Lecture Feedback System

CS39440 Major Project Report

Author: Morgan Jones ([mwj7@aber.ac.uk](mailto:mwj7@aber.ac.uk))

Supervisor: Dr. Chris Loftus (cwl@aber.ac.uk)

22nd April 2019

Version 1.0 (Draft)

This report is submitted as partial fulfilment of a MEng degree in Software Engineering (With Integrated Year In Industry) (G601)

Department of Computer Science

Aberystwyth University

Aberystwyth

Ceredigion

SY23 3DB

Wales, UK

Declaration of originality

I confirm that:

* This submission is my own work, except where clearly indicated.
* I understand that there are severe penalties for Unacceptable Academic Practice, which can lead to loss of marks or even the withholding of a degree.
* I have read the regulations on Unacceptable Academic Practice from the University’s Academic Registry (AR) and the relevant sections of the current Student Handbook of the Department of Computer Science.
* In submitting this work, I understand and agree to abide by the University’s regulations governing these issues.

Name Morgan Jones

Date 22/04/2019

Consent to share this work

By including my name below, I hereby agree to this project's report and technical work being made available to other students and academic staff of the Aberystwyth Computer Science Department.

Name Morgan Jones

Date 22/04/2019

Acknowledgements

I’d like to thank my supervisor Chris Loftus for his guidance and patience and those who helped me by trying out the system.

Abstract

Include an abstract for your project. This should be approximately 300 words.

The abstract is an overview of the work you have done. Highlight the purpose of the work and the key outcomes of the work.

Contents

1. Background, Analysis & Process 7

1.1. Background 7

1.1.1. Motivation 7

1.1.2. Research 7

1.2. Analysis 8

1.2.1. Requirements (Feature List) 9

1.3. Process 9

2. Design 11

2.1. Overall Architecture 11

2.1.1. Behavioural Diagrams 11

2.1.2. Structural Diagrams 11

2.2. Detailed Design 11

2.3. User Interface Design 12

3. Implementation 14

4. Testing 15

4.1. Overall Approach to Testing 15

4.2. Unit Testing 15

4.3. System Testing 15

4.4. User Testing 16

5. Critical Evaluation 17

5.1. Functional Evaluation 17

5.1.1. Requirements – DO I NEED THESE?? 17

5.1.2. Design 17

5.1.3. Meeting Needs of Users 17

5.2. Work to Extend the Project 17

5.3. Alternate Approach If Redoing the Project 17

5.4. Overall Evaluation 18

6. Annotated Bibliography 19

7. Appendices 20

A. Third-Party Code and Libraries 20

B. Ethics Submission 21

C. Feature List 22

D. Test Cases 23

# Background, Analysis & Process

## Background

The aim of this project is to build a responsive web application that will allow students in the university to give feedback and ask questions anonymously throughout a lecture or workshop.

The idea being that the students would connect to the system by entering a 6-digit code that would be read out at the start of the lecture, the lecturer taking the session could then respond to the feedback and questions as they are provided by perhaps explaining things further or re-visiting misunderstood material.

Any data given would be recorded by the system for review by a lecturer at some point in the future enabling the lecturer to easily see a summary of the feedback and adjust future content delivery accordingly.

### Motivation

The project interested me because it presented opportunity for me to learn and gain more experience with web development and in particular the use of a web framework technology, this was something I had struggled with in the past despite having an interest in and knowing that the ability to develop using a full-stack web framework is a valuable, if not essential, skill in the world of web development.

I was also inclined to take the project because of the accessibility of the problem domain. I myself have been a student for years therefore the transition of starting to think like my end users (students) was an easy one.

### Research

During my research I started recording all activity, links and ideas in a project diary that was heavily used during the first few weeks of the project and continued to be used throughout the project.

I knew I wanted to use a web framework approach because my product was to be a web app. The main three I considered were Ruby on Rails [1], Laravel and Django [2] because all three are popular and well established/supported tools. I chose Django in the end despite having previous experience with Ruby on Rails because I am most competent with the python programming language (Django being the web framework for python). This did mean I had to learn Django from scratch, but that learning experience was one of the reasons I chose the project in the first place.

