



National  
College *of*  
Ireland

# Classistant

## Classistant

### Technical Report

#### Abstract

Classistant is a Real-time Responsive Web Application which aims to increase Student – Lecturer engagement during classes by digitizing communications and offering anonymity options in order to negate social anxiety experienced by students

Paul Reid | x14552067  
paulreid96@gmail.com

# 1 Table of Contents

<b>1</b>	<b>TABLE OF CONTENTS .....</b>	<b>2</b>
	<b>EXECUTIVE SUMMARY .....</b>	<b>5</b>
<b>1.</b>	<b>INTRODUCTION .....</b>	<b>6</b>
1.1	BACKGROUND .....	6
1.2	AIMS .....	7
1.3	TECHNOLOGIES .....	8
1.4	STRUCTURE .....	10
1.4.1	System .....	10
1.4.2	Design and Architecture .....	10
1.4.3	Viability Survey .....	10
1.4.4	Conclusion .....	10
1.4.5	Further Development / Research .....	10
<b>2</b>	<b>SYSTEM.....</b>	<b>11</b>
2.1	FUNCTIONAL REQUIREMENTS .....	11
2.1.1	Use Case Diagram.....	11
2.1.2	Requirement 1 <Classistant – Create Lecturer - Use Case .....	11
2.1.3	Requirement 1 <Classistant – Enroll in Class - Use Case .....	13
2.1.4	Requirement 1 <Classistant – Join a Session - Use Case.....	14
2.2	NON-FUNCTIONAL REQUIREMENTS .....	15
2.2.1	Data Requirements .....	15
2.2.2	User Requirements .....	16
2.2.3	Environmental Requirements .....	16
2.2.4	Usability Requirements .....	16
2.3	DESIGN AND ARCHITECTURE .....	18
2.3.1	Classistant Application Architecture .....	18
2.3.2	Classistant Application Entity Relationship Diagram.....	19
2.3.3	Core Components.....	20
2.4	CODE SNIPPETS AND EXPLANATION.....	27
2.5	GRAPHICAL USER INTERFACE (GUI) LAYOUT .....	30
2.6	TESTING & EVALUATION.....	44
2.6.1	Iterative Testing .....	45
2.6.2	Usability Testing.....	45
2.6.3	Performance Testing.....	45
2.6.4	Browser Testing.....	45
2.6.5	Model Testing .....	46
<b>3</b>	<b>VIABILITY SURVEY .....</b>	<b>47</b>
3.1	INTRODUCTION .....	47
3.2	HAVE YOU EVER BEEN TOO AFRAID TO ASK A QUESTION IN A CLASS? .....	48
3.3	HAVE YOU EVER SAT THROUGH A CLASS WHERE YOU DIDN'T UNDERSTAND THE TOPIC? .....	48
3.4	DO YOU FIND IT DIFFICULT TO SPEAK IN FRONT OF A CLASS OR LARGE GROUP OF PEOPLE? .....	49
3.5	WOULD YOU USE A CLASSROOM ASSISTANT APP TO HELP GIVE REAL-TIME FEEDBACK TO YOUR TEACHER / LECTURER (E.G SHOW YOU DON'T UNDERSTAND A TOPIC OR ALLOW YOU TO ASK QUESTIONS, ANONYMOUSLY OR OTHERWISE)? .....	49
3.6	IF YES, WHAT PLATFORMS WOULD YOU USE THE APP ON? .....	50

3.7	FINALLY, ANY FUNCTIONALITY YOU'D LIKE TO SEE ADDED TO AN APPLICATION LIKE THIS?	50
3.8	CONCLUSIONS.....	51
<b>4</b>	<b>CONCLUSIONS.....</b>	<b>51</b>
<b>5</b>	<b>FURTHER DEVELOPMENT OR RESEARCH .....</b>	<b>52</b>
5.1	MORE SOPHISTICATED EMAIL SYSTEM.....	52
5.2	MORE SOPHISTICATED BADGE SYSTEM .....	52
5.3	ADDITIONAL INTERACTIONS .....	53
5.4	IMPLEMENTATION OF MULTIPLE GRAPHS / CHARTS .....	53
5.5	CUSTOMISABLE DASHBOARD.....	53
5.6	UPLOAD CUSTOM CLASS ICONS .....	53
5.7	USER PROFILE PICTURE .....	53
5.8	MESSAGING SYSTEM.....	53
5.9	INTEGRATION TO 3 <sup>RD</sup> PARTY SYSTEMS SUCH AS MOODLE .....	53
5.10	MOBILE APPLICATIONS.....	53
5.11	API IMPLEMENTATION .....	54
5.12	CLASS FORUMS.....	54
<b>6</b>	<b>REFERENCES.....</b>	<b>54</b>
6.1	WEBSITES USED.....	54
6.2	STACK OVERFLOW POSTS VIEWED .....	55
<b>7</b>	<b>APPENDIX.....</b>	<b>57</b>
	<b>OBJECTIVES .....</b>	<b>62</b>
<b>8</b>	<b>BACKGROUND .....</b>	<b>62</b>
<b>9</b>	<b>TECHNICAL APPROACH .....</b>	<b>63</b>
<b>10</b>	<b>SPECIAL RESOURCES REQUIRED.....</b>	<b>64</b>
<b>11</b>	<b>PROJECT PLAN.....</b>	<b>65</b>
<b>12</b>	<b>TECHNICAL DETAILS.....</b>	<b>66</b>
<b>13</b>	<b>EVALUATION .....</b>	<b>67</b>
	<b>EXECUTIVE SUMMARY.....</b>	<b>70</b>
<b>1</b>	<b>INTRODUCTION .....</b>	<b>71</b>
1.1	BACKGROUND .....	71
1.2	AIMS.....	72
1.3	TECHNOLOGIES .....	72
1.4	APPROACH.....	73
<b>2</b>	<b>SYSTEM.....</b>	<b>75</b>
2.1	REQUIREMENTS.....	75
2.1.1	<i>Use Case Diagram.....</i>	<i>75</i>
2.1.2	<i>Requirement 1 &lt;Classistant - Login – Use Case&gt;.....</i>	<i>76</i>
2.1.3	<i>Requirement 1 &lt;Classistant - Profile – Use Case&gt; .....</i>	<i>77</i>
2.1.4	<i>Requirement 1 &lt;Classistant – Classroom View – Use Case&gt; .....</i>	<i>78</i>
2.1.5	<i>Requirement 1 &lt;Classistant - Dashboard – Use Case&gt; .....</i>	<i>80</i>
2.2	NON-FUNCTIONAL REQUIREMENTS.....	81
2.2.1	<i>Performance/Response time requirement .....</i>	<i>81</i>
2.2.2	<i>Availability requirement.....</i>	<i>81</i>

2.2.3	<i>Recover requirement</i> .....	81
2.2.4	<i>Portability requirement</i> .....	82
2.2.5	<i>Extendibility requirement</i> .....	82
2.3	DESIGN AND ARCHITECTURE .....	83
2.4	IMPLEMENTATION.....	83
2.5	GRAPHICAL USER INTERFACE (GUI) LAYOUT .....	83
2.6	EVALUATION.....	86
<b>3</b>	<b>CONCLUSIONS</b> .....	<b>87</b>
<b>4</b>	<b>FURTHER DEVELOPMENT OR RESEARCH</b> .....	<b>88</b>
<b>14</b>	<b>MONTHLY JOURNALS</b> .....	<b>88</b>
14.1	SEPTEMBER MONTHLY REPORT .....	88
14.2	OCTOBER MONTHLY REPORT .....	88
15.1	NOVEMBER MONTHLY REPORT.....	89
15.2	DECEMBER MONTHLY REPORT.....	89
15.3	JANUARY MONTHLY REPORT.....	89
15.4	FEBRUARY MONTHLY REPORT.....	90
15.5	MARCH MONTHLY REPORT .....	90
15.6	APRIL MONTHLY REPORT.....	90
15.7	MAY MONTHLY REPORT.....	91

## Executive Summary

The Purpose of this Executive Summary is to give you an insight into what Classistant is and what it is trying to achieve.

Classistant is a Responsive Web Application which allows Students and Lecturers digitally interact with each other during class time. The goal is to digitise some of the typical social interaction in the classroom to remove social anxiety from students and allow them to speak out and ask questions without outing themselves to a large group of peers.

Classistant is light weight and portable and can be deployed on almost any cloud application service such as Heroku.

The Project was conceived during my time Lecturing for the Computing Support office at NCI. My experience made me realise how real the lack of questioning is, and how many students slip through the cracks, scraping by hoping to get a pass.

The application is developed using the Ruby on Rails framework and a number of technologies and libraries. The application uses Web sockets and ActionCable to create real-time environments in which students and lecturers can interact with each other through a number of different channels.

The end result of the project is to hopefully increase Student – Lecturer engagement and I hypothesise the average grade for any given class which implements Classistant to rise too, as students will be encouraged to ask questions and can do so Anonymously or otherwise.

Classistant in my view is the service Education doesn't realise its missing.

# 1. Introduction

## 1.1 Background

Throughout my entire academic career, I have seen time and time again the scenario of a Teacher or Lecturer delivering content when the class does not fully understand. There is always the dreaded question asked, “Does everybody understand?” or “Can I move on?”.

These questions are extremely difficult for students to answer and 9 times out of 10, a student will say nothing and allow the class to continue, despite their difficulties or not fully understanding the topics covered.

During my time as a Student at NCI I joined the Computing Support office in the College as part time staff. My work in the office largely consisted of teaching large classes, both online and offline and supporting them in any topic they were struggling with.

During my lectures, I noticed when I was teaching in person the participation from the class would be next to non-existent. Apart from the one or two students who were totally engrossed in the subject and loved their area of study, the majority of students present in the class would remain silent.

This came as a total surprise to me as my online lectures light up with questions and sometimes it became more of a question and answers session rather than a lecture. This really interested me and I began to speculate the reasons behind this occurring.

I believe students will not raise their hands and ask questions or say they don't understand due to social anxiety. Very few people will find it difficult to raise their hand in front of a class of 20 or 30, never mind 70, 150 or more. This is because students fear judgment from their peers, not knowing how others will react to their questions. Students feel like they are admitting defeat or being stupid by asking a lecturer to repeat content. This however, is absolutely not the case and chances are if one student feels this way in the class, many others secretly do too.

Upon talking to peers and staff at NCI, my theory was in line with their own. In the physical classroom, one student has to raise their hand and ask a question in front of all their peers, thus grinding the whole class to a halt. However, online, a student can simply type the question into a chat box from the comfort of their home and it will be answered at the next most convenient moment.

I found this absolutely fascinating and it drove me to try to develop an idea which could either prove or disprove the theory.

After thinking about the differences between teaching in person and digitally at length I decided to try make my project, Classistant (Read: Class Assistant), about trying to combine the best of both worlds. The concept is to create a virtual link between students and lecturers during in person class time which will allow students to interact with the lecturer delivering the content without drawing any attention to themselves.

I firmly believe a scenario where Students and Lecturers can interact digitally will increase the number of questions dramatically and as a direct result I would expect the average grade for a given class to rise too.

## 1.2 Aims

The Goal of Classistant is to increase Student – Lecturer engagement in the physical Classroom by digitising interaction in order to negate social anxiety as a barrier to communication.

This goal is achieved by developing a fully functional Web Application, which allows students and lecturers to log in and interact in real-time, using web sockets to maintain a connection to each other. The application is fully responsive to cater for the fact that almost every student today in Ireland carries a smartphone with them.

I want the application to contain functionality such that a User can register an account. All accounts will be made as a Student initially and one master account which is an Administrator will be able to then easily promote them to Lecturer. This adds a layer of security behind the application seeing as it is dealing with User's data.

The application will allow Lecturers to create classes which Students can enrol in using a unique key. Each class in the system will be able to hold multiple sessions. Each session will be a real-time environment which allows students and lecturers to easily interact using a chat system, question system as well as "interactions"

"Interactions" will be real-time prompts sent to Students to complete. This will consist of two types of polls, one which is based on pictures, and another which is based on text buttons. This will prompt the user to rate their understanding in the class and provide real-time data back to the lecturer. The responses will be graphed in real-time for the lecturer to see and gain a better understanding of the class and their performance. All data from a session can later be looked back on.

Students will earn badges in the system through participation which will be displayed on their profile.

Lecturers can see students who are struggling and send email notifications for support.

### 1.3 Technologies

The goal for the development side of Classistant was to create a fully functioning application in the most simple, light weight way possible. I am a firm believer that less is more and a large, cumbersome web application with extensive amounts of logic and scripts will simply not give the simple, premium experience Classistant strives to deliver.

#### **IDE**

The development of Classistant started in the Cloud Based IDE cloud9 which was unfortunately becoming phased out throughout the development of this project. Unwilling to pay for AWS, I moved development to JetBrains's RubyMine IDE. This is locally installed on my 2015 MacBook pro and licensed with a free student license.

#### **Framework**

The entire backend of the project is created using Ruby and the Ruby on Rails (RoR) framework. In my view, this was the best approach as it allows me to utilise my coding abilities in OOP languages like Java and mix it with Web Development.

#### **Languages & Libraries**

The application is not just solely developed in Rails though, it makes use of languages such as HTML, CSS, JavaScript, Coffeescript, SCSS as well as technologies such as Active Record and PostgreSQL. In order to make the application responsive I used Twitter's Bootstrap framework to implement a responsive grid system. Finally, I implemented a Gem called Chartkick which allows me to use the HighCharts Graphing Library but through Ruby code

#### **Deployment**

The application is deployed using the Heroku Cloud Deployment Service and deployment as well as development was done using Git & GitHub



**Table of Technologies and Tools**

Programming Languages	JQuery
	Ruby
	HTML5, CSS, SCSS, JavaScript
	CoffeeScript
Development Environments / Tools	Cloud9
	RubyMine
	Mac Terminal
	Git + GitHub
	Heroku
	Stylebot (Chrome Plugin)
Libraries / Gems	ChartKick
	HighCharts
	Twitter Bootstrap
	Devise
	Bootstrap Form For
Database	Active Record running PostgreSQL
Platform	Responsive Web Application

## **1.4 Structure**

Below are short descriptions of the main chapters found in this Report

### **1.4.1 System**

This section of the report gives a thorough breakdown of the requirements specification. Some of the requirements specification has not changed and therefore is not mentioned. This can be found in the Requirements Specification in the Appendix.

### **1.4.2 Design and Architecture**

This Section Details diagrams of the System's Architecture. This shows both the Higher-Level concept of the application as well as the Entity Relationship Diagram for the database.

This section also explains the key parts of the Application as well as interesting code snippets to further example the key parts.

### **1.4.3 Viability Survey**

The Viability Survey was carried out before the project had begun. This was simply an anonymous poll sent out to my peers, questioning them in the research area of the project.

### **1.4.4 Conclusion**

The Final Conclusion of the Report, detailing my findings and experiences

### **1.4.5 Further Development / Research**

A brief list of future features that could be added in order to improve Classistant.

## 2 System

### 2.1 Functional Requirements

#### 2.1.1 Use Case Diagram

Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.

The Use Case Diagram provides an overview of all functional requirements.

It is inferred that Lecturers have access to everything a Student does and an admin has access to everything a Lecturer has.

**Note: Not all Use Cases fit on one Diagram so the most important are shown.**

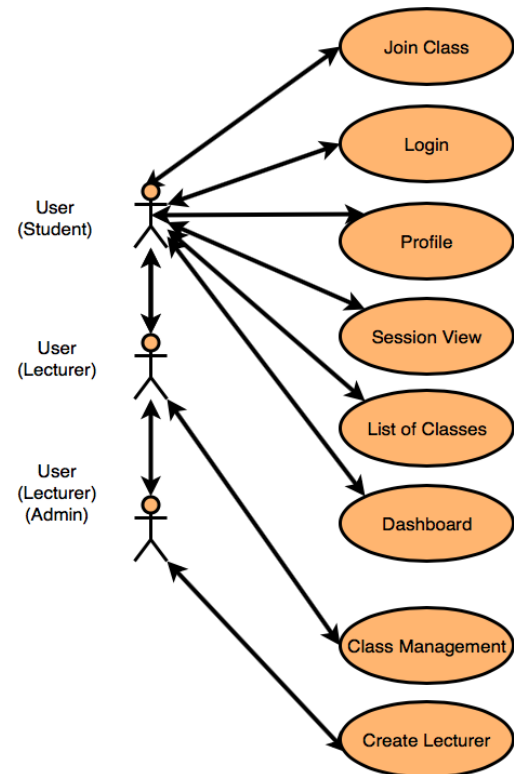


Figure 1 – Use Case Diagram

#### 2.1.2 Requirement 1 <Classistant – Create Lecturer - Use Case

##### 2.1.2.1 Description Priority

The Create Lecturer Use Case allows more than one Lecturer to be added into the System. Without this Use Case, only one lecturer would ever exist unless the seeds.rb is modified or the Rails console for the Application instance is used.

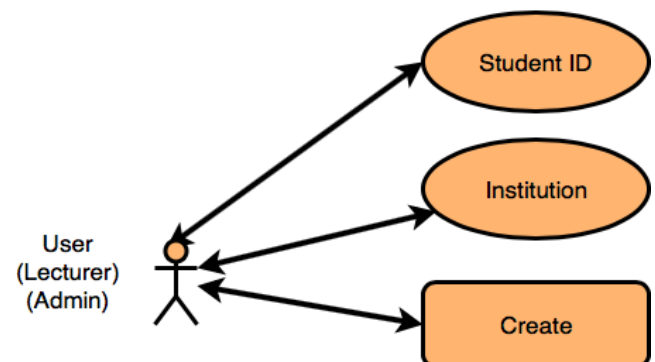


Figure 2 – Create Lecturer Use Case

##### 2.1.2.2 Use Case

##### Scope

The scope of this use case is to understand the Create Lecturer use case

## Description

This use case describes the process through which the User creates a new Lecturer in the System using an existing Student

## Flow Description

### Precondition

The User logged in is an Administrator and the Lecturer being created already has a student account.

### Activation

This use case starts when the User selects the “Create Lecturer” button on the dashboard.

### Main Flow

1. The Application presents the User with the Create Lecturer Form
2. The User enters in the Student ID of the Lecturer they wish to Promote
3. The User enters in the Institute for the new Lecturer
4. The User selects the “Add Lecturer” button
5. The System accepts the variables and creates a new lecturer
  - a. The System also deletes the old Student

### Alternate Flow

#### A1: Creation Error

1. The Application presents the User with the Create Lecturer Form
2. The User enters in the Student ID of the Lecturer they wish to Promote
3. The User enters in the Institute for the new Lecturer
4. The User selects the “Add Lecturer” button
5. The System rejects the data being passed from the form
6. The System redirects the user back to the form and displays any errors
  - a. The incorrect fields will be marked appropriately

### Termination

The System accepts the Data from the Form, a new Lecturer User is created and the old Student User for the Lecturer is destroyed.

### Post Condition

The User is redirected to the Dashboard

### 2.1.3 Requirement 1 <Classistant – Enroll in Class - Use Case

#### 2.1.3.1 Description Priority

The Enrol in Class Use Case allows a Student to enrol in a Class. This means the Student has access to the class and its data. This also allows the student to join Sessions related to the class.

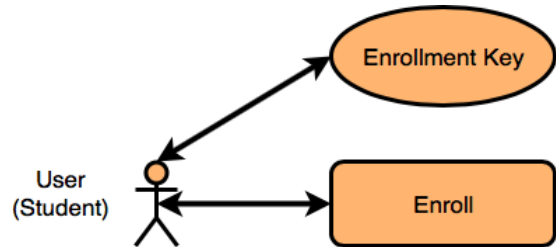


Figure 3 – Enroll in a Class Use Case

#### 2.1.3.2 Use Case

##### Scope

The scope of this use case is to understand the Enrol in Class Use Case

##### Description

This use case describes the process through which the user enrolls in a Class

##### Flow Description

##### Precondition

The User logged in is a Student.

##### Activation

This use case starts when the User selects the “Enroll in Class” button on the dashboard.

##### Main Flow

1. The Application presents the User with the Enroll in Class Form
2. The User enters in the Enrollment Key of the Class they wish to Join
3. The User selects the “Enroll” button
4. The System accepts the Enrollment Key and adds the Student to the Group

##### Alternate Flow

###### A1: Join Error

1. The Application presents the User with the Enroll in Class Form
2. The User enters in an invalid Enrollment Key
3. The User selects the “Enroll” button
4. The System rejects the Enrollment Key

5. The System redirects the User to the Enroll in Class form
6. The Form inputs show validations and errors messages display detailing the error

## Termination

The System accepts the Enrollment Key from the Form and the Student joins the Class

## Post Condition

The User is redirected to the Dashboard

## 2.1.4 Requirement 1 <Classistant – Join a Session - Use Case

### 2.1.4.1 Description Priority

The Join a Session Use Case allows a Student to join Sessions.

