu.edu"

The app handles all the exceptions thrown by the authentication framework.

**#131 Display the Take Quiz View**

**Description:**

As a teacher I will like for my student to press the Take Quiz button an display the TakeQuiz window for the station that the student is currently in.

**Acceptance Criteria:**

The button TakeQuiz display a new window

The window contain a short description of the station for the quiz selected

The new window contain three button to select the type of Quiz (fill in the blank, multiple choice or random)

**#132 Load full station information in the detail view**

**Description:**

As a teacher I would like the students to have a full view with the information of the selected station so they can learn and practice the lab.

**Acceptance Criteria:**

The detail view is populated with the list of terms

On tapping a term in the list, the app should update the ImageView next to the terminology list and show an image related to the term.

**#133 Disable the Logout option while taking a Quiz**

**Description:**

As a teacher I will like to cancel the logout option while the student is taking a quiz so that the students can’t interrupt the quiz.

**Acceptance Criteria:**

The student is taking a quiz, and the logout button is disable.

The student can logout after finish the quiz.

**#134 Allow the users to sign out from the app**

**Description:**

As a teacher I would like the app to allow the students to sign out from the app so that they can close their session.

**Acceptance Criteria:**

The app cleans all the information about the logged in student.

The app switches the detail view controller to the first one in all tabs.

The app displays the Login View Controller modally.

**#153 Take Multiple Choice Quiz**

**Description:**

As a developer I will like to create a multiple choice manner so that the student can see a question and selected from the 4 possible answers.

**Acceptance Criteria:**

The student can select the multiple choice button

The Multiple Choice button display a new TakeQuizWindow

The student see an image

The student see a question

The student select and answer, and can submit the answer for review

When the student submit all the answers, the Window displays a grade.

**#154 Take Fill in the Blank Quiz**

**Description:**

As a developer I will like to create a fill in the blank  manner so that the student can see a question and type a possible answers.

**Acceptance Criteria:**

The student can select the Fill in the Blank button

The Fill in the blank button display a new TakeQuizWindow

The student see an image

The student see a question

The student see an empty text field. The student types an answer. Then the student can submit the answer for review

When the student submit all the answers, the Window displays a grade.

**#155 Take a Random Quiz**

**Description:**

As a developer I will like to create a random quiz that will have multiple choice and fill in the blank  manner so that the student can see a question and selected from the 4 possible answers or type the possible answer, depending of the type of quiz.

**Acceptance Criteria:**

The student can select the Random button

The Fill in the blank button display a new TakeQuizWindow

The student see an image

The student see a question

The student see an empty text field or a multiple choice answers. The student types an answer or select an answer. Then the student can submit the answer for review

When the student submit all the answers, the Window displays a grade.

**#156 Add search capability for the terminology list**

**Description:**

As a student, I would like to be able to search for a specific term so that I can look for a specific term in order to work faster

**Acceptance Criteria:**

The TableView updates its content as the student types in the search field

If there's no term that matches the student input, display an empty table view with a message

If there's one or more terms that matches the student's input, display them in the TableView

**#161 Customize Terminology DetailViewController**

**Description:**

As a student I would like to quickly access the list of all the terminology for all the stations in all Labs so that I can have all the terminology for all the stations in one view.

**Acceptance Criteria:**

The DetailViewController should display a hierarchy of terms and different ways to organize such hierarchy

The DetailViewController should allow the student search for terms by entering search term.

The student should be able to filter the terms per lab.

**#173 Make the quizzes question and answers random**

**Description:**

As a teacher I want to make the quizzes more random so that the student don’t see a pattern in the questions, and every time that they open the quiz, they will see something different.

**Acceptance Criteria:**

The questions appear in  a random fashion

The possible answer appears in a random fashion

The correct answer is always included in the possible answers

All the labels or terms are asked during the quiz

**#174 Fix data for videos in the Lab 1**

**Description:**

As a developer I want to fix the video data in the Lab 1 so that the videos displayed in the video view for all the stations in Lab 1 are accurate

**Acceptance Criteria:**

The Json have all the videos for the 6 stations in Lab 1

All the videos play in all the stations

**#175 Fix quiz data for Lab 1**

**Description:**

As a developer I want to fix the data for the quizzes in Lab 1 so that the users can take an accurate quiz for all stations in the Lab 1

**Acceptance Criteria:**

All the quizzes in the Lab 1 stations display the correct images

All the quizzes in the Lab 1 display the correct questions

All the quizzes in the Lab 1 display all the correct terms

**#176 Personalize quiz settings**

**Description:**

As a student I want to change the settings of my quizzes so that I can take quizzes with time limits, quizzes for specific stations or labs.

**Acceptance Criteria:**

The student can change the time of the quiz.

The student can select to take the quiz of one station or multiple stations in one quiz.

The student can take the quiz for one lab or multiple labs in one quiz.

The student can choose to take it multiple choice, fill in the blank or both.

**#178 Create Take Quiz Tab View**

**Description:**

As a developer I want to fix the Quiz tab so that when the student clicks on it, they see a quiz setting option.

**Acceptance Criteria:**

The student clicks on the Quiz tab and the view changes displaying the quiz settings.

The view have a description of the View.

The student can click on the settings to get the view to change it.

If the student taps the Quiz tab again, the same view should be displayed.

**#184 Zoom in the pictures of the terms**

**Description:**

As a student I want to zoom in on the pictures of the term so that I can see with better detail the image.

**Acceptance Criteria:**

The student touch the image and the image open itself in another view.

The student can zoom in and out of the image

The student can close the view.

**#185 Zoom in a picture during a** quiz

**Description:**

As a student I want to touch a photo and open it in a separate view so that I can zoom in and out the picture.

**Acceptance Criteria:**

The student can touch the picture, and the picture will open in a different view.

The student can zoom in and out of the picture.

The student can close the separate view and return to the quiz.

The timer of the quiz keeps running in the zoom in picture.

**#186 Update student profile view to reflect changes in database**

**Description:**

As a teacher I would like my students to be able to check their progress in the user profile tab so they can determine which knowledge area they need to reinforce.

**Acceptance Criteria:**

The user profile tab displays information about the quizzes taken, quizzes grade, average.

The user profile tab must always show up to date information retrieved from the database.

**#187 Add full data on stations for Lab 1**

**Description:**

As a teacher I would like to have all the information for the stations in Lab 1 so that my students can start testing the application.

**Acceptance Criteria:**

Stations from Lab 1 display their respective information.

All stations should update the detail view controller with its own list of terms and images.

**Pending User Stories**

**#177 Save student's information in a local database**

**Description:**

As a developer I would like the app to save the student's information, such as quizzes taken, quizzes grades and general progress in a database so I can track of my progress.

**Acceptance Criteria:**

The app saves the student's information on a local database.

On log out, the app keeps all the student's data in the persistent framework.

**#179 Display a student profile view**

**Description:**

As a teacher I would like my students to have a profile view so they can have a general view of their progress, quizzes, grades, etc.

**Acceptance Criteria:**

The app displays a profile tab in the tab controller

The profile page contains the information of the student's progress

The profile page loads the information from the database structure

**Project Plan**

The main goal of the initial version of Technology Driven Active Learning in Human Laboratory was to implement a base product with the core features required to be able to serve as an interactive and media-rich lab manual. The main features allow students to review course materials and assess their knowledge. Being the first iteration of the app, future improvements and features need to be added.

The goal of this version was to develop an almost fully functional application, that would serve as a complementary tool in the teaching-learning process of both teacher and student. Future improvements upon this project should include features that truly allow the app to be a tool that improves the student's engagement and enriches their learning experience.

**Hardware and Software Resources**

***Hardware***

In order to develop and work with the application the student needs the following hardware:

* macOS powered computer, such as iMac, Macbook Pro, Macbook Air, Mac Mini, etc…
* An iPad to test on device (optional, but desirable)

***Software***

* Developers will need to be proficient in *Swift* programming language. Swift is not hard to learn for newcomers, language is very readable, almost English-like. Plenty of documentation and tutorials are available online.
* Knowledge of design patterns.
* Prior knowledge of object oriented programming is also required.

**Deployment requirements**

To deploy test or release versions, developers need to setup an Apple developer account in order to be able to distribute the app through one of the several ways Apple offers. An institutional account is preferred.

For testing on devices a simple user account is enough.

The Google Sign In framework requires setting up the application in its platform. For further reference please visit: <https://developers.google.com/identity/sign-in/ios/>

Ideally, the device should be updated to the latest iOS version. Release target in Xcode should be set to the latest version or to the version your intended release wants to support.

**Sprints Plan**

***Sprint 1***

**#118 Project template with TabBarController**

**Description:**  
As a developer I want the app to displays Labs information, takes Quizzes and contains Terminology information so that the students can study it in class.

**Acceptance Criteria:**  
The app displays the three tabs (Labs, Quizzes, Terminology.

**Related Tasks:**

#128 Add documentation in code

#126 Update documentation

#117 Setup app project template

***Sprint 2***

**#119 Display the Labs list**

**Description:**

As a developer I want that when the student taps on Labs the app will display the Labs list in the left table so that my students can see a list of the labs that the class have.

**Acceptance Criteria:**

On tap on Labs tab, display the Labs list in the table

Set as default view

**#120 Display popup to select desired station**

**Description:**

As a developer I want that on tapping the selected lab in the menu, the app will display a popup with a list of lab stations so that the students can see the stations that the selected lab have.

**Acceptance Criteria:**

The app shows a popup with a list of stations for the selected lab

**Related Tasks:**

**#127 Display PopUp menu**

**#121 Load Lab 1 information**

**Description:**

As a developer I want to save the Lab 1 information in a static data structure inside the app bundle so that the data is saved in the application to be use in the view displays.

**Acceptance Criteria:**

The terminology will be divided by stations and each Lab will have several stations

**Related Tasks:**

**#122 Create the data model for Lab 1**

**#123 Display Take Quiz button**

**Description:**

As a developer I want to display a button with a "Take Quiz" label so that the students will be able to click it and move to the take quiz view.

**Acceptance Criteria:**

A button with a "Take Quiz" label is shown.

**#124 Display Video button**

**Description:**

As a developer I want to display a button with a "Video" label so that when the students clicks on them they will be move to the station video view.

**Acceptance Criteria:**

A button with a "Video" label is shown.

**#125 Display station information**

**Description:**

As a developer I want that when the student taps on a station of a given Lab, the app will display information of that specific station in the Detail View Controller so that the students can select a station from the popup and see the data that the station have.

**Acceptance Criteria:**

The Detail View Controller will display the given station information for the given Lab.

**#135 Create a Welcome Screen**

**Description:**

As a teacher I would like my students to see a Welcome Screen from where they can Sign in using their PantherMail account so that only the university students can login to the application.

**Acceptance Criteria:**

The app displays a welcome screen when the student is not logged in.

The welcome screen display a title and a "Login with your PantherMail account".

***Sprint 3***

**#129 Load and Play Station Videos**

**Description:**

As a teacher I will like my student to be able to click on the Video button of a selected station an see a list of available videos. Then the student may select a video, and that video will be display.

**Acceptance Criteria:**

The Video Button Shows a screen with a list of available videos for the selected station.

The student select a video, and the video plays.

**Related Tasks:**

**#152 Choose a Video from the POPUP**

**#139 Close the PopUp when the User click close or outside the screen**

**#138 Play Youtube Video**

**#137 Load Youtube video into a new view**

**#136 Create PopUp when the Video Button is touch**

**#130 Allow app login through PantherMail account**

**Description:**

As a teacher I would like my students to be able to login using their PantherMail credentials so they can access the information in the app.

**Acceptance Criteria:**

A student logs in successfully using the PantherMail account.

The app allows the student to access the Labs information.

The app won't allow access to users signing in with accounts with domains other than "fiu.edu"

The app handles all the exceptions thrown by the authentication framework.

**Related Tasks:**

**#162 Add Login documentation**

**#148 Create and store an User object in the device database**

**#147 Restrict allowed user accounts to be in the "fiu.edu" domain**

**#140 Integrate Google/Sign-In pod in project**

**#131 Display the Take Quiz View**

**Description:**

As a teacher I will like for my student to press the Take Quiz button an display the TakeQuiz window for the station that the student is currently in.

**Acceptance Criteria:**

The button TakeQuiz display a new window

The window contain a short description of the station for the quiz selected

The new window contain three button to select the type of Quiz (fill in the blank, multiple choice or random)

**Related Tasks:**

**#141 Create the Take Quiz View**

**#132 Load full station information in the detail view**

**Description:**

As a teacher I would like the students to have a full view with the information of the selected station so they can learn and practice the lab.

**Acceptance Criteria:**

The detail view is populated with the list of terms

On tapping a term in the list, the app should update the ImageView next to the terminology list and show an image related to the term.

**Related Tasks:**

**#163 Add Load Station detail documentation**

**#151 Add a ImageView to show an image containing the term selected by the user**

**#150 Add a TableView**

**#149 Add images for Station 1 of Lab 1**

**#142 Add station terminology list to detail view**

**#134 Allow the users to sign out from the app**

**Description:**

As a teacher I would like the app to allow the students to sign out from the app so that they can close their session.

**Acceptance Criteria:**

The app cleans all the information about the logged in student.

The app switches the detail view controller to the first one in all tabs.

The app displays the Login View Controller modally.

**Related Tasks:**

**#146 Reverse all detail view controllers to the first screen**

**#145 Clean user information on logout confirmation**

**#144 Show popup after the user taps the Logout button**

**#143 Add logout button on the top right corner of the app's detail view**

***Sprint 4***

**#133 Disable the Logout option while taking a Quiz**

**Description:**

As a teacher I will like to cancel the logout option while the student is taking a quiz so that the students can’t interrupt the quiz.

**Acceptance Criteria:**

The student is taking a quiz, and the logout button is disable.

The student can logout after finish the quiz.

**#153 Take Multiple Choice Quiz**

**Description:**

As a developer I will like to create a multiple choice manner so that the student can see a question and selected from the 4 possible answers.

**Acceptance Criteria:**

The student can select the multiple choice button

The Multiple Choice button display a new TakeQuizWindow

The student see an image

The student see a question

The student select and answer, and can submit the answer for review

When the student submit all the answers, the Window displays a grade.

**Related Tasks:**

**#171 Finish Quiz**

**#170 Change Picture**

**#169 Update Grade**

**#168 Check Answer**

**#167 Display Possible Answers**

**#166 Display Picture**

**#165 Display Question**

**#164 Display Multiple Choice View**

**#154 Take Fill in the Blank Quiz**

**Description:**

As a developer I will like to create a fill in the blank  manner so that the student can see a question and type a possible answers.

**Acceptance Criteria:**

The student can select the Fill in the Blank button

The Fill in the blank button display a new TakeQuizWindow

The student see an image

The student see a question

The student see an empty text field. The student types an answer. Then the student can submit the answer for review

When the student submit all the answers, the Window displays a grade.

**Related Tasks:**

**#172 Create Text Field**

**#155 Take a Random Quiz**

**Description:**

As a developer I will like to create a random quiz that will have multiple choice and fill in the blank  manner so that the student can see a question and selected from the 4 possible answers or type the possible answer, depending of the type of quiz.

**Acceptance Criteria:**

The student can select the Random button

The Fill in the blank button display a new TakeQuizWindow

The student see an image

The student see a question

The student see an empty text field or a multiple choice answers. The student types an answer or select an answer. Then the student can submit the answer for review

When the student submit all the answers, the Window displays a grade.