There were many options for my choice of IDE because the standard approach was simply to use a text editor alongside folder/file navigation. The most common editors I saw used in the various Django tutorials I watched were either “Sublime Text 3” [3] or “atom” [4] text editors. I chose “atom” because it has all the functionality of sublime text but with the added benefit of being open source and therefore having many plugins available that I could make use of during the project; I installed a command line console plugin [5] that allowed me to run the server and access the database directly without having to switch desktop windows and a beautify plugin [6] that auto-formatted my HTML and python code.

I knew the system would have to be responsive because most student users would access it through their mobile phones during a lecture. My CSS skills are limited and therefore chose to use a CSS library to assist in this aspect of the project. Bootstrap was my choice of technology for assisting with the responsive design because the purpose of the technology is to “Build responsive, mobile-first projects on the web” [1]. I have also had some experience with older versions of it so had a feel for how it was used already.

The application will need to store data in a database. I had previous experience with MySQL, SQLite3 and PostgreSQL. I chose PostgreSQL [8] because it has the most advanced features and is therefore the most flexible, there is a lot supporting tutorials for using it with Django and it is the database technology I am most competent with having used it all throughout my year in industry.

I wanted to have the feedback display in a visual way and had previous experience using JavaScript along side the HTML canvas to produce client side graphics; it was my aim at the beginning of the project to use these skills to produce some form of visualisation of feedback in graphs or charts. Later in the project I decided to use a JavaScript library (chart.js) for my data visualisation because it required far less code to be written and tested by me.

As inspiration for my system I looked at a quiz system produced by a student (NAME HERE) as a major project in a previous year. It was a good place to start as it showed me the type of style and quality I should be aiming for. The simplistic style of the site and the way he presented data visually stood out to me as something I would want in my app. I also discussed with my supervisor the means of authenticating staff using a login and connection over LDAP to a university server, this was how the student achieved the staff login functionality on his project; a functionality I too would need to implement.

As result of this I tried reading the LDAP RFC specifications and investigated the use of a python library (ldap3) [9]that would function as an LDAP API for my Django application. My supervisor also forwarded me an email that was provided to (NAME HERE) with instructions on how to connect and process the data returned by the application level protocol (LDAP).

I then read the tutorials on writing a custom authentication backend to accommodate for my authentication via LDAP, this would still allow me to use Django’s built-in authentication system.

The system was to be used by the university. Every other web service the university provides is available in both Welsh and English because the university is bilingual. It seemed appropriate that my app should also be available in Welsh and English.

## Analysis

After the above-mentioned background preparation, the project direction was decided. The system was to be built in python following the Django web framework using a customised version of the atom text editor as an IDE. JavaScript was to be used to add some form of data visualisation and Bootstrap was to be used to ensure responsive design.

Sessions for lectures would only be managed by members of university staff this would be made possible by a user login functionality that utilised authentication over LDAP to a university server. Feedback would only be provided by those with an active 6-digit session code given out by a member of staff at the start of a lecture.

### Requirements (Feature List)

Shortly after my initial research and learning I produced a feature list; as is process convention in an FDD project. I broke the problem down into functionally valuable features and separated these features into four feature sets with estimated weeks for each feature set.

The four feature sets were:

* Authentication
* Lecture Management
* Session Management
* Providing Feedback

The content of these feature sets evolved throughout the project and were not all ordered by dependency; especially between the last two I found my self frequently jumping from working on Session Management features to working on Feedback features and visa-versa.

The authentication feature set started with only one feature which was LDAP authentication, this was because at the time I was unsure on the difficulty of the feature I later included internationalisation in this feature set due to feature being easier to implement than first expected.

The motivation for having sperate Lecture Management and Session Management came from the idea that a single lecture may be given many times to different classes.

Lecture Management is more basic CRUD functionality which expanded throughout the project to include search query functionality and functionality to work with PDF uploads.

Session Management is more to do with managing live sessions through a control panel style page this made use of JavaScript, jQuery and AJAX. This feature set also included the feature for visually displaying feedback data.