### 2.1.4.2 Use Case

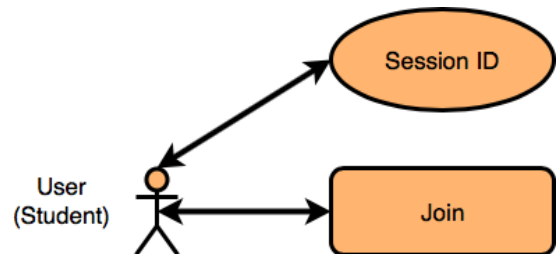


Figure 4 – Join a Session Use Case

## Scope

The scope of this use case is to understand the Join Session Use Case

## Description

This use case describes the process through which the user joins a Session

## Flow Description

## Precondition

The User logged in as a Student

## Activation

This use case starts when the User selects the “Join Session” button on the Dashboard.

## Main Flow

1. The Application presents the User with the Join Session Form
2. The User enters in the Session Key of the Session they wish to Join
3. The User selects the “Join Session” button

4. The System accepts the Session Key and adds the Student to the Session
5. The System redirects the User to the Session's page for the Session they tried to join

### **Alternate Flow**

#### **A1: Join Error**

1. The Application presents the User with the Join Session Form
2. The User enters an invalid Session Key of the Session they wish to Join
3. The User selects the "Join Session" button
4. The System rejects the Session Key and redirects the User back to the Form
5. The System displays error messages as well as highlights on the form any invalid inputs in any input fields.

### **Termination**

The System accepts the Session Key from the Form and the Student joins the Session

### **Post Condition**

The User is redirected to the ongoing Session

## **2.2 Non-Functional Requirements**

### **2.2.1 Data Requirements**

Data Requirements ended up being quite important for Classistant. Classistant is a Web Application and this means there are a number of input fields throughout the Application. Web Applications are highly susceptible to attacks due to the lack of input validation on their inputs and forms and due to how browsers and servers consume data.

As a result of this, Data input fields in the Application, where necessary, have validations applied to ensure nothing goes wrong. Thankfully, some of this is automagically handled by rails, but other unexpected inputs such as the lack of input in a certain field may cause an issue with the Database and cause data to not be saved.

This means, the Application will make sure all mandatory data will be provided before it will save. Unsatisfactory input Data will be thrown away and the User will be notified of the error using dynamic error handling and custom error forms that have been implemented.

## 2.2.2 User Requirements

In order to have access to the application, a User must have an account. An account can be obtained by following the registration link and filling out two short forms. This will create a new account for the User, create their profile and store their details in the database.

By default, one Master account is created. This is the Administration account and this can be used to promote Users who have registered to Lecturers. The system only allows Students to register to ensure a stricter security policy.

## 2.2.3 Environmental Requirements

### 2.2.3.1 Client Side

The User must have a device with an active internet connection which also supports JavaScript.

### 2.2.3.2 Server Side

When deployed, it is recommended that the System is created and maintained using a PostgreSQL Database. This is not mandatory unless the service you deploy to has explicit requirements. Heroku is an example of a Cloud service which enforces PostgreSQL.

The application must also have the database seed file executed after the database is generated in order to have access to the default administrator account.

The applications assets should be precompiled before deployment.

Finally, the application is dependent on a number of Gems, located in the Gemfile. All of these Gems must be installed before operation.

## 2.2.4 Usability Requirements

### 2.2.4.1 Cross Platform

The application is designed to work on any device. This is to cater for Students and Lecturers from all walks of life with access to varying devices, whether it be their smartphone in their pocket or a computer in-front of them in a lab.

This requirement is fulfilled by using Twitter's Bootstrap framework and CSS Media Queries in some specific cases.



#### **2.2.4.2 Modern Design Interface**

The interface for the application is intentionally minimalistic and incredibly simple. It is very rare in the application to have a bunch of data crammed into one box. This makes the User Experience incredibly pleasing and actually streamlined as it is clear where you can click buttons and navigate to.

The application follows Flat UI Design Principles and Colour Schemes. All the colours used in the application are recommended by Flat UI Designers. References to Flat UI websites can be found in the bibliography.

#### **2.2.4.3 Intuitive**

A big requirement in my eyes as was to have the System as intuitive as possibly. Having gone through 4 years of college and taught numerous classes during my time, it is clear not every Student / Lecturer is tech-savvy.

Thanks to the Modern Design and as a direct result of the implementation, the application uses bright and clear colours to direct Users towards clickable buttons. The application also Uses what I believe to be a simple navigation System whereby instead of crowding pages with back buttons the User can simply click their Dashboard button on the Navbar to go back home. This works perfectly because there's no large number of nested pages and each page is at maximum 3 clicks from the dashboard.

#### **2.2.4.4 Security**

User emails and passwords are stored securely through the Devise gem. This makes sure the data will not be easily exposed.

Access to certain pages is restricted. This means a Student cannot access the Lecturer or Admin side of the application just by knowing the address for the page.

## 2.3 Design and Architecture

### 2.3.1 Classistant Application Architecture

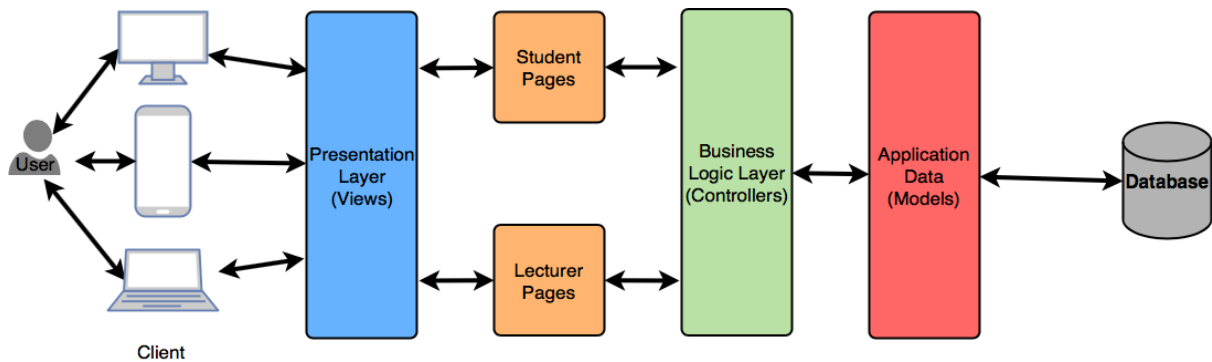


Figure 5 – Classistant Application Architecture

Classistant is designed using the Model View Controller (MVC) Principle. This is implemented by using Ruby on Rails.

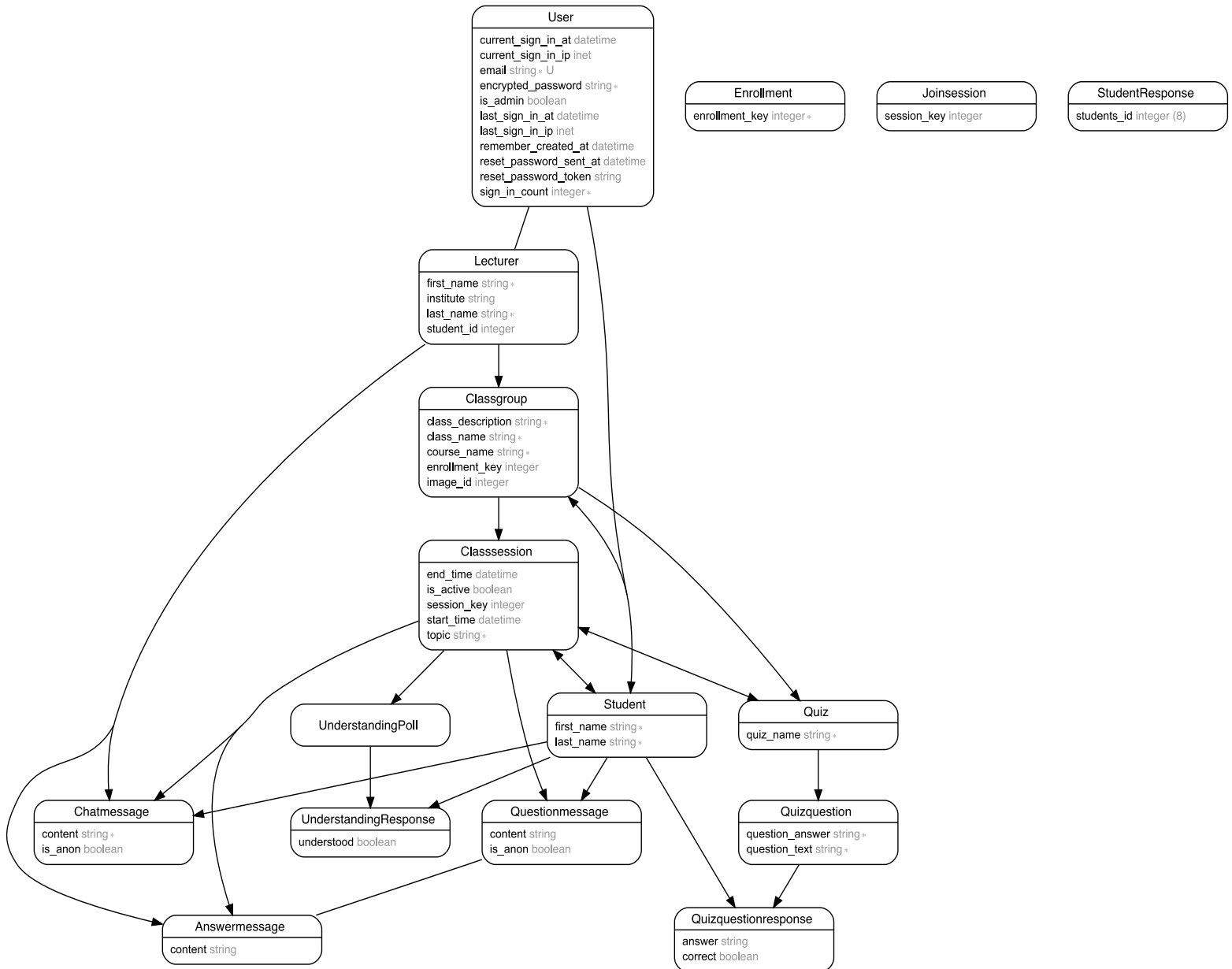
The application is intended to be run on any device with an active internet connection and JavaScript support. As shown in the Architecture Diagram, PCs, Laptops and Mobile Devices all support Classistant.

The flow of the application is actually really simple thanks to the MVC implementation and it ended up working perfectly to implement the System I wanted. Ruby on Rails operates based on a route file and exposes pages inside the application based on the contents of this file. This actually allows for certain pages to not be shown or accessible based on conditions. An example of this would be to not expose any navigation unless the User is signed in.

## 2.3.2 Classistant Application Entity Relationship Diagram

Figure 6 - Classistant Domain Model

Classistant domain model



### 2.3.3 Core Components

#### 2.3.3.1 Layouts

Layouts are essentially a template inside which your page is rendered. A good way to conceptualise this concept is almost like a container or box but in our case the contents is the page we want to render. This means generic styling can be applied application wide.

Classistant Utilises three layouts, each with slight tweaks for the optimum experience depending on the page. The layouts are called

*application.html.erb*  
*session.html.erb*  
*landing.html.erb*

*Note: More layouts exist in the application due to the existence of a mailer but the purpose is slightly different*

A perfect example of why separate layouts would be used is the application layout doesn't load all of the JavaScript for the application. The real-time part of the application uses special scripts to manage and maintain the live connection.

Therefore, a separate layout for the sessions is used (*session.html.erb*) which loads in the necessary JavaScript on the sessions page in order to make the session work.

#### 2.3.3.2 Views

Views in Ruby on Rails are essentially the Web Pages you wish to render. This concept is extremely confusing to develop with at first, especially when using Twitter's Bootstrap as you have to remember the code you are writing in a View file is wrapped in the Layout file. Each view has a corresponding route and controller which controls it.

#### 2.3.3.3 Controllers

Controllers determine what page should be rendered based on the action that is called. Controllers are typically mapped to a model but not always. By default, controllers have basic Create, Read, Update, Delete (CRUD) functionality. This however, is not mandatory and custom actions can be defined.

An example of a controller tied to a model would be the ClassSession Controller. This Controller controls the classsession model and allows for sessions to be created. Not only does the controller allow for CRUD functionality but it also

allows for sessions to be reviewed which is a custom action. This takes the route `classsessions/session_id/review`

In Classistant, the Dashboard Controller only contains one method and is perfect to show a small-scale example of what a controller can do and how it operates.

```
class DashboardController < ApplicationController
  def index
    if current_user.lecturer.nil?
      @student = current_user.student
      render 'student'
    else
      @lecturer = current_user.lecturer
      render 'lecturer'
    end
  end
end
```

The index method maps the view “index.html.erb” located inside the Dashboard controller in “app/views” Therefore, when a User tries to access that page, the method will execute.

#### 2.3.3.4 Models

Models are essentially like Objects in OOP. A model is a representation of your data structure for an object or table in the database. For example, a model can have attributes and validations for the model as well as relationships.

Classistant uses models for everything in the System and they are incredibly powerful. Thanks to the verbose use of models, Classistant is able to get the number of questions a Student has asked by simply looking at a Student, getting all the QuestionMessages associated with them and count the amount of associations.

A small example of a model is a chat message model

```
class Chatmessage < ApplicationRecord
  belongs_to :classsession
  belongs_to :student, optional: true
  belongs_to :lecturer, optional: true

  validates :content, presence: true
end
```

#### 2.3.3.5 Migrations

Migrations are used to create the tables which models map to. It is crucial that migrations map to models (Models need to define their relationships inside the class) or else the Application’s Database will not run. The filenames also need to

match which I unfortunately learned the hard way during the development of the application.

The chatmessage migration is perfect for showing what a small migration looks like.

```
class CreateChatMessages < ActiveRecord::Migration[5.1]
  def change
    create_table :chatmessages do |t|
      t.references :classsession, foreign_key: true
      t.references :student, foreign_key: true, null: true
      t.references :lecturer, foreign_key: true, null: true
      t.string :content
      t.boolean :is_anon
      t.timestamps
    end
  end
end
```

### 2.3.3.6 Gems

Gems are essentially plugins or libraries for your application. They are compiled before the application is launched and they allow for extended functionality on top of the Rails framework.

Gems are defined in the applications Gemfile. I used bundle to install my gems and keep them up to date.

An example of a Gem I used is ChartKick. ChartKick implements a graphing framework of your choice (Google Charts, Charts.js or HighCharts.js) and allows you to use the library using Ruby code and variables instead. This was hugely beneficial as it decreased the amount of code needed exponentially. To get an idea of scale, roughly 100 lines of previously coded chart configuration in JavaScript was replaced by 3 lines of embedded Ruby.

An example of gem declaration in the Gemfile is as follows

```
# Charting Library for Real-time Charts
gem 'chartkick'
```

*Note: The Comment is not essential but it is good practice.*

### 2.3.3.7 Partial

Partials are essentially a snippet of HTML code you can render inside a view. The idea of a partial is to help enforce the Don't Repeat Yourself (DRY) principle

of programming. This means if you use the same code in multiple places for a Web Page such as a Navbar or some kind of Title, you could save it in a partial file and call it from anywhere in the application to be rendered. An example of a partial file can be seen below.

```
<% @messages.each do |msg| %>
  <% if msg.lecturer.nil? %>
    <% if msg.is_anon %>
      <p><span class="chat-timestamp"><%= msg.created_at.strftime('%H:%M') %></span> Anonymous:
<%= msg.content %></p>
    <% else %>
      <p><span class="chat-timestamp"><%= msg.created_at.strftime('%H:%M') %></span> <%=
msg.student.full_name %>: <%= msg.content %></p>
    <% end %>
  <% else %>
    <p class="lecturer-message"><span class="chat-timestamp"><%= msg.created_at.strftime('%H:%M')
%></span> <span class="lecturer-name"><%= msg.lecturer.full_name %></span>: <%= msg.content
%></p>
  <%end %>
<% end %>
```

This partial is rendered inside a Div in a Class Session View. The partial simply builds / renders a list of all the messages that have been sent in the chat for the class thus far.

### 2.3.3.8 Channels

Channels allow for real-time communication between browser sessions. A channel works very similarly to MQTT Systems which I became extremely familiar with during the Internet of Things Stream. Channels are extremely complex to implement and definitely took the longest of all my features.

Sessions essentially work by first using CoffeeScript to establish a connection

```
#Subscribe to the Session
App.session = App.cable.subscriptions.create {channel: "SessionChannel", room: s_id},

  connected: ->
# Called when the subscription is ready for use on the server

  disconnected: ->
# Called when the subscription has been terminated by the server

  received: (data) ->
# Called when there's incoming data on the websocket for this channel
  payload = data['message']
```

In the Header of the Subscription we can see “room: s\_id”

This means we can subscribe to a specific room on the channel. This is fantastic and incredibly useful as we can separate out each Class Session into a different room. This mean messages will never accidentally get sent to the wrong Class Session.

Finally, Methods can be called to the Session's corresponding ruby file using the following syntax

```
@perform 'send_chat_message', message: payload, room_id: s_id
```

The 'send\_chat\_message' *directly* maps to the name of a function in the Ruby File. So this line of code would call the following method

```
class SessionChannel < ApplicationCable::Channel
  def subscribed
    stream_from "session_channel_#{params[:room]}"
  end

  def unsubscribed
    # Any cleanup needed when channel is unsubscribed
  end

  def send_chat_message(data)

    #Take in the Payload from the sent Message
    @payload = data['message']

    #Get the contents of the payload
    @content = @payload['content']
    @uid = @payload['uid']
    @utp = @payload['utp']
    @sid = @payload['sid']
    @anon = @payload['anon']

    #Parse the User ID and User Type (Checks if they are somehow Strings, if so, cancel the rest of the
    action)
    @uid = Integer(@uid) rescue -100
    @utp = Integer(@utp) rescue -100
    @sid = Integer(@sid) rescue -100

    #Cancelling in the case of String
    if @uid == -100 or @utp == -100 or @sid == -100

    else
      #This far means the message is A O K to send

      #Save the message to our DB
      @save_message = Chatmessage.new
      @save_message.content = @content

      #Get the Session for the Message and Associate them
      @session = Classsession.find(@sid)
      @save_message.classsession = @session

      #Get the User for the Message
      @user = User.find(@uid)

      if @utp == 191
        #lec
        @user = @user.lecturer
        @save_message.lecturer = @user
      end
    end
  end
end
```



```

elsif @utp == 4
  #stu
  @user = @user.student
  @save_message.student = @user
end

if @anon == "t"
  @save_message.is_anon = true
else
  @save_message.is_anon = false
end

if @save_message.save
  @payload['timestamp'] = @save_message.created_at.strftime('%H:%M')
  @payload['name'] = @user.full_name
  @payload['type'] = 'chat'
  ActionCable.server.broadcast "session_channel_#{@sid}", message: @payload
end

#End of Message being authentic and sending code
end
end

```

### 2.3.3.9 Mailers

Mailers can be used to send Emails from the Rails Application. I implemented Mailers to allow lecturers to notify students at the click of a button if they are not doing well in their class. This is split into Poor Attendance and Poor Understanding. Lecturers can see Students who are not doing well in their Class on the page for that class and there, they have the option to fire off the emails.

In order to support this feature, I created two new custom routes for the student's controller

```

def send_support
  @student = Student.find(params[:student_id])
  @classgroup = Classgroup.find(params[:classgroup])
  ClassassistantMailer.with(student: @student, classgroup: @classgroup).support_email.deliver_now
  redirect_to('/classgroups/' + @classgroup.id.to_s)
end

def send_attendance
  @student = Student.find(params[:student_id])
  @classgroup = Classgroup.find(params[:classgroup])
  ClassassistantMailer.with(student: @student, classgroup: @classgroup).attendance_email.deliver_now
  redirect_to('/classgroups/' + @classgroup.id.to_s)
end

```

Next, I added the routes to accommodate this in my routes.rb file.

```

resources :students do
  get :send_support
  get :send_attendance
end

```

Finally, I added the code for the Mailer to support sending both emails.

```
class ClassistantMailer < ApplicationMailer
  default from: "classistant@protonmail.com"

  def support_email
    @student = params[:student]
    @classgroup = params[:classgroup]
    @lecturer = @classgroup.lecturer

    @user = @student.user

    mail(to: @user.email,
         subject: 'Classistant | Notice from Lecturer: ')
  end

  def attendance_email
    @student = params[:student]
    @classgroup = params[:classgroup]
    @lecturer = @classgroup.lecturer
    @user = @student.user

    mail(to: @user.email,
         subject: 'Classistant | Notice from Lecturer: ')
  end
end
```

These Methods are mapped directly to layouts with the same name. An example is the following layout, support\_email.html.erb

```
<!DOCTYPE html>
<html>
<head>
  <meta content='text/html; charset=UTF-8' http-equiv='Content-Type'/>
</head>
<body>
<h1>This is an Urgent Notice from Classistant</h1>
<p>
  Dear <%= @student.full_name %>,
</p>
<p>
  It appears that you are not performing so well in <%= @classgroup.class_name %>
</p>
<p>
  Would it be possible for you to come meet me and we can discuss your performance in person. I want to
  ensure you are able to keep on top of things and don't fall behind
</p>
<p>
  Kind Regards,
</p>
<p>
  <%= @lecturer.full_name %>
</p>
</body>
</html>
```

## 2.4 Code Snippets and Explanation

### 2.4.1.1 Dashboard Controller

```
class DashboardController < ApplicationController
  def index
    if current_user.lecturer.nil?
      @student = current_user.student
      render 'student'
    else
      @lecturer = current_user.lecturer
      render 'lecturer'
    end
  end
end
```

The Dashboard Controller simply controls what Dashboard should be Displayed to the user when they navigate to the Dashboard URL. Each User is associated with a Student or Lecturer model, and this is how we can distinguish them.