**#156 Add search capability for the terminology list**

**Description:**

As a student, I would like to be able to search for a specific term so that I can look for a specific term in order to work faster

**Acceptance Criteria:**

The TableView updates its content as the student types in the search field

If there's no term that matches the student input, display an empty table view with a message

If there's one or more terms that matches the student's input, display them in the TableView

**Related Tasks:**

#159 Add a view on empty TableView to guide users

#158 Filter terms TableView content when the user inputs text on the search bar

#157 Add UISearchController on top of the terms TableView

***Sprint 5***

**#173 Make the quizzes question and answers random**

**Description:**

As a teacher I want to make the quizzes more random so that the student don’t see a pattern in the questions, and every time that they open the quiz, they will see something different.

**Acceptance Criteria:**

The questions appear in a random fashion

The possible answer appears in a random fashion

The correct answer is always included in the possible answers

All the labels or terms are asked during the quiz

**#176 Personalize quiz settings**

**Description:**

As a student I want to change the settings of my quizzes so that I can take quizzes with time limits, quizzes for specific stations or labs.

**Acceptance Criteria:**

The student can change the time of the quiz.

The student can select to take the quiz of one station or multiple stations in one quiz.

The student can take the quiz for one lab or multiple labs in one quiz.

The student can choose to take it multiple choice, fill in the blank or both.

**#178 Create Take Quiz Tab View**

**Description:**

As a developer I want to fix the Quiz tab so that when the student clicks on it, they see a quiz setting option.

**Acceptance Criteria:**

The student clicks on the Quiz tab and the view changes displaying the quiz settings.

The view have a description of the View.

The student can click on the settings to get the view to change it.

If the student taps the Quiz tab again, the same view should be displayed.

***Sprint 6***

**#161 Customize Terminology DetailViewController**

**Description:**

As a student I would like to quickly access the list of all the terminology for all the stations in all Labs so that I can have all the terminology for all the stations in one view.

**Acceptance Criteria:**

The DetailViewController should display a hierarchy of terms and different ways to organize such hierarchy

The DetailViewController should allow the student search for terms by entering search term.

The student should be able to filter the terms per lab.

**Related Tasks:**

**#191 Add an image view to display the image associated with a term**

**#190 Add a table view to hold a list of terms**

**#174 Fix data for videos in the Lab 1**

**Description:**

As a developer I want to fix the video data in the Lab 1 so that the videos displayed in the video view for all the stations in Lab 1 are accurate

**Acceptance Criteria:**

The Json have all the videos for the 6 stations in Lab 1

All the videos play in all the stations

**#175 Fix quiz data for Lab 1**

**Description:**

As a developer I want to fix the data for the quizzes in Lab 1 so that the users can take an accurate quiz for all stations in the Lab 1

**Acceptance Criteria:**

All the quizzes in the Lab 1 stations display the correct images

All the quizzes in the Lab 1 display the correct questions

All the quizzes in the Lab 1 display all the correct terms

**#184 Zoom in the pictures of the terms**

**Description:**

As a student I want to zoom in on the pictures of the term so that I can see with better detail the image.

**Acceptance Criteria:**

The student touch the image and the image open itself in another view.

The student can zoom in and out of the image

The student can close the view.

**#185 Zoom in a picture during a quiz**

**Description:**

As a student I want to touch a photo and open it in a separate view so that I can zoom in and out the picture.

**Acceptance Criteria:**

The student can touch the picture, and the picture will open in a different view.

The student can zoom in and out of the picture.

The student can close the separate view and return to the quiz.

The timer of the quiz keeps running in the zoom in picture.

**#186 Update student profile view to reflect changes in database**

**Description:**

As a teacher I would like my students to be able to check their progress in the user profile tab so they can determine which knowledge area they need to reinforce.

**Acceptance Criteria:**

The user profile tab displays information about the quizzes taken, quizzes grade, average.

The user profile tab must always show up to date information retrieved from the database.

**Related Tasks:**

**#188 Save student station review information**

**#187 Add full data on stations for Lab 1**

**Description:**

As a teacher I would like to have all the information for the stations in Lab 1 so that my students can start testing the application.

**Acceptance Criteria:**

Stations from Lab 1 display their respective information.

All stations should update the detail view controller with its own list of terms and images.

**Related Tasks:**

**#189 Load stations info into .json file**

**System Design**

The Technology Driven Active Learning in Human Anatomy Laboratory Ver 1.0 is an iPad application built with the MVVM architectural pattern in mind. For this initial version, all the application data is stored locally on the device in a text file with JSON format. A small, local and limited relational database is used in order to store some basic information about the student and its progress through the course materials.

**Architectural Patterns**

MVVM was the chosen architectural pattern to develop the application. MVVM stands for Model-View-Viewmodel. It was developed by Microsoft and aims to simplify event-driven development of user interfaces. Being a mobile application, events are a major component and have to be dealt with accordingly. MVVM is derived from the widely known and used MVC (Model-View-Controller).

In iOS development, the use of MVC pattern tends to produce highly bloated controllers, with a lot of responsibilities and sometimes high coupling with the view. MVVM seeks to reduce this controller overload, by adding a new viewmodel layer between the view and the model. The viewmodel responsibilities include, but are not limited to:

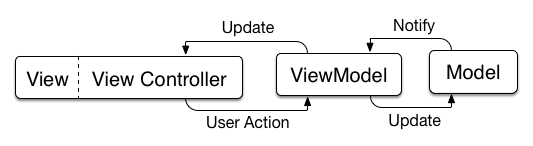
* Act as a value converter of data from the model to something that the view understands, thus relieving the controller or the view of this task.
* Contain properties that represent the state of the view at a certain moment.
* It is much easier to write tests for ViewModels than for Views. Testable without awkward UI automation and interaction.****

Figure 1 - MVVM design

**System and Subsystem Decomposition**

The application is divided into several subsystems, each one handling a critical business unit. In our case, this units are Labs, which are the main units of information. The course consist of six (6) Lab and each lab has six (6) stations. Each stations have several terms, images associated to those terms, videos and questions for the quizzes.

**Repository subsystem:** This subsystem handles everything related to loading and organizing the information from the app's main repository, namely the JSON file containing the courses materials, into easy to handle structures. Provides classes that readily retrieve and format the information as needed by all the subsystems that depend on it.

**Station subsystem:**  This subsystem organizes and display the course materials in an organized way and provides methods to load the different stations from the different labs.

**Quizzes subsystem:** This subsystem main function is to provide the students access to the app's knowledge assessment methods and its customization options.

**Terminology subsystem:** Acts as a consolidated knowledge base of all the terms from all the stations and all the labs. It provides methods for filtering through the entire database.

**Authentication subsystem:** Provides mechanisms to authenticate FIU students through their PantherMail account.

**User profile subsystem:** Handles everything related to the user in the application. Saving stats, displaying progress information, authentication operations.

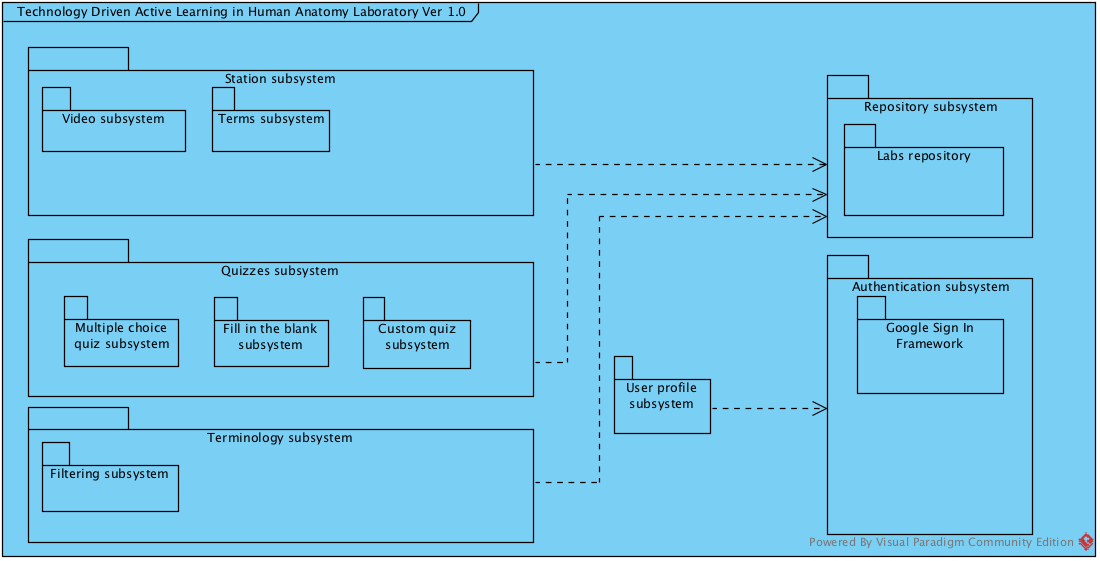


Figure 2 - Package diagram

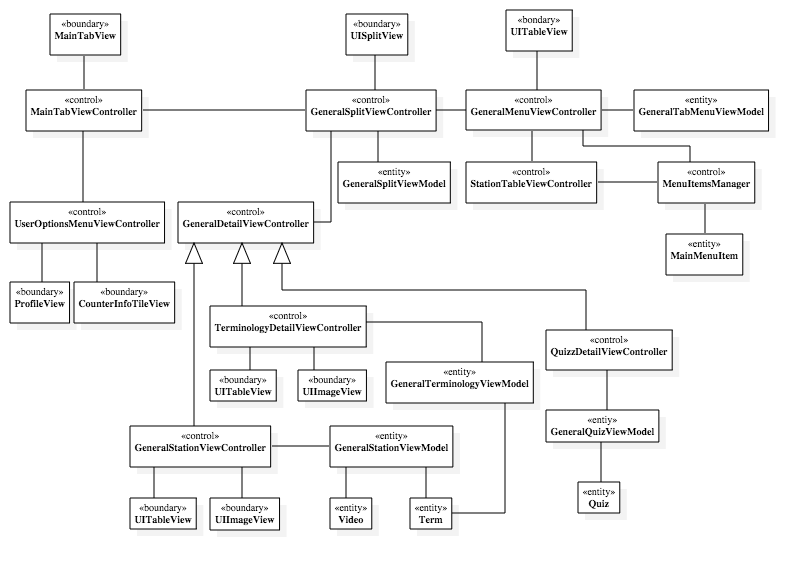
****

Figure 3 - Minimal class diagram

**Design Patterns**

Two particular patterns proved to be particularly useful during the development of the application:

* **Repository pattern**: This pattern was chosen due to the technical decision of storing the data in a file with JSON format. The knowledge of the persistence and the mapping from the elements in the file to actual data models are safely handled by the repository. The repository is comprised of the file with JSON format and the classes that handle the data loading and formatting.
* **Delegation pattern:** The delegation pattern is perhaps the most widely used by Objective-C and Swift. It is heavily adopted by Apple and developers for Apple platforms. It is a design pattern that shifts responsibilities from one class to another, thereby creating separation of concerns. In practice, delegation is most often used as a way for one class to communicate to another class.

**System Validation**

**#129 Load and Play Video User Story**

**Unit Test**

|  |
| --- |
| **Test case ID**: US\_129\_001 |
| **Test Title/Name**: Video Pop Up Display |
| **Test Description:** Test that when the station have more than one video, the system will call the video pop up view controller. |
| **Pre-condition**:  1-The system have more than 1 video.  2- The user clicks in the Video Button. |
| **Test Steps**:  1- The system check if there is more than one video. |
| **Expected Result**:  The system recognize the 2 videos, and displayed the Pop Up |
| **Actual result**: PASS |

**Integration Test**

|  |
| --- |
| **Test case ID**: US\_129\_002 |
| **Test Title/Name**: Display Video View of selected video from the Pop Over |
| **Test Description:** Test that when the user clicks on a video from the pop up video view, the window change to the video’s selected description. |
| **Pre-condition**:  1-The system have more than one video  2- The user clicked the Video Button.  3- The system displayed a pop up with a list of all the videos of the station. |
| **Test Steps**:  1- The user clicks on a video from the pop over |
| **Expected Result**:  The window change to display the video’s description of the video that the user selected.In this case Video 1. |
| **Actual result**: PASS |

**User Story #130 Allow app login through PantherMail account**

**Unit test**

|  |
| --- |
| **Test case ID**: US\_130\_001 |
| **Test Title/Name**: |
| **Test Description:**  Ensure that the student can login through FIU its assigned email address. |
| **Pre-condition**:  Student is logged out of the application. Login view is shown in the app. |
| **Test Steps**:  The user taps the "Login with your PantherMail account " button on the center of the login view controller. On the web view with Google Sign in flow, the user enters his / hers panther mail credentials. |
| **Expected Result**:  The login view controller gets dismissed and the app shows the detail view controller of the "Labs" tab, this is the default behavior. It also shows the user full name in the bottom right of the detail view controller. |
| **Actual result**: PASS |

**Integration test**

|  |
| --- |
| **Test case ID**: US\_130\_002 |
| **Test Title/Name**: |
| **Test Description:**  Ensure that the student cannot login through another Google account. |
| **Pre-condition**:  Student is logged out of the application. Login view is shown in the app. |
| **Test Steps**:  The user taps the "Login with your PantherMail account " button on the center of the login view controller. On the web view with Google Sign in flow, the user enters his / hers personal Gmail credentials. |
| **Expected Result**:  The app displays a popup warning the user that only accounts belonging to "fiu.edu" domain are allowed to authenticate in the application. |
| **Actual result**: PASS |

**User Story #132 Load full station information in the detail view**

**Unit test**

|  |
| --- |
| **Test case ID**: US\_132\_001 |
| **Test Title/Name**: |
| **Test Description:**  Ensure that the detail view gets updated when the students select a station from a Lab. |
| **Pre-condition**:  The table view with the Labs and Stations menu loaded all the information correctly. The user is logged in and is located in the "Labs" tab. |
| **Test Steps**:  The user taps the "Labs" button on the top left of the view controller, then select a Lab from the displayed table, then selects a station from the displayed table within a popup. |
| **Expected Result**:  The detail view controller should update its view and display a table view containing all the station terms in the left and a empty image view on the right. It should also display a button with a "Video" label and another with a "Take Quiz" label. |
| **Actual result**: PASS |

**Integration test**

|  |
| --- |
| **Test case ID**: US\_132\_002 |
| **Test Title/Name**: |
| **Test Description:**  Ensure that the detail view controller doesn't get updated if the user cancels the selection of anu station or any lab. |
| **Pre-condition**:  The table view with the Labs and Stations menu loaded all the information correctly. The user is logged in and is located in the "Labs" tab. |
| **Test Steps**:  The user taps the "Labs" button on the top left of the view controller, then select a Lab from the displayed table, then taps on the "Close" button on the top right in the displayed table within a popup. |
| **Expected Result**:  The popup window containing all the stations from the Lab gets dismissed and the detail view controller doesn't get updated |
| **Actual result**: PASS |

