Waterford Institute of Technology

INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

Uathcruthú

H.Dip. Sci. – Computer Science (KCOSC)
Department of Computing & Mathematics
WIT

Project – Report
Cathal Henchy – 20091405
18 April 2022

Supervisor: Clodagh Power

1 Declaration of Authenticity

I declare that the work which follows is my own, and that any quotations from any sources (e.g. books, journals, the internet) are clearly identified as such by the use of ‘single quotation marks’, for shorter excerpt and identified italics for longer quotations. All quotations and paraphrases are accompanied by (date, author) in the text and a fuller citation is the bibliography. I have not submitted the work represented in this report in any other course of study leading to an academic award.

Student [illegible] Date 18th April 2022
Work Place Mentor ________________________________ Date ________________________

2 Acknowledgments

This project would not have been possible without the help of a few people that I would like to acknowledge:

I would like to thank my supervisor Clodagh Power for her help and continued support.

I would like to thank the ‘customer’ Colm Dunphy for providing me with this idea and helping to scope the project.

3 Preface

In Autumn of 2021, when tasked with coming up with a subject to base this project on, I struggled to conceive any feasible, valid applications. I struggled to the point that it became incumbent on my provisional supervisor at the time (Colm Dunphy) to suggest something for me. I initially suggested some sort of database for hurling and football statistics, which a user could execute queries upon. As Colm pointed out, this would be entirely unrealistic given the need for large amounts of data acquisition – data that probably doesn't *formally* exist anywhere in the first place – which would be difficult to obtain for a number of reasons.

Colm then suggested I create a process that would automate a manual task that he had to carry out himself, on the processing of these very project submissions themselves, for the H.Dip. Given that I have an inherent love for the concept of automating repetitive tasks, and that prior to enrolling in this course I'd implemented this interest to great success at work using excel (Which would be the closest thing to programming experience that I had prior to starting the H.Dip.), I was chomping at the bit to get started.

The practical nature of this concept also appealed greatly to me. The fact that this application would (hopefully – at the time) be actually used in practise (within days of writing the code!), served as great motivation to get moving. The practical, applied nature of this concept, and the fact that it was essentially a reality before I even got started served as a great incentive, and having a practical – not theoretical – goal to aim for, was a great psychological boost to get started.

4 Contents

1 Declaration of Authenticity 2
2 Acknowledgments 3
3 Preface 4
5 Abstract 6
6 Abbreviations 7
7 Figures 8
8 Tables 9
9 Keywords 9
10 Uathcruthú 10
10.1 Introduction 10
10.1.1 Background 10
10.1.2 Framework, Language & IDE 12
10.2 Application Walkthrough 13
10.2.1 Modelling 14
10.2.2 Student User Experience 16
10.2.3 Administrator User Experience 23
10.3 Problems Encountered/Bugs Remaining 35
10.3.1 Image Manipulation 35
10.3.2 Handbook Creation Time Out 37
10.3.3 Access to PDF Files 37
10.3.4 Miscellaneous 38
11 Conclusion 40
11.1 Summary 40
11.2 Key Learnings 40
12 Appendices 41
Appendix A: External Project Links 41
Appendix B: Application Models 42
Appendix C: Controllers/Utilities & Functions 43
13 Bibliography 44

5 Abstract

As stated on the cover page, this is the report for my project for the H.Dip. in Science in Computer Science (KCOSC) at WIT. All such projects (including this one) have a submission process which involves submitting a number of data items related to the project (e.g., project title, project type etc.) on Moodle. After all of the above has been submitted, the course administrators manually disseminate this data into a 'showcase' document whereby the work of the student can be exhibited on one page in a user-friendly, concise manner.

The purpose of this project was to automate this process, by creating a web application that consisted of a web-form that the user (student) could use to submit their work in the exact same way as they currently do.

However, this application would then reassemble all submissions into a standalone showcase file – thus saving the college admin from doing so.

6 Abbreviations

IDE	Integrated Development Environment
PDF	Portable Document Format
URL	Uniform Resource Locator
NDA	Non-Disclosure Agreement
URI	Uniform Resource Identifier
DevOps	Development and Operations
XSS	Cross-site Scripting
SQL	Structured Query Language
TCP	Transmission Control Protocol
IP	Internet Protocol
ICT	Information Communication Technology
EWD	Enterprise Web Development
CS&N	Computer Systems and Networks
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
CSS	Cascading Style Sheets
JS	JavaScript
I/O	Input/Output
CPU	Central Processing Unit
UX	User Experience
CRUD	Create, Read, Update, Delete
API	Application Programming Interface
IoT	Internet of Things
DRY	Don't Repeat Yourself
GIF	Graphics Interchange Format
UI	User Interface

7 Figures

Figure 1: Sample Showcase Page 10
Figure 2: Sample Showcase Page with Deliverables labelled 11
Figure 3: Creation of Submission Object........ 16
Figure 4: Student Submission Form with no data entered, but previously submitted data rendered as placeholders........ 17
Figure 5: Example of if statements used to prevent blank web fields from overwriting previously submitted data........ 18
Figure 6: functions required to submit an image without persisting stale data 18
Figure 7: Function that hides the project type drop down menu, unhides the project type text field, and checks the (hidden) 'projectTypeOther' checkbox 19
Figure 8: Project Type Input field before, during and after “Other” option has been selected........ 20
Figure 9: Submission Output View........ 20
Figure 10: Function that hides all non-relevant fields, if the NDA checkbox is selected........ 21
Figure 11: Student Submission Form after the NDA checkbox has been selected........ 22
Figure 12: Submission Output View after all data has been entered, and the NDA checkbox has been selected 22
Figure 13: Submission Output View after no non-essential data has been entered, and the NDA checkbox has been selected 23
Figure 14: adminSubmission Creation 24
Figure 15: Admin Home Page 25
Figure 16: Admin submission form for student 26
Figure 17: Admin submission form for student with Project Type field now editable........ 27
Figure 18: Admin submission form for student if their submission is protected under NDA 28
Figure 19: Student PDF File........ 28
Figure 20: Student PDF file protected under NDA........ 29
Figure 21: Error message, generated due to omission of data required for PDF creation........ 29
Figure 22: Project Type Other Screen 30
Figure 23: Admin Submission Form 31
Figure 24: Extract of loop that creates user submission list in Admin PDF file 33
Figure 25: Admin PDF File 33
Figure 26: Changing parameter format to maintain consistency, and compatibility with all functions 34
Figure 27: Snippet of Image Data URI for relatively small image (generated by EZGIF.COM (Anon., 2017)) 35
Figure 28: Addition of three images using the image-data-URI module to first extract the base-64 Data URL, and then the jsPDF addImage() function 36
Figure 29: Example of the inclusion of a project type (Web App) for which there are no submissions to list 38
Figure 30: Example of the relative coordinate scheme, using quotients instead of multiples for return values 39

8 Tables

Table 1: Moodle Submission Format prior to Project Implementation 10
Table 2: Summary of iterations 13
Table 3: Attributes of Submission Object 15

9 Keywords

JavaScript, Node.js, Hapi, jsPDF, pdf-merger-js, Cloudinary, image-data-uri

10 Uathcruthú

10.1 Introduction

10.1.1 Background

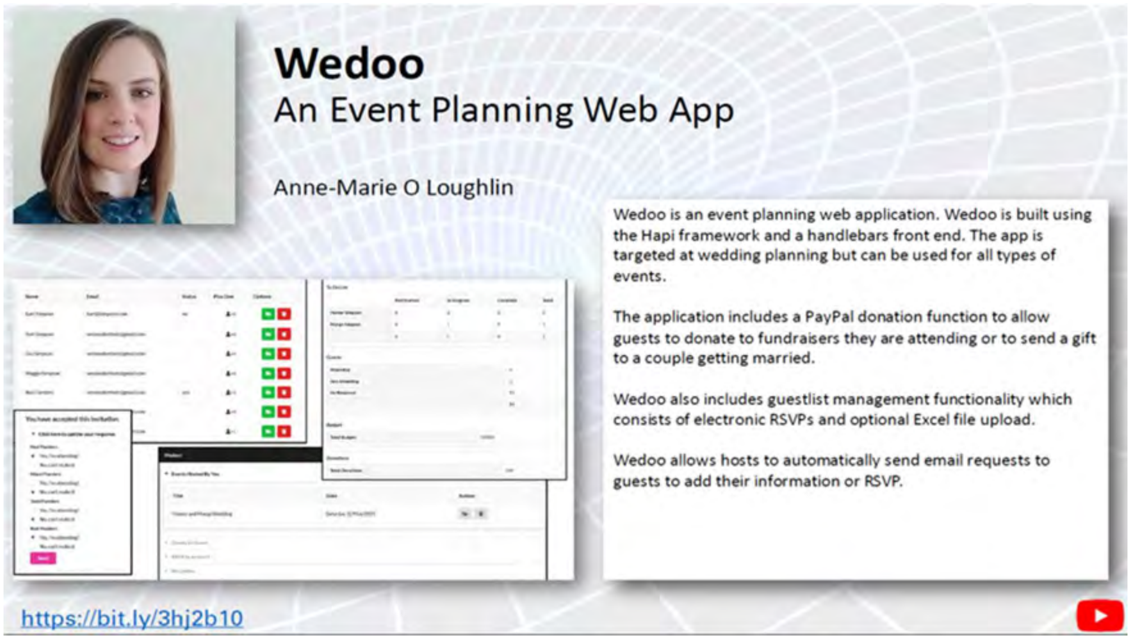
As stated on the cover page, this is the report for my project for the H.Dip. in Computer Science at WIT. All such projects (including this one) undergo a submission process that involves submitting a number of descriptive data items that are manually processed into a showcase handbook (as a means for summarising each individual project, and to serve as a guide for the panel of reviewers for each student's presentation).

The student submits these to Moodle by dragging and dropping three files as per Table 1.

Deliverable	Submission Format
Student's name	
Project Title	
Descriptive Title	
Description of project	Document File (.pdf, .docx, .txt etc.)
Project landing page URL	
Demonstration Video URL	
Project Type	
Personal Digital Photograph	Image File (.png, .jpg etc.)
Project Image	Image File (.png, .jpg etc.)