The above code is an incredibly simple yet effective conditional statement which reads

*If the current user is not a lecturer, render the student view*

*Otherwise, render the lecturer view.*

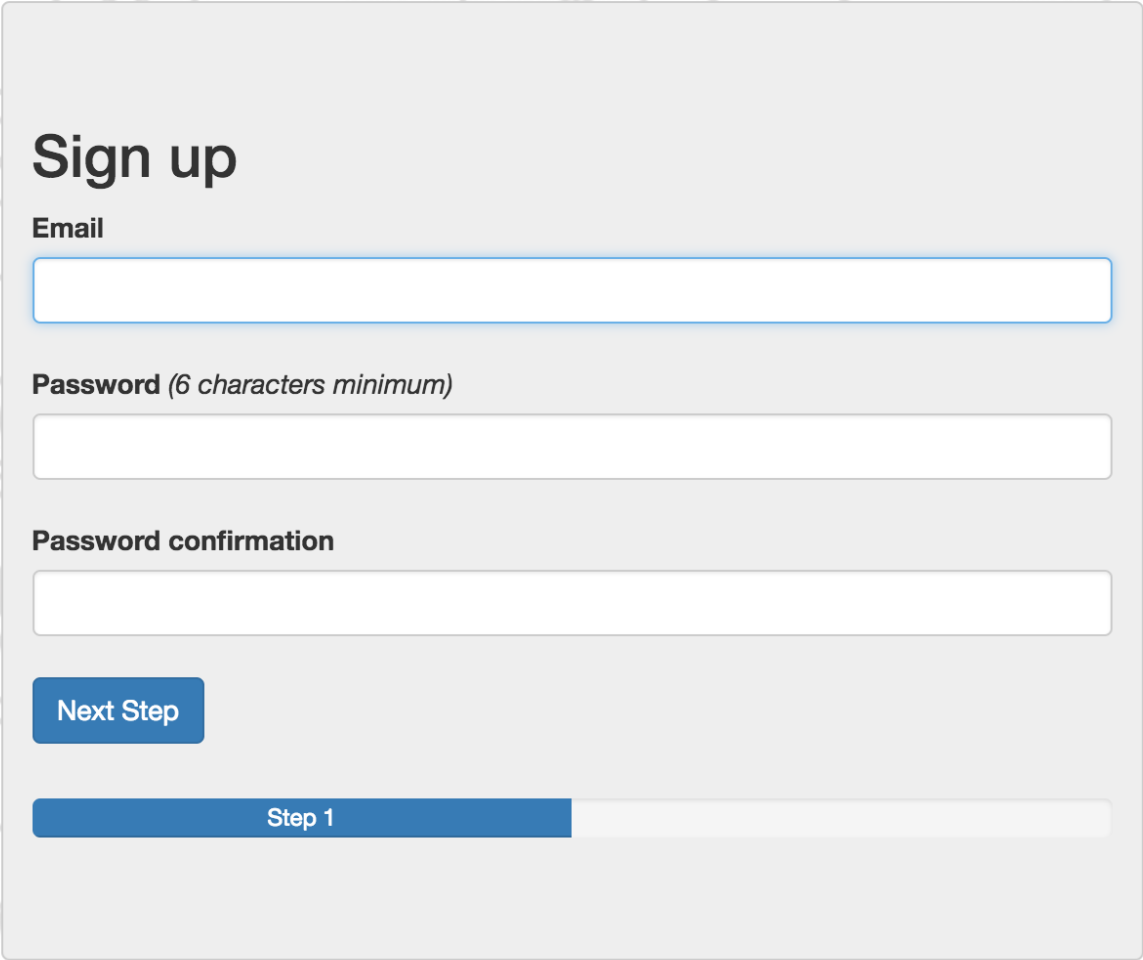
The line

```
@student = current_user.student
```

Just saves the instance of the Student into a variable called @student. We can later access this from the Student's Dashboard view if needs be. The same applies for the @lecturer declaration in the Lecturer view.

It is worth noting at this point that current\_user is a singleton which contains the instance of the User signed in.

### 2.4.1.2 Enforcing Full User Registration



The image shows a 'Sign up' form titled 'Sign up' in a large, bold font. Below the title are three input fields: 'Email', 'Password (6 characters minimum)', and 'Password confirmation'. Each field is a simple white rectangle with a thin blue border. Below the 'Password confirmation' field is a blue button with the text 'Next Step' in white. At the bottom of the form is a progress bar consisting of a blue segment on the left and a grey segment on the right. The blue segment contains the text 'Step 1' in white. The entire form is set against a light grey background with a faint, repeating pattern of the word 'Classistant'.

*Sign Up Form (Step 1) - Classistant*

When Registering, a User must complete two steps to Registration. First, the User types in their email and password they wish to use.

Once the user clicks “Next Step”, the User model (If the inputs are valid) will be created in the Database. Next, we need to create an instance of a Student to associate to the Model.

**You're almost there..**

**First Name**

**Last Name**

**Complete Registration**

**Step 2**

*Sign Up Form (Step 2) - Classistant*

However, at this point, the User could technically navigate away from this page and not create an instance of the Student to associate to a User, thus breaking the application.

To solve this, there is a Controller that controls the whole Application through which everything is routed. So I simply added a method which would check for the association to exist, and if it didn't, it would reroute the User back to this page

```
class ApplicationController < ActionController::Base
  protect_from_forgery with: :exception
  before_action :authenticate_user!, :check_user_has_identity

  def check_user_has_identity
    if user_signed_in? and current_user.lecturer.nil? and current_user.student.nil? and not request.fullpath
      == '/students/new'
      redirect_to '/students/new' unless request.fullpath == '/students'
    end
  end
end
```

This code snippet is located in the ApplicationController class. It reads

*Before the User changes page, authenticate the user and check that they have an identity (Student / Lecturer)*

Then there is a method below which checks if there is a User signed in (So you don't get caught out as a new User on the landing page) and if the user has no Identity, if so, send them back to the form (Assuming they aren't already going there)

## 2.5 Graphical User Interface (GUI) Layout

This section contains various screenshots from the Classistant UI. The screenshots range from Laptop Screens, to phones and tablets. Naturally, not *every single screen* of the System is shown as it would simply be too much. Instead, there's a hand-picked number of important screens shown.

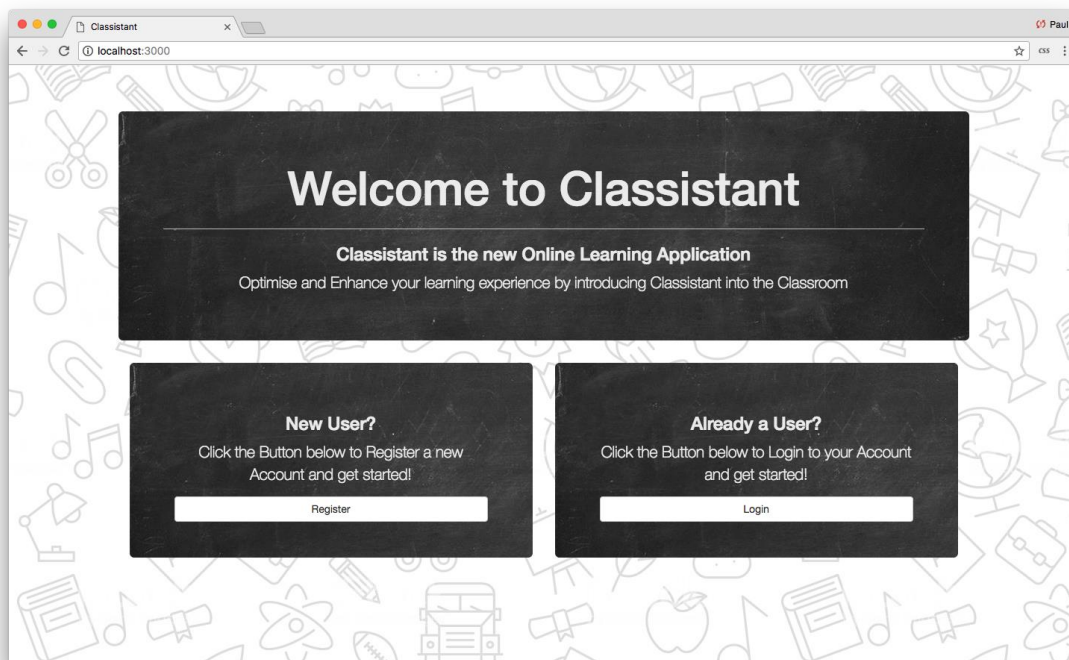


Figure 7 - Classistant Landing Page (Not Signed In, Laptop)

The screenshot shows a web browser window with the address bar displaying 'localhost:3000/users/sign\_up'. The page has a navigation bar with links for 'Classistant', 'Dashboard', 'Register', and 'Login'. The main content area is a sign-up form titled 'Sign up'. It includes three input fields: 'Email', 'Password (6 characters minimum)', and 'Password confirmation'. Below the password fields is a blue 'Next Step' button. At the bottom of the form is a progress bar with a blue segment labeled 'Step 1'. The background of the page is a light gray with a repeating pattern of various educational icons.

Figure 8 - Classistant Sign Up Page (Laptop)

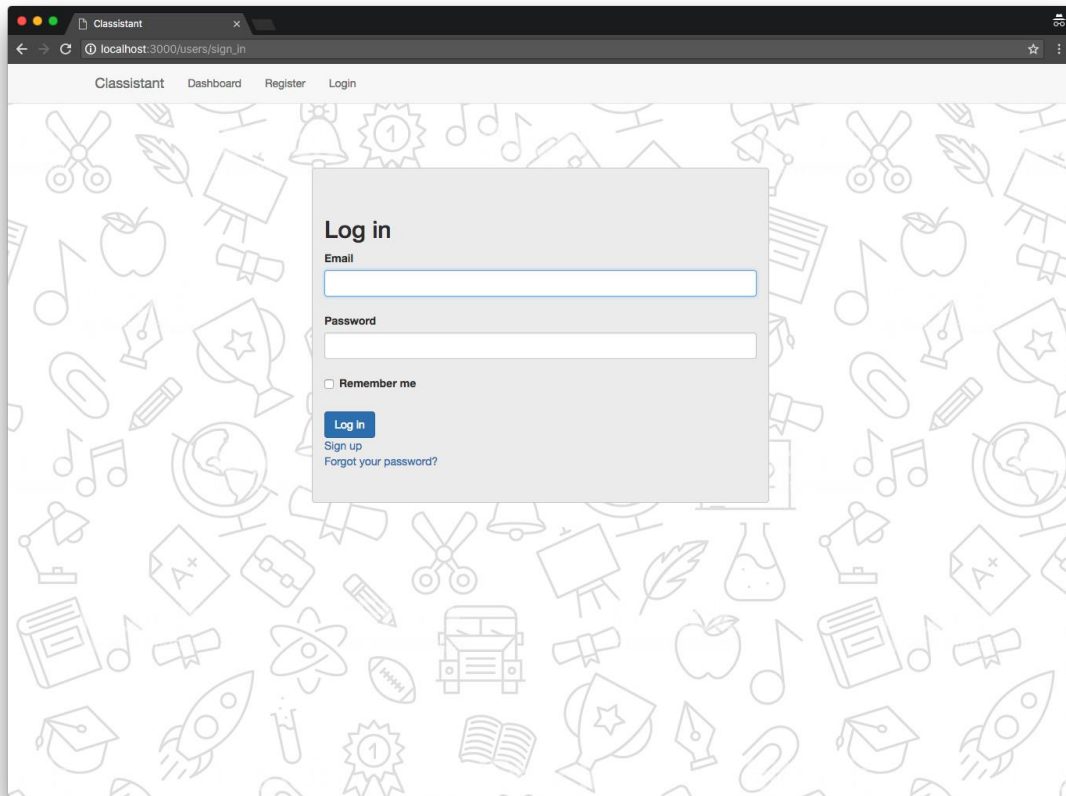


Figure 9 - Classistant Sign Up Page (Laptop)



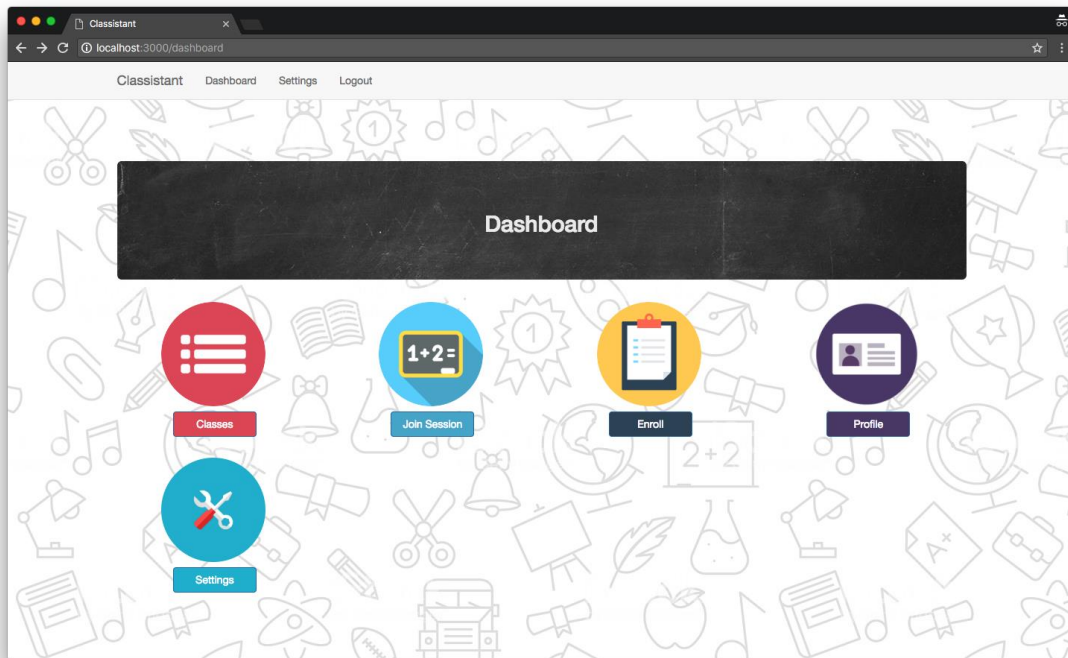


Figure 10 - Classistant Student Dashboard (Laptop)

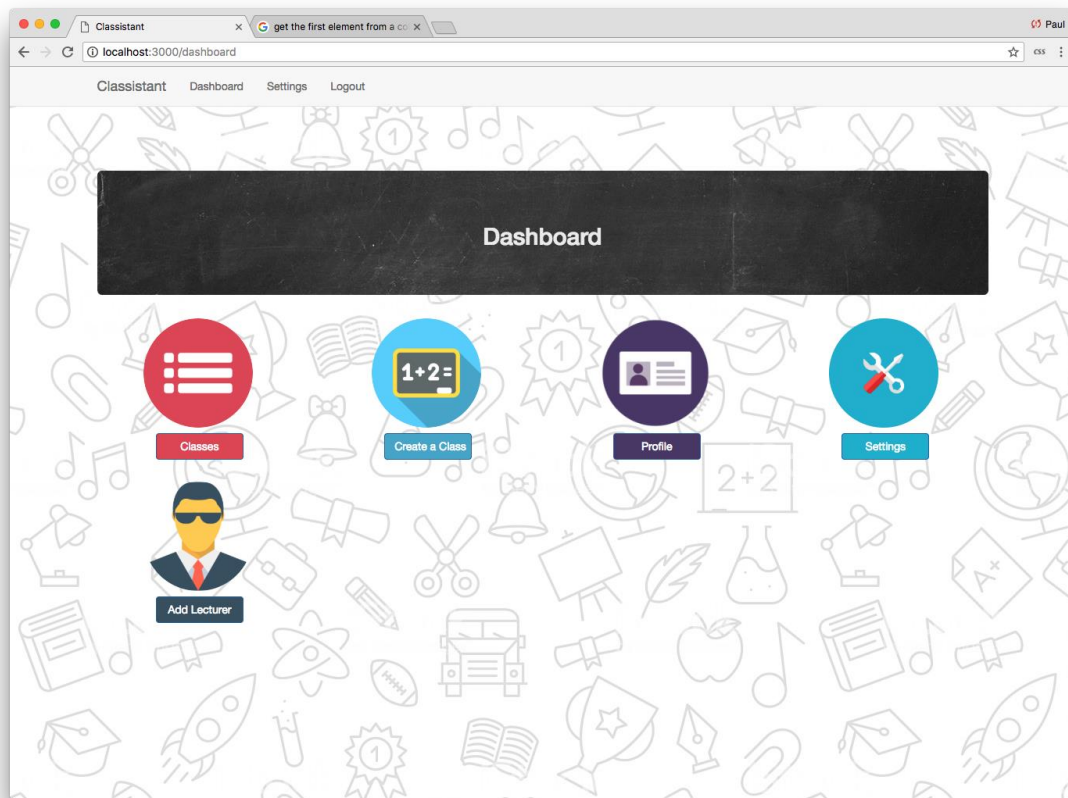


Figure 11 - Classistant Admin Dashboard (Laptop)

Classistant | Dashboard | Settings | Logout

## Settings

### Edit Details

Email  
paulreid@mail.com

Password (leave blank if you don't want to change it)

6 characters minimum

Password confirmation

Current password (we need your current password to confirm your changes)

Update

Back to Dashboard

### Cancel my account

Unhappy?

Cancel my account

Figure 12 - Classistant Settings Page (Laptop)

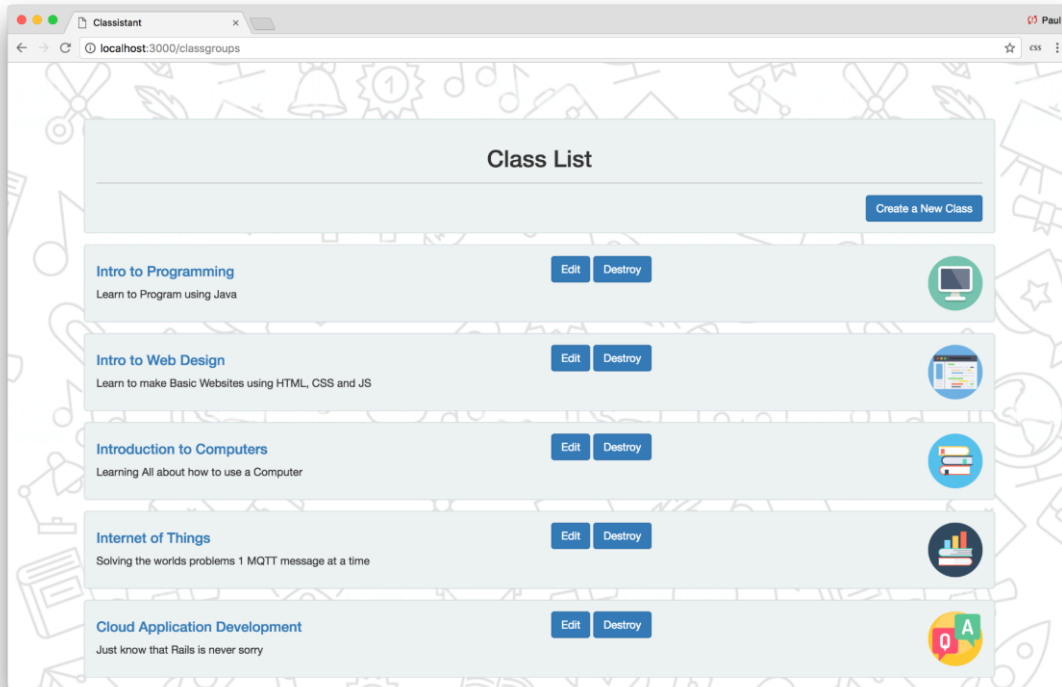


Figure 13 - Classistant Lecturer Class List (Laptop)