Providing Feedback was meant to be the simpler UI as it was intended for use by students on mobiles. It required connecting to an active session and maintaining user data through use of a session on the server.

## Process

I originally intended to follow a personalised version of feature-driven development (FDD) as my engineering process. I chose FDD because it is recommended when building a project which is well defined and would allow me to get a lot of design out of the way during the first three FDD process steps (AKA iteration zero). This approach worked well at the start of my project as it resulted in me producing a range of high-level design diagrams to model the system which served as a useful starting point to which only incremental updates were necessary throughout the project. I also produced an ordered feature list that functioned as a requirements specification and reference point for noting my own progress.

As the project progressed, I found that my progress tracking to be unaligned with FDD because my iterations did not follow the standard 6 FDD iterative milestones. I was working through the features on my feature list but was writing code, updating design and tests at the same time. This hinted at my adoption of XP-style iterations and caused me to abandon my attempts at maintaining an FDD progress tracking report.

Into the second half of the project I found that I was almost entirely writing code and unit tests with very little updates to the formal test or design. This apparent decrease in discipline was due I think to my attempt to make unplanned changes that I was often unsure of and resulted in failures/rollbacks or minor advancements. It is for this reason I have kept this report in the structure of a plan-based project because from an honest perspective the project structure now reflects that with investment in design, implementation and testing in that order.

# Design

## Overall Architecture

Towards the start of the project I produced several high-level UML designs to model the system; these diagrams were updated as the project progressed. I used Visual Paradigm community edition to produce the diagrams.

### Behavioural Diagrams

My behavioural diagrams display the system functionality and detail the intended user workflow.

I used a use-case diagram to brainstorm the functions of the system both the staff and student users would want to access.

I used activity diagrams to describe in detail the set of actions and decisions users would take when using the application.

### Structural Diagrams

I started with an entity relationship diagram describing the underlying data the system would need to store in the database. Focusing on data was recommended to me by my tutor during my year in industry.

I had to redesign the database in the middle of the project to reduce redundant data, make the addition of extra functionality easier and allow the use of more Django conventions.

* I made use of relations produced by Django’s authentication contribution therefore removed attributes from the lecture relation.
* Added a ‘Time’ relation allowing feedback sessions to be re-started by introducing multiple start and end times.
* Shifted attributes from the Lecture relation to the Session relation resulting in more flexibility of the functioning of the application.
* Removed my ‘enum’ relations for the type of feedback options available by instead using a simple varchar type attribute on the feedback table which was restricted only by the hard-coded values supplied to a ChoiceField on a Django model class. This is more conventional for fixed drop-down options in Django.

The component architecture of the system is quite basic, I had originally intended to write a desktop program to go along with the system but that feature never got implemented.

## Detailed Design

I found detailed design difficult and delayed it for most the project. My original goal was to auto-generate it using some tool, I tried to use the pygraphviz python package to do this however ran into an issue to do with missing C libraries that I could not resolve.

There was a general lack of documentation for the standard of how to model Django applications I think due to that fact that the code you write into a Django application is mostly not class structured albeit the MVC framework is.

A common recommendation for low-level design was to describe the Object-Relational mapping classes with a class diagram. This is only obvious aspect of the code requiring a class diagram of its own since all ORM in Django is always defined in classes.

This in my opinion does not as that much value as it is close to structure described in the database, as it would be considering these classes are used to build the migrations that in turn build the database.

Below is the class diagram describing the ORM classes from my applications models.py file. These are the model part of the MVC design pattern on which Django framework is built.

## User Interface Design

During my research and while learning Bootstrap I build mock UI designs of how I imagined different pages of the site would look. This allowed me to have a talking point with my supervisor and others about the look of the site it also allowed me to have a detailed starting point for my HTML templates.