Table 1: Moodle Submission Format prior to Project Implementation

The course administrator then manually downloads these files, browses through them to ensure that all data is present & correct (reverting to the student if not, e.g., the student hasn't exceeded the 100-word limit on the project description etc.), ensures all data is compatible (i.e. the image files are the correct format etc.) and transfers it into the desired showcase file. This file now serves as a showcase or an exhibition/summary of the work carried out by that student on that project:



Wedoo

An Event Planning Web App

Anne-Marie O Loughlin

Wedoo is an event planning web application. Wedoo is built using the Hapi framework and a handlebars front end. The app is targeted at wedding planning but can be used for all types of events.

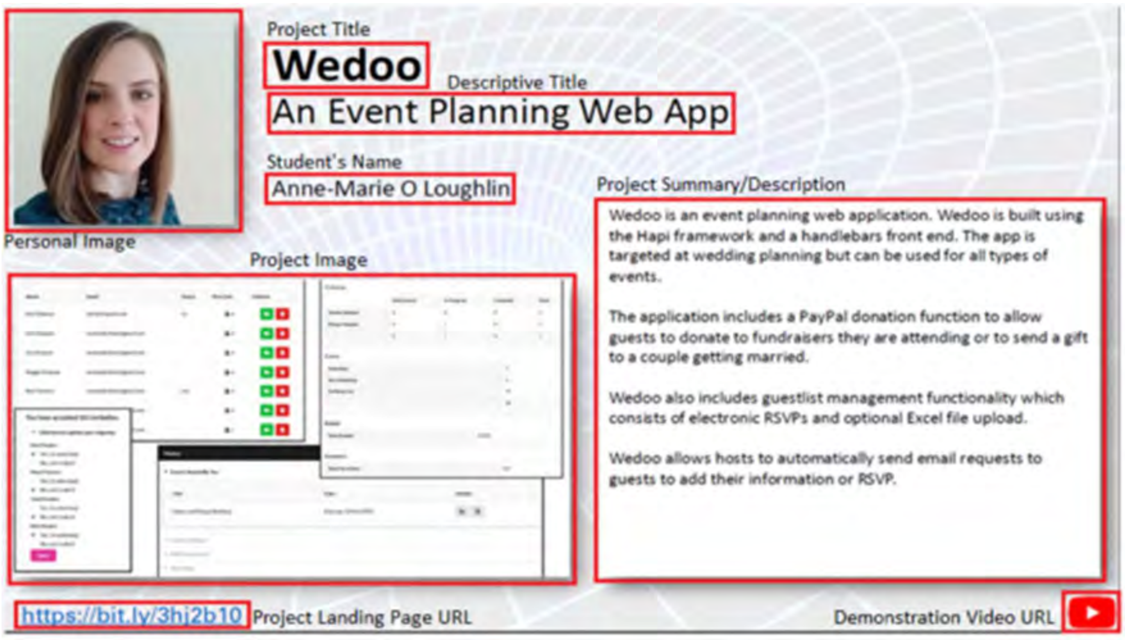
The application includes a PayPal donation function to allow guests to donate to fundraisers they are attending or to send a gift to a couple getting married.

Wedoo also includes guestlist management functionality which consists of electronic RSVPs and optional Excel file upload.

Wedoo allows hosts to automatically send email requests to guests to add their information or RSVP.

https://bit.ly/3hj2b10

Figure 1: Sample Showcase Page


Project Title
Wedoo
Descriptive Title
An Event Planning Web App
Student's Name
Anne-Marie O Loughlin
Personal Image
Project Image
Project Summary/Description

Wedoo is an event planning web application. Wedoo is built using the Hapi framework and a handlebars front end. The app is targeted at wedding planning but can be used for all types of events.

The application includes a PayPal donation function to allow guests to donate to fundraisers they are attending or to send a gift to a couple getting married.

Wedoo also includes guestlist management functionality which consists of electronic RSVPs and optional Excel file upload.

Wedoo allows hosts to automatically send email requests to guests to add their information or RSVP.

https://bit.ly/3hj2b10 Project Landing Page URL
Demonstration Video URL

Figure 2: Sample Showcase Page with Deliverables labelled

This process has many problems. There is very little data validation on Moodle. All that exists is the maximum file size (2GB) and the number of files allowed per submission (ten). Therefore, the student can submit whatever file they want (under 2GB) using the drag and drop method. Any issues with the files that have been uploaded would therefore lead to the administrator having to revert back to the student.
This approach is also subjective in nature. Even if the student hands up all files in the correct format, there is nothing to stop them from compiling their text data in the file in whatever order that they choose. The administrator thus has the arduous task of having to manually process multiple different files of varying format (.txt, .docx etc.), each of which could have the data set in different order, separated by spaces, separated by line breaks, one piece of data per page, etc. The scope for error on the part of the administrator is therefore huge (e.g., creating a showcase file with the URL landing page given as the project title).
Ultimately, this all leads to an extremely labour-intensive process, with a lot of time spent on the processing of each submission, but still no guarantee that it is submitted or processed correctly.

Uathcruthú is a Node.js web application that serves to automate this process. It does so by transforming the user interface from a drag and drop feature, to a webform. Instead of drafting a document file with all of the text data, the student instead inputs this data into the web form, and inputs both image files also.

10.1.2 Framework, Language & IDE

Throughout the H.Dip, I had gained a decent grasp of Java, JavaScript and Python from the Prog., ICT I/EWD, and CS&N/DevOps modules respectively, and thought it best to limit my options to these languages, as it seemed unfeasible to learn a new language from scratch. And from this selection I ruled out Python, as it is more orientated towards the server-side, and in order to process HTML, CSS and JS requests (client-side processes), very complex frameworks would be required (Johansson, 2020). Java also didn't appeal to me, as it seems to be a broader applied language, with capabilities in desktop application development, enterprise-scale development and back-end development, whereas JavaScript is a simple language that is more focused on front-end web development and is specialised in browser compatibility and HTML interaction (Rose, 2019). JavaScript is an interpreted, scripting language that can be executed at the run-time and doesn't require pre-compilation (unlike Java). Its code can be written straight into HTML (Brewster, 2020), and given that my app is more focused on the front-end HTML, JavaScript seemed like the best, simplest and most economic option.

When exploring the available JavaScript frameworks, I studied the attributes and features of a couple of different environments, but Node.js immediately jumped out as being most appropriate.

Node.js - which was covered extensively in the EWD module – is one of the most popular JavaScript platforms at present:

- It supports JavaScript at both the front-end *and* the back-end (Dziuba, 2020), which greatly simplifies things for my application.
- It allows for the development of highly scalable applications (Dziuba, 2020), which is a very important consideration for Uathcruthú, because if this application does see the light of day in WIT, it may well be used for other submission types, and undoubtedly, the course administrators will edit and update their showcase formats as time goes on, thus necessitating scalability.
- It is a non-blocking I/O, single threaded, asynchronous runtime environment (Dziuba, 2020). This means that it allows for multiple processes to be processed simultaneously, but only while using the one thread that makes all communication between the server and the client, while all requests are being processed or waiting in the Event Queue (Hamedani, 2018). As this would be an I/O driven application with many clients accessing it simultaneously, and not a CPU-intensive application (which would result in longer wait times to process CPU heavy requests), the asynchronous approach is ideal.

For selecting the appropriate Node.js framework to go with, I ultimately chose Hapi, as it is works more with plug-ins, as opposed to Express which relies on middleware for parsing data etc. (Swersky, 2018). This would give Hapi an edge, as the entire front-end of the Uathcruthú application is a webform that needs input data to be parsed. Also, although I'd not yet established how I was going to differentiate between users (i.e., Uathcruthú could have been a plug-in to Moodle, it could have use OAuth authorisation to Moodle, or it could have been its own standalone application that had its own authorisation), I thought the authorisation features already contained within Hapi (Dhaduk, 2021) were a safe bet, to keep all eventualities covered.

10.2 Application Walkthrough

This project consisted of a number of incremental, iterative releases, with each release taking on some new product backlog items, or refining/improving previously released ones.

Table 1 below provides a concise (not all patches etc. are included) high-level summary overview of the development of Uathcruthú, including all major and minor releases.

Development Phase	Version	Features
Pre-Release	v0.0.0-alpha	User can submit data
	v0.1.0-alpha	Data Persistence in database
		User data can be viewed
	v0.2.0-alpha	Creation of Student PDF file
	v0.3.0-alpha	PDF file is downloadable
	v0.4.0-alpha	Refined the PDF file creation
		Project Type' Drop-down for Student, and Administrator can edit 'Other' project types on each submission (if 'Other' has been selected)
		Deadline can be set using a drop-down calendar, and the student's view is affected by the date set
		Selection of 'NDA' checkbox automatically changes webform input options, and redacts any previously submitted data
	v0.5.0-alpha	Creation of Admin PDF file
		Creation of Handbook PDF file
Working Release	v1.0.0	Refined all existing functionality for first working release
	v1.1.0	Handbook is no longer created based on Student and Admin data that is in the cache, and creation of said data is no longer a pre-requisite to Handbook creation
	v2.0.0	Bug Fixes
	v2.1.0	Conditional submission completion colour scheme
	v2.1.1	README.md

Table 2: Summary of iterations

10.2.1 Modelling

Initially, the structure of the process was focused on the student only, with the intention being that the student would input their data, and then generate the PDF file based on this. The PDF file would be accessible to the administrator, either by calling the same function that the student called to create it, or by accessing the pre-generated file from a database.

A student model (labelled as ‘user’, as that was the only user intended at the time) was developed with basic attributes. The student creates a ‘submission’, which necessitated a submission model. Initially this model consisted of all the data that would be rendered on the PDF file, the project type, and the id of the submitter (i.e., the student whose file this pertained to) however, as the application development progressed, a few Booleans were added, which helped control certain views and functions (more on this in Section 10.2.2).