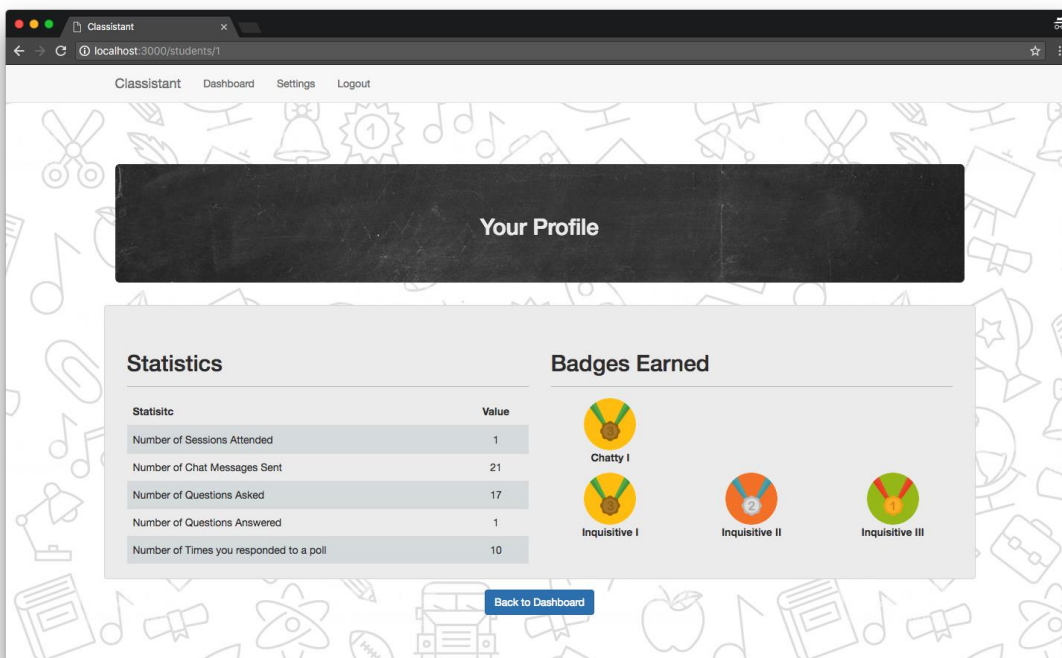


Figure 14 - Classistant Student Profile (Laptop)

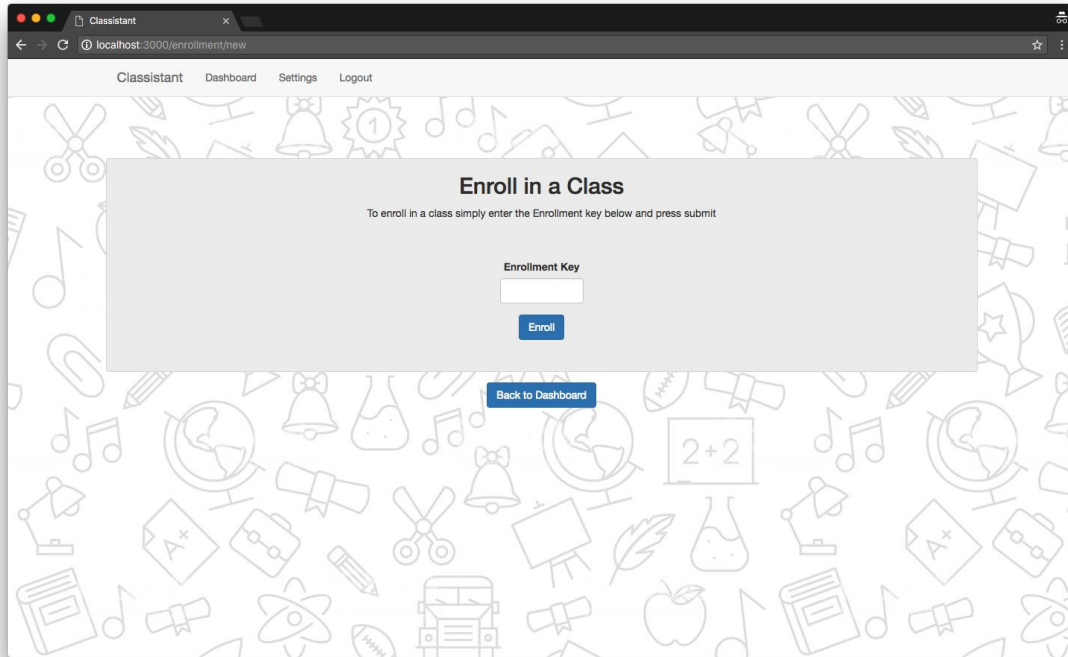


Figure 15 - Classistant Enroll in a Class (Laptop)

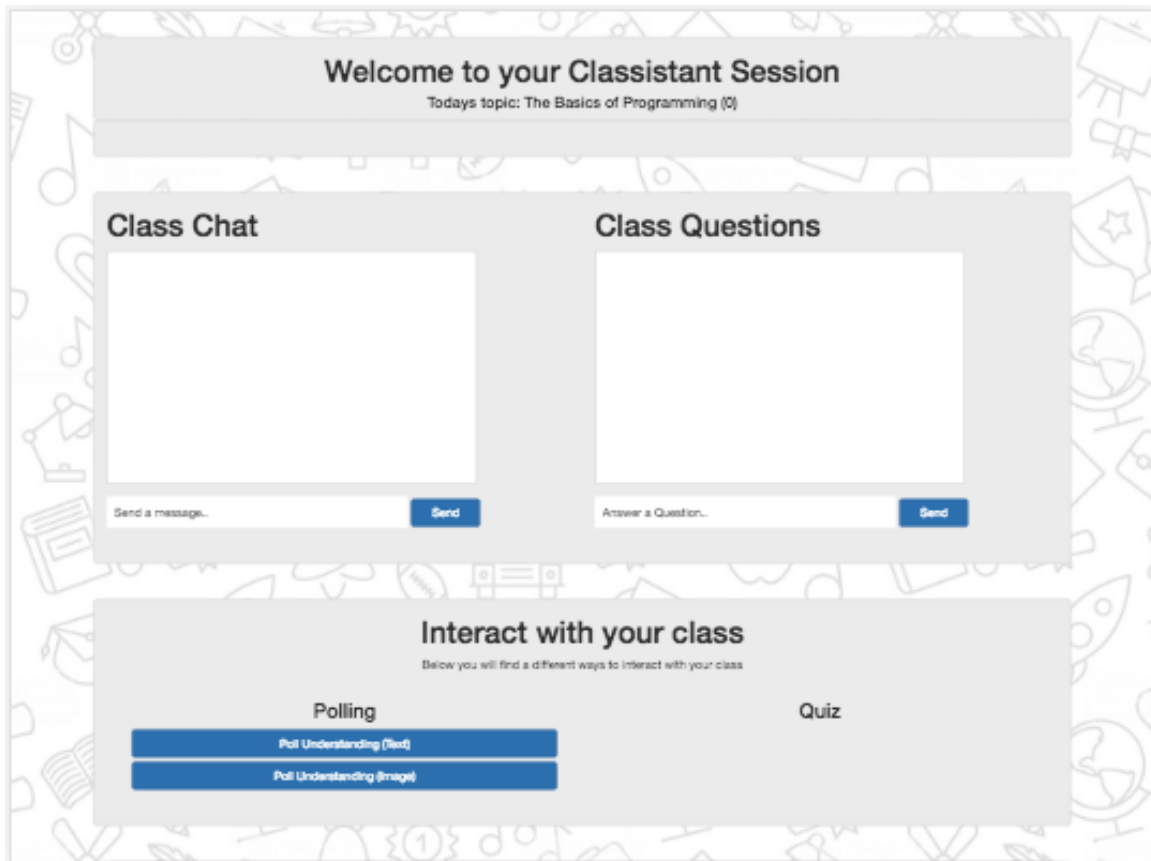


Figure 16 - Classistant Lecturer Class Session (iPad)

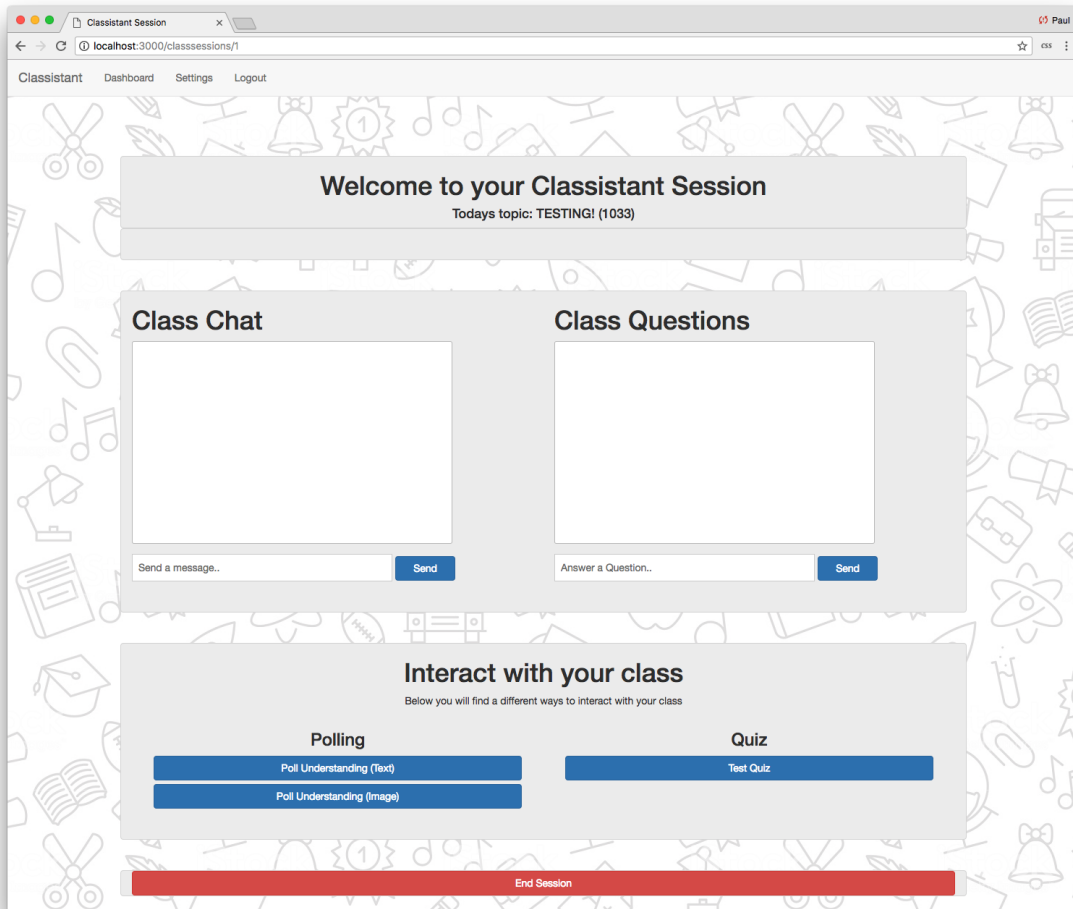


Figure 17 - Classistant Lecturer Class Session View (Laptop)

The screenshot displays the 'Classistant' web application in a browser window. The page is titled 'Intro to Programming - Overview' and shows course details: Course: BSHC, Lecturer: Paul Reid, and Enrollment Key: 9109. Below this is a 'List of Sessions' table with columns for Session Key, Session Topic, Session Start Time, Session End Time, Attendance, Review, and Join. The table lists three sessions: 'The Basics of Programming', 'TESTING!', and 'Testing!'. Below the sessions is a 'List of Students' section showing one student, Aaron Meaney, with columns for Student ID, Name, Attendance, Understanding, Quiz Performance, Send Attendance Email, and Quiz Performance. The bottom section is titled 'Quizzes' and shows a list of quizzes for the class, with a 'Create a Quiz' button.

### Intro to Programming - Overview

Course: BSHC  
Lecturer: Paul Reid  
Enrollment Key: 9109  
"Learn to Program using Java"

[Edit Class](#) [Delete Class](#)

### List of Sessions

[Start new Session](#)

Session Key	Session Topic	Session Start Time	Session End Time	Attendance	Review	Join
0	The Basics of Programming	05/12/18 - 11:32	05/12/18 - 12:32	0	<a href="#">Review</a>	
1033	TESTING!	05/12/18 - 12:01		1	<a href="#">Review</a>	<a href="#">Join</a>
2747	Testing!	05/13/18 - 18:17		1	<a href="#">Review</a>	<a href="#">Join</a>

### List of Students

Number of Students Enrolled: 1

Student ID	Name	Attendance	Understanding	Quiz Performance	Send Attendance Email	Quiz Performance
1	Aaron Meaney	2 / 3	0 / 11	3 / 115	<a href="#">Send Attendance Email</a>	<a href="#">Send Performance Email</a>

### Quizzes

Below is a list of Quizzes for this class

[Create a Quiz](#)

Quiz Name	Delete Button Goes here
Test Quiz	

Figure 18 - Classistant Lecturer Class View (Laptop)

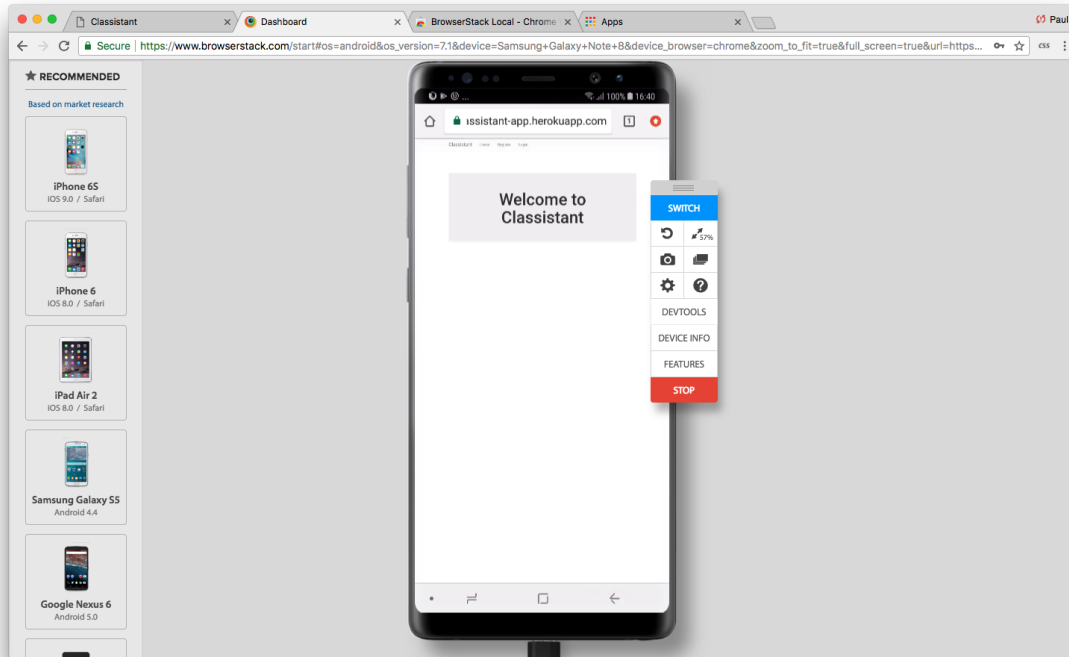


Figure 19 - An older UI Page being Tested using Browser Stack

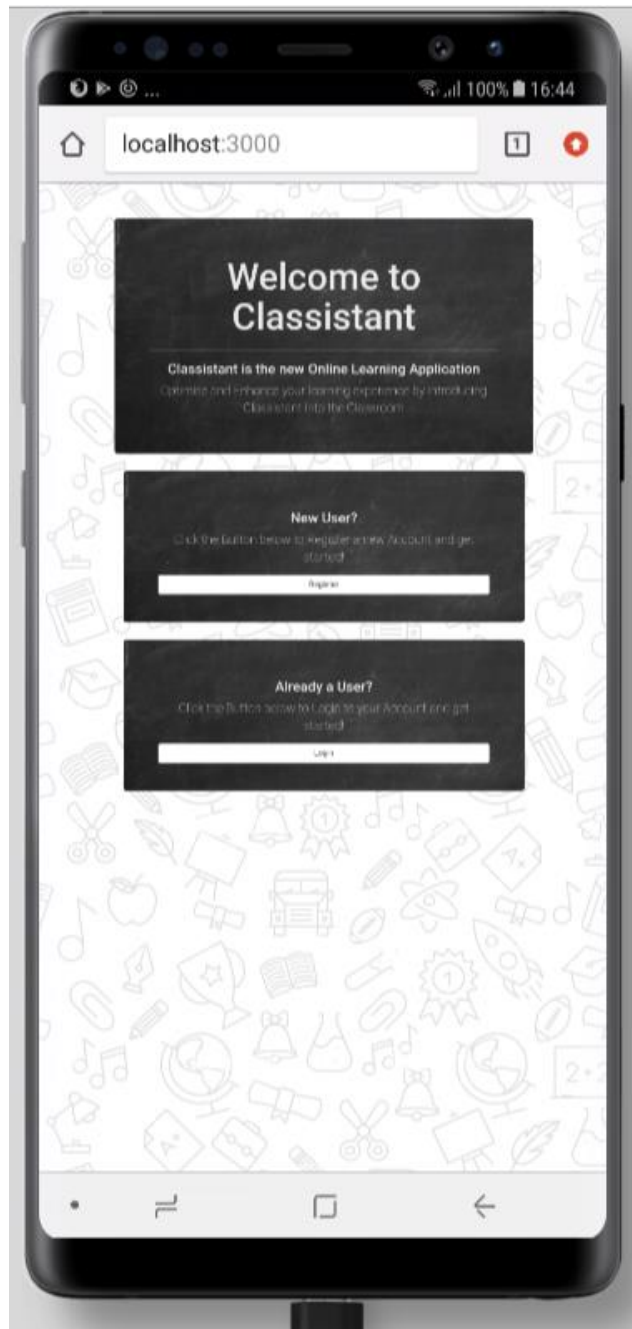


Figure 20 - Classistant Landing Page (Galaxy Note 8)



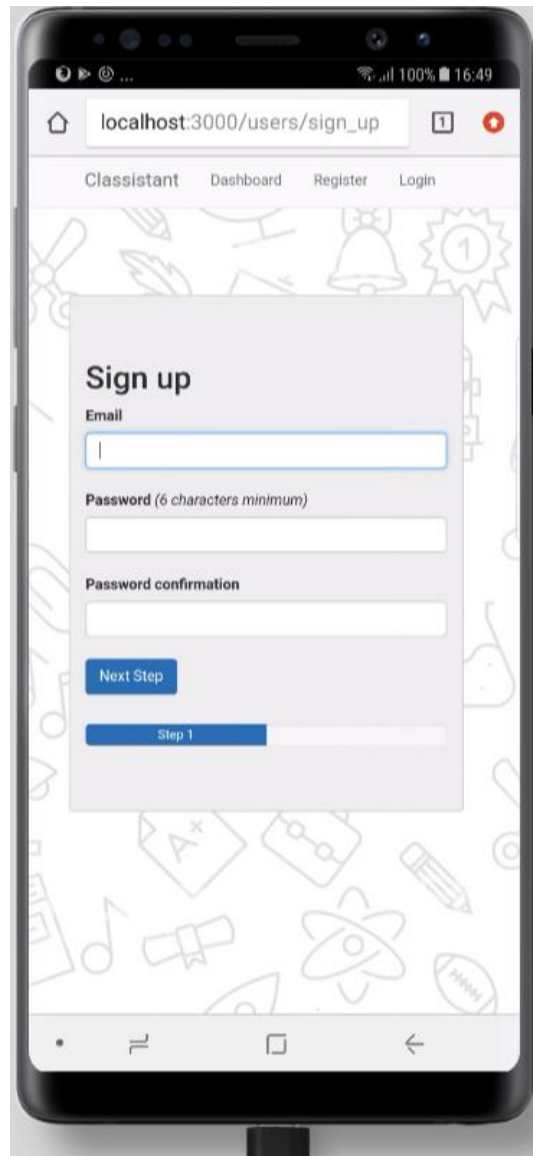


Figure 21 - Classistant Sign up Page (Galaxy Note 8)

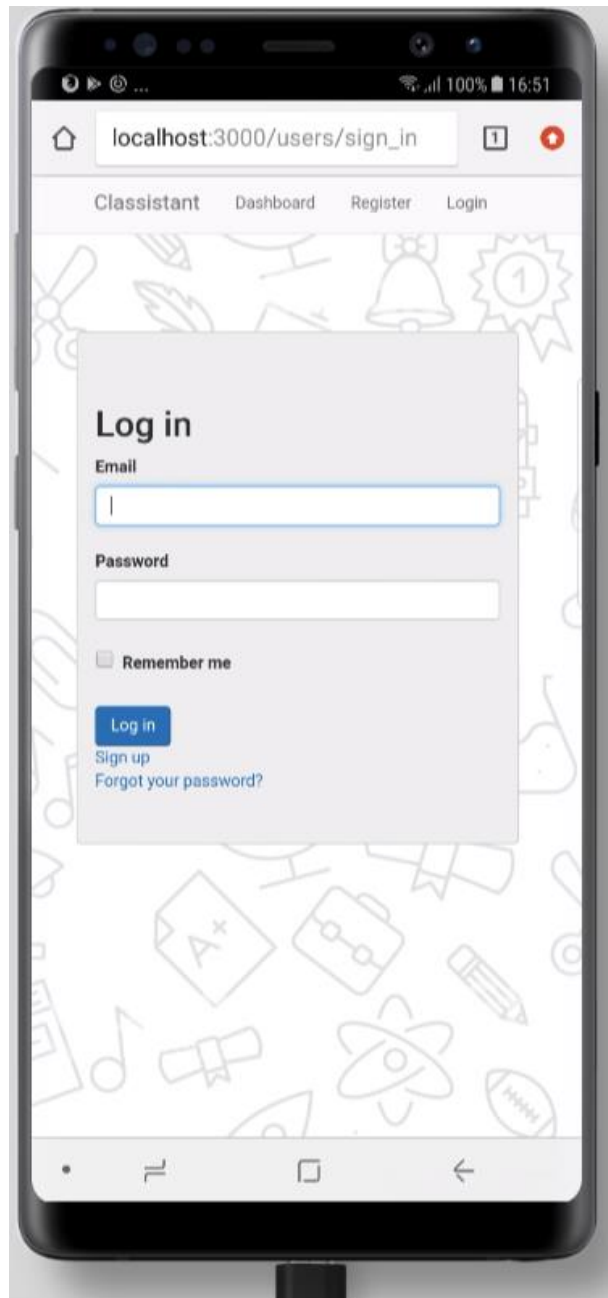


Figure 22 - Classistant Login Page (Galaxy Note 8)