The design of the UI evolved over time as the exact functionality of the pages became clearer. Below is a comparison between the original UI design and the final result.

|  |  |
| --- | --- |
| Figure 1 Login Page Original | Figure 2 Login Page Final |
|  | |
| Figure 3 Lecture List Page Original | Figure 4 Lecture List Page Final |
|  | |
| Figure 5 New/Edit Lecture Page Original | Figure 6 New/Edit Lecture Page Final |
|  | |
| Figure 7 Lecture Detail Page Original | Figure 8 Lecture Detail Page Final |
| Figure 9 Lecture Detail Page (Canvas Graphs at bottom) Original | Figure 10 Feedback Detail Page Final |
|  | |
| Figure 11 Connect Page Original | Figure 12 Connect Page Final |
|  | |
| Figure 13 Feedback Page Original | Figure 14 Feedback Page Final |

One of the main changes between my original design and the final design is the simplification of the interface, which is mainly achieved through removing unneeded text and separating sections more clearly. TALK ABOUT WHAT I LEARNED FROM READING ‘Don’t Make Me Think’ HERE

# Implementation

During implementation my code was debugged using the pdb interactive source code debugger [10] on the command line; the pdb module is part of the python standard library. I created a python virtual environment [11] for my project to have only the dependencies required for the project to run. I used pip to install and manage the project dependencies [12]. The requirements.txt file in my project directory is a file that can be used by pip to install all the python packages my project depends upon at once by use of the ‘pip install -r requirements.txt’ command.

I tried out my system in google chrome on my desktop and mobile device. I used chromes developer tools to debug my JavaScript and view the page style and HTML DOM.

I used git and GitHub for version control and backup during the project. I used a master branch for working commits, a development branch for regular commits and a learning branch for trying out new ideas and spike testing.

## Lecture Management

## Session Management

### Lecture Detail Page

The Lecture Detail page displays

### Feedback Detail Page

To display feedback data on the feedback detail page I planned to use vanilla JavaScript to draw on the HTML canvas. When I came to implement this feature I found an easy to use JavaScript library chart.js that build good quality responsive charts. I used this library to create the bar and pie charts in my application. Use of this library saved me having to write and test a lot of code.

## Providing Feedback

# Testing

## Overall Approach to Testing

For testing I built up a set of unit tests and test cases throughout the project. I then planned to run individual user usability tests to gather feedback on the application. The plan was to do a full documented run through of all the test cases before usability tests and then a full run through of test cases after implementing the changes inspired by the usability test results.

## Unit Testing

My python code was unit tested according to Django’s built in testing framework. As is convention in Django I have structured my unit tests inside a /tests folder with each file of implementation code having a corresponding file of test code. The test file matches the name of the code file with ‘test\_’ prepended to the file name. Django’s unit tests use a Python standard library module: unittest. This module defines tests using a class-based approach therefore every class or function of code in my project has an associated unit test class to test it.

I ran the tests through the command line using the ‘python manage.py test’ command.