**User Story #153 Take Multiple Choice Quiz**

**Unit Test**

|  |
| --- |
| **Test case ID**: US\_153\_001 |
| **Test Title/Name**: Test Check Answer Func  for Correct Answer |
| **Test Description:** Test the Check Answer functionality for a correct selected answer. |
| **Pre-condition**:  1-The user selected an answer  2- The user clicked in the Check Answer Button |
| **Test Steps**:  1- The checkAnswerAction func is called  2- The CheckAnswerAction call the checkAnswer func  3- The checkAnswer func verify if the answer is correct  4- The func return its boolean result. |
| **Expected Result**:  The boolean returned by the checkAnswer function must be true |
| **Actual result**: PASS |

|  |
| --- |
| **Test case ID**: US\_153\_002 |
| **Test Title/Name**: Test Check Answer Func  for Incorrect Answer |
| **Test Description:** Test the Check Answer functionality for a incorrect selected answer |
| **Pre-condition**:  1-The user selected an answer  2- The user clicked in the Check Answer Button |
| **Test Steps**:  1- The checkAnswerAction func is called  2- The CheckAnswerAction call the checkAnswer func  3- The checkAnswer func verify if the answer is correct  4- The func return its boolean result. |
| **Expected Result**:  The boolean returned by the checkAnswer function must be false |
| **Actual result**: PASS |

**Integration Test**

|  |
| --- |
| **Test case ID**: US\_153\_003 |
| **Test Title/Name**: Test Answer Question |
| **Test Description:** Test the Check Answer functionality for a correct selected answer and when the user clicks on the Next button to change to a different question. |
| **Pre-condition**:  1-The user is in the Multiple Choice Quiz View |
| **Test Steps**:  1- The user clicks on an answer.  2- The user clicks on the Check Answer button.  3- The system verify the answer. |
| **Expected Result**:  The system display a “Correct message”, and the Next Button is displayed. |
| **Actual result**: PASS |

|  |
| --- |
| **Test case ID**: US\_153\_004 |
| **Test Title/Name**: Test Answer Question |
| **Test Description:**Test the Check Answer functionality for a Incorrect selected answer. |
| **Pre-condition**:  1-The user is in the Multiple Choice Quiz View |
| **Test Steps**:  1- The user clicks on an answer.  2- The user clicks on the Check Answer button.  3- The system verify the answer. |
| **Expected Result**:  The system display an “Incorrect message”. |
| **Actual result**: PASS |

|  |
| --- |
| **Test case ID**: US\_153\_005 |
| **Test Title/Name**: Test Next Button |
| **Test Description:**Test the Next Button functionality for displaying a new question with new possible answers. |
| **Pre-condition**:  1- The user is in the Multiple Choice Quiz View.  2- The user selected an answer.  3- The answer selected is the correct one.  4- The button Next is displayed. |
| **Test Steps**:   * The system check if there is more question to ask.   + If there are more questions to ask: * The system call the setupView func and the reload data. * The new question data is send to the tableview and the quesiton label   + If there are no more questions to ask: * The system call the Final Grade View |
| **Expected Result**:  The system display Next Question, and Next Possible Answers |
| **Actual result**: PASS |

|  |
| --- |
| **Test case ID**: US\_153\_006 |
| **Test Title/Name**: Test the Next Button functionality for displaying the Final Grade View |
| **Test Description:**Test the Next Button functionality for displaying a new question with new possible answers. |
| **Pre-condition**:  1- The user is in the Multiple Choice Quiz View.  2- The user selected an answer.  3- The answer selected is the correct one.  4- The button Next is displayed.  5- The user is in the last question of the quiz. |
| **Test Steps**:  1- The system check if there is more question to ask.   1. If there are more questions to ask:    1. The system call the setupView func and the reload data.    2. The new question data is send to the tableview and the quesiton label 2. If there are no more questions to ask:    1. The system call the Final Grade View |
| **Expected Result**:  The system display change to the Final Grade View. |
| **Actual result**: PASS |

**User Story #156 Add search capability for the terminology list**

**Unit test**

|  |
| --- |
| **Test case ID**: US\_156\_001 |
| **Test Title/Name**: Test Search Bar |
| **Test Description:**  Ensure that the table view filters the results according to the user input in the search bar. |
| **Pre-condition**:   1. The table view loaded all the terms correctly. 2. The user is logged in and selected a lab and a station from that lab. |
| **Test Steps**:   1. The user enters, through the device's onscreen keyboard, the letters, keywords, terms or words to search for. |
| **Expected Result**:  The table view should update its data as the user enter characters. If not single term is found to match the input by the user, then the table view should be displayed empty. If one or more terms match, then the table view should only show those terms. |
| **Actual result**: PASS |

|  |
| --- |
| **Test case ID**: US\_156\_002 |
| **Test Title/Name**: Test Search Bar for special characters. |
| **Test Description:**  Ensure that the search bar on top of the table view doesn't accept special characters or numbers, just a character in the set of lower and uppercase alphabet letters, the characters '/' and blank space. |
| **Pre-condition**:   1. The table view loaded all the terms correctly. 2. The user is logged in and selected a lab and a station from that lab. |
| **Test Steps**:   1. The user tries to enter any non-allowed character through the device's onscreen keyboard. |
| **Expected Result**:   1. The search bar doesn't allow the character to be entered. 2. No change is made to the table view or its content. |
| **Actual result**: PASS |

**User Story #176 Personalize quiz settings**

**Unit Test**

|  |
| --- |
| **Test case ID**: US\_176\_001 |
| **Test Title/Name**:  Change the time for the quiz |
| **Test Description:**Change the time for the quiz. From 60 mins to 20 mins. |
| **Pre-condition**:  1-The user is in logging with a correct user name  2- The user clicks on the Quiz Button in the main tab bar. |
| **Test Steps**:  The user clicks in the Slider, and move it down until the number display is 20. |
| **Expected Result**:  The time variable is set to 20 |
| **Actual result**: PASS |

|  |
| --- |
| **Test case ID**:  US\_176\_002 |
| **Test Title/Name**:  Don’t change the time for the quiz. |
| **Test Description:** Change the time from 60 to 20, then click in the logout button. |
| **Pre-condition**:  1-The user is in logging with a correct user name  2- The user clicks on the Quiz Button in the main tab bar. |
| **Test Steps**:   1. The user clicks in the Slider, and move it down until the number display is 20. 2. The user click in the Logout button. A message display. |
| **Expected Result**:  The user is logout. The time is not save. |
| **Actual result**: PASS |

|  |
| --- |
| **Test case ID**:  US\_176\_003 |
| **Test Title/Name**:  Select Lab 1 for the quiz. |
| **Test Description:**  Select Lab 1 for the quiz. |
| **Pre-condition**:  1-The user is in logging with a correct user name  2- The user clicks on the Quiz Button in the main tab bar.  3- The user clicks on the Select Labs button. |
| **Test Steps**:   1. The user clicks in the Lab 1. 2. The user click in the Save Button. A pop Message appears. The message asks the user if he/she wish to alter the stations. 3. The user click in the No button. |
| **Expected Result**:  The labSelected array contain a new element. |
| **Actual result**: PASS |

|  |
| --- |
| **Test case ID**:  US\_176\_004 |
| **Test Title/Name**: Select Stations 1 and 2 from lab 1 for the customize quiz. |
| **Test Description:**  The user selected Lab 1, but wishes to only be testes in station 1 and 2. |
| **Pre-condition**:  1-The user is in logging with a correct user name  2- The user clicks on the Quiz Button in the main tab bar.  3- The user clicks on the Select Labs button. |
| **Test Steps**:   1. The user clicks in the Lab 1. 2. The user click in the Save Button. A pop up message appears. The message asks the user if he/she wish to alter the stations. 3. The user click in the Yes button. 4. The user selects Station 1 and Station 2 5. The user clicks the Save Button. A pop up message appears asking the user if he wishes to continue with the selected stations. 6. The student clicks Yes |
| **Expected Result**:  The labSelected array contain a new element. The stationSelected contain 2 elements. |
| **Actual result**: PASS |

|  |
| --- |
| **Test case ID**:  US\_176\_005 |
| **Test Title/Name**:  Logout while selecting stations. |
| **Test Description:**  Select stations 1 and 2 for lab 1. Instead of clicking on Save Button, the user clicks on Logout. |
| **Pre-condition**:  1-The user is in logging with a correct user name  2- The user clicks on the Quiz Button in the main tab bar.  3- The user clicks on the Select Labs button. |
| **Test Steps**:   1. The user clicks in the Lab 1. 2. The user click in the Save Button. A pop Message appears. The message asks the user if he/she wish to alter the stations. 3. The user click in the Yes button. The Station View is displayed 4. The user selects stations 1 and 2 5. The user clicks the logout button. A confirmation message is displayed. The user clicks Yes. |
| **Expected Result**:  The labSelected array and the stationsSelected array is empty. |
| **Actual result**: PASS |

**Integration Test**

|  |
| --- |
| **Test case ID**:  US\_176\_006 |
| **Test Title/Name**: Create a Multiple Choice quiz. |
| **Test Description:**  The user selects to take a quiz for 20 minutes, that will include only term questions and it will have both: A multiple choice and fill in the blank. The quiz will be for lab 1 station 1 and 2 only. |
| **Pre-condition**:  1-The user is in logging with a correct user name  2- The user clicks on the Quiz Button in the main tab bar. |
| **Test Steps**:   1. The user slide the timer from 60 to 20. 2. The user clicks in the term segment. 3. The user clicks in the Select Lab button. The view changes to the select labs view. 4. The user clicks in Lab1. 5. The user clicks in the Save Button. A message box appears asking the user if he wish to change the stations for the quiz. 6. The user clicks Yes. The view changes to the select Station View. 7. The user selects station 1 and 2. 8. The user clicks on the Save Button. A confirmation message appears. Asking the user if he/she wish to user the selected stations for the quiz 9. The user clicks the Yes button. The view changes to the Customize the Quiz View. Now a table is fill with the lab and stations selected. The user clicks on the Make it Multiple Choice button. |
| **Expected Result**:  The quiz is for 20 minutes, the question are only for stations 1 and 2 of the lab 1. The type of question is a term and the possible answers are the label of the term. |
| **Actual result**: PASS |

|  |
| --- |
| **Test case ID**:  US\_176\_007 |
| **Test Title/Name**:  Change Tabs while creating a Multiple Choice quiz. |
| **Test Description:**  The user selects to take a quiz for 20 minutes, that will include only terms questions and it will have both: A multiple choice and fill in the blank. The quiz will be for lab 1 station 1 and 2 only. Before pressing the Make it Multiple Choice Only Button the user changes tab to the lab tab in the button of the screen. |
| **Pre-condition**:  1-The user is in logging with a correct user name  2- The user clicks on the Quiz Button in the main tab bar. |
| **Test Steps**:   1. The user slide the timer from 60 to 20. 2. The user clicks in the term segment. 3. The user clicks in the Select Lab button. The view changes to the select labs view. 4. The user clicks in Lab1. 5. The user clicks in the Save Button. A message box appears asking the user if he wish to change the stations for the quiz. 6. The user clicks Yes. The view changes to the select Station View. 7. The user selects station 1 and 2. 8. The user clicks on the Save Button. A confirmation message appears. Asking the user if he/she wish to user the selected stations for the quiz 9. The user clicks the Yes button. The view changes to the Customize the Quiz View. Now a table is fill with the lab and stations selected. The user clicks on the Lab Tab Button. 10. The view changes to the Lab Tab view. |
| **Expected Result**:  The view changes to the Lab tab view. |
| **Actual result**: PASS |

**User Story #185  Zoom in a picture during a quiz**

**Unit Test**

|  |
| --- |
| **Test case ID**: US\_185\_001 |
| **Test Title/Name**:   Test Zoom in Button in Multiple Choice Quiz View |
| **Test Description:**  Test that the Zoom Button call another zoom imageview when it is pressed. |
| **Pre-condition**:  1- The user is log in  2- The user is in the Multiple Choice Quiz View. |
| **Test Steps**:  1- The user clicks on the zoom button  2- The system call the function zoomImageAction()  3- The system call the self.segueWithIdentificator (“zoomImageViewController”)  4- The system display the Zoom ImageView. |
| **Expected Result**:  The system is in the zoom imageview. |
| **Actual result**: PASS |

|  |
| --- |
| **Test case ID**: US\_185\_002 |
| **Test Title/Name**:  Test Zoom in Button in Fill in the Blank Quiz View. |
| **Test Description:**  Test that the Zoom Button call another zoom imageview when it is pressed. |
| **Pre-condition**:  1- The user is log in  2- The user is in the Fill in the Blank Quiz View. |
| **Test Steps**:  1- The user clicks on the zoom button  2- The system call the function zoomImageAction()  3- The system call the self.segueWithIdentificator (“zoomImageViewController”)  4- The system display the Zoom ImageView. |
| **Expected Result**:  The system is in the zoom imageview. |
| **Actual result**: PASS |

|  |
| --- |
| **Test case ID**: US\_185\_002 |
| **Test Title/Name**:   Test Zoom the cancel button in the Multiple Choice Quiz View |
| **Test Description:**  Test that the cancel Button call the segue function. |
| **Pre-condition**:  1- The user is log in  2- The user is in the Multiple Choice  Quiz View. |
| **Test Steps**:  1- The user clicks on the zoom button  2- The system call the function zoomImageAction()  3- The system call the self.segueWithIdentificator (“zoomImageViewController”)  4- The system display the Zoom ImageView.  5- The user press the cancel button.  6- The system call the func cancelAction ()  7- The system check if isQuiz == true  8- The system call the unwidToQuizView |
| **Expected Result**:  The system returns to the Multiple Choice Quiz View. |
| **Actual result**: PASS |

|  |
| --- |
| **Test case ID**: US\_185\_003 |
| **Test Title/Name**:  Test Zoom the cancel button in the Fill in the Blank Quiz View |
| **Test Description:** Test that the cancel Button call the segue function. |
| **Pre-condition**:  1- The user is log in  2- The user is in the Fill in the Blank  Quiz View. |
| **Test Steps**:  1- The user clicks on the zoom button  2- The system call the function zoomImageAction()  3- The system call the self.segueWithIdentificator (“zoomImageViewController”)  4- The system display the Zoom ImageView.  5- The user press the cancel button.  6- The system call the func cancelAction ()  7- The system check if isQuiz == true  8- The system call the unwidToQuiz |
| **Expected Result**:  The system returns to the Fill in the Blank Quiz View. |
| **Actual result**: PASS |

**Integration Test**

|  |
| --- |
| **Test case ID**: US\_185\_004 |
| **Test Title/Name**: Test Image Zoom In. |
| **Test Description:**  Test the Zoom in when the user pinch the screen. |
| **Pre-condition**:  1-The user is in the zoom imageview. |
| **Test Steps**:  1- The user pinches the image. |
| **Expected Result**:  The image is zoom in. |
| **Actual result**: PASS |

|  |
| --- |
| **Test case ID**: US\_185\_005 |
| **Test Title/Name**:   Test the timer after the zoom is pressed in the Multiple Choice View. |
| **Test Description:**  Test the if the timer stays running despite been in the zoom image view. |
| **Pre-condition**:  1-The user is in the Multiple Choice View. |
| **Test Steps**:  1- The user clicks the zoom button.  2- The system displays the zoom image view.  3- The user waits a minute in the zoom image view.  4- The user press the cancel button.  5- The system displays the multiple choice quiz view. |
| **Expected Result**:  The timer is still running, one minute less than before pressing the zoom button. |
| **Actual result**: PASS |

**Glossary**

**View**: The screen that is displaying the iPad.

**View Controller**: The class in charge of controlling the view of that is displaying the iPad.

**Pinch**: Action on an iPad where the user touches the image with two fingers, and by closing them, the image zoom in.

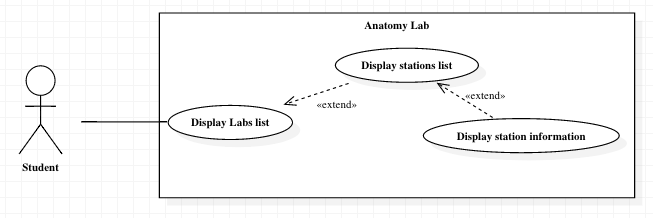
**Tab**: The buttons in the bottom bar of the application.