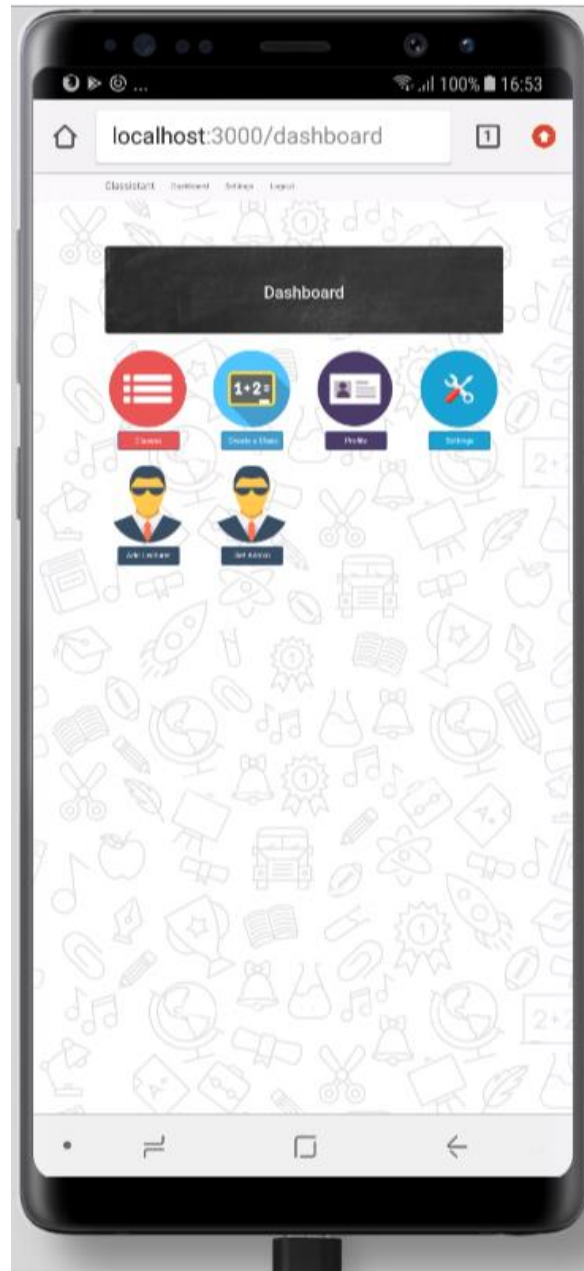


Figure 23 - Classistant Admin Dashboard Page (Galaxy Note 8)

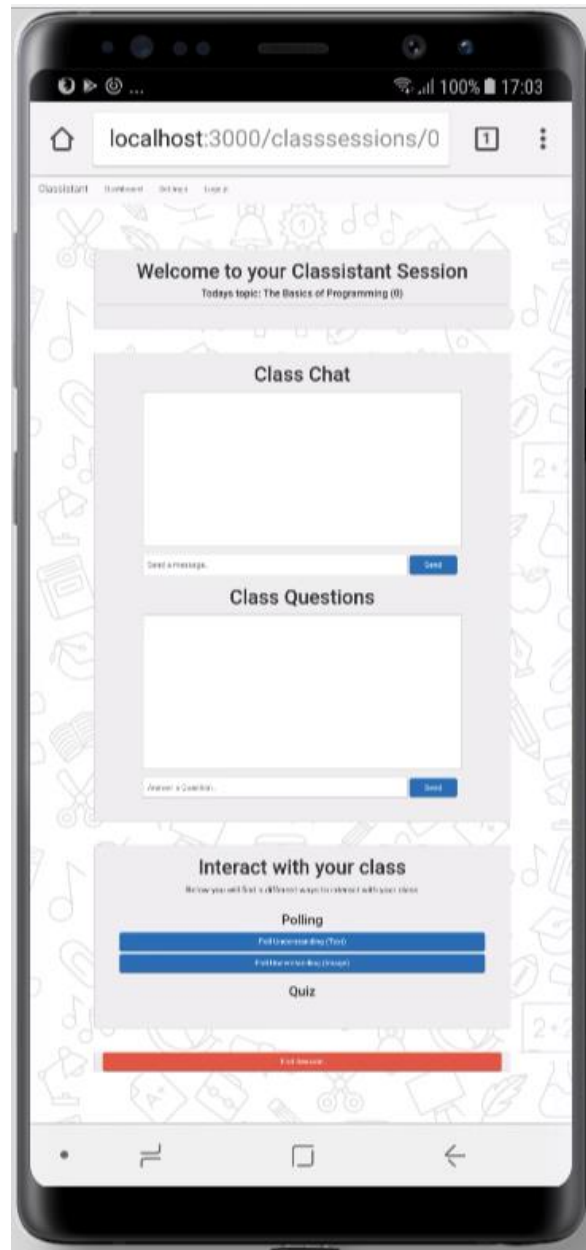


Figure 24 - Classistant Lecturer Class Session Page (Galaxy Note 8)

## 2.6 Testing & Evaluation

Classistant is difficult in some respects to test due to it being a Web Application. However, I did test the application in a number of different ways which will be outlined below.

### 2.6.1 Iterative Testing

During the development of the Application I would start with a very small piece of logic and test it to ensure it works doing sanity tests and just trying to break it. I would then slowly build upon the logic piece by piece until I reached my final goal. This was actually extremely useful and effective and allowed me to make very few mistakes in my logic for the Application during Development

### 2.6.2 Usability Testing

During Development and post development, I had various people attempt to navigate my Application and provide me with constructive feedback. Participants ranged in technical ability from my mother who isn't very tech-savvy, to my girlfriend who can use a phone and computer right up to my brother who works in the IT Industry as well as my peers in class.

Overall, this had a huge impact on UI design and helped me streamline the application down to exactly what I need.

The process for this type of testing was simple. I would sit the User in-front of my laptop with the Application landing screen and ask them to explore the application and provide me with feedback. I would note down the feedback in a copy book I have and later incorporate it into my design.

### 2.6.3 Performance Testing

Due to the Application being Web Based and having real time interactions, I was curious to see if the Application would break or throttle with a large number of Users. I opened numerous tabs and logged into the Application (Incognito so sessions weren't tracked) and got some peers from my class and previous project groups to do the same and we registered and logged in multiple accounts.

We decided to try overload the messaging system by posting numerous posts at the same time. This only showed us that the Application worked as expected and I was thoroughly pleased that the way I implemented Web sockets worked without difficulty.

### 2.6.4 Browser Testing

I tested the Application and its code across multiple browsers. I personally use a mix between Google Chrome, Firefox and Safari depending on my device as well as the default web app for my phone. All of these browsers worked perfectly with no issues (Thanks to using some universal CSS) and all of my tests were successful. I didn't even attempt to try anything older like IE as I would have no interest in supporting it fully. That said, the application doesn't have a large amount of browser specific code / rules.

Luckily, a previous job I had as a Web Developer left me with a BrowserStack account which is a website that emulates devices through the browser and lets you test your Websites on them. I tried the Galaxy Note 3, Note Tab, Various iPhones as well as my own Phone, a Samsung Galaxy Note 8 and my iPad.

Again, all of the results were (surprisingly) positive and no browser had any main issues. Even my main monitor for development which is a 2K 34" Ultra-wide (LG34UC97C) Displays the website fine.

### 2.6.5 Model Testing

Using the Power of the seeds.rb file in Rails, I was able to write and attempt to create a large quantity of instances for various models. Thankfully, certain ones designed to fail, failed and did not persist, thus showing me that the models were set up correctly.

Below is an example of a seeds file

```
@admin = User.new(id: 9999, email: "paulreid@mail.com", password: "123456", is_admin: true)
@paul = Lecturer.new(first_name: "Paul", last_name: "Reid", institute: "NCI", user_id: 9999)
@admin.lecturer = @paul

@paul.save
@admin.save

# Creating Students to Populate the System

@student = User.new(id: 9090, email: "paulreid96@gmail.com", password: "123456", is_admin: false)
@aaron = Student.new(first_name: "Aaron", last_name: "Meaney", user_id: 9090)
@student.student = @aaron

@aaron.save
@student.save

# Creating Classes to Populate the System

@itp = Classgroup.new(id: 0, class_name: "Intro to Programming", course_name: "BSHC",
  class_description: "Learn to Program using Java", enrollment_key: 9109, image_id: 1,
  lecturer_id: 1)
@itp.students << @aaron
@itp.save

@iwd = Classgroup.new(id: 1, class_name: "Intro to Web Design", course_name: "BSHC",
  class_description: "Learn to make Basic Websites using HTML, CSS and JS",
  enrollment_key: 9101, image_id: 2, lecturer_id: 1)
@iwd.save

@itc = Classgroup.new(id: 2, class_name: "Introduction to Computers", course_name: "BSHC",
  class_description: "Learning All about how to use a Computer", enrollment_key: 9102,
  image_id: 3, lecturer_id: 1)
@itc.save

@iot = Classgroup.new(id: 3, class_name: "Internet of Things", course_name: "BSHC",
  class_description: "Solving the worlds problems 1 MQTT message at a time",
```

```

enrollment_key: 9103, image_id: 4, lecturer_id: 1)
@iot.save

@cad = Classgroup.new(id: 4, class_name: "Cloud Application Development", course_name: "BSHC",
                     class_description: "Just know that Rails is never sorry", enrollment_key: 9104, image_id: 5,
                     lecturer_id: 1)
@cad.save

# Creating a Example Session for the System

@ses = Classsession.new(id: 0, topic: "The Basics of Programming", classgroup_id: 0, session_key:
0000)
@ses.start_time = "2018-05-12 11:32:41"
@ses.end_time = "2018-05-12 12:32:41"
@ses.save

```

## 3 Viability Survey

### 3.1 Introduction

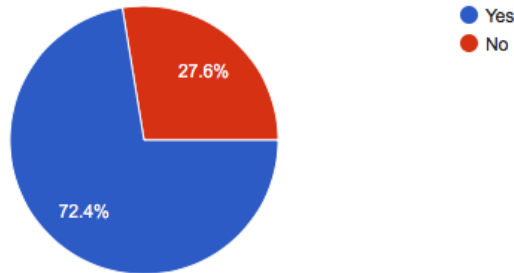
During the early stages of the projects conception, I ran an anonymous survey amongst peers in my college to gauge where students stood in relation to my theory. The results were shocking in some ways but to be totally honest, fairly in line with what I expected overall. The survey was a huge success and really helped when it came to motivation for the project.

The Survey was conducted using Google Forms and ended up with a total of 29 Responses, which is quite high for my class. Each response was Anonymously revealed to me and all the results were aggregated into graphs for me to read.

### 3.2 Have you Ever been too afraid to ask a question in a class?

Have you ever been too afraid to ask a question in a class

29 responses

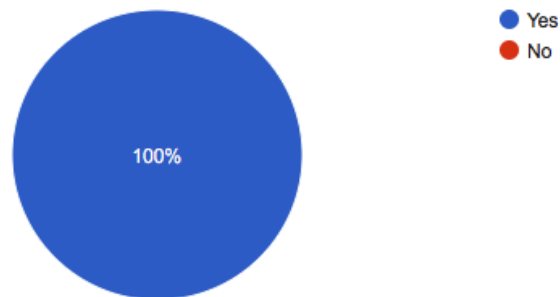


This question is fairly self-explanatory. It is worth noting almost  $\frac{3}{4}$  of the class have been too afraid to ask a question at some point in time. This is an extremely disappointing figure to see from the perspective of someone who has taught many classes. This should definitely be the exact opposite way around and this *needs* to be formally addressed by education institutions.

### 3.3 Have you ever sat through a class where you didn't understand the topic?

Have you ever sat through a class where you didn't understand the topic

29 responses



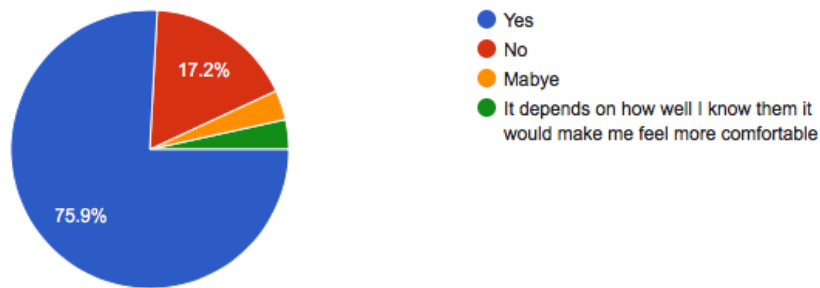


This Question honestly came as a complete surprise. As we can see 100% of the participants voted yes which is just crazy. The total lack of communication between student and lecturers is outright ridiculous and no one benefits from this situation. This is a disaster waiting to happen.

### 3.4 Do you find it difficult to speak in front of a class or large group of people?

Do you find it difficult to speak in front of a class or large group of people

29 responses

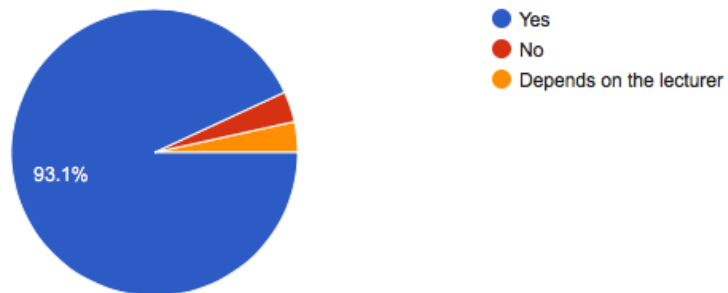


I think this is an important statistic to have as we can see the problem is not just specific to Classes. This could mean Students perhaps need to be trained for better presentation skills at an earlier stage in life. This would be interesting to try and solve, looking back at primary school or at the very least secondary school.

### 3.5 Would you use a classroom assistant app to help give real-time feedback to your teacher / lecturer (e.g show you don't understand a topic or allow you to ask questions, anonymously or otherwise)?

Would you use a classroom assistant app to help give real time feedback to your teacher / lecturer (e.g show you do or don't understand a topic or allow you to ask questions, anonymously or otherwise)

29 responses

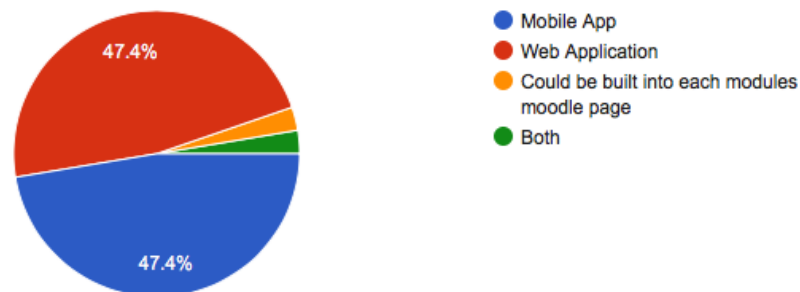


Out of a total of 29 Students, only two students didn't say yes. I think this is a fair indicator to say there is a gap in the market or a need for an application like Classistant to help optimise the learning experience for everyone involved.

### 3.6 If yes, what platforms would you use the app on?

If yes, what platforms would you use the app on

28 responses



This question was a huge surprise. I was absolutely shocked. In this day and age I did not expect the divide so evenly split. I would've assumed it would be 75 / 25 mobile to web. This was a pleasant surprise as it showed that Web Applications are still relevant in today's market and encouraged me to make Classistant a Web Application

### 3.7 Finally, any functionality you'd like to see added to an application like this?

## Finally, any functionality you'd like to see added to an application like this?

8 responses

Forum (2)
Chat functionality
Rate answers, rate a lecture, rate a lecturer etc. Maybe somewhat shared virtual environment, where you can see questions others are asking, rate that questions relevance or provide your own answer?
Sample quizzes
<ul style="list-style-type: none"> <li>- Give students who understand the topic a means to alert the lecturer to possible trouble areas that they managed to figure out but know the class are still struggling with</li> <li>- Allow students to opt in and out of standing up for demonstrations. Then the lecturer can select volunteers from the pool of students who opted in</li> <li>- Snapchat integration. Dunno how, just do it</li> </ul>
Maybe have a list of the lecture topics that are to be discussed. Then a flag beside each one indicating if a student needs it to be explained in more detail.
Possibly an are to ask after the class if not wanting to disrupt it. But still to shy to ask

Above was an open-ended question left at the end. This question received 8 responses and was interesting to see the ideas other people had conceived which could benefit my application. I will admit, most I didn't even initially consider so this was beneficial and eye opening.

### 3.8 Conclusions

The survey in my view was of huge benefit and helped me answer a number of questions I had going into the project.

#### Q1 – Was there a requirement for an application like Classistant?

## 4 Conclusion

Classistant set out as a project to create a Web Application that would allow for Students and Lecturers to interact in real-time. This was always the core goal from the start. Naturally with any project, since conception the scope has grown beyond reasonable achievement. However, having said that, I firmly believe Classistant is a successful project and achieved what I initially set out to do.

The main goal as mentioned was real-time interactions and that feature is fully implemented with no faults or errors. This to me marks it as a success.

I think over the course of the project though, there are many more hugely beneficial features which could be implemented to improve the User Experience and I feel almost disappointed in myself for not achieving all of these stretch goals.

Although I say Classistant is a successful project, I am left questioning if it is a complete product and honestly, I think the answer is no. Unfortunately, going into my Final year as one of the top students, I was unable to perform to the best of my abilities due to a number of factors, mostly to do with my personal life.

This left me at a huge disadvantage and quite literally months behind my peers. I started the year in a traffic accident and ended it with a sold childhood home, packed and ready to be left behind. The year has been quite frankly, emotional and quite a difficult journey but I'm glad I at least showed perseverance and pushed through to get it finished instead of deferring. I think it is unfortunate however, that due to the weighting of the project, a first is realistically unachievable, despite a peak average grade of 86%.

After putting the situation into context, I am happy to conclude my report by saying the Classistant project was a success, a journey and a wonderful learning experience. Although it didn't quite reach its full potential, the experience I gained on my journey to the end has been unrivalled. I managed to learn Ruby on Rails, PostgreSQL, Coffeescript and a number of other languages in literally weeks and develop a fully working application which is deployed to the cloud.

I have no doubt if Classistant was to go forward into the future as a live product, it would grow and succeed in providing the sought-after experience/functionalities and it would prove my initial theory which lead to the conception of the project.

## **5 Further Development or Research**

### **5.1 More Sophisticated Email System**

I would love to implement a more dynamic email system which is properly implemented using the Observer pattern. This would listen out for events in the system (Such as a class finishing)

### **5.2 More Sophisticated Badge System**

I think creating custom badges for classes would be a fantastic idea. Lecturer definable badges with custom triggers for achievement. I would also create models for the badges to simplify the logic immensely

### 5.3 Additional Interactions

It would be interesting to try implement different interactions a Lecturer can have with a student and potentially some reverse actions that a student can post to a lecturer.

### 5.4 Implementation of multiple Graphs / Charts

Inclusion of more types of charts rather than the standard two in the system would be a nice feature. The way I envisioned this working was to have a bootstrap tabbed / pill view where you can click between the graphs in different views.

### 5.5 Customisable Dashboard

The ability to dynamically rearrange your dashboard and have the layout save would be fantastic. Customising the background image would be a nice touch as well.

### 5.6 Upload custom class icons

An image upload field on a class form would be a nice addition so lecturers can set their own class icons. I believe there are gems to make this incredibly simple.

### 5.7 User profile picture

Allow users to upload a profile picture in a similar manner to Class Icons

### 5.8 Messaging system

Implement an internal mail system for outside of class communications with students and lecturers

### 5.9 Integration to 3<sup>rd</sup> party systems such as Moodle

Allow Classistant to be readily available and accessible through Moodle, perhaps on a classes page, or automatically set up based on a Moodle classes configuration.

### 5.10 Mobile Applications

Developing phone applications would probably increase the incentive to use the Application instead of relying on users to go out of their way to open the website, login and get started.

## 5.11 API Implementation

Given an appropriate authorisation key, an organisation, such as NCI, could consume data from the application for reporting. This would include but not be limited to

- Student attendance
- Student performance
- Data relates to Class Sessions

## 5.12 Class Forums

Forums would allow students to interact with not only the lecturer but with each other outside of class and might be an interesting way for students to cooperate and learn together.