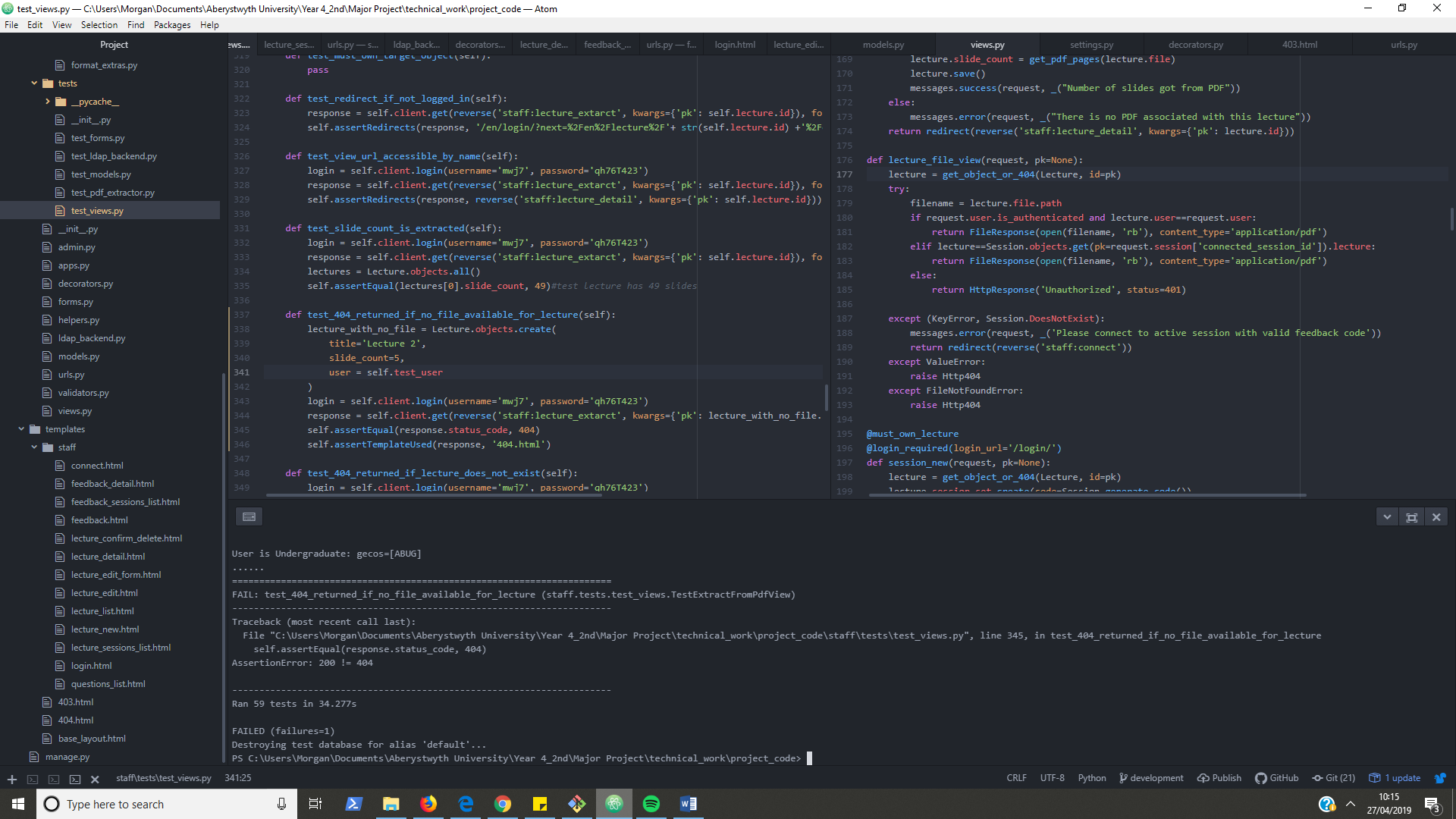


Figure 15 Example of unit testing project through command line while developing inside atom text editor

## System Testing

The test cases for this system are included in appendices.

Every feature on the feature list has a corresponding test scenario; all test cases for a given scenario collectively test a feature has been implemented correctly. Testing is mapped to requirements by each feature in the feature list having a test scenario ID from the test case table.

## User Testing

I performed supervised user testing of the system with individuals. The document for this testing is in the appendices.

A list of tasks was performed by my volunteers some tasks as staff users and some tasks as student users. While the volunteers were performing user tasks I was acting as a staff member to control sessions. While the volunteers were performing staff tasks I was acting as a student user to provide feedback.

I had half my volunteers run through tasks using a desktop and half run through the tasks using a mobile. I didn’t have all volunteers use both mobile and desktop because once they had used the system once they would already know its layout and that felt like cheating.

# Critical Evaluation

## Functional Evaluation

### Requirements – DO I NEED THESE??

### Design

### Meeting Needs of Users

## Work to Extend the Project

The addition of a desktop python program that could display feedback data to a lecturer without the browser having to be open may add value to the system. This program would make use of a web API of my application which could be implemented by use of the Django REST framework.

My program has feedback options that I have defined myself. This is limiting as the feedback that can be given is pre-restricted. It would be an improvement if users running feedback sessions could define their own feedback forms that the application would use allowing each member of staff to tailor feedback options to their individual needs.

## Alternate Approach If Redoing the Project

The most difficult aspect of my project was getting the functionality I wanted on the client side. I ran into multiple issues due to an increase in the amount of JavaScript and jQuery I had to add to the project. In the end I stripped away functionality to get it to a good enough working state.