Midway through the project, as the student PDF file creation was realised, the scope was increased to include the functionality of merging all student PDF files together, and the submission of additional administerial data that could also be added to this PDF file merger – thus creating the ‘Handbook’. This necessitated the addition of two new models: the ‘admin’ and ‘adminSubmission’ models, that closely resembled the ‘user’ and ‘submission’ models respectively.

Attribute	Purpose	Input Method	Type
Student First Name	Rendered on Student PDF File.	User Object counterparts are declared as such when the student registers, or when an unregistered Student logs in for the first time.	String
Student Last Name			
Project Title		Inputted by Student through web-form text fields.	
Descriptive Title			
Summary			
Project URL			
Video URL		Inputted by Administrator through web-form text field.	
Personal Photo		Uploaded by Student through web-form file upload fields.	
Project Image			
Project Type	Used to categorise each submission PDF and thus group together in Handbook.	Selected by Student from web-form drop-down menu.	
Submitter (The student)	Links Submission to Student.	User Object is declared as such when the Student registers, or when an unregistered Student logs in for the first time.	Object
NDA	If true, only the personal photo is required for submission. Any previously entered data is excluded from PDF.	Selected by Student using web-form checkbox.	Boolean
Project Type is Other	If true, the student is prompted to specify details.	Declared as true if student selects “Other” from web-form drop-down menu.	
	If true, the administrator has the ability to edit the specified project type that was entered by the student.		
Presentation Time	Rendered on admin pages of Handbook PDF File.	Selected by Administrator using web-form drop-down calendar.	String
	Originally intended to be used to order submissions lists on admin pages, but this was never realised. Can be implemented in future releases.		
Submission is Incomplete	If true, submission is excluded form Handbook PDF File.	Declare as true if the ‘presentation time’ or any of the required data for rendering on user PDF file is not entered.	Boolean
	If true, the submission listing on the Admin Home screen is highlighted in red.		

Table 3: Attributes of Submission Object

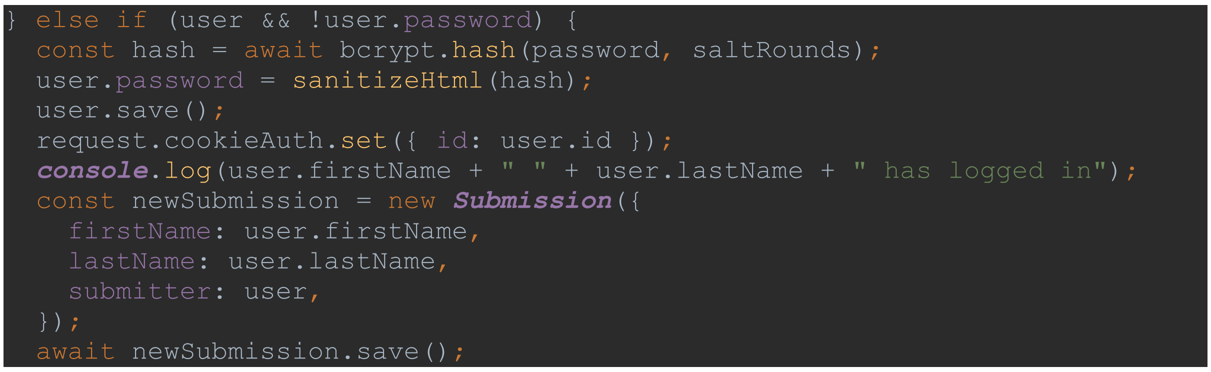
10.2.2 Student User Experience

My key focus on this application was Student UX simplicity. I wanted to have as little functionality as possible for the student. The student logs in, uploads their data, can edit previously uploaded data and that's it. So basically the 'CRU' of CRUD would be at the disposal of the student and nothing else.

10.2.2.1 Prevention of 'Stale' Submissions

Given that the student was only ever going to make a single submission, I didn't want to create a situation whereby the creation of multiple submissions would be possible, and the need to delete obsolete or unwanted submissions would be needed. In order to prevent the creation and storage of superfluous data, I thought the best approach was to strip the student of the ability to actually create anything directly. They would only be able to update an existing submission, and all submissions that they would 'make' – including the first one, and all subsequent ones – would actually be updates or edits to the original one.

The 'original submission' - i.e., the only submission ever created for each student – is created when the student registers, or when an unregistered student logs in for the first time, so all students on the database have one corresponding submission. As this creation happens in the background, no attributes (Boolean or otherwise) are declared, only those that are directly related to the actual existence of the student (i.e., student first name, student last name, and the student object itself), with all others remaining undefined, until explicitly defined by the student and administrator, or implicitly defined based on the actions of the student and administrator (some of the Boolean attributes):

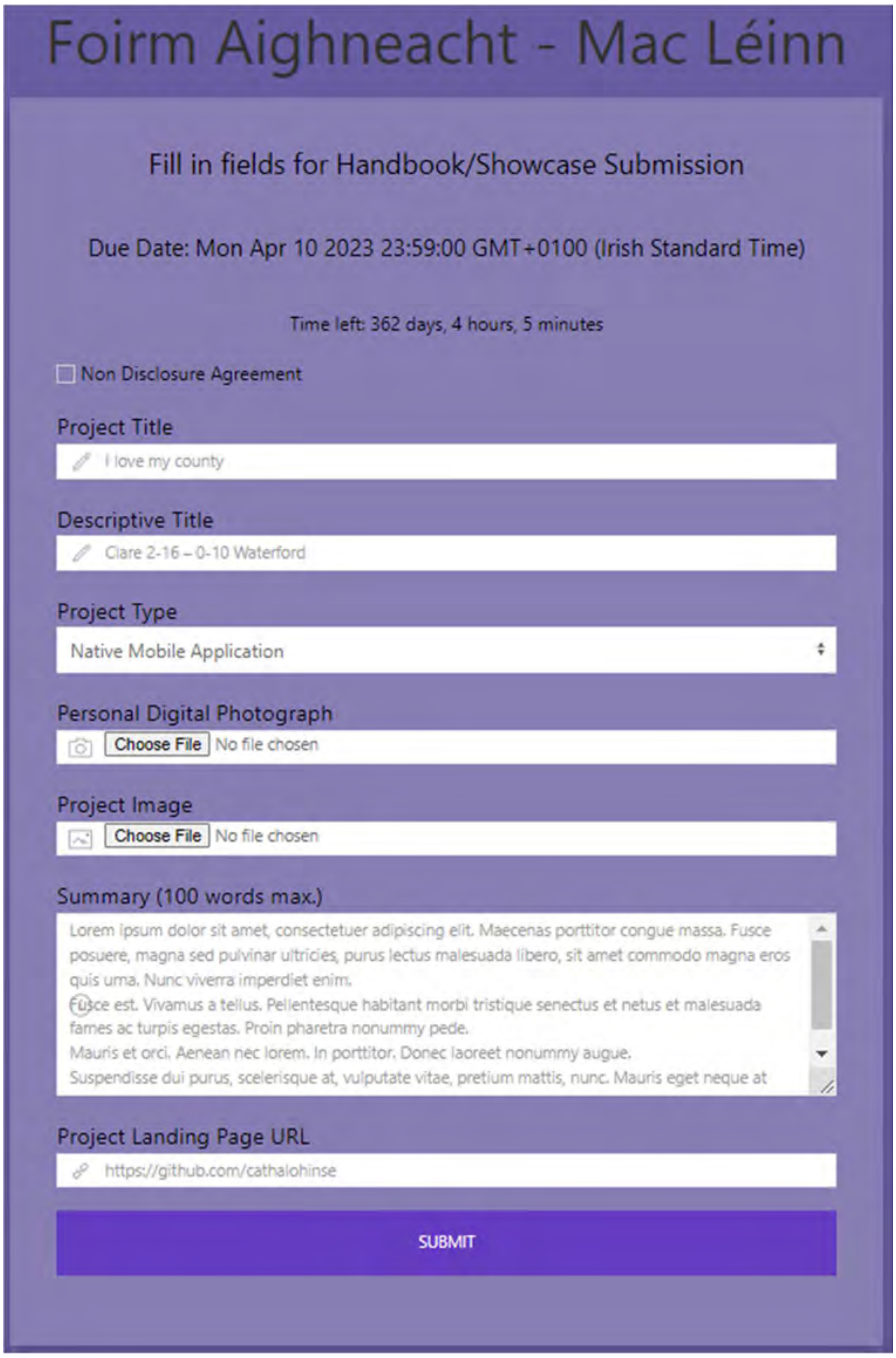


```
} else if (user && !user.password) {
  const hash = await bcrypt.hash(password, saltRounds);
  user.password = sanitizeHtml(hash);
  user.save();
  request.cookieAuth.set({ id: user.id });
  console.log(user.firstName + " " + user.lastName + " has logged in");
  const newSubmission = new Submission({
    firstName: user.firstName,
    lastName: user.lastName,
    submitter: user,
  });
  await newSubmission.save();
```

Figure 3: Creation of Submission Object

10.2.2.2 User/Submission Object Trade-Off

Originally, instead of defining first name and last name attributes for the submission object, I just used the pre-existing corresponding attributes from the user object. Seeing as they are intrinsically linked, and because the user object is directly linked to the submission object anyway, it seemed counter-intuitive to create excess data and complicate the submission object. However, as time went on in the application development, the simplification of the submission object was offset by the increasingly common need to link user object id to each view requiring the use of the user first name and last name. In order to create PDF files, and to render user name details on screen, not only was the submission data required, but so was the user data. It therefore seemed like the best approach was to duplicate the user name attributes to the submission object.