# 6 References

## 6.1 Websites Used

- Baimas, V. (2018). *flatuicolorpicker : Best Flat Colors For UI Design*. [online] flatuicolorpicker : Best Flat Colors For UI Design. Available at: <http://www.flatuicolorpicker.com/> [Accessed 1 Jan. 2018].
- Blog.heroku.com. (2018). *Real-Time Rails: Implementing WebSockets in Rails 5 with Action Cable*. [online] Available at: [https://blog.heroku.com/real\\_time\\_rails\\_implementing\\_websockets\\_in\\_rails\\_5\\_with\\_action\\_cable](https://blog.heroku.com/real_time_rails_implementing_websockets_in_rails_5_with_action_cable) [Accessed 4 Apr. 2018].
- Chartkick.com. (2018). *Chartkick - Create beautiful JavaScript charts with one line of Ruby*. [online] Available at: <https://www.chartkick.com/> [Accessed 21 Apr. 2018].
- Coffeescript.org. (2018). *CoffeeScript*. [online] Available at: <http://coffeescript.org/> [Accessed 9 Apr. 2018].
- CSS-Tricks. (2018). *jQuery with CoffeeScript | CSS-Tricks*. [online] Available at: <https://css-tricks.com/jquery-coffeescript/> [Accessed 8 May 2018].
- Edgeguides.rubyonrails.org. (2018). *Action Cable Overview — Ruby on Rails Guides*. [online] Available at: [http://edgeguides.rubyonrails.org/action\\_cable\\_overview.html#client-server-interactions-subscriptions](http://edgeguides.rubyonrails.org/action_cable_overview.html#client-server-interactions-subscriptions) [Accessed 12 Apr. 2018].
- Flaticon. (2018). *Flaticon, the largest database of free vector icons*. [online] Available at: <https://www.flaticon.com/> [Accessed 1 Jan. 2018].

- Getbootstrap.com. (2018). *Bootstrap · Content moved*. [online] Available at: <https://getbootstrap.com/docs/4.0> [Accessed 1 Jan. 2018].
- GitHub. (2018). *Best method to update chart data dynamically · Issue #304 · ankane/chartkick*. [online] Available at: <https://github.com/ankane/chartkick/issues/304> [Accessed 20 Apr. 2018].
- Guides.rubyonrails.org. (2018). *Active Record Query Interface — Ruby on Rails Guides*. [online] Available at: [http://guides.rubyonrails.org/active\\_record\\_querying.html](http://guides.rubyonrails.org/active_record_querying.html) [Accessed 3 Apr. 2018].
- jquery.org, j. (2018). *jQuery API Documentation*. [online] Api.jquery.com. Available at: <http://api.jquery.com/> [Accessed 1 Jan. 2018].
- June 13, 2. (2018). *Make Easy Graphs and Charts on Rails with Chartkick — SitePoint*. [online] SitePoint. Available at: <https://www.sitepoint.com/make-easy-graphs-and-charts-on-rails-with-chartkick/> [Accessed 19 Apr. 2018].
- Mark Otto, a. (2018). *CSS · Bootstrap*. [online] Getbootstrap.com. Available at: <https://getbootstrap.com/docs/3.3/css/> [Accessed 1 Jan. 2018].
- Pluralsight.com. (2018). *Creating a chat using Rails' Action Cable | Pluralsight*. [online] Available at: <https://www.pluralsight.com/guides/creating-a-chat-using-rails-action-cable> [Accessed 4 Apr. 2018].
- Ruby-for-beginners.rubymonstas.org. (2018). *Instance variables | Ruby for Beginners*. [online] Available at: [http://ruby-for-beginners.rubymonstas.org/writing\\_classes/instance\\_variables.html](http://ruby-for-beginners.rubymonstas.org/writing_classes/instance_variables.html) [Accessed 1 Apr. 2018].
- W3schools.com. (2018). *HTML Color Picker*. [online] Available at: [https://www.w3schools.com/colors/colors\\_picker.asp](https://www.w3schools.com/colors/colors_picker.asp) [Accessed 1 Jan. 2018].

## 6.2 Stack Overflow Posts viewed

<https://stackoverflow.com/questions/36455077/find-records-whose-attributes-after-some-change-is-equal-to-something>

<https://stackoverflow.com/questions/13644562/dynamically-add-embedded-ruby-with-jquery-html>

<https://stackoverflow.com/questions/13831601/disabling-and-enabling-a-html-input-button/13831737>

<https://stackoverflow.com/questions/23610012/how-to-bind-an-jquery-onclick-event-with-coffeescript-in-rails>

<https://stackoverflow.com/questions/16267577/change-content-of-a-p-tag-using-jquery>

<https://stackoverflow.com/questions/3025784/rails-layouts-per-action>

<https://stackoverflow.com/questions/17765249/generate-migration-create-join-table>

<https://stackoverflow.com/questions/36926816/actioncable-how-to-use-dynamic-channel>

<https://stackoverflow.com/questions/8108511/how-to-access-instance-variables-in-coffeescript-engine-inside-a-slim-templates>



## 7 Appendix

In order of Appearance

1. Project Proposal & Project Plan
2. Project Requirements Specification
3. Monthly Journals









BSc (Hons) in Computing

Internet of Things

2017/2018

# CLASSISTANT

## Technical Report

### Abstract

*Classistant is a Real-time Responsive Web Application which aims to increase Student – Lecturer engagement during classes by digitizing communications and offering anonymity options in order to negate social anxiety experienced by students*

Paul Reid | x14552067  
paulreid96@gmail.com

## Objectives

### 8 Background

Throughout my entire academic career, I have seen time and time again the scenario of a Teacher or Lecturer teaching content when the class does not fully understand. There is always the dreaded question asked, “Does everybody understand?” or “Can I move on?”.

These questions are extremely difficult for students to answer and 9 times out of 10, a student will say nothing and allow the class to continue, despite their difficulties or not fully understanding the topics covered.

I believe students will not raise their hands and ask questions or say they don’t understand due to social anxiety. Very few people will find it difficult to raise their hand in front of a class of 20 or 30, never mind 70, 150 or more. This is because students fear judgment from their peers, not knowing how others will react to their questions. Students feel like they are admitting defeat or being stupid by asking a lecturer to repeat content. This however, is absolutely not the case and chances are if one student feels this way in the class, many others secretly do too.

During my time studying as a Computing student in college, I have worked as a Support Lecturer in the Colleges Computing Support Office. Some of my classes and support sessions that I have had to give, have been online. This means my screen is streamed live as well as my microphone, and I can deliver content to anyone who logs into my class.

I have personally found that when I am delivering content online, I get a large amount of questions from almost every student in the class. I believe this to be because of the lack of any social anxiety of sitting in a physical room full of real people. Students can simply hide behind their screen and type away and the only thing anyone else will see is some text on a screen. Compare this to any in-person classes I deliver and I would get very few questions, maybe 5 or 6 per session as opposed to 30 or 40, sometimes even more.

To rectify this issue, I have come up with the idea of a Virtual Classroom (VC) and Teaching Assistant. The idea of my project is to allow students to anonymously (or otherwise, if chosen), can interact with the person delivering content, in a real-time virtual environment.

The application will allow a Virtual Classroom (VC) to be created. The lecturer who is delivering content to the class will be able to manage the room and the students in their class will be able to join and interact with the lecturer. The virtual room will allow students to provide real-time feedback on content, through the use of “Yes/No” questions, quizzes, challenges, and many more features.

Creating such a scenario where every student has the chance to interact with the lecturer anonymously or otherwise, without having to physically draw attention to themselves, should increase the amount of student interaction and feedback exponentially.

## 9 Technical Approach

### **Research**

When I initially conceived the idea for my project, I carried out some research in the area of my project. I examined existing systems and products that were similar and worked out what I considered to be the pros and cons of each. I then asked peers as well as colleagues (Lecturers) what their thoughts and experiences were using 3rd party software to assist in the teachings of their classes. I then ran a Survey amongst my peers in my class to how they would react. The responses I received were extremely positive.

### **Requirements Definition**

Upon concluding my research, I decided to formally lay out what my requirements would be for the System. I also arranged formal meetings with the Lecturer in charge of the Student Support Office who had been wanting to do a similar style project for years. I made the decision to use him as a client for my project and elicited some of my requirements through him.

### **Prototype Development & Technology Research**

The application needs to be lightweight fast and run in real-time in a web based environment. I intend to trial and error various implementations of this type of functionality to see which is the most flexible, easy to use and stable as well as scalable.

Once I have determined the approach I want to take, I will begin developing a prototype which focuses on simple functionality, purely to convey the idea of my project. The hope is to be able to instil the vision of the final product.

### **Milestones**

I have set myself a series of Milestones / Deliverables that I will aim to achieve during the development of my Project. This will help keep me on track for the duration of the development of my project.

Once I begin prototyping and Developing my end solution, I will know the full extent of what I need to achieve in order to deliver an end product.

### **Trials & Testing**

The final stage of my project will be to test the full project, seeing if I can overload and break the System, as well as perform simple testing to find basic bugs or errors such as typos or flaws in logic.

I will then hopefully trial the System with a small group of students to see if they find the System easy to use and beneficial.

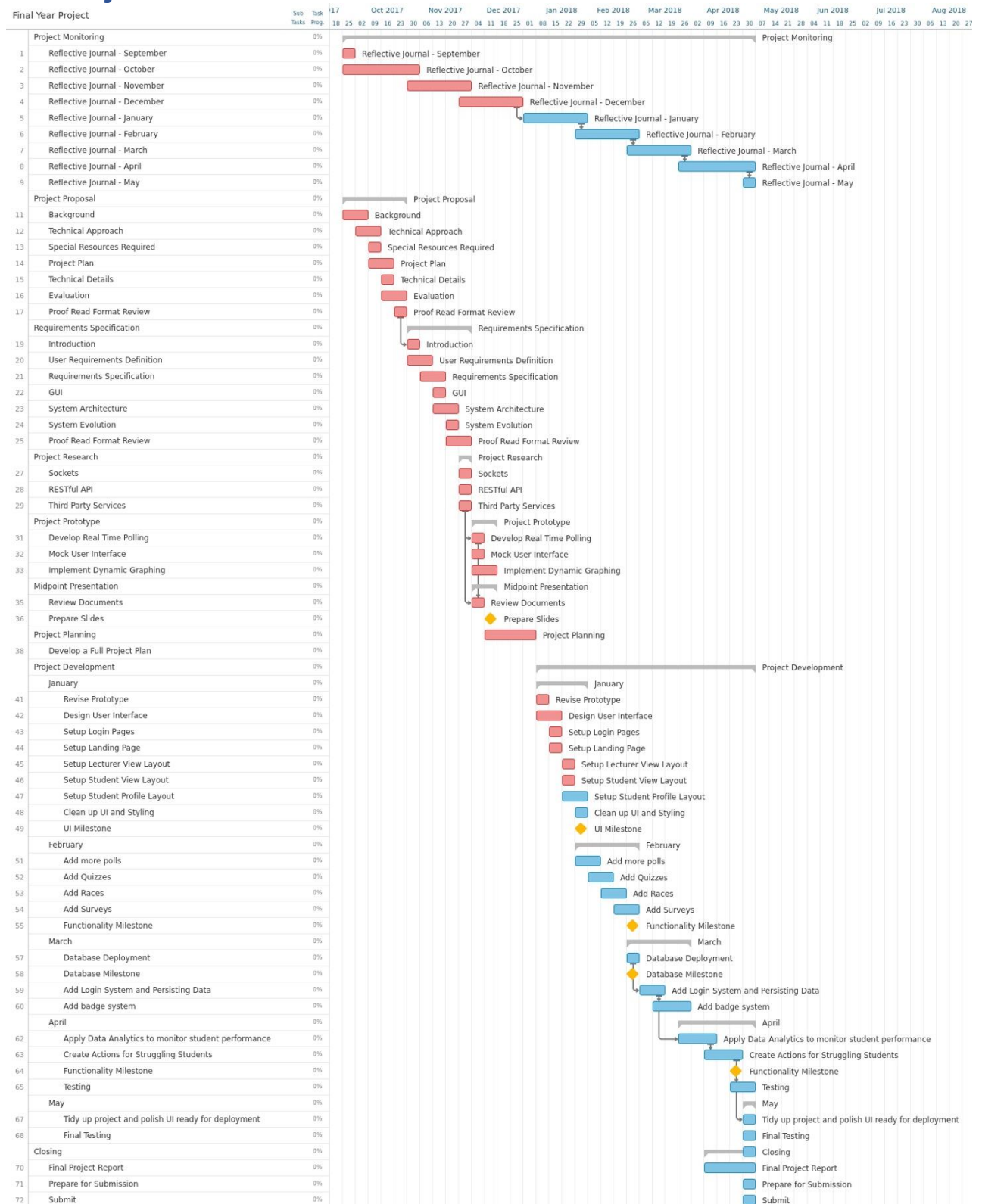
## **10 Special resources required**

I will be using Cloud9 to develop my project. However, this may severely limit my features so I may need to purchase a domain or server to host my project on so I can develop the full suite of functionality that I am aiming for.

I am also using the Service PubNub. As I get into the development of my project I may need to consider getting a premium account.



# 11 Project Plan



## 12 Technical Details

Due to the nature of the project, the ability to consume and display information in real-time is crucial. In order to implement real time data consumption on a web page I have to create a program that uses sockets or an API.

I researched into the approaches and did some trials and I found using a message based service was the best suited for what I want. This allows me to rapidly develop new functionality by just changing the content of any messages I send to the system.

The implementation uses HTML for the front end of my application. The back end is written in JavaScript, JQuery. My database will be made using SQL and most importantly, I am using a service called PubNub to handle my message system.

PubNub I found is a perfect fit for my needs as it operates in real-time, allows me to integrate with multiple languages, gives me total control over the messages and their format and contents and it also allows me to scale the system up.

The basic premise of the implementation is that a lecturer creates a virtual classroom which the students then subscribe to. This subscription is allowing me to send data between lecturers and students and allows them to interact.

The system will know what classes are on what “Channel” in the System and will only send data relevant to classes on the classes channel. This basically means there will never be an overlap of data or data being send to the wrong location.

This also allows me to just create more channels or codes for classes and I can listen out for any messages being sent and only read messages relevant to the subscription.

The front end of my application as mentioned, is developed in HTML. I am using Twitter Bootstrap to create a responsive web application. This should allow for the use of my application on mobile devices.

I also make use of JQuery to add some fluidity to my User Interface. The goal of JQuery combined with Bootstrap is to create a simple, aesthetically pleasing User Interface that is naturally intuitive to use no matter who the user is. This means no training would be required to use the System.

The Application also uses Highcharts, a Graphing framework that allows for dynamic charts and graphs to be displayed and updated on a Web Page.

Programming Languages	JavaScript
-----------------------	------------

	JQuery
	SQL
	HTML, CSS
	Cloud9
Development Environments	SQL Workbench
	Notepad++
	MySQL
Database Platform	Responsive Web Application

## 13 Evaluation

I plan to perform unit tests on the back-end code, to ensure it works as expected. These will be small tests of specific parts of functionality such as functions and methods, to insure, given a specific input, a desired output / outcome is returned.

Once the back-end is reaching a point of completion, I will then run integration tests to make sure PubNub is working fine with my framework as well as the Database is connecting and adding / removing without any issues.

The majority of System Evaluation however, will be manual testing. Due to the nature of the System it will require me to login as a Lecturer and Student and simply test the interactions and functionality.

I also plan to stress test the System to see if I can overload or overwhelm it with requests and see if I can find an upper limit of class size. I feel because I am using a free service, it may not handle 100 simultaneous requests with ease, although I may have found a workaround.

Finally, I plan to try run a test "Class" with some peers to see how the System performs and if they actually find the System easy to use / Intuitive, if they would actually use the System in a real-life scenario and elicit comments and feedback.

National College of Ireland  
BSc in Computing  
2017/2018

Paul Reid  
14552067  
paulreid96@gmail.com

Classistant

Technical Report



# Table of Contents

<b>1</b>	<b>TABLE OF CONTENTS .....</b>	<b>2</b>
	<b>EXECUTIVE SUMMARY.....</b>	<b>5</b>
<b>1.</b>	<b>INTRODUCTION.....</b>	<b>6</b>
1.1	BACKGROUND .....	6
1.2	AIMS.....	7
1.3	TECHNOLOGIES .....	8
1.4	STRUCTURE .....	10
<b>2</b>	<b>SYSTEM.....</b>	<b>11</b>
2.1	REQUIREMENTS.....	11
2.1.1	<i>Functional Requirements .....</i>	<i>Error! Bookmark not defined.</i>
2.1.2	<i>Data Requirements .....</i>	15
2.1.3	<i>User Requirements .....</i>	16
2.1.4	<i>Environmental Requirements .....</i>	16
2.1.5	<i>Usability Requirements .....</i>	16
2.2	DESIGN AND ARCHITECTURE .....	18
2.2.1	<i>Classistant Application Architecture .....</i>	18
2.2.2	<i>Classistant Application Entity Relationship Diagram.....</i>	19
2.2.3	<i>Core Components.....</i>	20
2.3	CODE SNIPPETS AND EXPLANATION.....	27
2.4	GRAPHICAL USER INTERFACE.....	30
2.5	TESTING .....	44
2.6	EVALUATION.....	ERROR! BOOKMARK NOT DEFINED.
<b>3</b>	<b>CONCLUSIONS.....</b>	<b>51</b>
<b>4</b>	<b>FURTHER DEVELOPMENT OR RESEARCH .....</b>	<b>52</b>
<b>5</b>	<b>REFERENCES.....</b>	<b>54</b>
5.1	WEBSITES USED.....	54
5.2	STACK OVERFLOW POSTS VIEWED .....	55
<b>6</b>	<b>APPENDIX.....</b>	<b>57</b>
	<b>OBJECTIVES .....</b>	<b>62</b>
<b>7</b>	<b>BACKGROUND .....</b>	<b>62</b>
<b>8</b>	<b>TECHNICAL APPROACH .....</b>	<b>63</b>
<b>9</b>	<b>SPECIAL RESOURCES REQUIRED.....</b>	<b>64</b>
<b>10</b>	<b>PROJECT PLAN.....</b>	<b>65</b>
<b>11</b>	<b>TECHNICAL DETAILS.....</b>	<b>66</b>
<b>12</b>	<b>EVALUATION .....</b>	<b>67</b>
	<b>EXECUTIVE SUMMARY.....</b>	<b>70</b>
<b>1</b>	<b>INTRODUCTION .....</b>	<b>71</b>
1.1	BACKGROUND .....	71
1.2	AIMS.....	72
1.3	TECHNOLOGIES .....	72
1.4	APPROACH.....	73

<b>2</b>	<b>SYSTEM.....</b>	<b>75</b>
2.1	REQUIREMENTS.....	75
2.1.1	<i>Use Case Diagram.....</i>	75
2.1.2	<i>Requirement 1 &lt;Classistant - Login – Use Case&gt;.....</i>	76
2.1.3	<i>Requirement 1 &lt;Classistant - Profile – Use Case&gt; .....</i>	77
2.1.4	<i>Requirement 1 &lt;Classistant – Classroom View – Use Case&gt; .....</i>	78
2.1.5	<i>Requirement 1 &lt;Classistant - Dashboard – Use Case&gt; .....</i>	80
2.2	NON-FUNCTIONAL REQUIREMENTS .....	81
2.2.1	<i>Performance/Response time requirement .....</i>	81
2.2.2	<i>Availability requirement .....</i>	81
2.2.3	<i>Recover requirement.....</i>	81
2.2.4	<i>Portability requirement .....</i>	82
2.2.5	<i>Extendibility requirement.....</i>	82
2.3	DESIGN AND ARCHITECTURE .....	83
2.4	IMPLEMENTATION.....	83
2.5	GRAPHICAL USER INTERFACE (GUI) LAYOUT .....	83
2.6	EVALUATION.....	86
<b>3</b>	<b>CONCLUSIONS .....</b>	<b>87</b>
<b>4</b>	<b>FURTHER DEVELOPMENT OR RESEARCH .....</b>	<b>88</b>

## Executive Summary

The Purpose of this Executive Summary is to give you an insight into what Classistant is and what it is trying to achieve. Classistant is a Responsive Web Application which allows Students and Lecturers digitally interact with each other during class time as well as outside of class. The goal is to digitise some of the typical social interaction in the classroom to remove social anxiety from students and allow them to speak out and ask questions without outing themselves to a large group of peers.

The reason for this is that Students are reportedly more likely to ask questions when they don't have to hold up their hand or bring a classroom to a total halt while they ask a question. Students feel that by asking a question they are admitting defeat or painting themselves as inferior than the rest but in reality, if the student has a question, chances are they are not the only one with that question.

The Application is developed to be a responsive Web Based Application. The interactions between Students and Lecturers happen in real time and the Lecturer can see feedback from the class as well as questions being asked, responses to Polls or Quizzes they may be running and so much more.

This data is all collected so lecturers can later report on the data. Seeing a trend of a student constantly voting they “Do not Understand” the content being delivered is vital as it can allow the Institution to reach out to that student and provide any help they need to bring them back up to speed.

# 1 Introduction

## 1.1 Background

Throughout my entire academic career, I have seen time and time again the scenario of a Teacher or Lecturer teaching content when the class does not fully understand. There is always the dreaded question asked, “Does everybody understand?” or “Can I move on?”.

These questions are extremely difficult for students to answer and 9 times out of 10, a student will say nothing and allow the class to continue, despite their difficulties or not fully understanding the topics covered.