These problems would not have been an issue if I had taken a different approach with the technologies I had used. I found myself trying to develop two pages that were themselves needing to be more like control panel style single page apps with heavy use of jQuery code and AJAX calls.

The functionality I wanted could be developed much easier if using a client-side view framework instead of fixed templates build on the server-side. I invested time during the project attempting to fix this by trying to learn and integrate react.js into my project. I found it difficult to get working and due to already falling behind my self-set targets decided to discard my changes and continue with my original plan of using Django templates.

If doing the project again I would almost certainly start using react.js integrated with Django from the get-go. I would convert the Django app into a pure web API using the Django REST framework to serialise all data into JSON; then use react.js as an API client to dynamically build and reload components of the UI.

In the final weeks of the project I have been reading about vue.js a client-side framework similar to react but far easier to get started with for beginners due to less initial configuration being required. If I had discovered it earlier, I could have added vue.js into my project for use on the two pages that are client-code heavy; which could have improved the quality of my project considerably.

## Overall Evaluation

I think the project went well overall. I had to refactor continuously throughout as I learnt better ways to implement things using more advanced topics of the Django framework. I learnt about problems I was not aware of at the start and ways to solve them using new techniques and technologies.

The project game me more exposure to web development and in that respect was a success. In my opinion the application is functional as a responsive web application and meets the requirements of a basic feedback system. I think the project in its current state would serve as a good starting point for developing a more robust and advanced system if I could start again taking what I have learned and my reflections into consideration.

The lack of discipline in my engineering process is something that makes the project weak when compared to the standard of a professional software project. I believe the decrease in discipline stems from an uncertainty with the requirements, technologies used to implement them and general lack of experience working on large projects. This is something that comes in time and this project has done nothing but help by providing me with a valuable learning experience.

# Annotated Bibliography

|  |  |
| --- | --- |
| [1] | “Ruby On rails Guides,” [Online]. Available: https://guides.rubyonrails.org/. [Accessed 26 January 2019]. |
| [2] | “Django documentation,” Django Software Foundation, [Online]. Available: https://docs.djangoproject.com/en/2.1/. [Accessed 26 January 2019]. |
| [3] | “Sublime Text,” [Online]. Available: https://www.sublimetext.com/. [Accessed 22 April 2019]. |
| [4] | “Atom Homepage,” [Online]. Available: https://atom.io/. [Accessed 22 April 2019]. |
| [5] | “Atom Terminal-Plus,” [Online]. Available: https://atom.io/packages/terminal-plus. [Accessed 2019 January 30]. |
| [6] | “Atom Beautify,” [Online]. Available: https://atom.io/packages/atom-beautify. [Accessed 2019 April 22]. |
| [7] | “Bootstrap Documentation Introduction,” Bootstrap, [Online]. Available: https://getbootstrap.com/docs/4.0/getting-started/introduction/. [Accessed 31 January 2019]. |
| [8] | “PostgreSQL 11.1 Documentation,” The PostgreSQL Global Development Group, [Online]. Available: https://www.postgresql.org/docs/11/index.html. [Accessed 26 January 2019]. |
| [9] | “The ldap3 Project,” [Online]. Available: https://ldap3.readthedocs.io. [Accessed 28 January 2019]. |
| [10] | “Documentation of pdb module,” [Online]. Available: https://docs.python.org/3/library/pdb.html. [Accessed 28 04 2019]. |
| [11] | “The Django Project,” [Online]. Available: https://www.djangoproject.com/. [Accessed 2019 04 27]. |
| [12] | “Get Bootstrap,” [Online]. Available: https://getbootstrap.com/. [Accessed 27 April 2019]. |

# Appendices

* 1. Third-Party Code and Libraries

**Django** – This is the framework I have used to build my application. It is open source and available from the Django Software Foundation [10]. The framework is released using a BSD license. The framework was used without modification.

**Bootstrap** – This is a CSS framework used to add style and responsive design to my application. It is open source and available the Bootstrap Core Team [12]. The framework is released under the MIT License. The framework was used without modification.