Foirm Aighneacht - Mac Léinn

Fill in fields for Handbook/Showcase Submission

Due Date: Mon Apr 10 2023 23:59:00 GMT+0100 (Irish Standard Time)

Time left: 362 days, 4 hours, 5 minutes

☐ Non Disclosure Agreement

Project Title
I love my county

Descriptive Title
Clare 2-16 – 0-10 Waterford

Project Type
Native Mobile Application

Personal Digital Photograph
Choose File No file chosen

Project Image
Choose File No file chosen

Summary (100 words max.)
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Maecenas porttitor congue massa. Fusce posuere, magna sed pulvinar ultricies, purus lectus malesuada libero, sit amet commodo magna eros quis urna. Nunc viverra imperdiet enim.
Fusce est. Vivamus a tellus. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Proin pharetra nonummy pede.
Mauris et orci. Aenean nec lorem. In porttitor. Donec laoreet nonummy augue.
Suspendisse dui purus, scelerisque at, vulputate vitae, pretium mattis, nunc. Mauris eget neque at

Project Landing Page URL
https://github.com/cathalohinse

Figure 4: Student Submission Form with no data entered, but previously submitted data rendered as placeholders

```
if (submissionEdit.projectTitle !== "") {
  submission.projectTitle = sanitizeHtml(submissionEdit.projectTitle);
}
if (submissionEdit.descriptiveTitle !== "") {
  submission.descriptiveTitle =
sanitizeHtml(submissionEdit.descriptiveTitle);
}
```

Figure 5: Example of if statements used to prevent blank web fields from overwriting previously submitted data

10.2.2.3 Maintenance of Cloudinary Database - Prevention of 'Stale' Images

The image data is stored in string format. This works through the *Cloudinary* API, which stores the submitted image, and returns the URL of the image location. In continuity with my efforts to prevent the database from being overloaded with unnecessary submissions, I also wanted to avoid the Cloudinary database from being swamped with stale data, so this required a somewhat more complex function that would delete the existing entry associated with the specific user/submission object. In order to use the 'delete' API, the existing images string attribute is fetched, and then public id is extracted from it and used as a parameter in the delete function. Then the new image submission is uploaded to Cloudinary, with the resulting URL string saved within the submission object in the database:

```
if (submission.personalPhoto !== undefined) {
  const personalPhotoFileName = await submission.personalPhoto.substr(
    submission.personalPhoto.lastIndexOf("/") + 1
  );
  const personalPhotoPublic_id = await personalPhotoFileName.substr(0,
personalPhotoFileName.indexOf("."));
  await ImageStore.deleteImage(personalPhotoPublic_id);
}
const personalPhotoResult = await
ImageStore.uploadImage(submissionEdit.personalPhoto);
const personalPhotoUrl = await personalPhotoResult.url;
submission.personalPhoto = await personalPhotoUrl;
```

Figure 6: functions required to submit an image without persisting stale data

10.2.2.4 Project Type 'Other'

If the user selects 'Other' from the Project Type *drop-down list*, the drop-down list is then hidden, and a new *text field* appears, with an onscreen prompt to specify the name of the 'Other' project type, by inputting it into said text field. This view also contains a hidden 'projectTypeOther' checkbox, that is checked. This declares the projectTypeOther as true, which in turn will allow its field in the admin forms to be edited. This is controlled by a simple <script> function within the .hbs file that is triggered by an onchange event:

```
<div id="projectType">
    <label style="color: black; font-size:120%" class="uk-form-label"
for="form-stacked-select">Project Type</label>
    <div class="uk-form-controls">
        <select id="projectTypeSelect" name="projectType" class="uk-select"
onchange="otherPleaseSpecify()">
            {{#if submission.projectType}}
                <option id="currentSelection"
value="{{submission.projectType}}"
selected>{{submission.projectType}}</option>
            {{/if}}
            {{#unless submission.projectType}}
                <option id="currentSelection" value="" disabled selected
hidden>Select Project Type</option>
            {{/unless}}/>
        </select>
    </div>
</div>
```

```
<script type="text/javascript">
    function otherPleaseSpecify(){
        if(document.getElementById("projectTypeSelect").value === "Other"){
            document.getElementById("pleaseSpecify").hidden = false;
            document.getElementById("pleaseSpecifyInput").disabled = false;
            document.getElementById("projectType").hidden = true;
            document.getElementById("projectTypeSelect").disabled = true;
            document.getElementById("projectTypeOther").checked = true;
        } else
if("{{projectTypes}}".includes(document.getElementById("projectTypeSelect")
.value) === true){
            document.getElementById("pleaseSpecify").hidden = true;
            document.getElementById("pleaseSpecifyInput").disabled = true;
            document.getElementById("projectType").hidden = false;
            document.getElementById("projectTypeSelect").disabled = false;
            document.getElementById("projectTypeOther").checked = false;
        } else{
            document.getElementById("projectTypeOther").checked = true;}
    }
</script>
```

Figure 7: Function that hides the project type drop down menu, unhides the project type text field, and checks the (hidden) 'projectTypeOther' checkbox

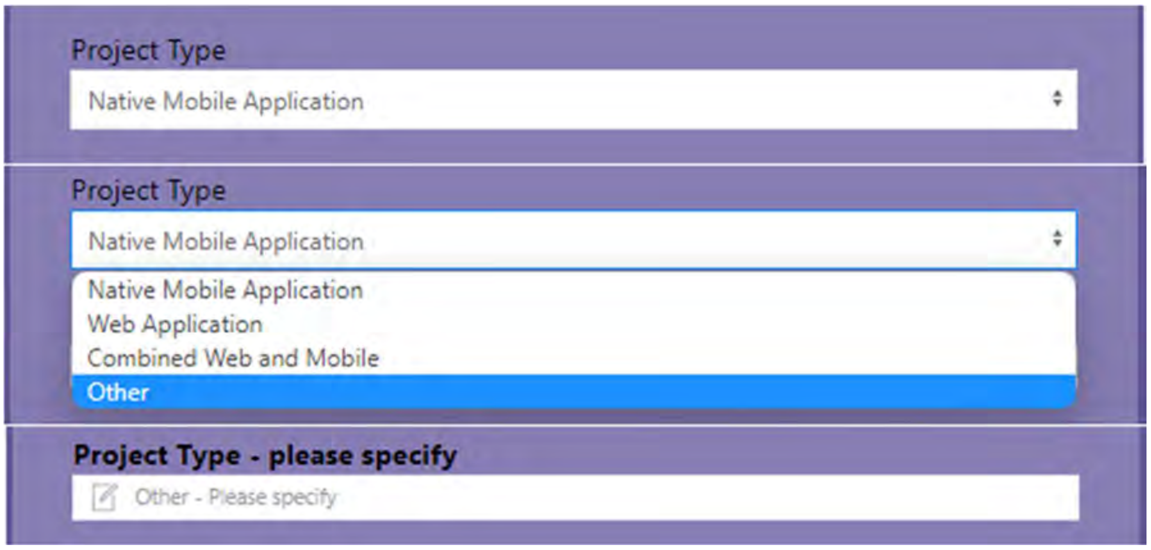
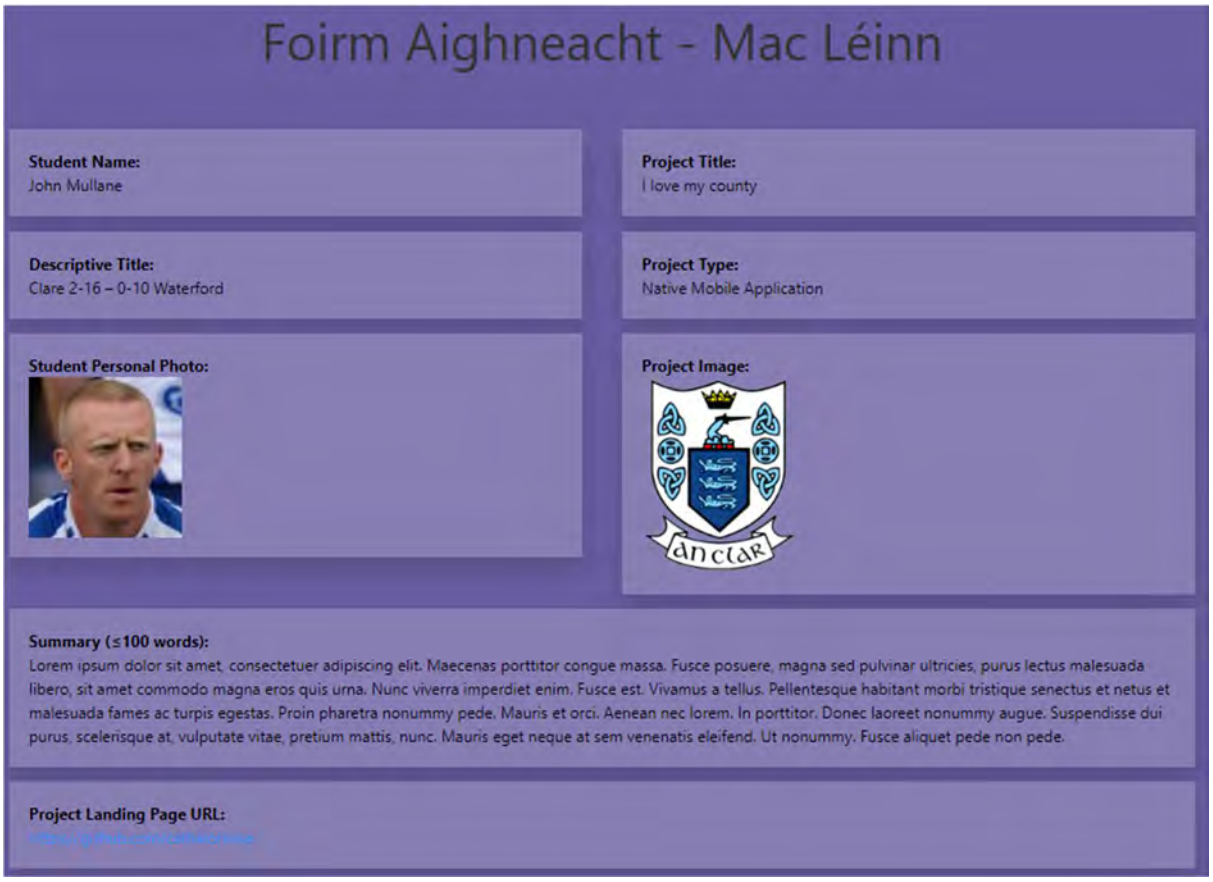

Project Type
Native Mobile Application
Project Type
Native Mobile Application
Native Mobile Application
Web Application
Combined Web and Mobile
Other
Project Type - please specify
Other - Please specify

Figure 8: Project Type Input field before, during and after "Other" option has been selected

After the student has submitted their data (whether that be every available field, or just a select few fields), they select the 'Submit' button, and are then redirected to the submission output screen:



Foirm Aighneacht - Mac Léinn

Student Name:
John Mullane

Project Title:
I love my county

Descriptive Title:
Clare 2-16 – 0-10 Waterford

Project Type:
Native Mobile Application

Student Personal Photo:

Project Image:

Summary (≤100 words):
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Maecenas porttitor congue massa. Fusce posuere, magna sed pulvinar ultricies, purus lectus malesuada libero, sit amet commodo magna eros quis urna. Nunc viverra imperdiet enim. Fusce est. Vivamus a tellus. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Proin pharetra nonummy pede. Mauris et orci. Aenean nec lorem. In porttitor. Donec laoreet nonummy augue. Suspendisse dui purus, scelerisque at, vulputate vitae, pretium mattis, nunc. Mauris eget neque at sem venenatis eleifend. Ut nonummy. Fusce aliquet pede non pede.