I believe students will not raise their hands and ask questions or say they don’t understand due to social anxiety. Very few people will find it difficult to raise their hand in front of a class of 20 or 30, never mind 70, 150 or more. This is because students fear judgment from their peers, not knowing how others will react to their questions. Students feel like they are admitting defeat or being stupid by asking a lecturer to repeat content. This however, is absolutely not the case and chances are if one student feels this way in the class, many others secretly do too.

During my time studying as a Computing student in college, I have worked as a Support Lecturer in the Colleges Computing Support Office. Some of my classes and support sessions that I have had to give, have been online. This means my screen is streamed live as well as my microphone, and I can deliver content to anyone who logs into my class.

I have personally found that when I am delivering content online, I get a large amount of questions from almost every student in the class. I believe this to be because of the lack of any social anxiety of sitting in a physical room full of real people. Students can simply hide behind their screen and type away and the only thing anyone else will see is some text on a screen. Compare this to any in-person classes I deliver and I would get very few questions, maybe 5 or 6 per session as opposed to 30 or 40, sometimes even more.

To rectify this issue, I have come up with the idea of a Virtual Classroom (VC) and Teaching Assistant. The idea of my project is to allow students to anonymously (or otherwise, if chosen), can interact with the person delivering content, in a real-time virtual environment.

The application will allow a Virtual Classroom (VC) to be created. The lecturer who is delivering content to the class will be able to manage the room and the students in their class will be able to join and interact with the lecturer. The virtual room will allow students to provide real-time feedback on content, through the use of “Yes/No” questions, quizzes, challenges, and many more features.

Creating such a scenario where every student has the chance to interact with the lecturer anonymously or otherwise, without having to physically draw attention to

themselves, should increase the amount of student interaction and feedback exponentially.

## 1.2 Aims

The scope of the project is to develop a Web Application for the purposes and needs outlined above.

The primary goal is to create an application that provides a Student with an outlet to interact with the class he is currently sitting in without the fear or anxiety of raising his hand or stopping the class to ask a question.

## 1.3 Technologies

Due to the nature of the project, the ability to consume and display information in real-time is crucial. In order to implement real time data consumption on a web page I have to create a program that uses sockets or an API.

I researched into the approaches and did some trials and I found using a message based service was the best suited for what I want. This allows me to rapidly develop new functionality by just changing the content of any messages I send to the system.

The implementation uses HTML for the front end of my application. The back end is written in JavaScript, JQuery. My database will be made using SQL and most importantly, I am using a service called PubNub to handle my message system.

PubNub I found is a perfect fit for my needs as it operates in real-time, allows me to integrate with multiple languages, gives me total control over the messages and their format and contents and it also allows me to scale the system up.

The basic premise of the implementation is that a lecturer creates a virtual classroom which the students then subscribe to. This subscription is allowing me to send data between lecturers and students and allows them to interact.

The system will know what classes are on what "Channel" in the System and will only send data relevant to classes on the classes channel. This basically means there will never be an overlap of data or data being send to the wrong location.

This also allows me to just create more channels or codes for classes and I can listen out for any messages being sent and only read messages relevant to the subscription.

The front end of my application as mentioned, is developed in HTML. I am using Twitter Bootstrap to create a responsive web application. This should allow for the use of my application on mobile devices.

I also make use of JQuery to add some fluidity to my User Interface. The goal of JQuery combined with Bootstrap is to create a simple, aesthetically pleasing User



Interface that is naturally intuitive to use no matter who the user is. This means no training would be required to use the System.

The Application also uses Highcharts, a Graphing framework that allows for dynamic charts and graphs to be displayed and updated on a Web Page.

Programming Languages	JavaScript
	JQuery
	SQL
	HTML, CSS
Development Environments	Cloud9
	SQL Workbench
	Notepad++
Database	MySQL
Platform	Responsive Web Application

## 1.4 Approach

### Research

When I initially conceived the idea for my project, I carried out some research in the area of my project. I examined existing systems and products that were similar and worked out what I considered to be the pros and cons of each. I then asked peers as well as colleagues (Lecturers) what their thoughts and experiences were using 3rd party software to assist in the teachings of their classes. I then ran a Survey amongst my peers in my class to how they would react. The responses I received were extremely positive.

### Requirements Definition

Upon concluding my research, I decided to formally lay out what my requirements would be for the System. I also arranged formal meetings with the Lecturer in charge of the Student Support Office who had been wanting to do a similar style project for years. I made the decision to use him as a client for my project and elicited some of my requirements through him.

### Prototype Development & Technology Research

The application needs to be lightweight fast and run in real-time in a web based environment. I intend to trial and error various implementations of this type of functionality to see which is the most flexible, easy to use and stable as well as scalable.

Once I have determined the approach I want to take, I will begin developing a prototype which focuses on simple functionality, purely to convey the idea of my project. The hope is to be able to instil the vision of the final product.

### **Milestones**

I have set myself a series of Milestones / Deliverables that I will aim to achieve during the development of my Project. This will help keep me on track for the duration of the development of my project.

Once I begin prototyping and Developing my end solution, I will know the full extent of what I need to achieve in order to deliver an end product.

### **Trials & Testing**

The final stage of my project will be to test the full project, seeing if I can overload and break the System, as well as perform simple testing to find basic bugs or errors such as typos or flaws in logic.

I will then hopefully trial the System with a small group of students to see if they find the System easy to use and beneficial.

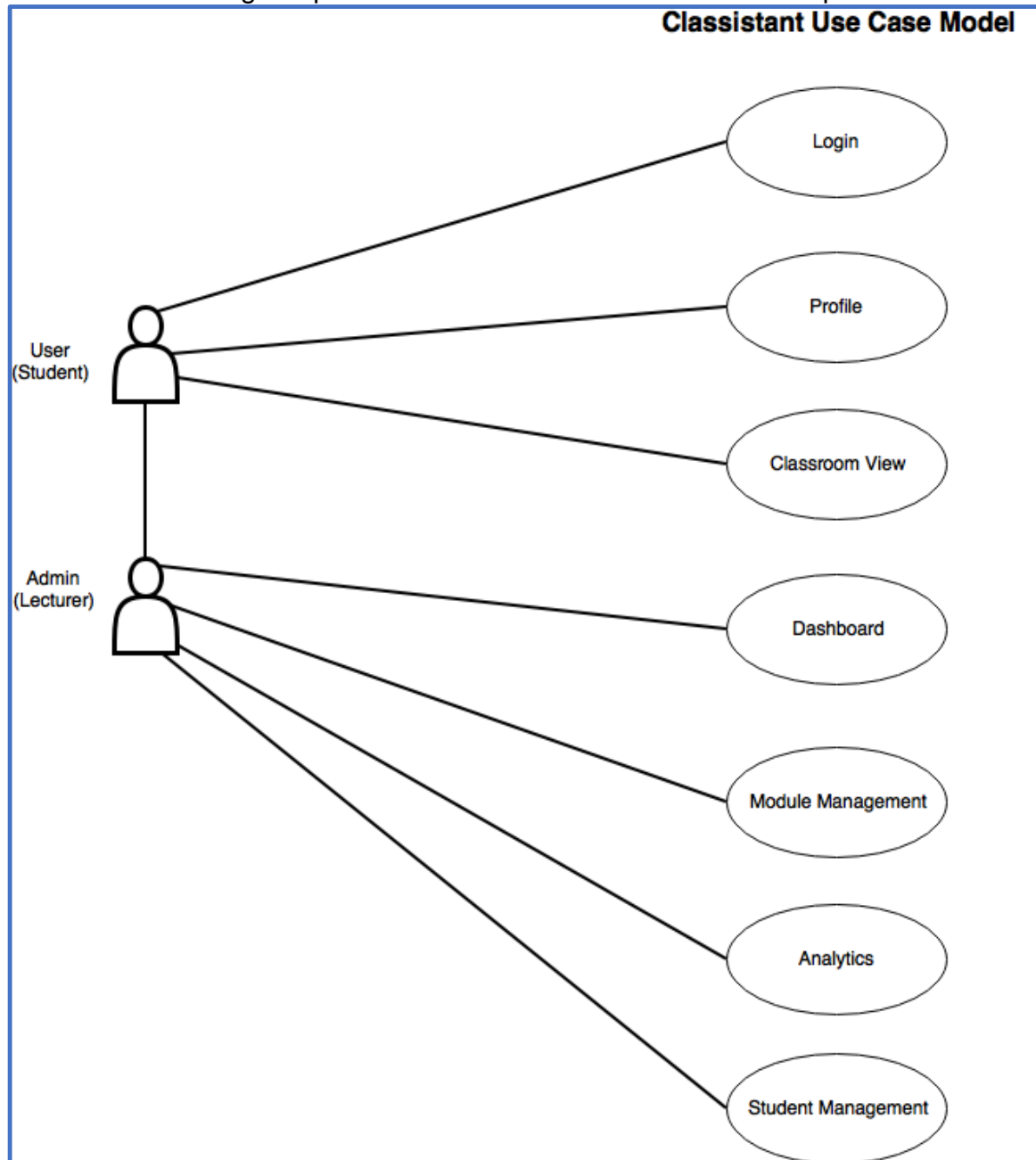
## 2 System

### 2.1 Requirements

#### 2.1.1 Use Case Diagram

Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.

The Use Case Diagram provides an overview of all functional requirements.



## 2.1.2 Requirement 1 <Classistant - Login – Use Case>

### 2.1.2.1 Description & Priority

The Login use case covers the functionality of the main landing page of the Classistant Application. This provides the means to authenticate a user's details and allow them access to the system and its core functionality. This is the only point of access.

### 2.1.2.2 Use Case

Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.

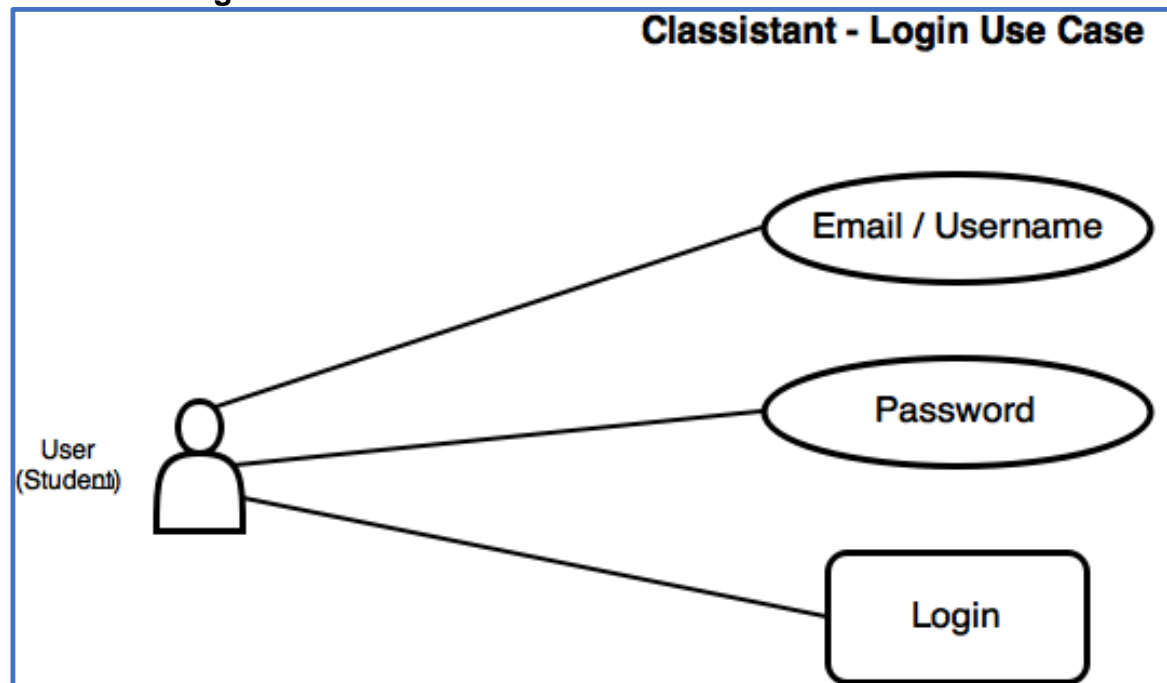
#### Scope

The scope of this use case is to define the Login use case

#### Description

This use case describes the process of a user logging into the application.

#### Use Case Diagram



#### Flow Description

#### Precondition

The user is not logged in and needs to partake in a class.

#### Activation

This use case starts when the User opens the Classistant website

#### Main flow

1. The Web Application presents the User with the login page
2. The User puts in their Username
3. The User puts in their Password
4. The User clicks the Login Button
5. The System accepts the Users credentials as valid and authenticates the user, progressing them to the correct screen

#### Alternate flow

**A1 : Login Error – Invalid Credentials**

1. The Web Application presents the User with the login page
2. The User puts in their Username
3. The User puts in their Password
4. The User clicks the Login Button
5. The System rejects the User credentials as they are invalid, and prompts the user to re-enter their credentials and try again.

**Termination**

The system accepts the Users credentials and progresses the User to the correct screen.

**Post condition**

The User Progresses to the next step of their requirement

**2.1.3 Requirement 1 <Classistant - Profile – Use Case>****2.1.3.1 Description & Priority**

The Profile Use Case describes the available options to the User upon successfully logging into their Account (See Login Use Case).

**2.1.3.2 Use Case**

Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.

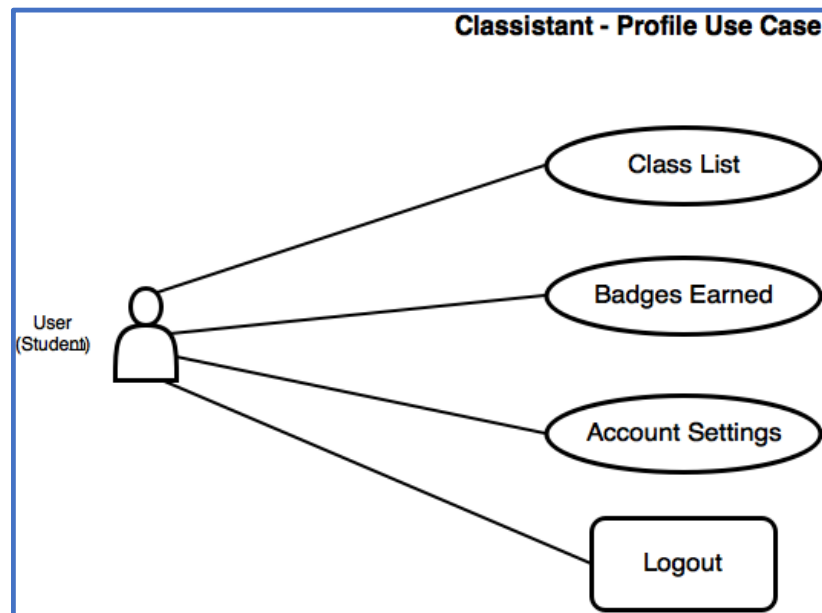
**Scope**

The scope of this use case is to elaborate on the options presented to the User when they are looking at their Profile

**Description**

This use case describes the actions behind each option presented on the profile

**Use Case Diagram**



### Flow Description

#### Precondition

The User has successfully completed the Login Use Case

#### Activation

This use case starts when the User is presented with the Profile Page

#### Main flow

1. The System presents the User with the options for "Class List", "Badges Earned", "Account Settings" and "Logout"
2. The user selects one of the presented options
3. The System redirects the user to the appropriate Page for that option

#### Alternate flow

A1 : Logout

1. The User decides they want to log out
2. The User clicks the Logout button
3. The user is Logged out of the System

#### Termination

The system responds correctly to the selection of the User and redirects them accordingly to the respective page

#### Post condition

The User Progresses to the next Use Case relevant to their selection

## 2.1.4 Requirement 1 <Classistant – Classroom View – Use Case>

### 2.1.4.1 Description & Priority

The Classroom View Use Case describes the available options to the User upon opening the Classroom View

#### 2.1.4.2 Use Case

Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.

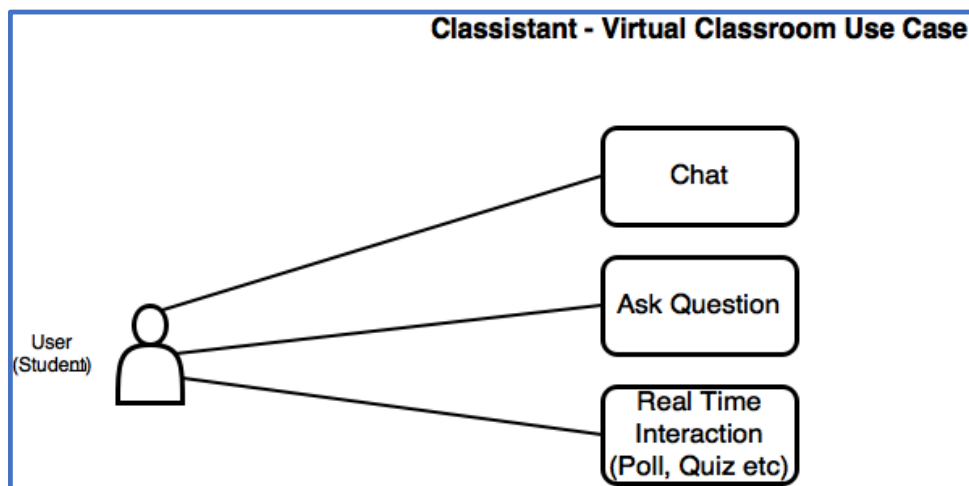
##### Scope

The scope of this use case is to elaborate on the options presented to the User when they are looking at the Classroom View

##### Description

This use case describes the actions behind each option presented in the Classroom View

##### Use Case Diagram



##### Flow Description

##### Precondition

The User has successfully completed the Login Use Case

##### Activation

This use case starts when the User is presented with the Classroom View

##### Main flow

1. The User is presented with a View of the Virtual Classroom created by the Admin (Lecturer)
2. The User interacts with the Polls, Quizzes, Surveys, Races, Chat, Questions
3. The User receives interactions and responses in real-time from their Admin (Lecturer)

##### Alternate flow

A1 : Exiting

1. The User redirects themselves away from the Classroom View page

2. The page is closed

### Termination

The system responds correctly to the actions of the User until the session is ended by an Admin.

### Post condition

The System waits for another Session to begin

## 2.1.5 Requirement 1 <Classistant - Dashboard – Use Case>

### 2.1.5.1 Description & Priority

The Dashboard Use Case describes the available options to the Admin upon opening the Dashboard Page

### 2.1.5.2 Use Case

Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.

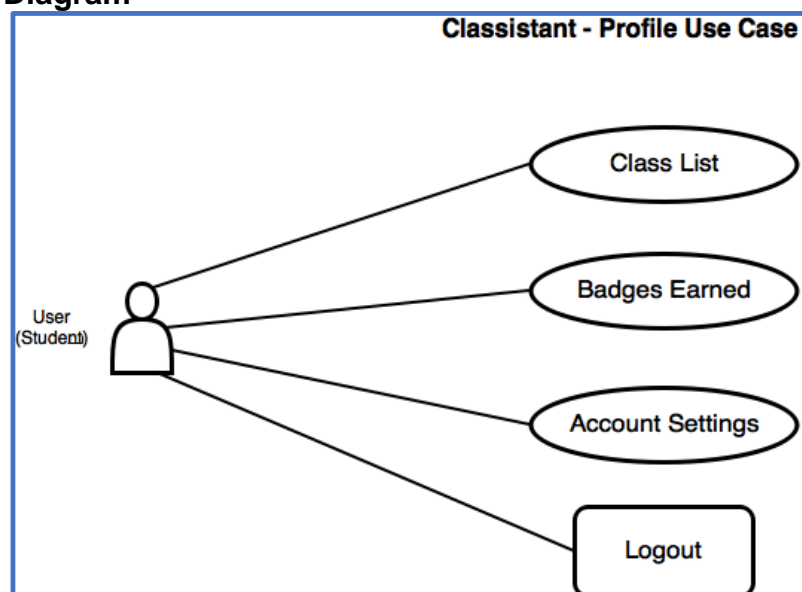
### Scope

The scope of this use case is to elaborate on the options presented to the Admin when they are looking at their Dashboard

### Description

This use case describes the actions behind each option presented on the Dashboard

### Use Case Diagram



### Flow Description



**Precondition**

The Admin has successfully completed the Login Use Case

**Activation**

This use case starts when the User is presented with the Dashboard Page

**Main flow**

1. The System presents the Admin with the options for “Class List”, “Manage Students”, “Manage Modules”, “Account Settings” and “Logout”
2. The Admin selects one of the presented options
3. The System redirects the Admin to the appropriate Page for that option

**Alternate flow**

A1 : Logout

4. The Admin decides they want to log out
5. The Admin clicks the Logout button
6. The Admin is Logged out of the System

**Termination**

The system responds correctly to the selection of the Admin and redirects them accordingly to the respective page

**Post condition**

The Admin Progresses to the next Use Case relevant to their selection

## 2.2 Non-Functional Requirements

Specifies any other particular non-functional attributes required by the system. Examples are provided below. **Remove the requirement headings that are not appropriate to your project.**

### 2.2.1 Performance/Response time requirement

It is important for the System to perform efficiently. The premise of the System is to allow for real-time interaction between Students and their Lecturers / Teachers. This means the System must have an incredibly fast response time in order to meet the aforementioned needs.