**Appendix**

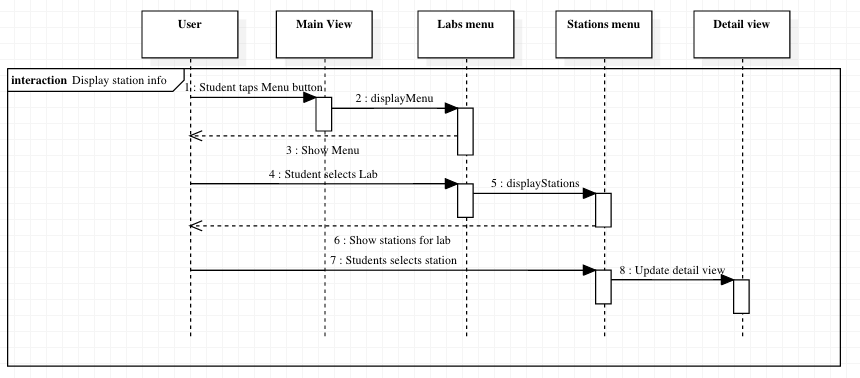
**Appendix A - UML Diagrams**

**User Story #119 #125 Display station information**

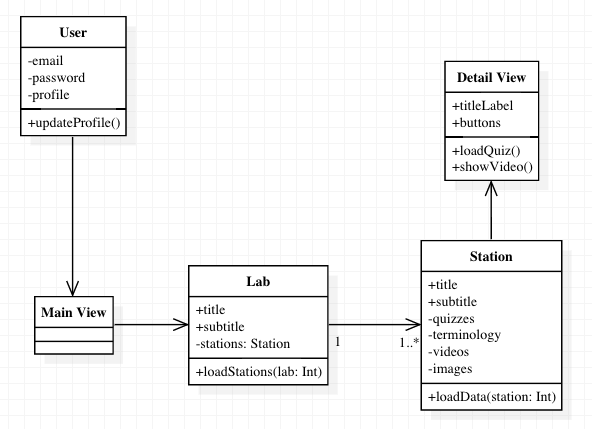
**Use Case Diagram**



**Sequence Diagram**

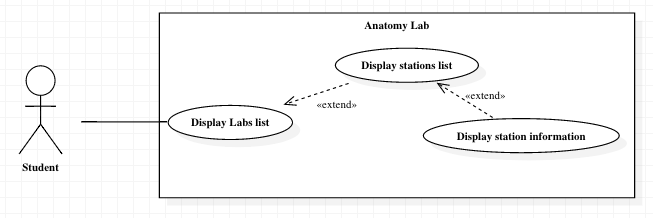


**Class Diagram**

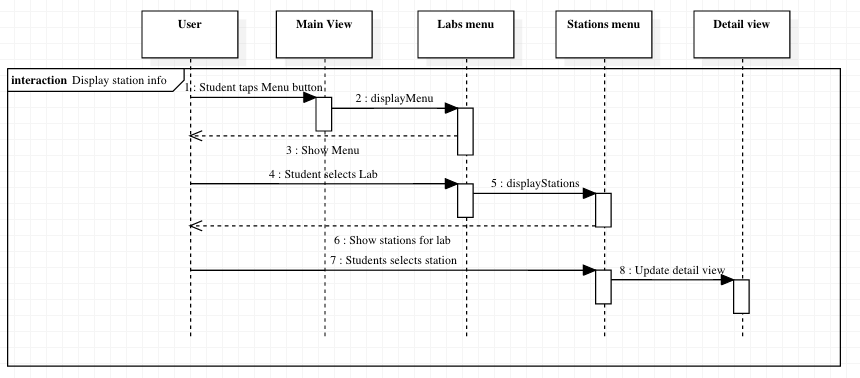


**User Story #120 Display popup to select desired station**

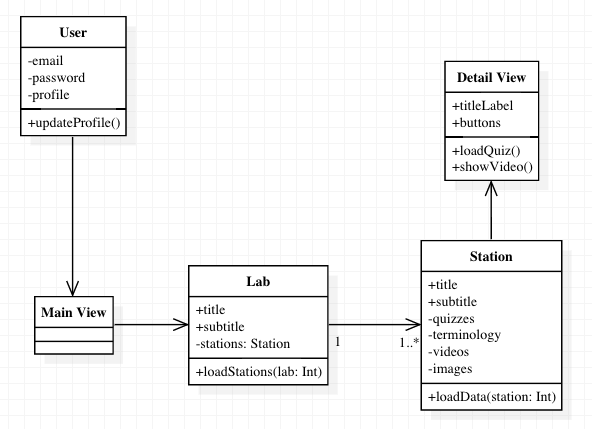
**Use Case Diagram**



**Sequence Diagram**

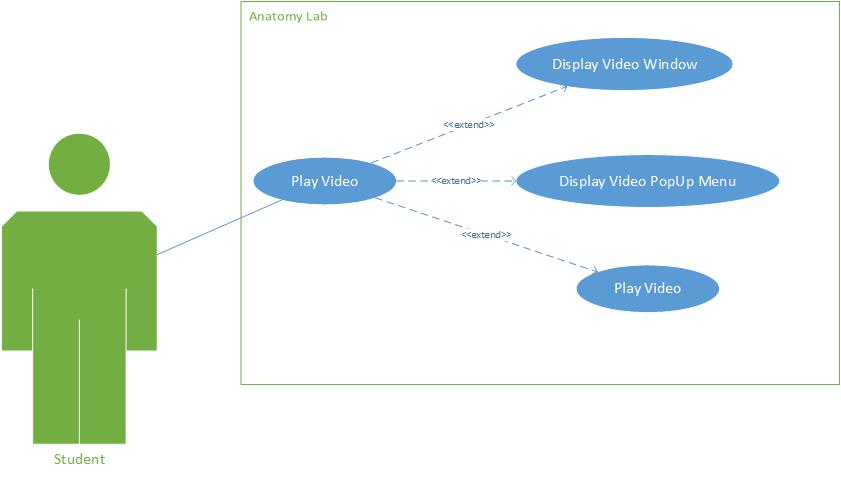


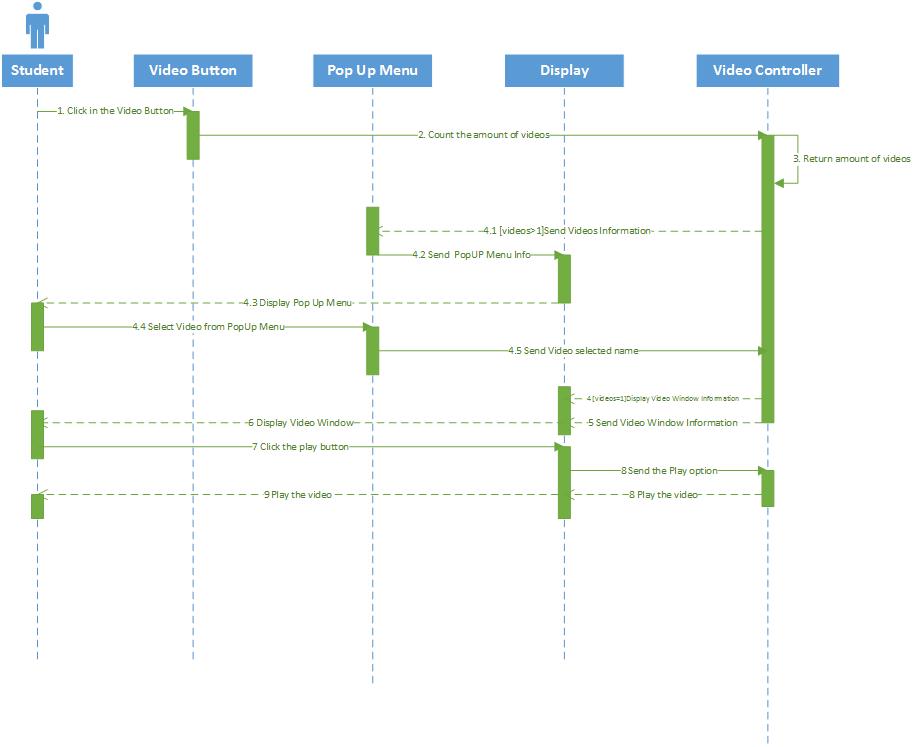
**Class Diagram**

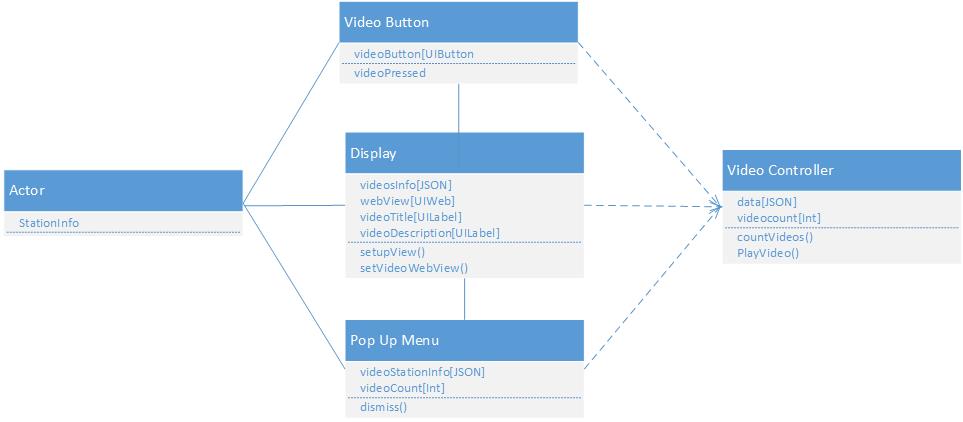


**User Story #129 Load and Play Video**

**Use Case Diagram**

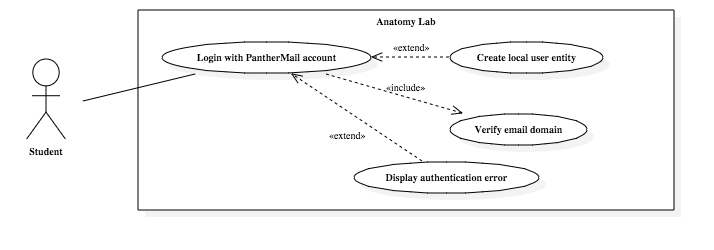


**Sequence Diagram**

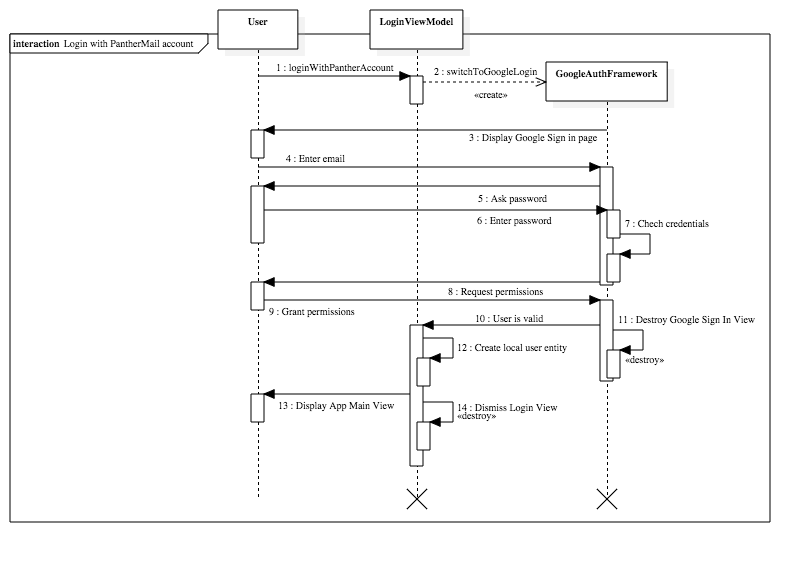
**Class Diagram**

**User Story #130 Allow app login through PantherMail account**

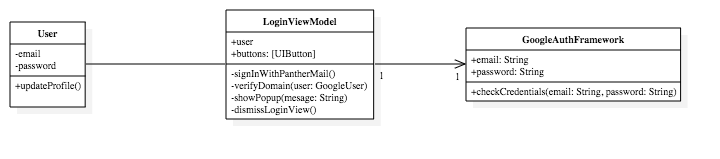
**Use Case Diagram**



**Sequence Diagram**

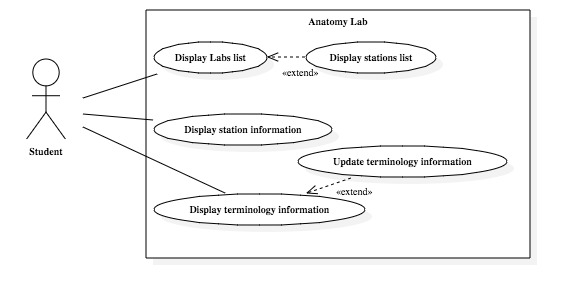


**Class Diagram**

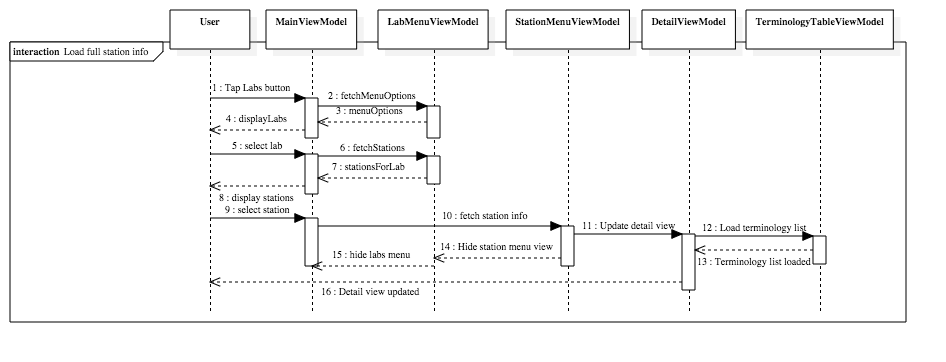
****

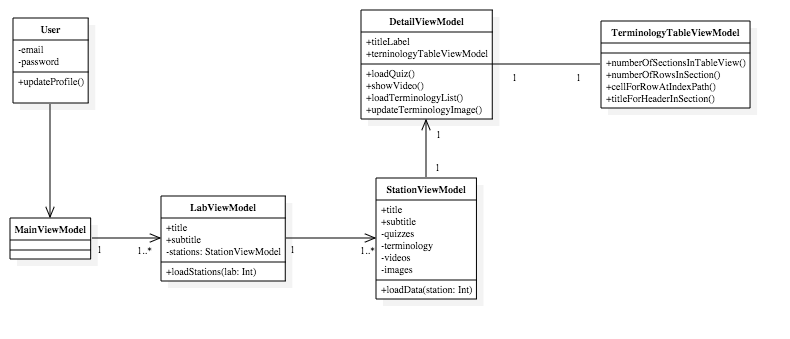
**User Story #132 Load full station information in the detail view**