Project Landing Page URL:
[illegible]

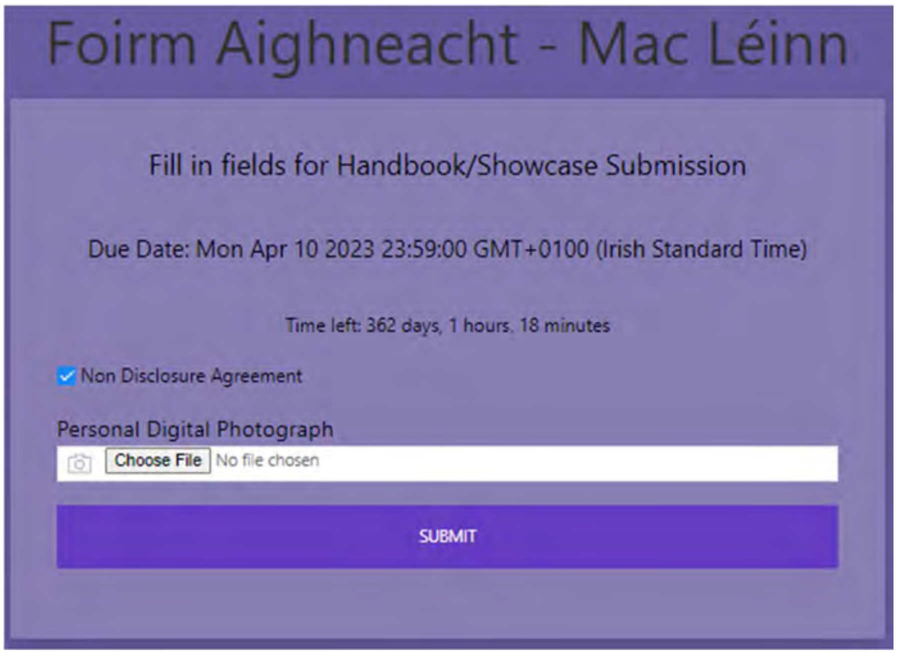
Figure 9: Submission Output View

10.2.2.5 NDA

If the student selects the “Non Disclosure Agreement” checkbox, a similar function to the “otherPleaseSpecify” function outlined above is called, that hides all input fields apart from the Personal Digital Photograph field, and the ‘nda’ attribute is declared as true. The declaration of nda as ‘true’ has multiple impacts on the generation of the PDF file, most notably that it removes the stipulation that all submission data must be entered in order for the student PDF to be included in the handbook PDF, with the only data required being the Personal Digital Photograph:

```
<label for="nda" style="color: black">
    <input class="uk-checkbox" type="checkbox" id="nda" name="nda"
value="true" onclick="disableField()"> Non Disclosure Agreement
</label>
<!--If the user checks the NDA checkbox, this function will hide all
irrelevant fields (i.e. data that
is not required to be submitted as it's covered under NDA).-->
<script type="text/javascript">
    function disableField(){
        document.getElementById("pleaseSpecify").hidden = false;
        if(document.getElementById("nda").checked === true){
            document.getElementById("projectTitle").hidden = true;
            document.getElementById("descriptiveTitle").hidden = true;
            document.getElementById("projectType").hidden = true;
            document.getElementById("pleaseSpecify").hidden = true;
            document.getElementById("projectImage").hidden = true;
            document.getElementById("summary").hidden = true;
            document.getElementById("projectUrl").hidden = true;
            document.getElementById("videoUrl").hidden = true;
        }else{
            document.getElementById("projectTitle").hidden = false;
            document.getElementById("descriptiveTitle").hidden = false;
            document.getElementById("projectType").hidden = false;
            document.getElementById("pleaseSpecify").hidden = true;
            document.getElementById("projectImage").hidden = false;
            document.getElementById("summary").hidden = false;
            document.getElementById("projectUrl").hidden = false;
            document.getElementById("videoUrl").hidden = false;
        }
    }
</script>
```

Figure 10: Function that hides all non-relevant fields, if the NDA checkbox is selected



Foirm Aighneacht - Mac Léinn

Fill in fields for Handbook/Showcase Submission

Due Date: Mon Apr 10 2023 23:59:00 GMT+0100 (Irish Standard Time)

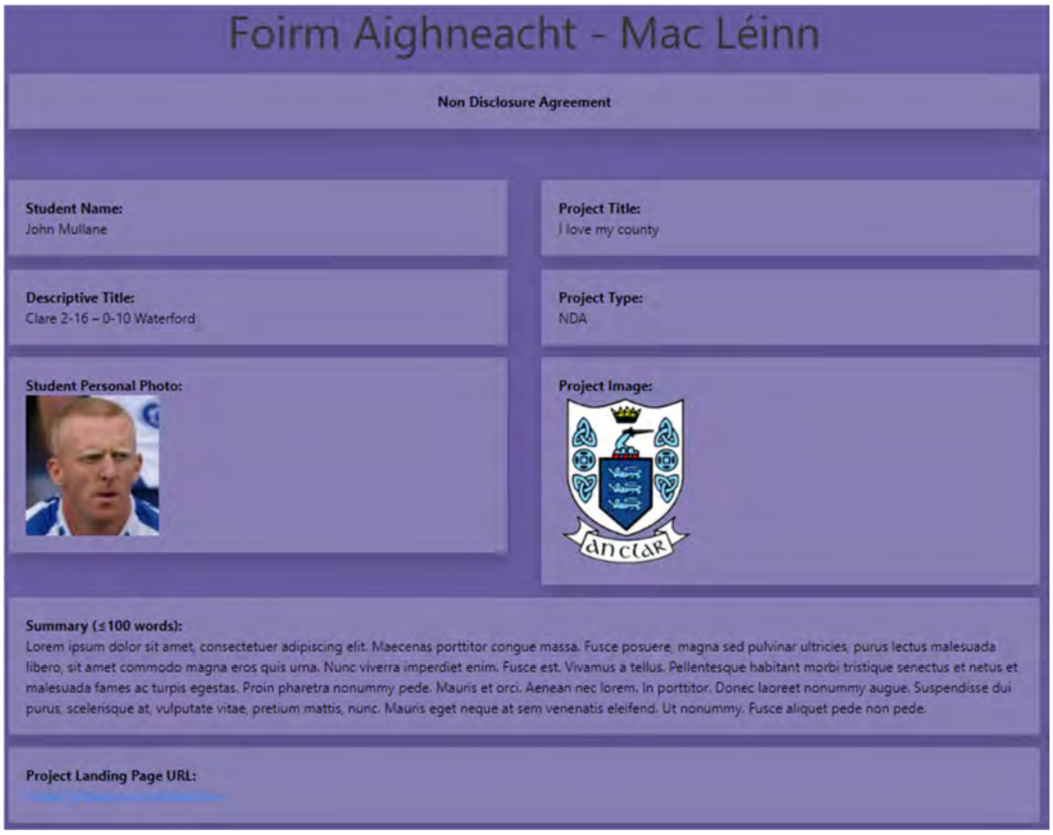
Time left: 362 days, 1 hours, 18 minutes

✓ Non Disclosure Agreement

Personal Digital Photograph

Figure 11: Student Submission Form after the NDA checkbox has been selected

As stated previously, I have intentionally not included any delete functionality (apart from cloudinary database maintenance), and therefore, the selection of the NDA checkbox only serves to redact any previously submitted data from any resulting PDF files and make it non-essential or required data for submission completion. Any previously entered data persists in the database, and also in the submission output view. This is to allow for the student to revert to not having an NDA, and not therefore having to reinput any previously entered data.