### 2.2.2 Availability requirement

Ideally, the System should be available almost 24/7. However, this would be down to the individual deployment of the System on an Institution by Institution basis. Due to the nature of Education, classes could be running at any hour of the day so the System must be online and available, ready to support any and all classes.

### 2.2.3 Recover requirement

The System should take redundancy into account and allow for backing up and recovery of all Student data lost in the event of a disaster.

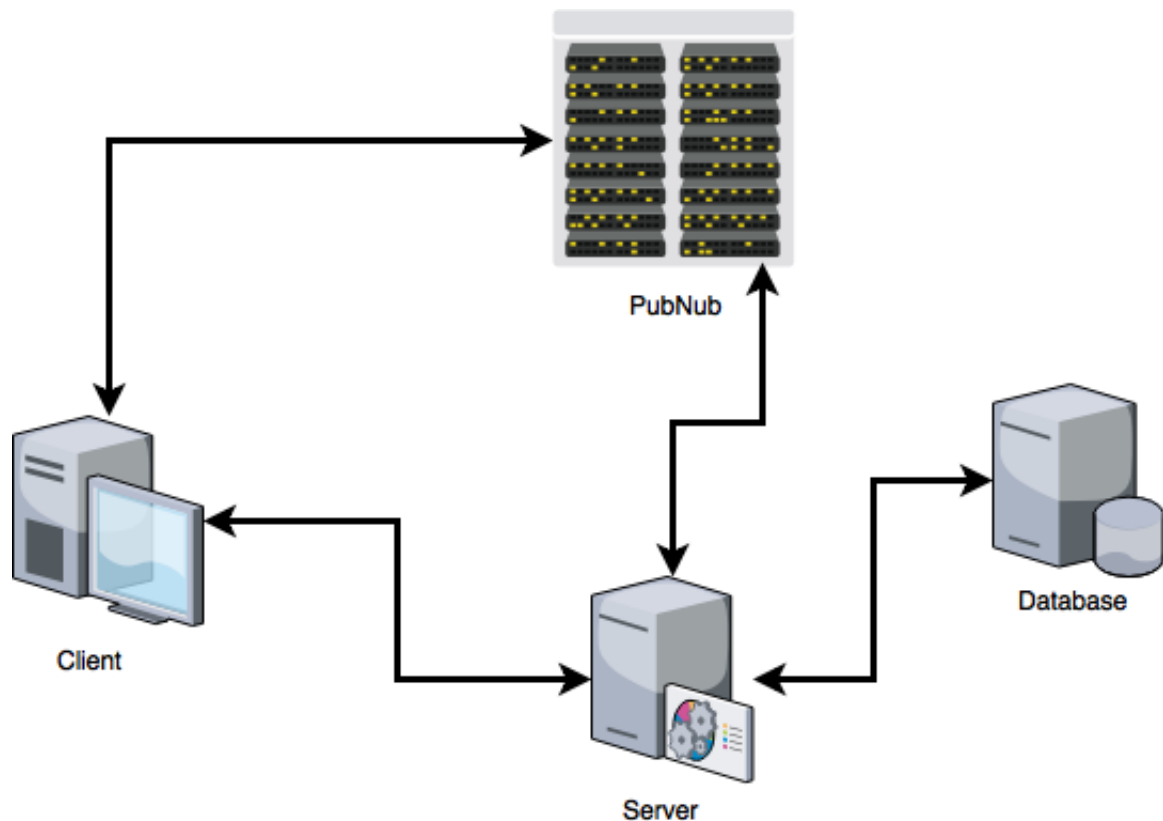
#### **2.2.4 Portability requirement**

The System is a light weight Web Based Application and Portability is essential in order for the System to achieve its light weight goal as well as being easy to use.

#### **2.2.5 Extendibility requirement**

The System should be designed so any extensions of functionality are easy to implement without having to refactor the whole System.

## 2.3 Design and Architecture



Above is a Diagram of the Systems Architecture. The reason I have chosen this Architecture is because it is a lightweight, and simple Architecture. I have tried to keep it as simple as possible so it can be deployed without a lot of exterior resources required.

This Architecture also allows me to quickly and easily carry out the exact functionality I have outlined, allowing the Web based Client to interact with the Web Server in real time with little to no delay.

## 2.4 Implementation

Describe the main algorithms/classes/functions used in the code. Consider to show and explain interesting code snippets where appropriate.

## 2.5 Graphical User Interface (GUI) Layout

## Lecturer Dashboard View

Classistant

Search...

Dashboard

Settings

Profile

Help

Overview

Reports

Analytics

Export

Nav item

Nav item again

One more nav

Another nav item

More navigation

Nav item again

One more nav

Another nav item

Dashboard

Label

Something else

Label

Something else

Label

Something else

Label

Something else

Section title

#	Header	Header	Header	Header
1,001	Lorem	ipsum	dolor	sit
1,002	amet	consectetur	adipiscing	elit
1,003	Integer	nec	odio	Praesent
1,003	libero	Sed	curus	ante
1,004	dapibus	diam	Sed	nisi
1,005	Nulla	quis	sem	at
1,006	nibh	elementum	imperdiet	Duis
1,007	sagittis	ipsum	Praesent	mauris
1,008	Fusce	nec	tellus	sed
1,009	augue	semper	porta	Mauris
1,010	massa	Vestibulum	lacinia	arcu

## Lecturer Classroom View – Pre Poll

Classistant Dashboard

Secure | https://classistant-x14552067.c9users.io/graph.html

Star

OS

Paul

Classistant

Search...

Dashboard

Settings

Profile

Help

Overview

Reports

Analytics

Export

Nav item

Nav item again

One more nav

Another nav item

More navigation

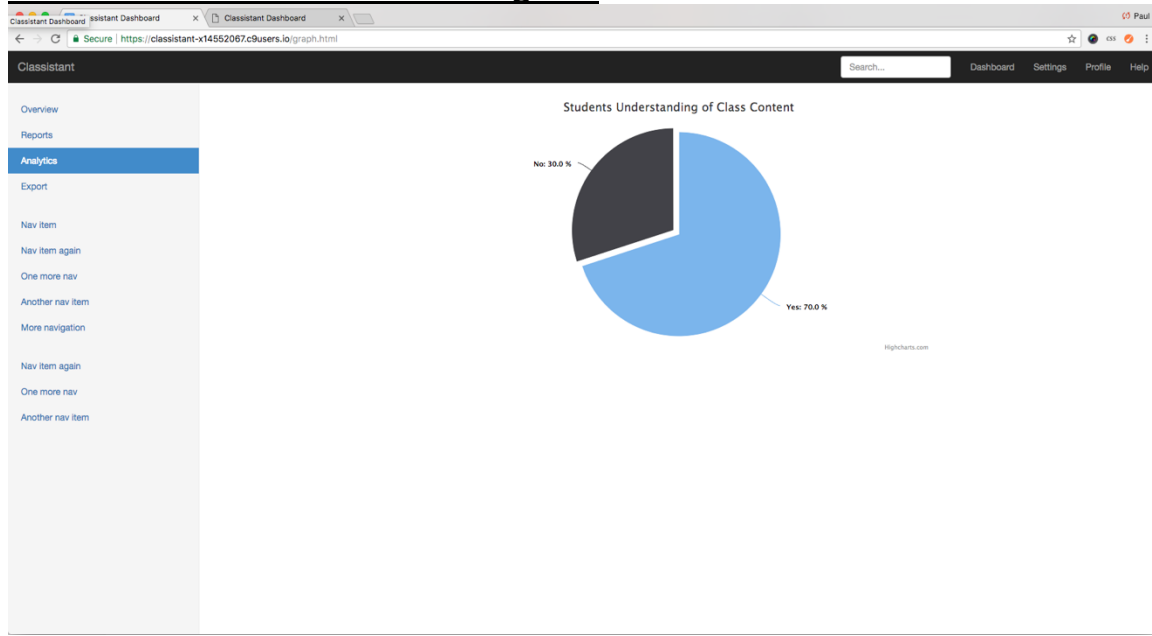
Nav item again

One more nav

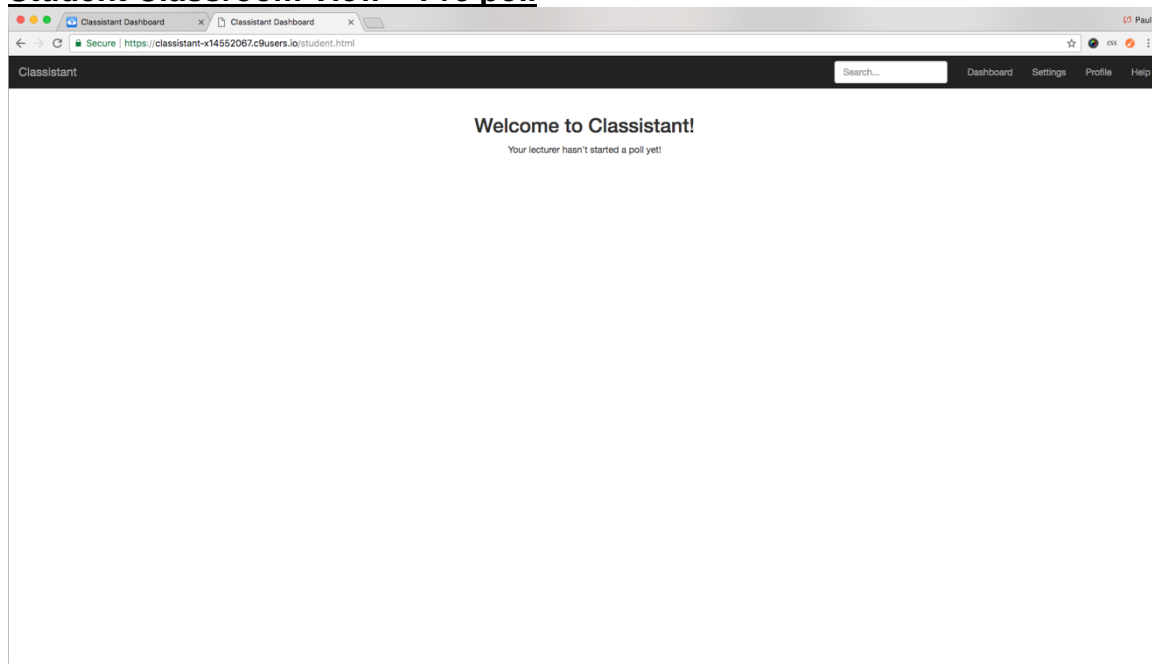
Another nav item

Poll Class Understanding

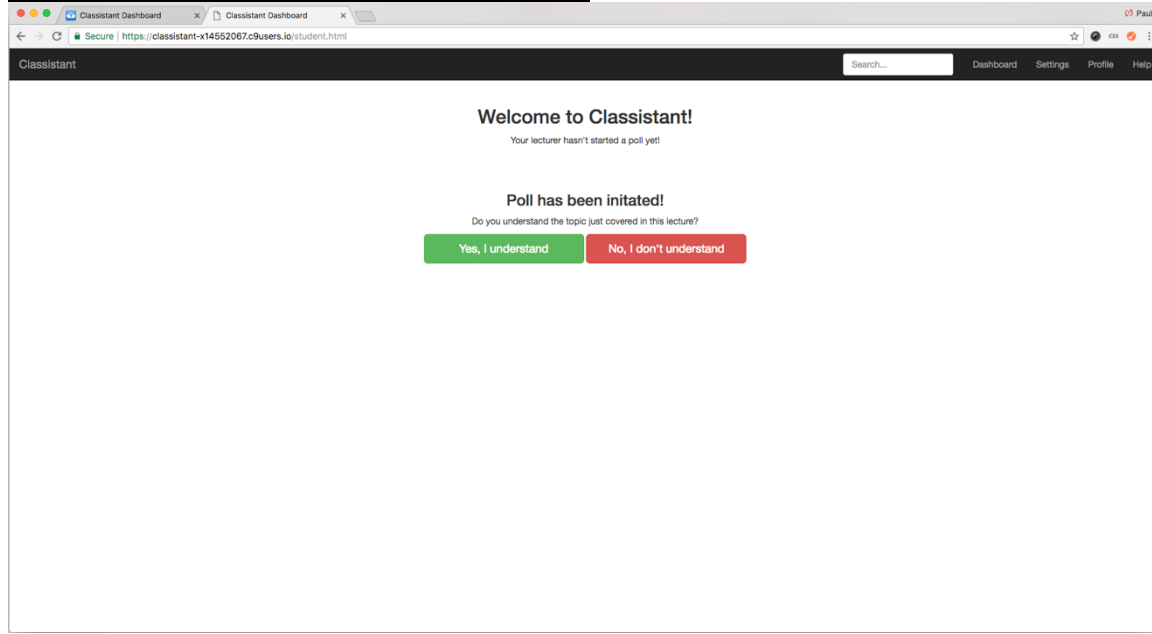
## Lecturer Classroom View – During Poll



## Student Classroom View – Pre poll



## Student Classroom View – During Poll



## 2.6 Evaluation

I plan to perform unit tests on the back-end code, to ensure it works as expected. These will be small tests of specific parts of functionality such as functions and methods, to ensure, given a specific input, a desired output / outcome is returned.

Once the back-end is reaching a point of completion, I will then run integration tests to make sure PubNub is working fine with my framework as well as the Database is connecting and adding / removing without any issues.

The majority of System Evaluation however, will be manual testing. Due to the nature of the System it will require me to login as a Lecturer and Student and simply test the interactions and functionality.

I also plan to stress test the System to see if I can overload or overwhelm it with requests and see if I can find an upper limit of class size. I feel because I am using a free service, it may not handle 100 simultaneous requests with ease, although I may have found a workaround.

Finally, I plan to try run a test "Class" with some peers to see how the System performs and if they actually find the System easy to use / Intuitive, if they would actually use the System in a real-life scenario and elicit comments and feedback.

### 3 Conclusions

The Project, like any other project, has many Advantages and Disadvantages.

#### **Advantages**

- Increase Student < - > Lecturer Interaction
- Web Based - Works on Phone, Tablet or PC
- Responsive
- Real Time – You can see how your class is performing in real-time
- Analytics on responses allows actions to be taken on struggling students and provide extra help

#### **Disadvantages**

- No Mobile Application
- Inconvenient to some (Opening web page every time you need to use it)
- No API
- No Integrations with 3<sup>rd</sup> Party Systems (Moodle etc.)

#### **Opportunities**

- Ability to sell the System / Subscriptions as a product to Education Instruments
- Develop an API to Interface with 3<sup>rd</sup> Party Apps
- Integration into Services like Moodle
- Develop Mobile Applications for both Android and iOS

#### **Limits**

- Application is Web Based
- Majority of the Application is developed with JavaScript
- Class sizes of hundreds may slow System

## 4 Further development or research

With more Research, Time and Resources the Project has the scope to become something far larger. The System could be sold on a Subscription model to any Educational Institute.

The Project could be developed as a full Application for both Android and iOS. The project could have a full-fledged Security System with Encryption. An API could be developed to allow the Systems data to be consumed and outputted to other services, as well as allow the application to consume data from other services.

The System could be integrated into a Service such as Moodle so all of the core functionality could be accessed and used from the Moodle page of the class

Appendix

## 14 Monthly Journals

### 14.1 September Monthly Report

Unfortunately, this report is written retrospectively. The majority of September, for me, was spent in hospital. I was travelling in a car as a passenger when another car blindly pulled out in front. Thankfully, my father, a retired Garda of 30 years, had his training kick in and he totally anticipated the situation and avoided a collision in a safe and controller manner.

The unfortunate side effect however, was a severe case of Whiplash for me and a damaged muscle all through my neck. I was taken by ambulance to Beaumont A&E where I was treated and prescribed bed rest.

It wasn't until late September I fully regained agility for the upper half of my body. Even then, the pain from moving was substantial but life had to go on.

### 14.2 October Monthly Report

This month I actually regained movement to a tolerable extent and I was able to attend college. It is unfortunate that despite continuous communication from people on my behalf, the college labelled me as a "Dosser" and considered me a no show, despite numerous emails sent from my student account detailing the situation. It is nice to know how students with a consistently high grade average, achieving well over a first class honours every year, are easily forgotten about and mistreated.

This month to be honest was incredibly stressful. No work has yet to be done on my final project as projects are already due for my modules!



## 15.1 November Monthly Report

This month was my first month of productivity! I was able to finally get allocated an informal slot to pitch my project presentation. Although this was unfair and left me at a disadvantage as I didn't have a panel to pitch to, Lisa, my project supervisor is an absolute god send and she has really set me on track! She has motivated me to no end and made me see light at the tunnel. I really am lucky to have been paired with Lisa and dealing with her has been an absolute pleasure.

My Project has been approved and I will begin research into implementations alongside catching up work from other modules. It is unfortunate that all my work has just cascaded forward and it's a near impossible balancing act. I will hopefully have my prototype ready in time for the midpoint!

## 15.2 December Monthly Report

This year of college has been the absolute worst by far! I was coming into the semester so excited and ready to work and I am just deflated, both by the lack of understanding from NCIs Machine like Administration, at the best of times, and then the sheer amount of deadlines.

I had to code a full AI for Chess, and Chess to play itself in a week, whereas my peers were given over a month. I did it, but it's just soul destroying.

Thankfully, I managed to get my Midpoint extended until after my Christmas Exams. It looks like things may be doable. I just need to get through my currents CAs.

I researched into my implementation methods and have been weighing up my options. It looks like I am going to use Pubnub's JavaScript MQTT Library to handle all of my real-time interactions.

## 15.3 January Monthly Report

I'm feeling good about this month. Lisa has been going above and beyond and has my mind in a good place. I have a very basic prototype which in my view, is *perfect* for demonstrating the exact concept and end goal I wish to achieve, which is the goal of the Midpoint. It looks like I am ready to present.

The prototype is a simple website, built using just HTML, JS, CSS and Pubnub. All development has been done in Cloud9. The application has a lecturer dashboard and allows a lecturer to go into a classroom and publish data to students. Students will see a poll being pushed to them in real-time, and can subsequently respond in real-time. The lecturers Page has a graph which dynamically updates as poll responses come in.

The presentation is closer to the end of the month. Wish me luck.

## 15.4 February Monthly Report

The midpoint went about how I expected, although the marks were disappointing to say the least. This has left me with a lack of motivation as students with far less work put in for their midpoint (read: no work) got a significantly higher mark.

However, life must go on. I must now balance my deliverables for my modules, teaching support sessions part time to afford my commute to college and my final year project. This is going to be a fun year.

## 15.5 March Monthly Report

This month I have made very little progress with my final year project. My aim is to try finish all my deliverables for other modules to give myself a good enough run at the software project.

It is unfortunate but my family must sell our home. This home has been my home since I was a kid. Unfortunately, due to the health of my parents, a lot of work and preparation will have to be carried out by me.

This leaves me with an unfortunate situation of now balancing Work to afford college, Final Year Project, College Deliverables for modules and now packing up the house and preparing for the sale of our home.

I am in good spirits though and I know Lisa has got my back and has total faith in me, which is a boost. I have been reading into the implementation more and I personally find thinking about a project at length and mapping the whole implementation in my head overtime is hugely beneficial and will pay off tremendously.

## 15.6 April Monthly Report

Wow, its April already! Time this year is absolutely flying. I have stopped teaching as college is less frequent and I have saved enough to afford my remaining commutes.

I have totally scrapped the project. Yep, you read that right, in the bin.

I have been exposed to Ruby on Rails through my Cloud Application Development module and it is a total eye opener and jaw dropper. I have never seen a framework as powerful as Rails and I am so excited to get at it and develop the application.

I have decided although Pubnub is viable and adds complexity in so far as using external services, I feel like using Sockets with Rails will be significantly better, smoother, seamless and in my view, even more complex and you have to code a lot of the functionality by hand that Pubnub would give for free. However, although Pubnub offers real-time capabilities, it will not let me access application code behind the scenes in real-time without the use of sockets *anyway* so it renders itself useless for my purpose.

## 15.7 May Monthly Report

WOW. May. The final month. I never thought I'd see the day I'd be sitting down to write my final Report to be honest.

First off, the house is sold!

The journey has been extremely tough but I am happy with myself for sticking true to my guns and working through tough times. It has helped build me as a character, both by adding my first grey hairs to my body, and by strengthening/toughening when it comes to dealing with stress and focusing.

All of my projects are uploaded except for my Final Year Project. I cannot begin to describe the adrenaline rush it gives me knowing I am almost at the finish line.

For my final sign off in my reports, would like to dedicate all my hard work and efforts during these times to my family, who have supported me day in days out, morally, emotionally and by bringing me endless amounts of cups of tea.

Finally, I don't think I would've made it here today without the help and support of my loving girlfriend of nearly 5 years, Bethany. Without Beth, the project would've simply been an impossible mountain of work covered by stress and personal life. Beth has stuck by my side through thick and thin and I owe her the world and more. She truly has changed my life for the better. If you ever end up reading this Beth, I love you so much and this is all for you.

## 16 Link to Project GitHub

<https://github.com/x14552067/FinalYearProject>