**jQuery** – This is a JavaScript Library used for DOM manipulation and event handling on the client-side. It is open source and available from the jQuery Foundation [15]. The library is released using the MIT License. The library was used without modification.

**Django-autofixture** – This is a python package written by Gregor Müllegger for use with the Django framework that allows me to generate random data which I used while testing the system. It is open source and available from the Python Software Foundation [16]. The library is released using the BSD License. The library was used without modification.

**Django-bootstrap3**– This is a python package written by Dylan Verheul for use with the Django framework that allows bootstrap to work better with Django. It is open source and available from the Python Software Foundation [16]. The library is released using the BSD License. The library was used without modification.

**Ldap3**– This is a python package written by Giovanni Cannata it provides a ldap API to my system that is needed for authentication. It is open source and available from the Python Software Foundation [16]. The library is released using the GNU Lesser General Public License. The library was used without modification.

**Psycopg2** – This is a python package written by Federico Di Gregorio it is PostgreSQL database adapter that allows my application to use a PostgreSQL database. It is open source and available from the Python Software Foundation [16]. The library is released using the GNU Library or Lesser General Public License. The library was used without modification.

**PyPDF2** – This is a python package used in my application to extract meta data from uploaded pdf files. It is open source and available from the Python Software Foundation [16]. The library is released using the BSD License. The library was used without modification.

**Chart.js** – This is a JavaScript library used to create responsive pie and bar charts in my application. It is open source and available from GitHub [15]. The library is available under the MIT License. The library was used without modification.

* 1. Ethics Submission

This appendix includes a copy of the ethics submission for the project. After you have completed your Ethics submission, you will receive a PDF with a summary of the comments. That document should be embedded in this report, either as images, an embedded PDF or as copied text. The content should also include the Ethics Application Number that you receive.

* 1. Feature List

|  |  |  |
| --- | --- | --- |
| **Feature** | **Done?** | **Test Scenario** |

**Authenticating User – Feature Set – 1 Week – (20th-27th February)**

|  |  |  |
| --- | --- | --- |
| Login form for staff with authentication through use of LDAP server via python LDAP API | Done | TS01 |
| Add I18n with Welsh localisation | Done | TS02 |

**Lecture Management – FS– 2 Weeks – (27th-13th)**

|  |  |  |
| --- | --- | --- |
| Create new lecture | Done | TS03 |
| List & delete any/all lectures   * Only those created by the currently logged in user | Done | TS04 |
| View each lecture data | Done | TS05 |
| Edit each lecture data | Done | TS06 |
| Search the list of lectures for a lecture with a specific title or date | Done | TS07 |
| Have download PDF functionality for students and staff on the app | Done | TS08 |
| Extract the meta-data of a lecture/workshop   * Number of Slides | Done | TS09 |

**Session Management – FS – 1 week – (13th – 22nd March)**

|  |  |  |
| --- | --- | --- |
| Start a feedback session for a lecture | Done | TS10 |
| Generate the 6-digit code for a session | Done | TS11 |
| View the questions of a session as they are asked | Done | TS12 |
| View all questions previously asked | TODO | TS13 |
| Stop a feedback session for a lecture | Done | TS14 |
| Delete a feedback session | Done | TS15 |
| Mark an asked Question as Answered | Done | TS16 |
| Toggle Questions for a session | Done | TS17 |
| Merge a session with the immediately previous session |  |  |
| Merge a session with the immediately next session |  |  |
| Use AJAX to update page data with having to reload page |  |  |
| Graphically display the feedback of a session using JavaScript and canvas on separate page | Done |  |

**Providing Feedback - FS – 2 weeks– user Testing Upon Completion**

|  |  |  |
| --- | --- | --- |
| Connect the users to a session   * Only if they have the session code and the session is active | Done |  |
| Associate Feedback to a specific lecture slide or provide overall feedback | Done |  |
| Change feedback for a specific lecture slide | Done |  |
| Ask a question to the person running the session | Done |  |
| Delete a previously asked question | Done |  |

* 1. Test Cases