**Use Case Diagram**



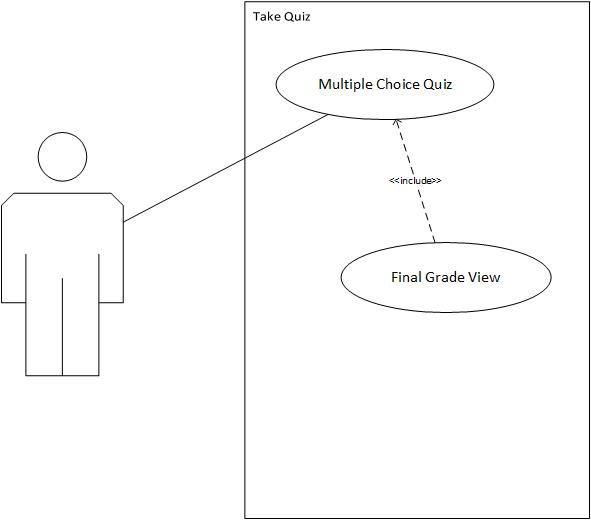
**Sequence Diagram**

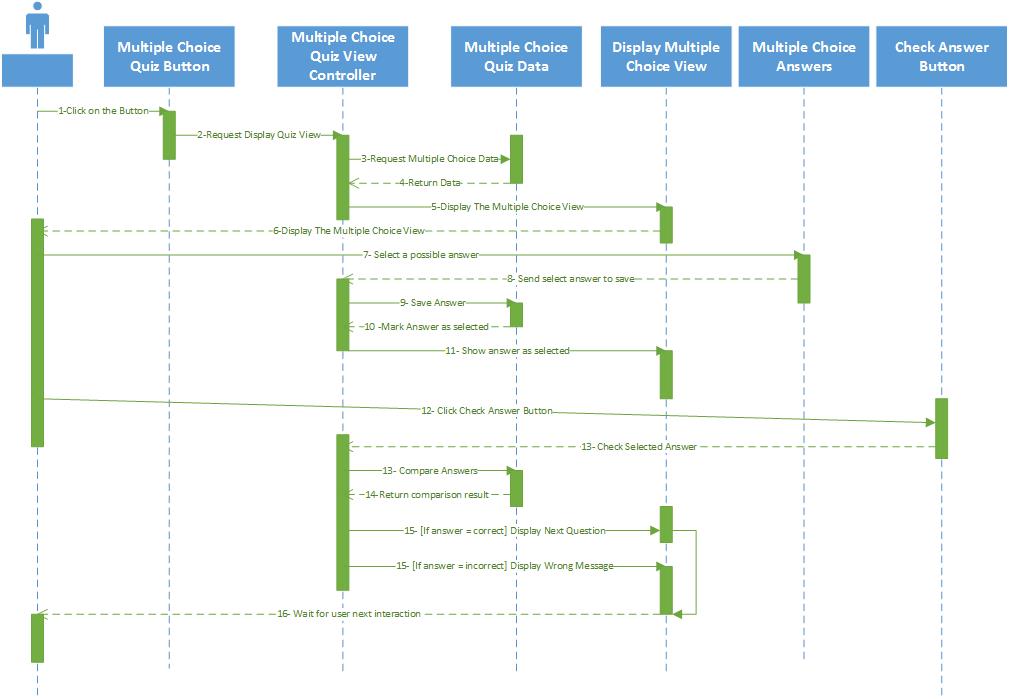


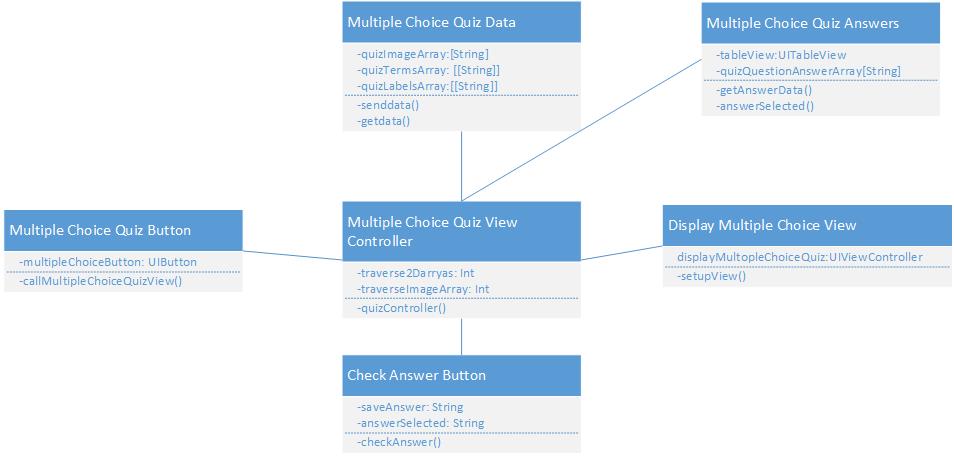
**Class Diagram**

**User Story #153 Take Multiple Choice Quiz**

**Use Case Diagram**

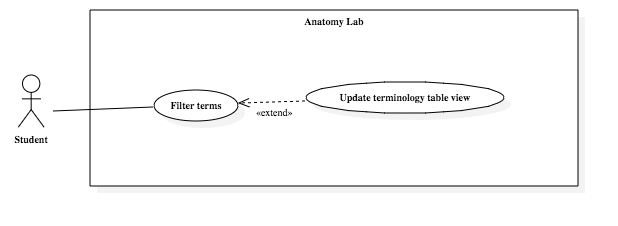


**Sequence Diagram**

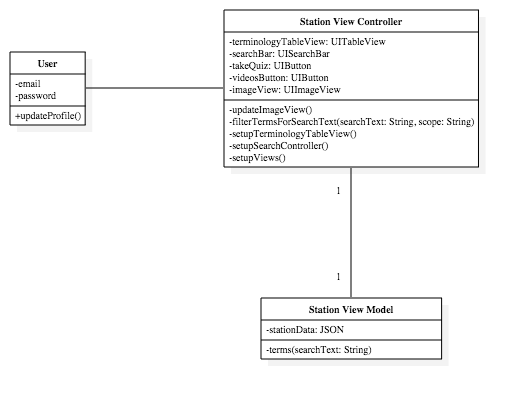
**Class Diagram**

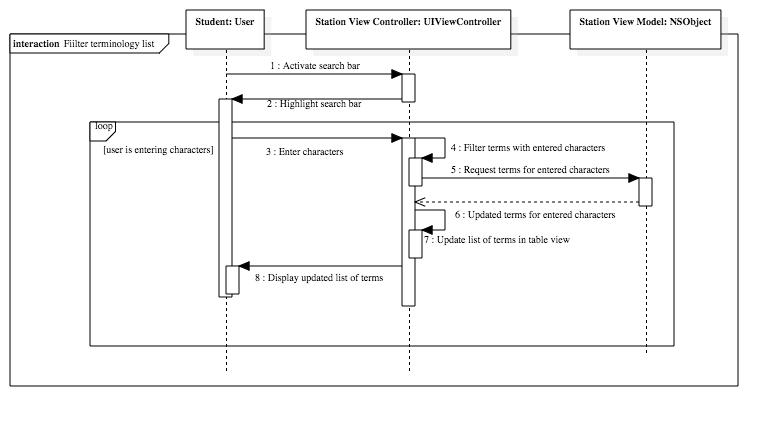
**User Story #156 Add search capability for the terminology list**