Foirm Aighneacht - Mac Léinn

Non Disclosure Agreement

Student Name:
John Mullane

Project Title:
I love my county

Descriptive Title:
Clare 2-16 – 0-10 Waterford

Project Type:
NDA

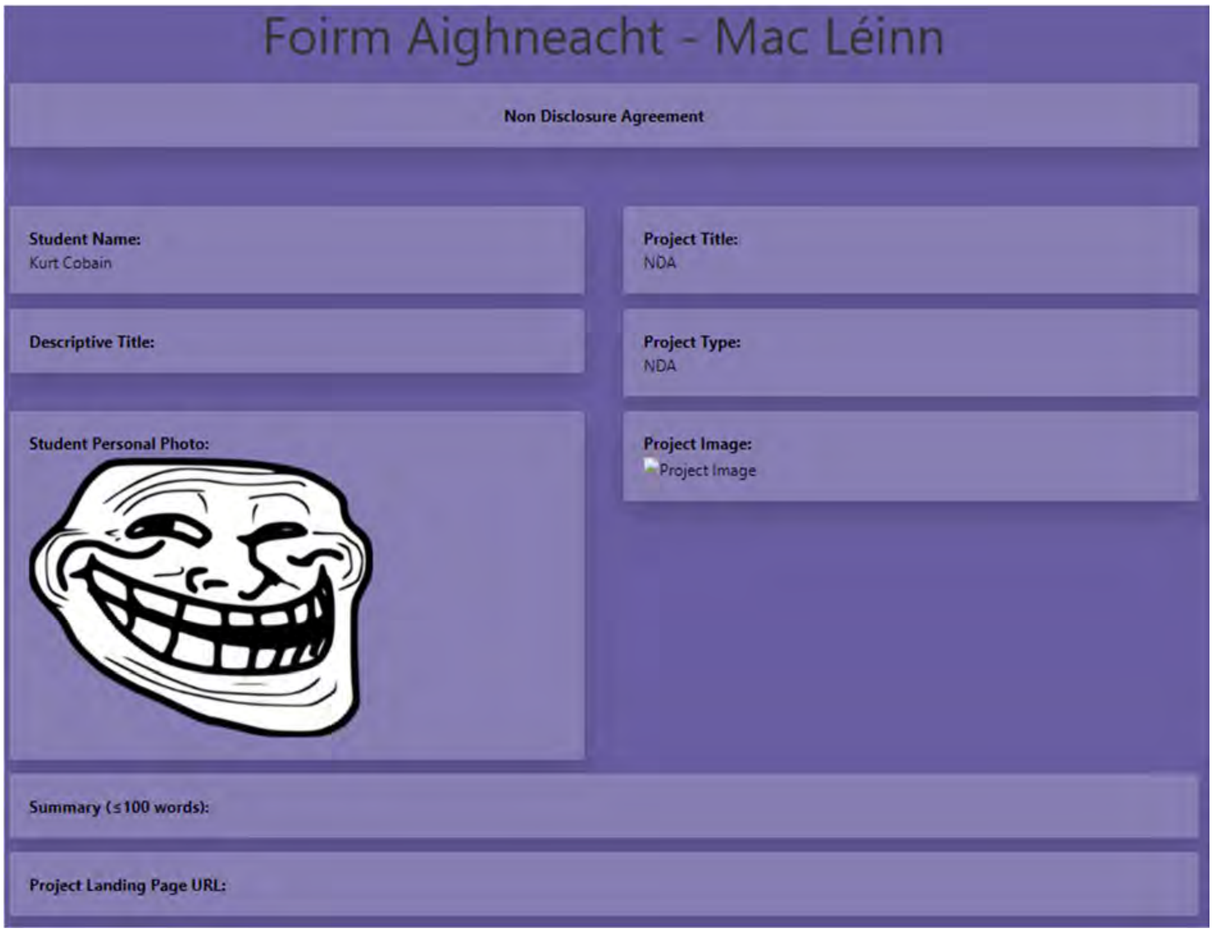
Student Personal Photo:

Project Image:

Summary (≤100 words):
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Maecenas porttitor congue massa. Fusce posuere, magna sed pulvinar ultricies, purus lectus malesuada libero, sit amet commodo magna eros quis urna. Nunc viverra imperdiet enim. Fusce est. Vivamus a tellus. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Proin pharetra nonummy pede. Mauris et orci. Aenean nec lorem. In porttitor. Donec laoreet nonummy augue. Suspendisse dui purus, scelerisque at, vulputate vitae, pretium mattis, nunc. Mauris eget neque at sem venenatis eleifend. Ut nonummy. Fusce aliquet pede non pede.

Project Landing Page URL:
[illegible]

Figure 12: Submission Output View after all data has been entered, and the NDA checkbox has been selected


Foirm Aighneacht - Mac Léinn

Non Disclosure Agreement

Student Name:
Kurt Cobain

Project Title:
NDA

Descriptive Title:

Project Type:
NDA

Student Personal Photo:

Project Image:
Project Image

Summary (≤100 words):

Project Landing Page URL:

Figure 13: Submission Output View after no non-essential data has been entered, and the NDA checkbox has been selected

10.2.3 Administrator User Experience

This is where the complexities of the application are realised.

The administrator has three roles:

- Edits and makes additions to the student submission
- Creates the 'Admin Submission' object, which consists of:
    - All data that is rendered on the admin handbook PDF file – basically the front and back pages, and all pages that group each student together
    - The submission deadline, which is optional, and if implemented it revokes access to the submission form for the student once the deadline has passed
- Creates the handbook PDF file

10.2.3.1 Prevention of 'Stale' AdminSubmissions

In keeping with the same approach above of not creating excessive amounts of stale data, only one adminSubmission is created, and all subsequent adminSubmissions are edits made to that original one. However, for the creation of adminSubmissions, a different approach was taken to that of student submissions. As outlined in above in Section 10.2.2.1, the student submission is created for each student, either when they first register, or if seeded students log in for the first time. Because I initially designed this application to only ever have one single adminSubmission (because it is used to create one handbook only, and that handbook only needs one such submission to define all the relevant data therein), I didn't want there to be a situation whereby if multiple administrators signed up, or were seeded into the application db, they would each have their own adminSubmission. I therefore thought of a different method for creating the adminSubmission. When the administrator navigates to the "ADMIN PDF FORM", an array is created, consisting of 'all' adminSubmissions in the database. If the number of elements in this array is zero (i.e., if it is the first time navigating to the form and no adminSubmission exists), then a new adminSubmission is created. Upon subsequent returns to this form, this same function is called, and because the number of submissions will no longer equal zero, the adminSubmission at position [0] – i.e., the first and only adminSubmission created – is called upon.

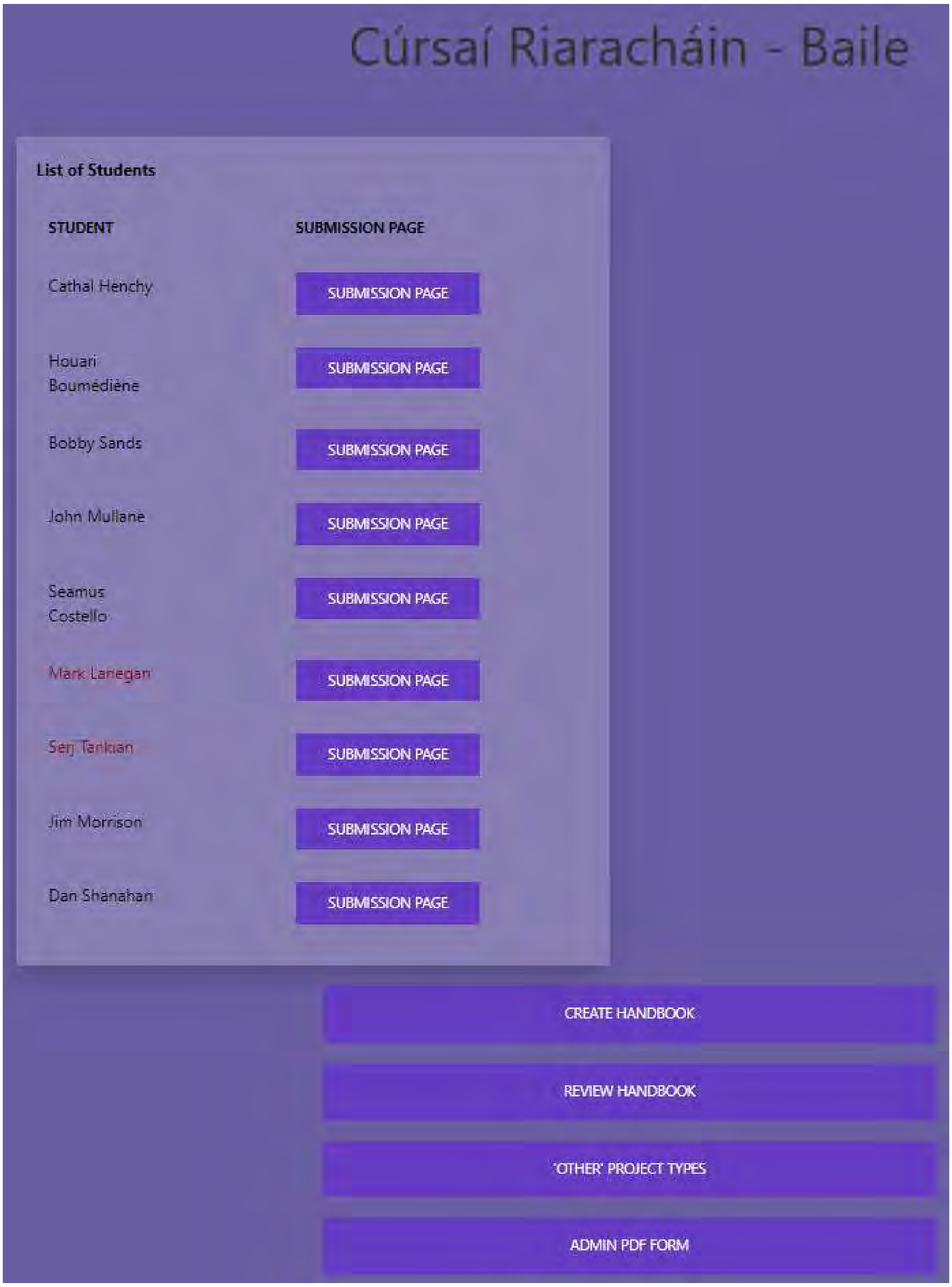
```
const adminSubmissions = await AdminSubmission.find();
var adminSubmission = null;

if (adminSubmissions.length === 0) {
  var adminSubmission = new AdminSubmission({
    submitter: admin,
  });
  console.log("Creating new Admin pdf (" + adminSubmission + ")...");
} else {
  var adminSubmission = await adminSubmissions[0];
  console.log("Editing existing Admin pdf (" + adminSubmission._id +
")...");
}
```

Figure 14: adminSubmission Creation

At first this seemed like a great approach, and it worked perfectly. However, as the development progressed, I began to think more in terms of scalability, and thought that this application should be accessible for multiple administrators working on multiple projects. I therefore created a link between the adminSubmission and the administrator, just like with the student submission and the student – I.e., I gave the adminSubmission a 'submitter' attribute, which was the administrator object. This was to be continued by binding all submissions and adminSubmissions to the same project environment, but due to project deadlines etc, this never materialised.

10.2.3.2 List of Submissions

When the administrator logs in, they are taken to the admin home screen. Here, there is a list of all students in the database. This list is actually accessing the list of submissions – not users – and what is listed is the 'submission.firstName' and 'submission.lastName' of each submission, as outlined in Section 0 above, it is eliminating the need to fetch both user id and submission id for each view and each function. Each one is displayed in either black or red text, depending on whether or not all necessary data has been inputted for inclusion in the handbook PDF. This 'necessary' data is partly dictated on whether or not they are protected under NDA (as specified in Section 0 above), and also includes the admin submission data (The Video URL and the Presentation time – this is covered below). So, if any student is highlighted in red, then either the student themselves, or the administrator needs to input further data for completion, and they will not be included in the handbook PDF, if it is generated prior to this.



Cúrsaí Riaracháin - Baile

List of Students

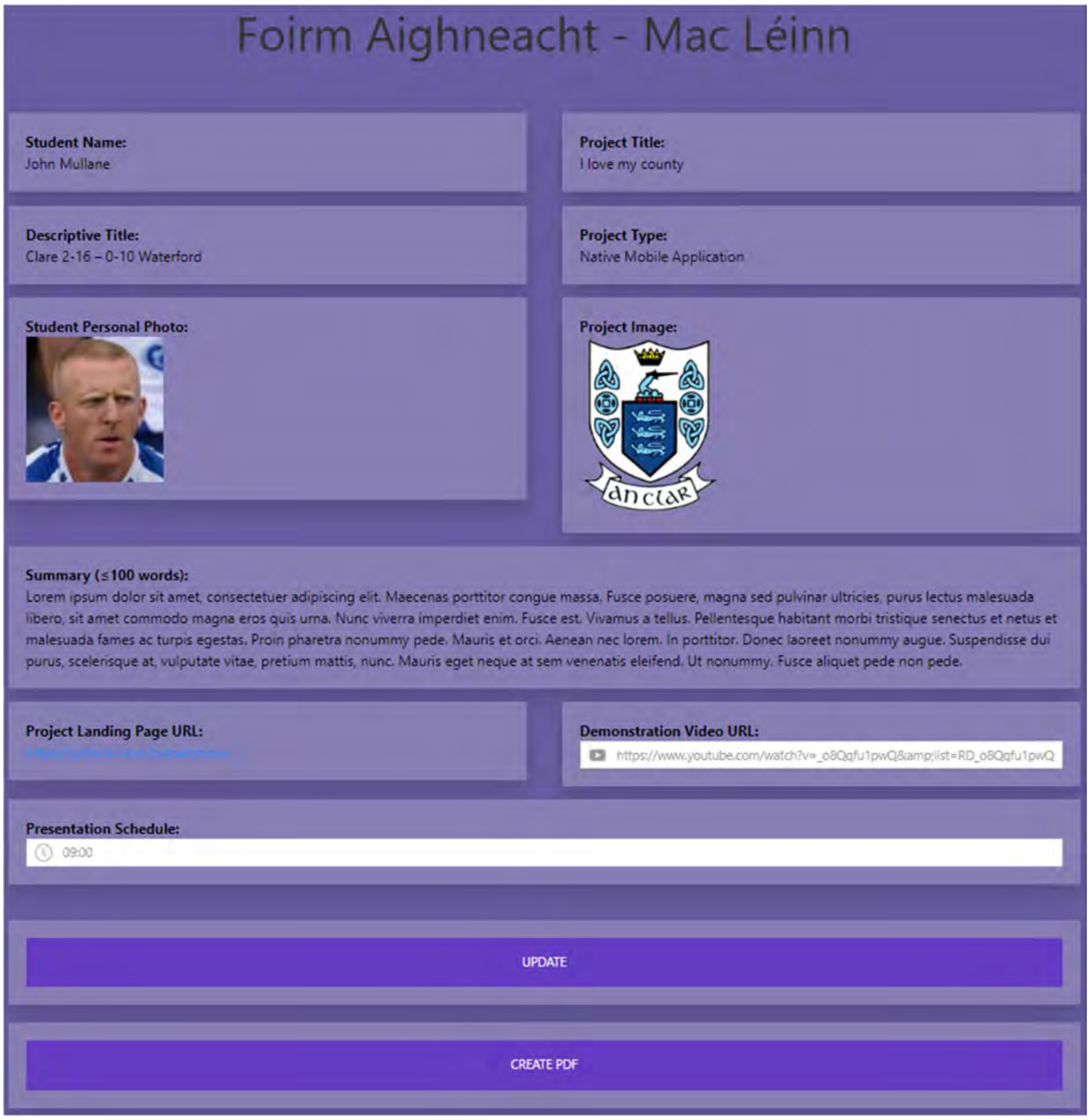
STUDENT	SUBMISSION PAGE
Cathal Henchy	SUBMISSION PAGE
Houari Boumédiène	SUBMISSION PAGE
Bobby Sands	SUBMISSION PAGE
John Mullane	SUBMISSION PAGE
Seamus Costello	SUBMISSION PAGE
Mark Lanegan	SUBMISSION PAGE
Serj Tankian	SUBMISSION PAGE
Jim Morrison	SUBMISSION PAGE
Dan Shanahan	SUBMISSION PAGE

Figure 15: Admin Home Page

10.2.3.3 Student Submission by Admin Page

If the ‘Submission Page’ for any student is selected, the user is taken to the admin submission form for that student. That closely resembles that of the submission output page from the student UX. If the student has *not* selected the NDA checkbox for their submission, then there are two additional form fields:

- Video URL
- Presentation time



Foirm Aighneacht - Mac Léinn

Student Name:
John Mullane

Project Title:
I love my county

Descriptive Title:
Clare 2-16 – 0-10 Waterford

Project Type:
Native Mobile Application

Student Personal Photo:

Project Image:

Summary (≤100 words):
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Maecenas porttitor congue massa. Fusce posuere, magna sed pulvinar ultricies, purus lectus malesuada libero, sit amet commodo magna eros quis urna. Nunc viverra imperdiet enim. Fusce est. Vivamus a tellus. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Proin pharetra nonummy pede. Mauris et orci. Aenean nec lorem. In porttitor. Donec laoreet nonummy augue. Suspendisse dui purus, scelerisque at, vulputate vitae, pretium mattis, nunc. Mauris eget neque at sem venenatis eleifend. Ut nonummy. Fusce aliquet pede non pede.

Project Landing Page URL:
[illegible]

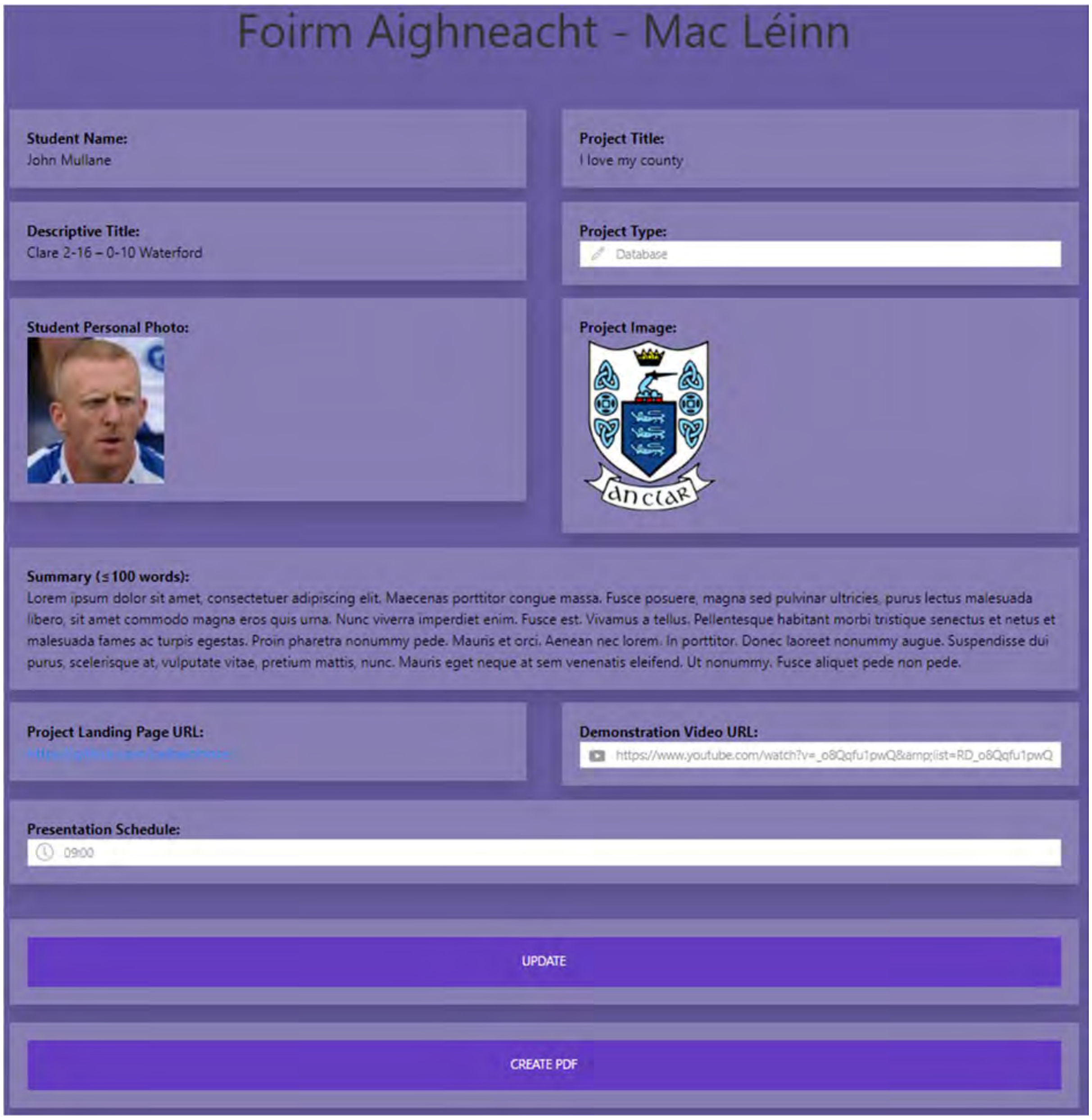
Demonstration Video URL:
https://www.youtube.com/watch?v=_o8Qqfu1pwQ&amp;list=RD_o8Qqfu1pwQ

Presentation Schedule:
09:00

Figure 16: Admin submission form for student

Both of these are text fields. The reason that the video URL is inputted here by the administrator and not upstream of this by the student is because the handbook submission data is simply required to be submitted prior to the Video URL. Therefore, the onus is on the administrator to include this on behalf of the student (provided of course that the student has actually submitted their video URL in time, via Moodle).