**Use Case Diagram**



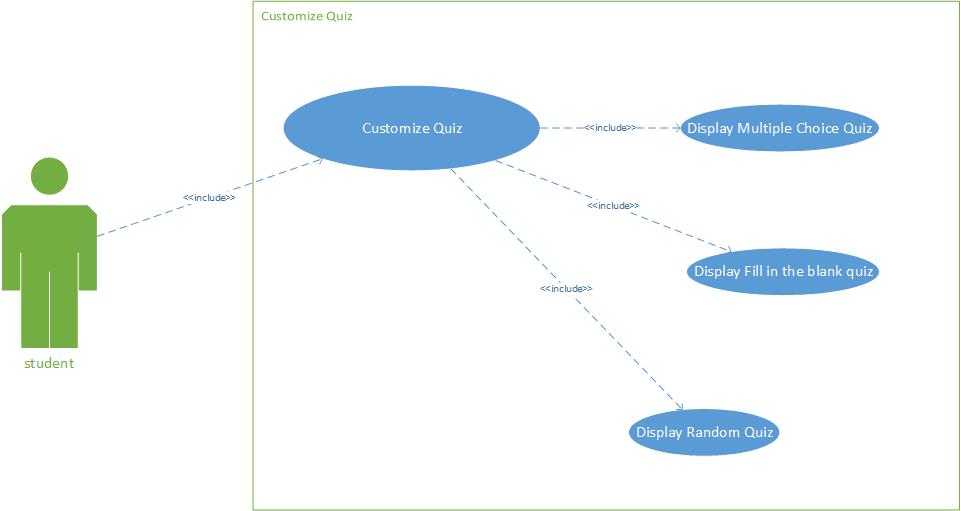
**Class Diagram**

****

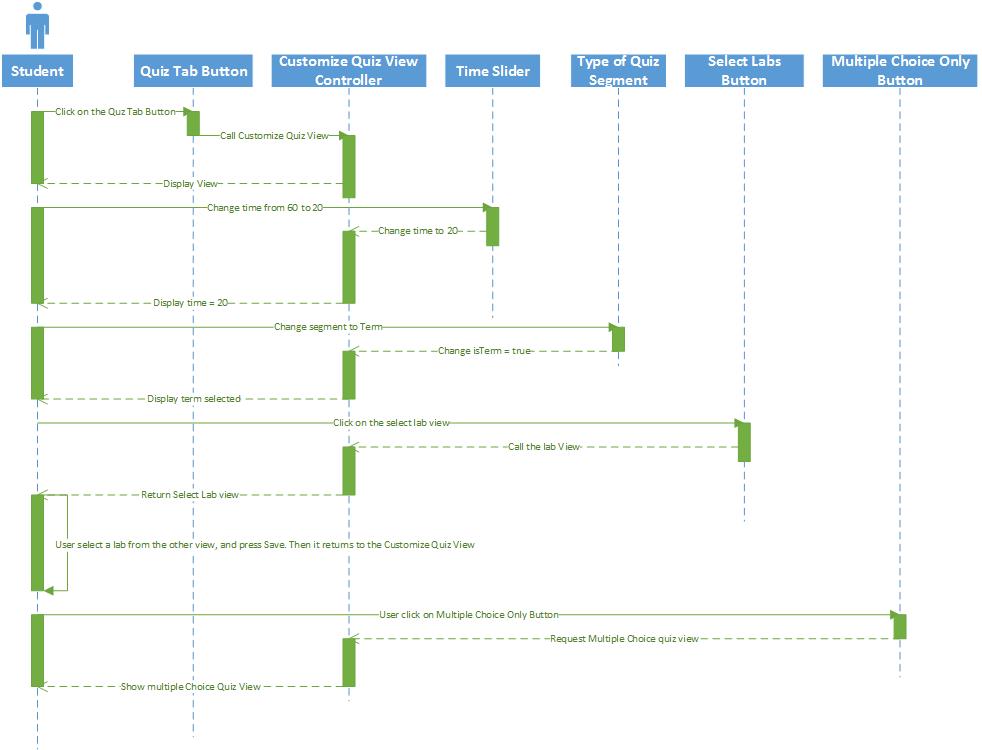
**Sequence Diagram**

**User Story #176 Personalize quiz settings**

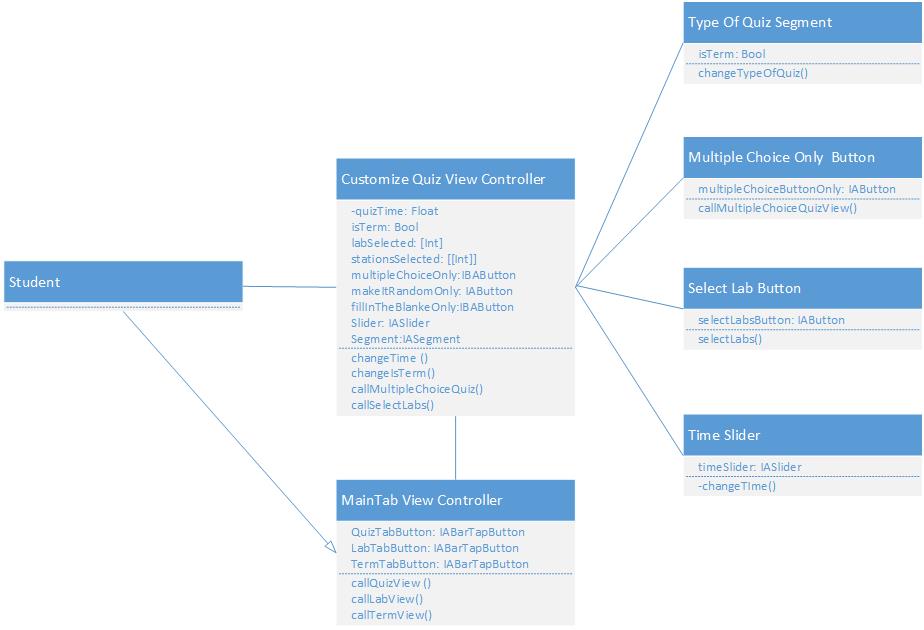
**Use Case Diagram**



**Sequence Diagram**

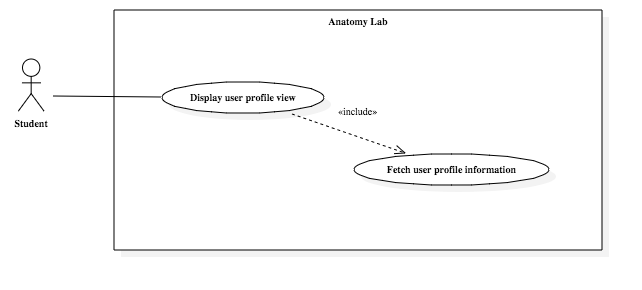


**Class Diagram**

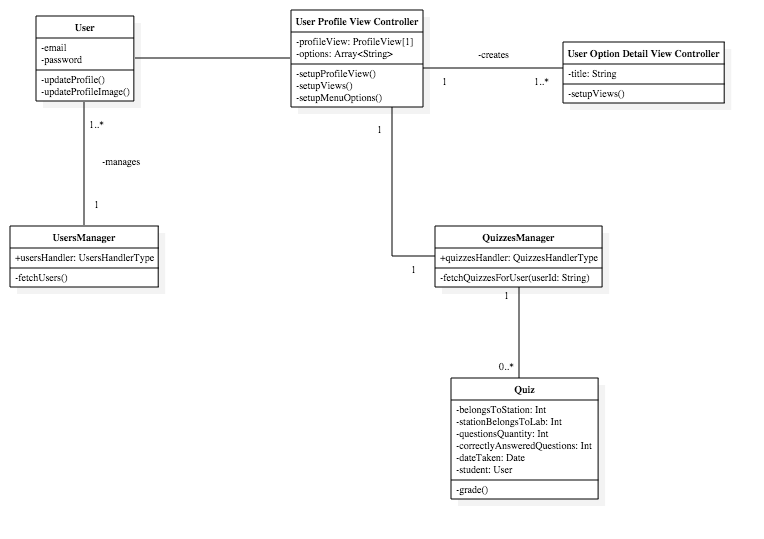


**User Story #179 Display a student profile view**

**Use Case Diagram**

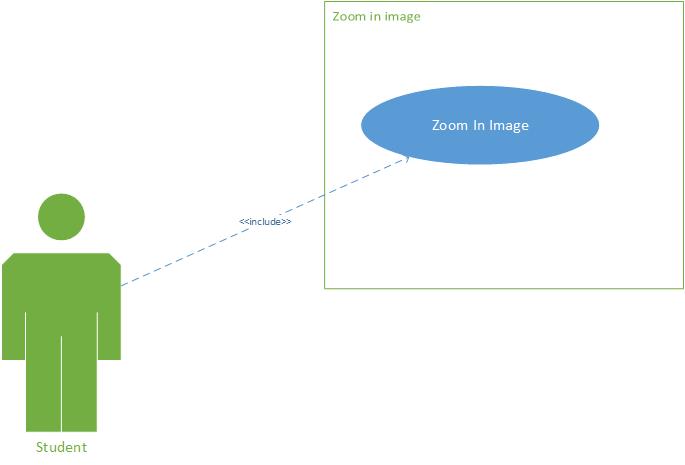


**Class Diagram**

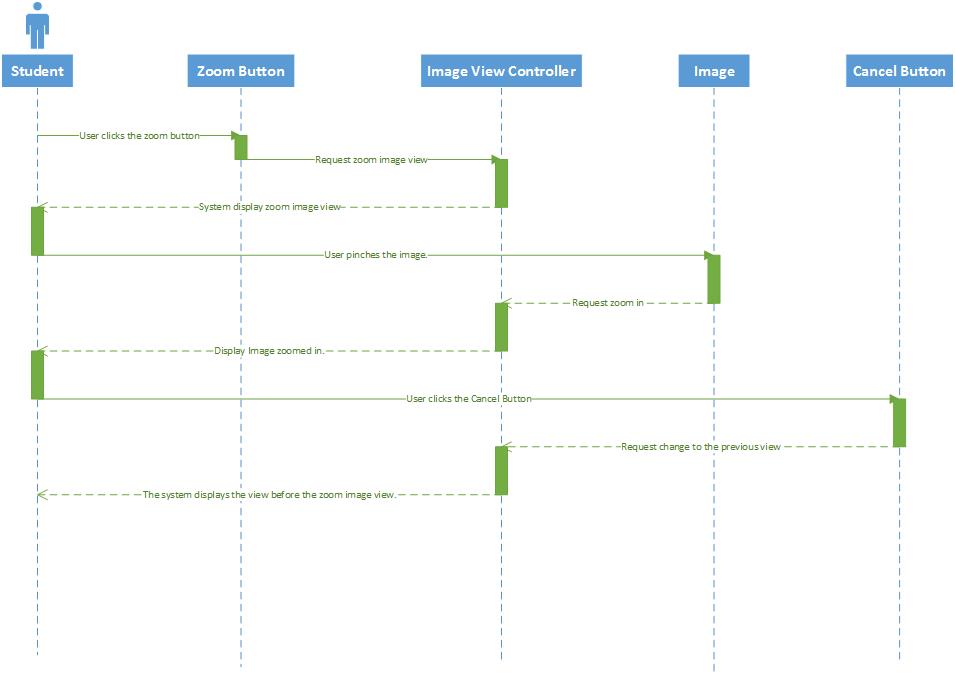
****

**User Story #185  Zoom in a picture during a quiz**

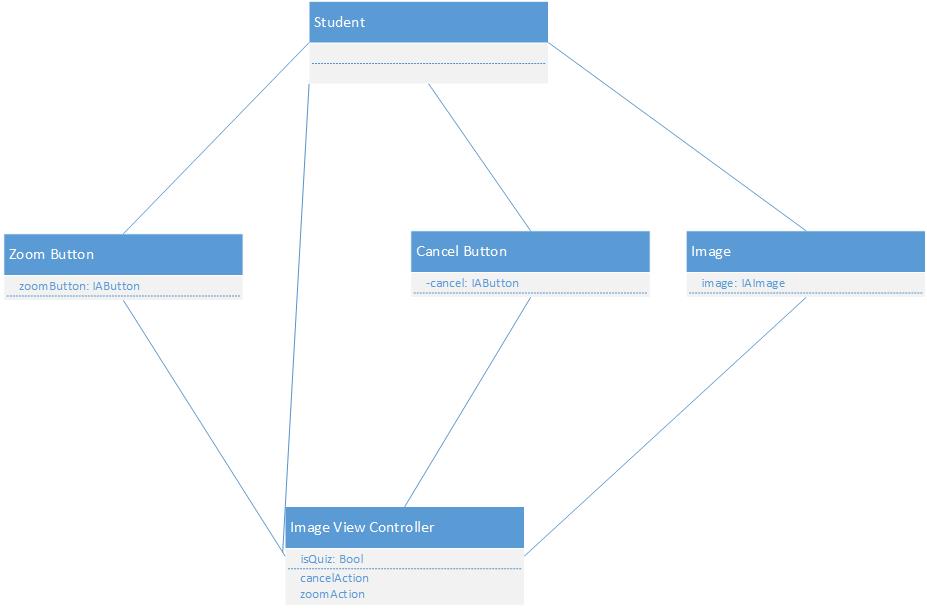
**Use Case Diagram**

****

**Sequence Diagram**

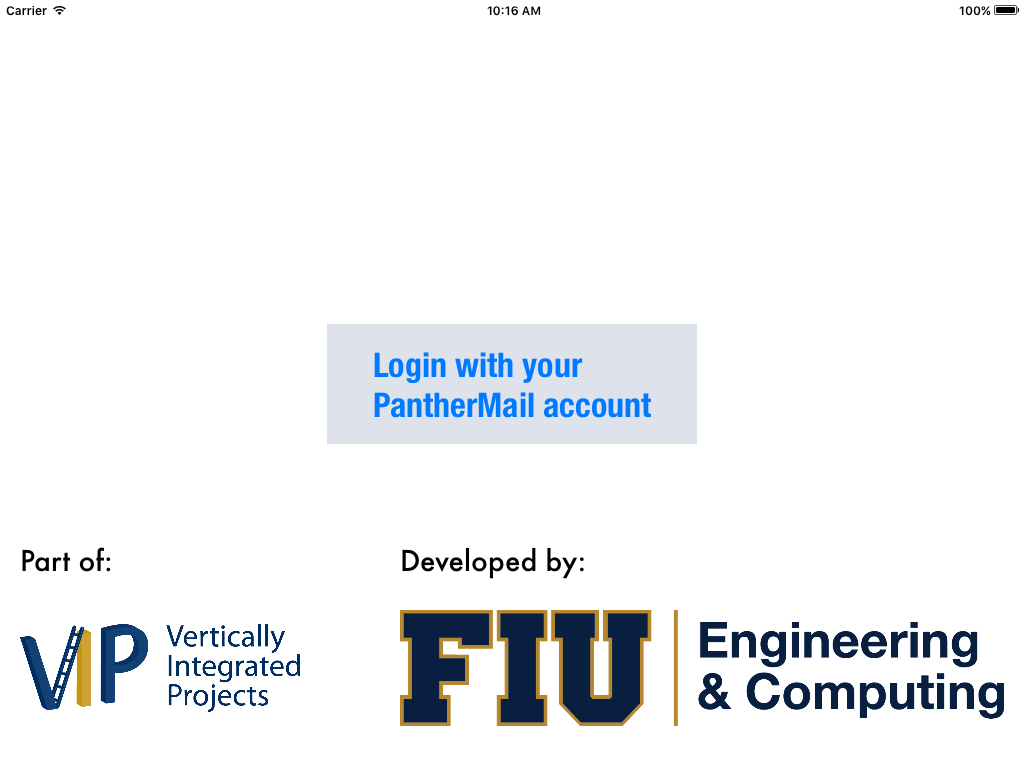
****

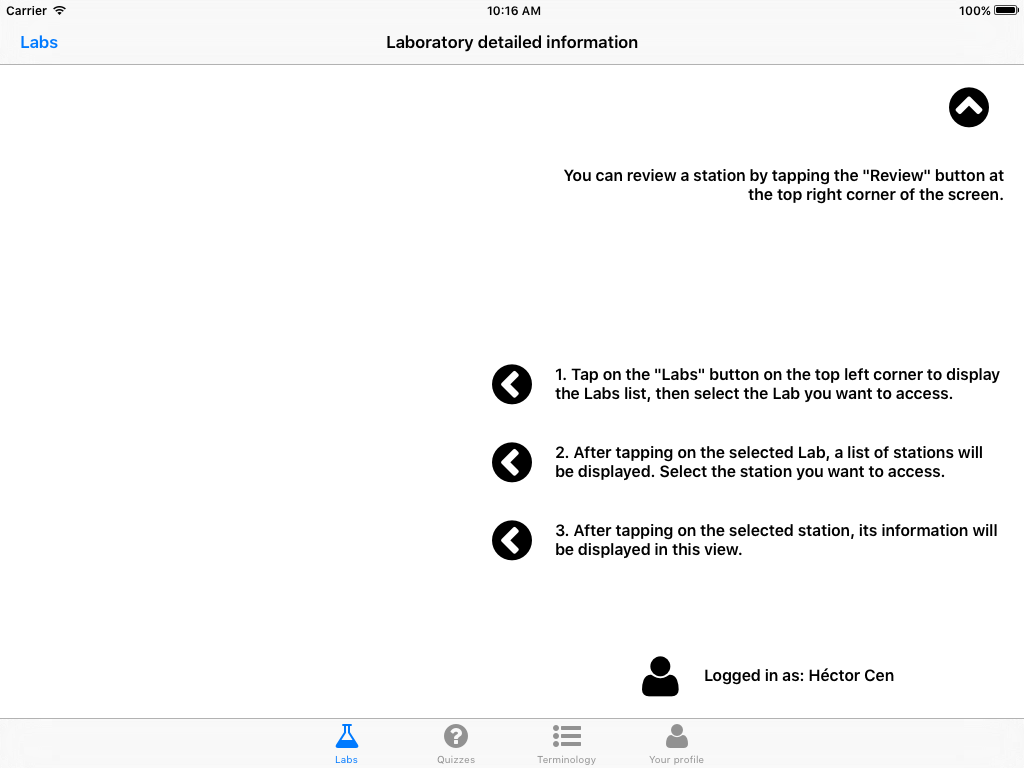
**Class Diagram**

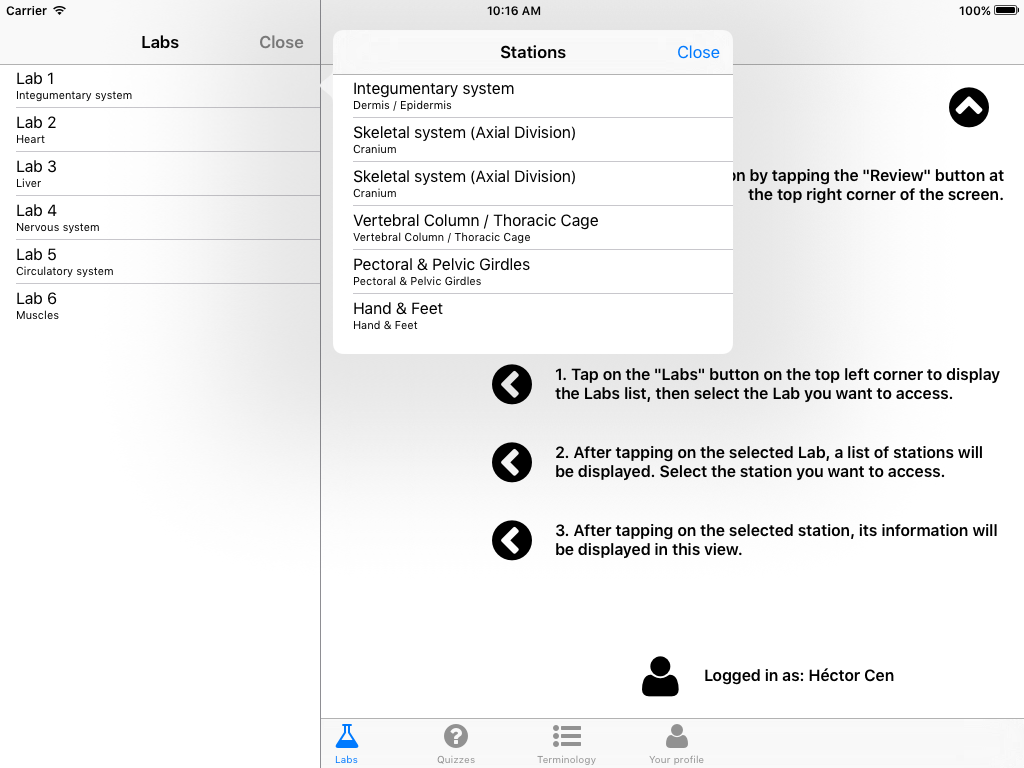


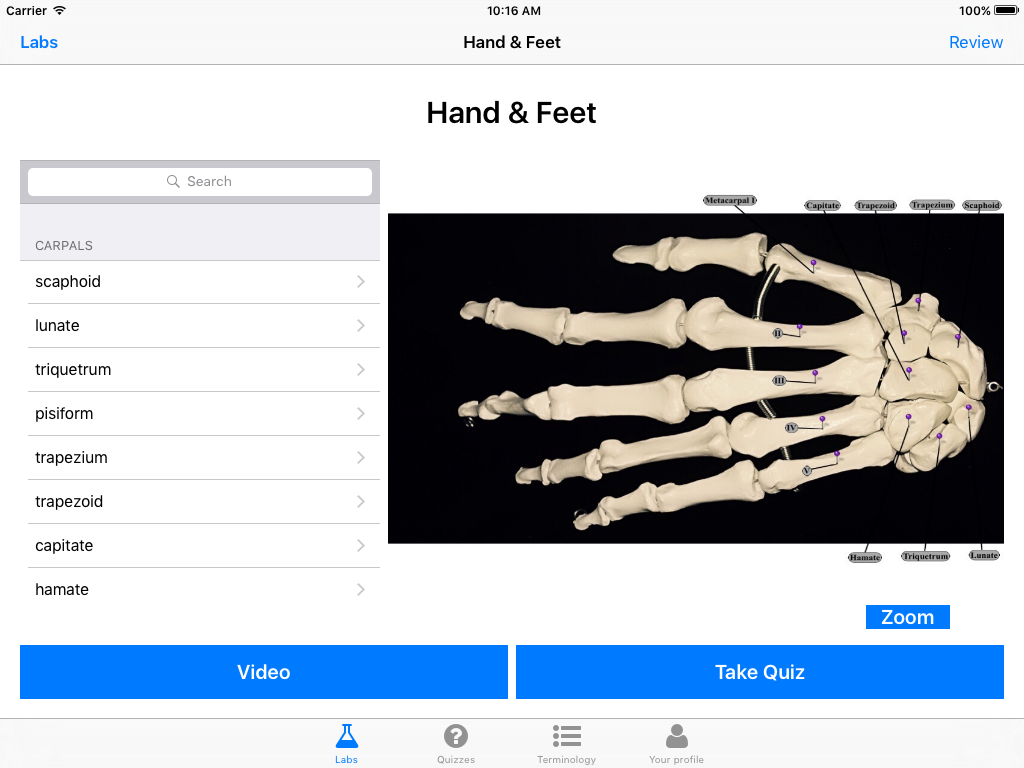
**Appendix B - User Interface Design**

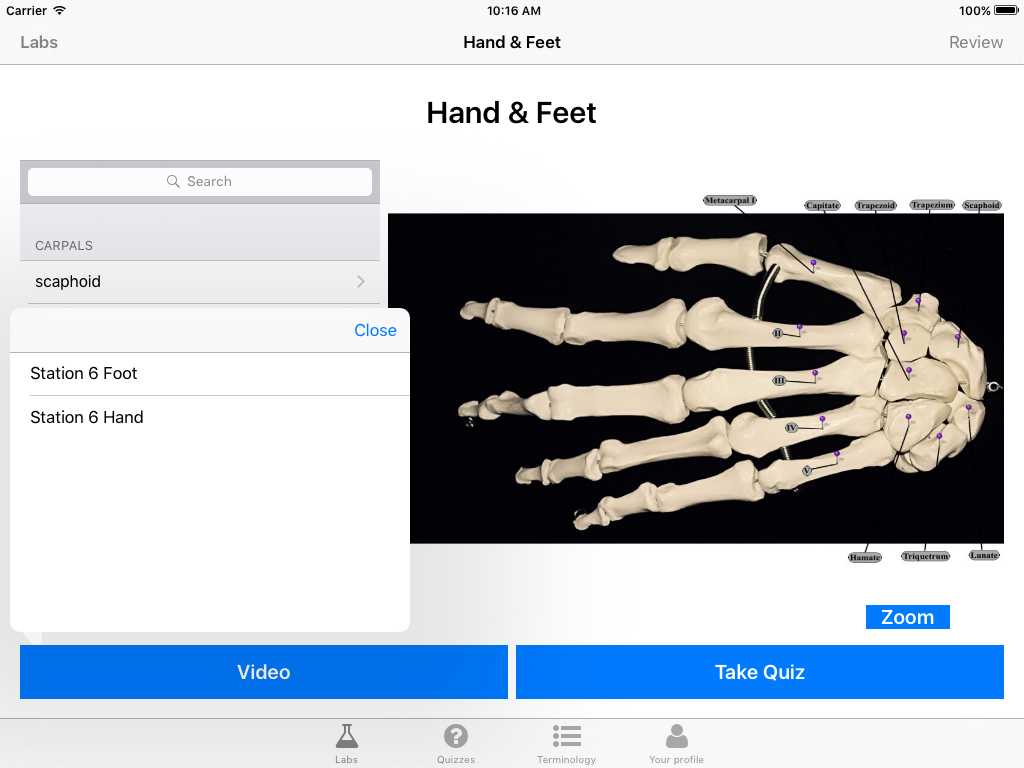
1- Log in Screen

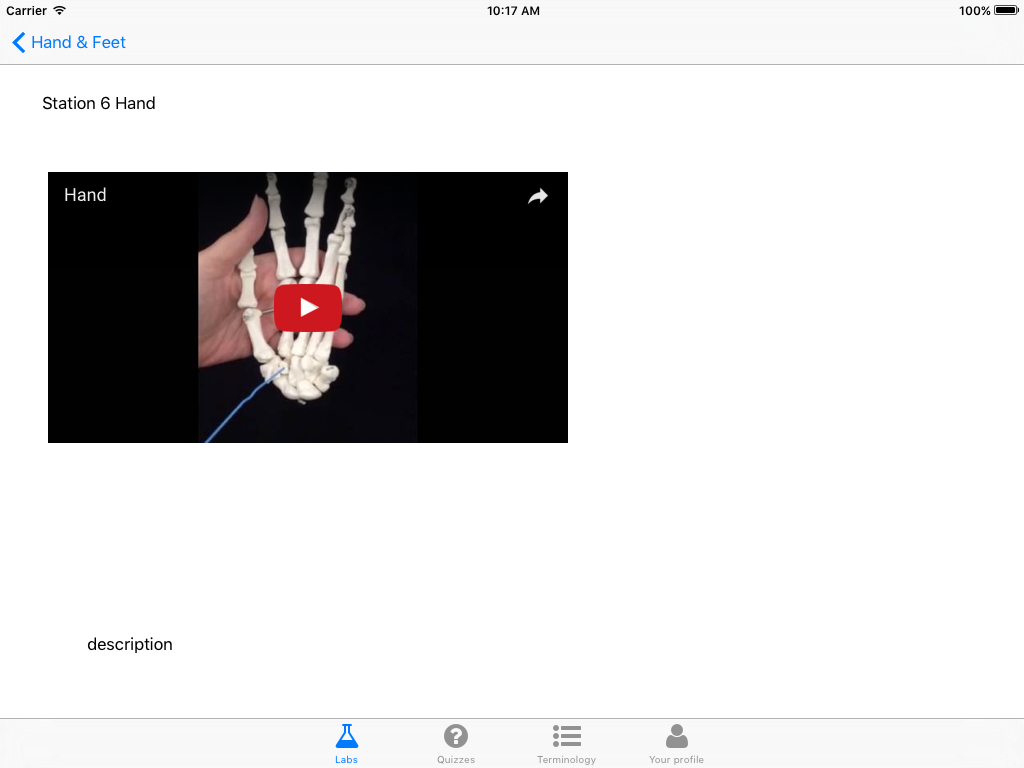


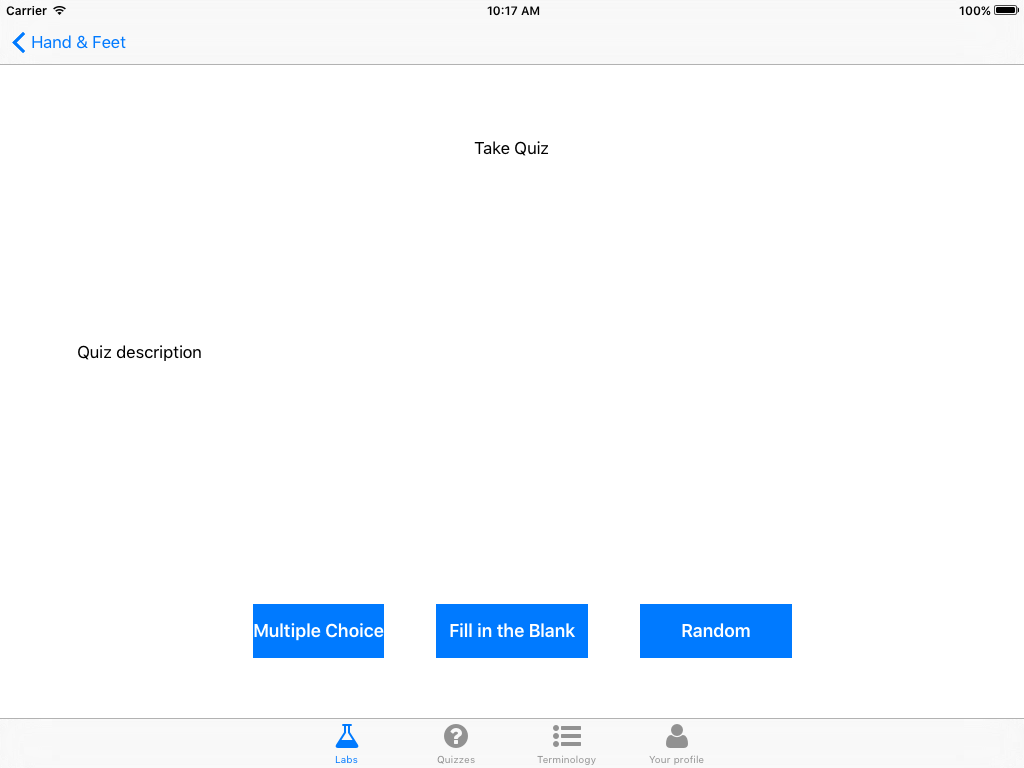
2- Welcome Page Screen 

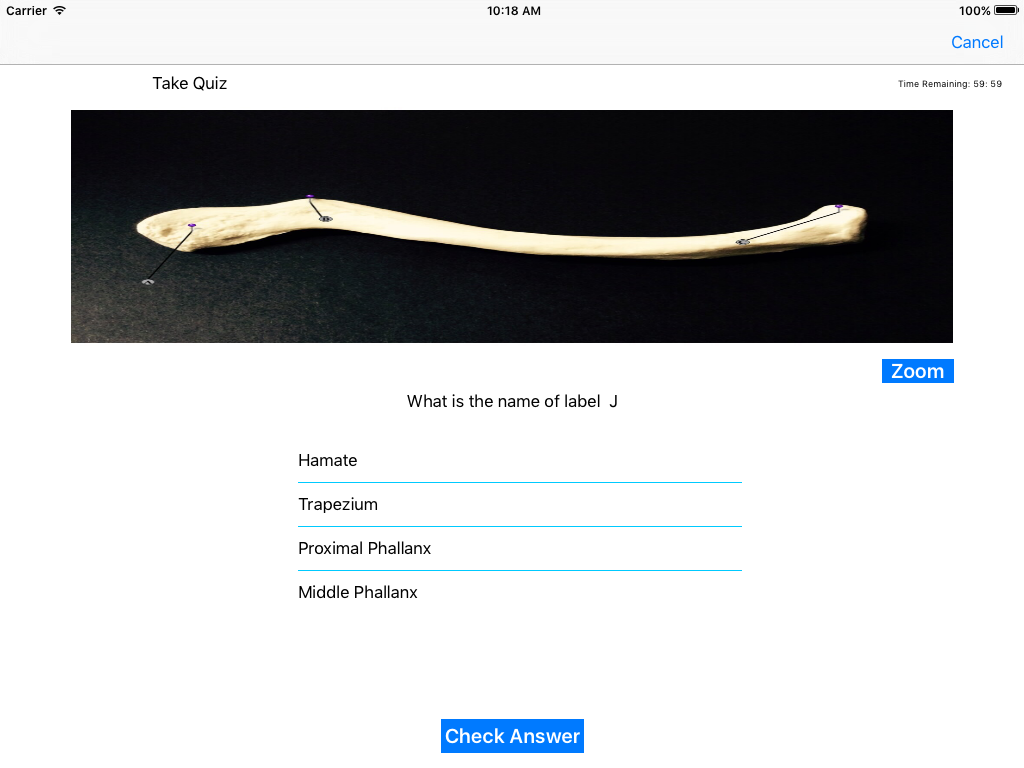
3- Labs table and Station table Pop Up

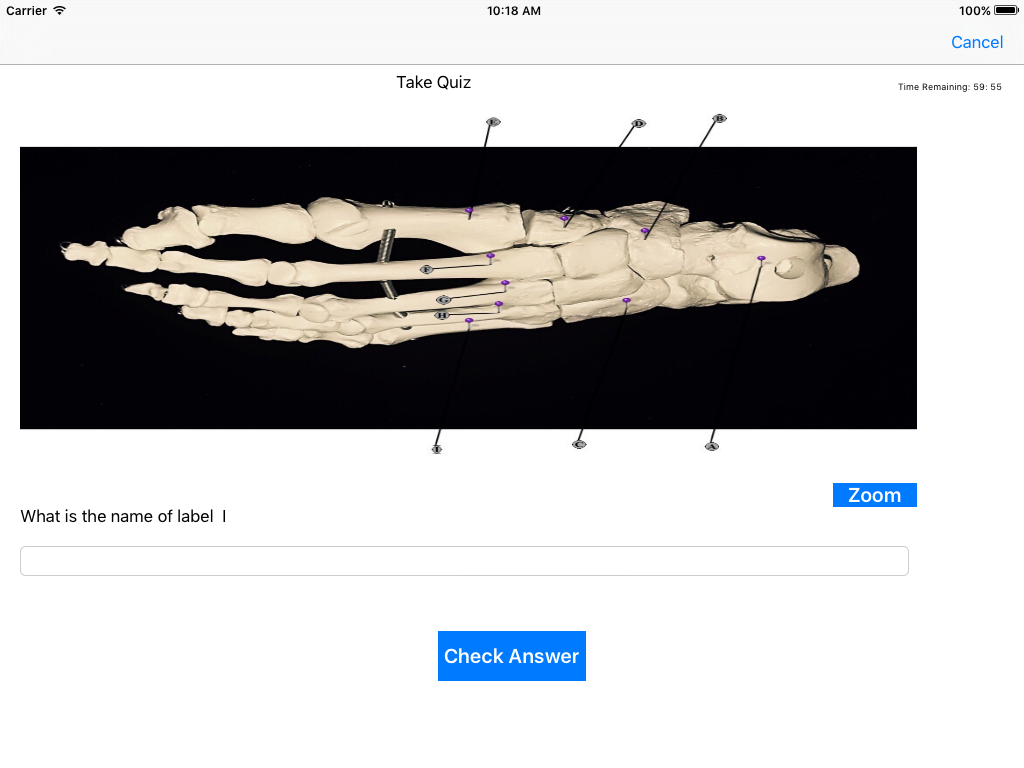
4- Screen of a station selected

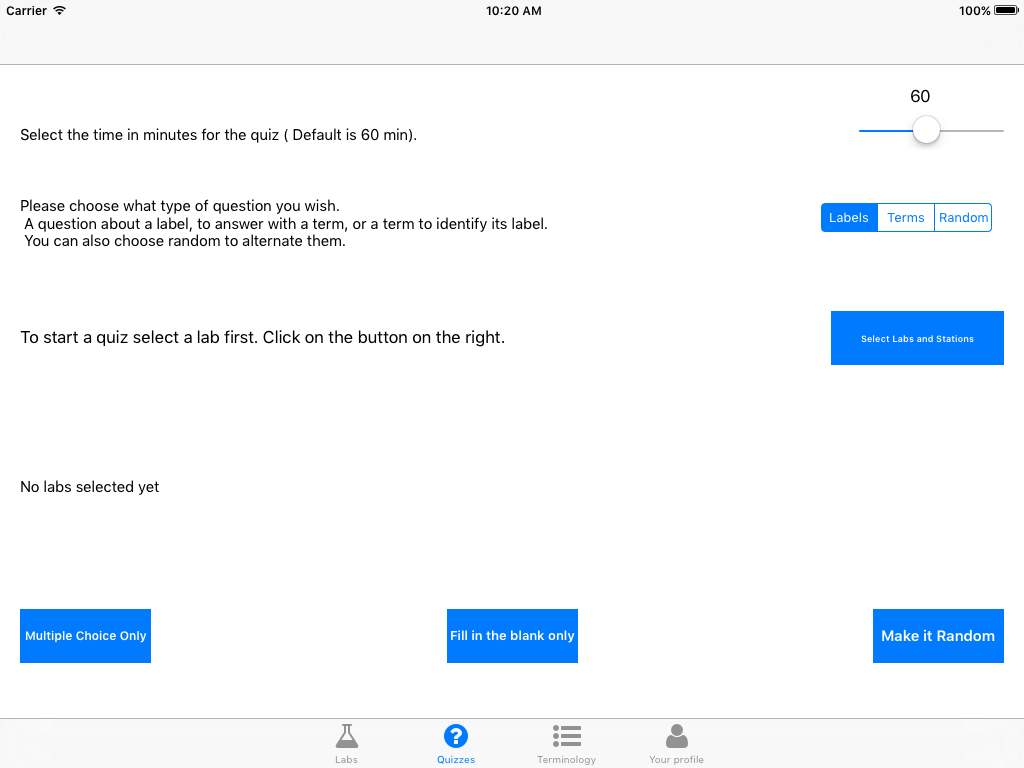
5- Video table pop up (When there is more than one video per station)

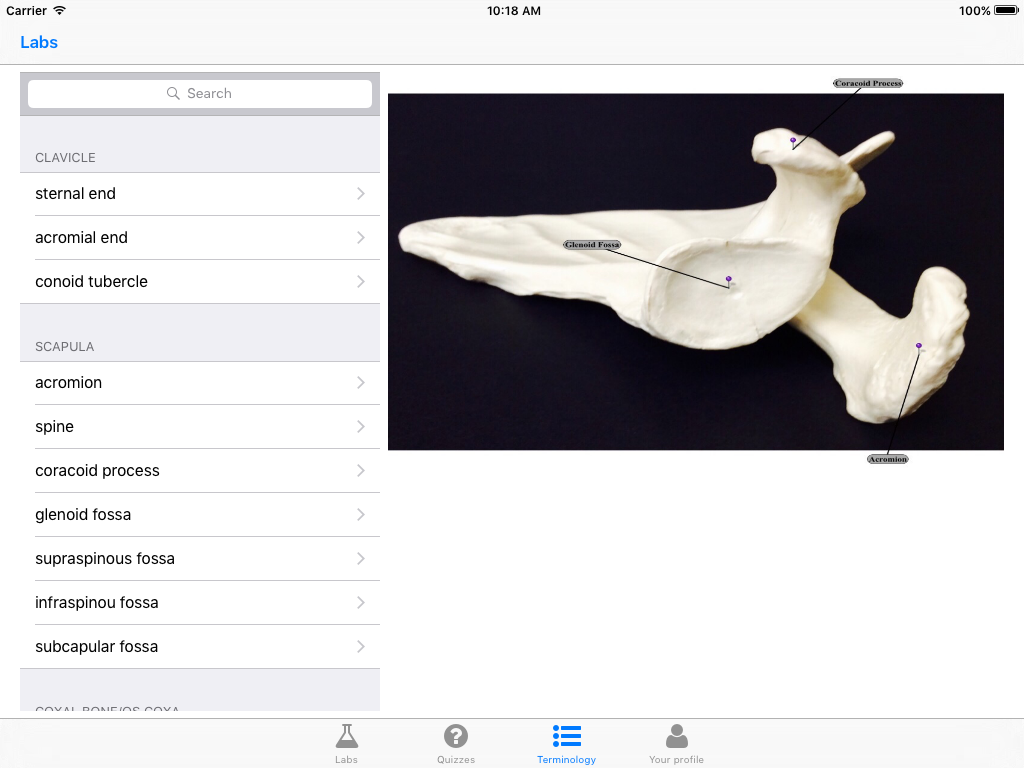
6- Video selected screen

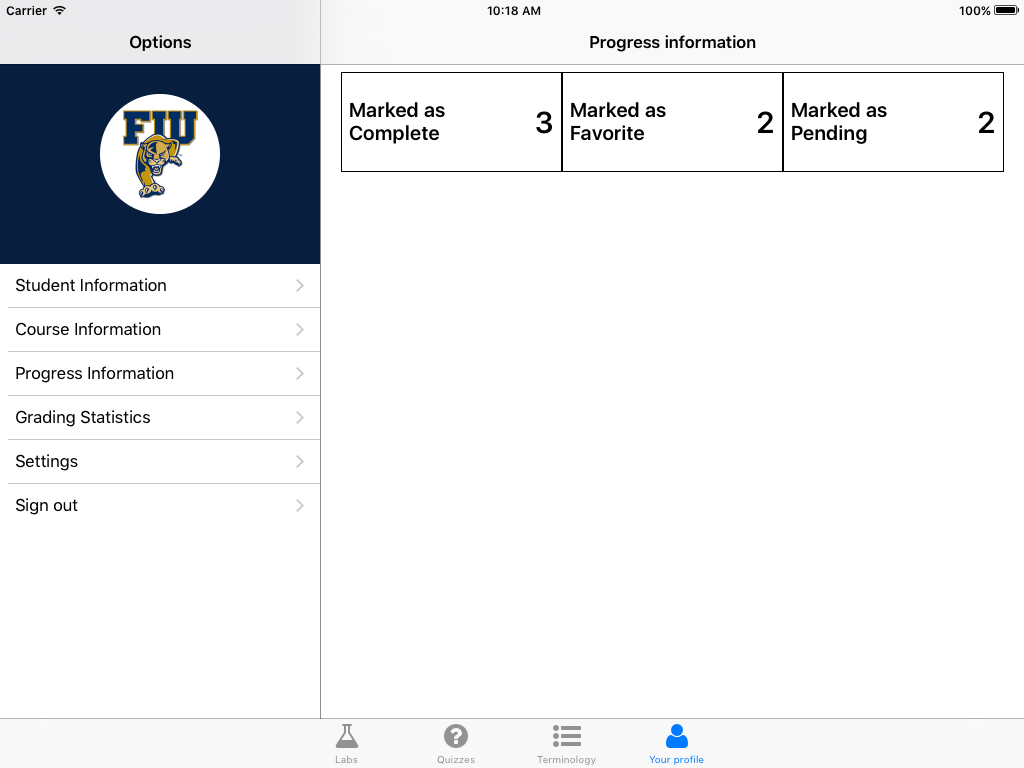
7- Take Quiz Screen. ( When the user clicks the Take Quiz button in the station info screen 5).

8- Multiple Choice Quiz Screen

9- Fill in the blank quiz screen.

10- Quiz Customize screen ( When the user click the Quiz Tab) 

11- Terminology view ( When the user clicks the Term tab) 

12- User Profile Screen ( When the user clicks in the profile tab) 

**Appendix C - Sprint Review Reports**

**Sprint 1:**

Attendees: Lisa Brinn, Héctor Cen, Darian Mendez

Start time: 14:20

End time: 14:30

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners: All.

User Story: #118 Project template with TabBar Controller

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Spring Planning meeting.

None

**Sprint 2:**

Attendees: Lisa Brinn, Héctor Cen, Darian Mendez

Start time: 13:45

End time:  14:00

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners: All.

User Story: #119 Display the Labs list.

User Story: #120 Display PopUp to select desired station.

User Story: #121 Load Lab 1 information.

User Story: #123 Display Take Quiz Button.

User Story: #124 Display Video Button.

User Story: #126 Display Station Information

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Spring Planning meeting.

None

**Sprint 3:**

Attendees: Lisa Brinn, Héctor Cen, Darian Mendez

Start time: 10:30

End time:  10:45

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners: All.

User Story: #129 Load and Play Station Videos.

User Story: #130 Allow app login through PantherMail.

User Story: #131 Display the TakeQuiz View.

User Story: #132 Load full station information detail.

User Story: #134 Allow the users to sign out from the app.

User Story: #135 Create a Welcome Screen

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Spring Planning meeting.

None

**Sprint 4:**

Attendees: Lisa Brinn, Héctor Cen, Darian Mendez

Start time: 10:15

End time:  10:45

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners: All.

User Story: #153 Take Multiple Choice Quiz

User Story: #154 Take Fill in the Blank Quiz

User Story: #155 Take a Random Quiz

User Story: #156 Add search capability for the terminology list

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Spring Planning meeting.

User Story:

#160 Customize Quizzes DetailViewController

#161 Customize Terminology DetailViewController

How this should be reflected on the user story definition in Mingle:

The user story description is changed and moved to the carryover backlog.

**Sprint 5:**

Attendees: Lisa Brinn, Héctor Cen, Darian Mendez

Start time: 11:15

End time:  11:30

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners: All.

User Story: #173 Make the quizzes questions and answers  random

User Story: #176 Personalize Quiz Settings

User Story: #178 Create Take Quiz Tab View

User Story: #177 Save student's information in a local database

User Story: #179 Display a student profile view

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Spring Planning meeting.

User Story:

#175 Fix quiz data for lab 1

#176 Fix data for videos in Lab 1

#161 Customize Terminology DetailViewController

How this should be reflected on the user story definition in Mingle:

The user story description is changed and moved to the carryover backlog.

**Sprint 6:**

Attendees: Lisa Brinn, Héctor Cen, Darian Mendez

Start time: 10:00

End time:  10:30

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners: All.

User Story: #175 Fix quiz data for Lab 1

User Story: #174 Fix data for videos in Lab 1

User Story: #184 Zoom in the pictures for the term

User Story: #185 Zoom in the picture during a quiz

User Story: #161 Customize Terminology DetailViewController

User Story: #186 Update student profile view to reflect changes in the database

User Story: #187 Add full data on stations for Lab 1

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Spring Planning meeting.

User Story:

None

**Appendix D - User Manuals, Installation/Maintenance Document, Shortcomings/Wishlist Document and other documents**

### Sprint Retrospective:

**Sprint 1:**

Attendees: Lisa Brinn, Héctor Cen, Darian Mendez

Start time: 14:00

End time: 14:20

What went wrong?

Did we do a good job estimating our team's velocity?

Yes

Did we do a good job estimating the points (time required) for each user story?

Fairly.

Did each team member work as scheduled?

Yes

What went right?

Basic design was accepted.

How to address the issues in the next sprint?

How to improve the process?

Do a better division of the work to be done by the members. Increase the communication with the product owner. Product owner should send over the materials quickly.

How to improve the product?

Receive input from end users whenever a basic product is available to test.

**Sprint 2:**

Attendees: Lisa Brinn, Héctor Cen, Darian Mendez

Start time: 13:30

End time: 13:45

What went wrong?

Did we do a good job estimating our team's velocity?

Yes

Did we do a good job estimating the points (time required) for each user story?

Yes

Did each team member work as scheduled?

Yes

What went right?

Features developed  were accepted.

How to address the issues in the next sprint?

How to improve the process?

Increase the communication with the product owner.

How to improve the product?

Receive input from end users whenever a basic product is available to test.

**Sprint 3:**

Attendees: Lisa Brinn, Héctor Cen, Darian Mendez

Start time: 10:45

End time: 11:00

What went wrong?

Did we do a good job estimating our team's velocity?

Yes

Did we do a good job estimating the points (time required) for each user story?

Good.

Did each team member work as scheduled?

Yes

What went right?

All the user stories were developed and accepted by the product owner.

How to address the issues in the next sprint?

How to improve the process?

We need to improve the communication with the product owner so we can get access to the material faster.

How to improve the product?

Receive input from end users whenever a basic product is available to test.

**Sprint 4:**

Attendees: Lisa Brinn, Héctor Cen, Darian Mendez

Start time: 10:45

End time: 11:00

What went wrong?

Did we do a good job estimating our team's velocity?

No

Did we do a good job estimating the points (time required) for each user story?

Fair.

Did each team member work as scheduled?

No

What went right?

Some user stories were developed and tested.

How to address the issues in the next sprint?

How to improve the process?

We need to show the features to the product owner before the end of the sprint.

How to improve the product?

Load real information to the app in order to test in a realistic manner.

**Sprint 5:**

Attendees: Lisa Brinn, Héctor Cen, Darian Mendez

Start time: 11:30

End time: 11:45

What went wrong?

Did we do a good job estimating our team's velocity?

No

Did we do a good job estimating the points (time required) for each user story?

Fair.

Did each team member work as scheduled?

No

What went right?

Most of the user stories were developed and tested.

How to address the issues in the next sprint?

How to improve the process?

We need to improve the way that we explain the product owner how we need the app data.

How to improve the product?

Load real information to the app in order to test in a realistic manner.

Add a more professional look & feel to the application.

**Sprint 6:**

Attendees: Lisa Brinn, Héctor Cen, Darian Mendez

Start time: 10:30

End time: 10:45

What went wrong?

Did we do a good job estimating our team's velocity?

Yes

Did we do a good job estimating the points (time required) for each user story?

Good.

Did each team member work as scheduled?

Yes

What went right?

Most of the user stories were developed and tested.

How to address the issues in the next sprint?

How to improve the process?

None

How to improve the product?

Load real information to the app in order to test in a realistic manner.

Add a more professional look & feel to the application.

**Installation**

***Development environment setup***

In order to develop apps for iOS, using native languages such as *Swift* or *Objective-C*, developers need an OS X powered computer, such as iMac, Macbook Pro, Macbook Air, Mac Mini, etc…

Recommended settings:

* Latest version of **macOS** (as of the date of this document's completion: *Sierra*)
* Latest version of **Xcode** (as of the date of this document's completion: 8.1)

Additionally, to setup the project's development environment, users have to install *Cocoapods.* *Cocoapods* is a dependency manager for Swift/Objective-C projects. It readily handles third party libraries and frameworks in an almost transparent way for the developer.

To install *Cocoapods*, run the following command in a terminal:

sudo gem install cocoapods

The output should reflect the result of the operation. After installing *Cocoapods*, you can setup the project in order to start working on it. To accomplish that:

* Open a Terminal and head to the root folder of the project (the one containing the Podfile file)
* Run: pod install --verbose. You should see the installation output in the terminal screen
* After the installation of the pods completes. Open Finder and go the the project folder, you should see a .xcworkspace file. This is the one you have to open, not the .xcodeproj.
* After you open the workspace, you should see two projects in Xcode's left side menu, Anatomy Lab and Pods. The coding is done in the files under Anatomy Lab, not under Pods.

**NOTE:** It is possible that you need to update the Google Developers credentials to work with the Google Sign In framework. This was set up using a personal account.

You can now work on the project.

**Maintenance**

***Updating third party libraries***

Third party libraries should be updated to their latest stable version. However, considerations have to be made before updating third party libraries:

* Regularly check the developer's documentation regarding the library's support targets, such as language version and/or target OS version, missing features, known bugs, etc…
* In very rare occasions, third party libraries may conflict with each other

**Shortcomings / Wishlist**

***Wishlist***

* A restful API (a typical webapp) should be developed in order to improve the current features as well as to add new ones. Potential improvements might include and are not limited to:
  + User information syncing across devices
  + Full blown relational database interface and features
  + Capability for faculty to edit app's information through several interfaces
  + Communication with existing FIU systems
* Improved graphs and information to be displayed in the user profile view
* Addition of gamification elements in order to incentivize students engagement and productivity
* The app should show the model pictures without the tags. There should be a section where the students can add the tag themselves after watching the videos.
* The videos should play side by side with the picture to tag, this way the student can watch the video and tag the pictures at the same time.
* Addition of rotating images or 360 degree views of the models.
* Streamline the process of setting view controllers in the detail part of the UISplitViewController.
* Replace some simple classes with structs
* Adopt and extensively use *protocols*, it goes hand to hand with MVVM pattern and development in iOS particularly.
* Leverage the power of the compiler in order to safely write code, especially in the following scenarios:
  + Safely unwrapping optionals
  + Using enums to safely treat elements identified through Strings, such as: Table view cell identifiers, Storyboard segues, Storyboard IDs, etc...
* Migration from Swift 2.3 to the latest Swift version in order to harness new and potentially useful features of language updates. (Version 3.0.1 is the latest stable release, as of this document's completion date)

***Shortcomings***

* MVVM pattern needs some refinement. Some code ended up in places it should not be. Model code in Viewmodel classes and vice versa.
* Some subclasses are not really that necessary and add extra complexity and might be discarded.
* UI definitely needs improvement. Better implementation of some controls and overall UI design needed.

**References**

The following tutorials helped us to understand and develop our iPad application using Swift in Xcode:

* Itunes store: Developing iOS 9 Apps with Swift by Stanford:

<https://itunes.apple.com/us/course/developing-ios-9-apps-swift/id1104579961>

* Vea Software Tutorials youtube channel:  <https://www.youtube.com/channel/UCAHVgG4R3JBzO9fUDvQ0M7g>