If the student has selected a project type of 'Other', and thus set the 'projectTypeOther' Boolean as true, then the 'Project Type' field is editable:



Foirm Aighneacht - Mac Léinn

Student Name:
John Mullane

Project Title:
I love my county

Descriptive Title:
Clare 2-16 – 0-10 Waterford

Project Type:
Database

Student Personal Photo:

Project Image:

Summary (≤100 words):
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Maecenas porttitor congue massa. Fusce posuere, magna sed pulvinar ultricies, purus lectus malesuada libero, sit amet commodo magna eros quis urna. Nunc viverra imperdiet enim. Fusce est. Vivamus a tellus. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Proin pharetra nonummy pede. Mauris et orci. Aenean nec lorem. In porttitor. Donec laoreet nonummy augue. Suspendisse dui purus, scelerisque at, vulputate vitae, pretium mattis, nunc. Mauris eget neque at sem venenatis eleifend. Ut nonummy. Fusce aliquet pede non pede.

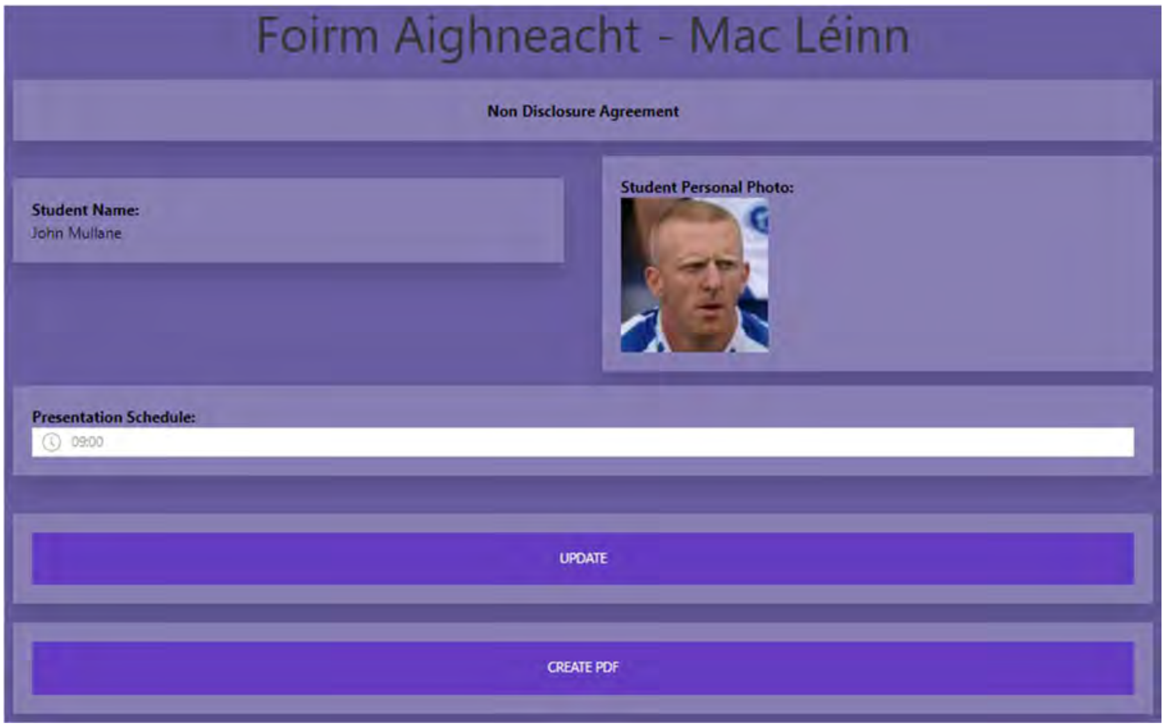
Project Landing Page URL:
[illegible]

Demonstration Video URL:
https://www.youtube.com/watch?v=_o8Qqfu1pwQ&amp;list=RD_o8Qqfu1pwQ

Presentation Schedule:
09:00

Figure 17: Admin submission form for student with Project Type field now editable

If the student *has* selected the NDA checkbox, then this view is restricted to the necessary data only (NDA disclaimer, name, personal digital photograph and presentation time), and any data that has previously been entered and visible to the student, is not visible to the administrator.



Foirm Aighneacht - Mac Léinn

Non Disclosure Agreement

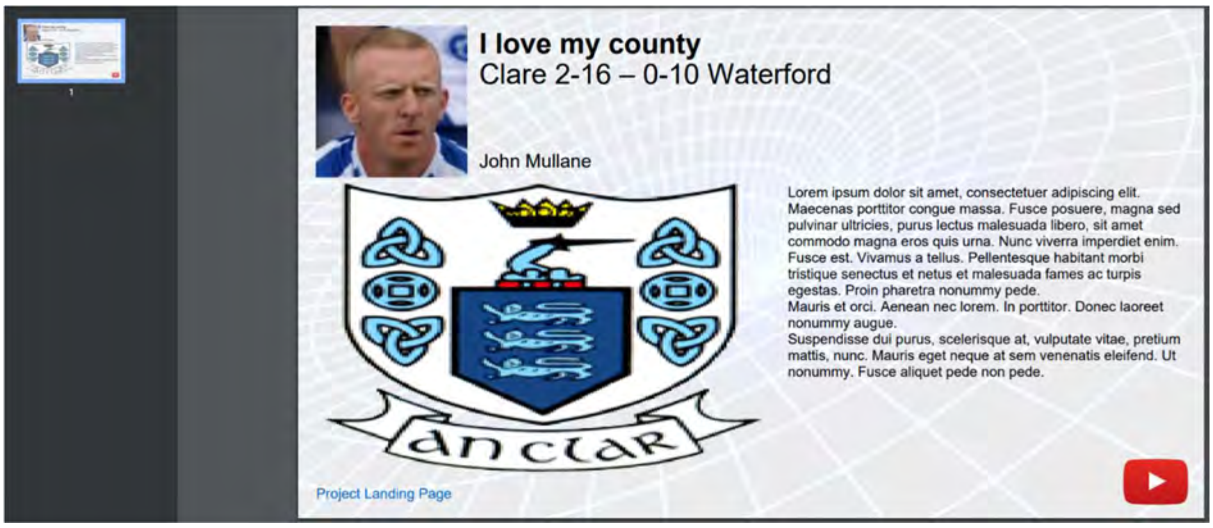
Student Name:
John Mullane

Student Personal Photo:

Presentation Schedule:
09:00

Figure 18: Admin submission form for student if their submission is protected under NDA

As just outlined, this form accepts either one or two text fields, depending on NDA status. In the case of it not being protected under NDA, and requiring both fields to be populated, it acts similarly to the student web form, in that it does not need to be fully complete prior to submission. Whenever the administrator wants, they can update the database by selecting the 'Update' button. Here, they also have the ability to review what this submission PDF file / this submission page of the handbook would look like on its own, by selecting the 'CREATE PDF' button (Note, both bottom corners contain embedded links to the project landing page and the demonstration video page):



I love my county
Clare 2-16 – 0-10 Waterford

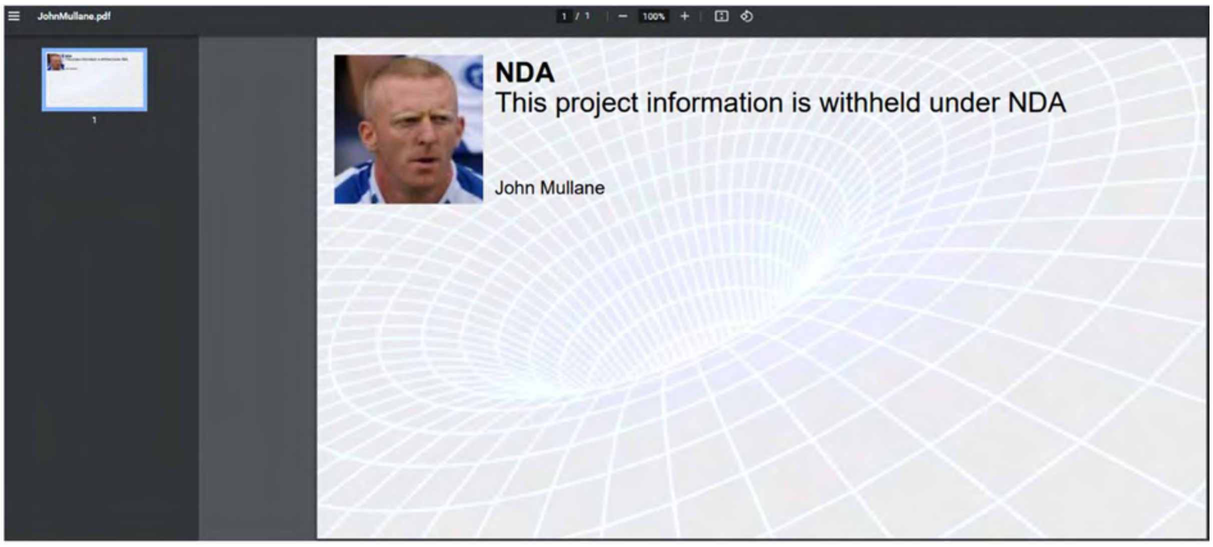
John Mullane

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Maecenas porttitor congue massa. Fusce posuere, magna sed pulvinar ultricies, purus lectus malesuada libero, sit amet commodo magna eros quis urna. Nunc viverra imperdiet enim. Fusce est. Vivamus a tellus. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Proin pharetra nonummy pede.
Mauris et orci. Aenean nec lorem. In porttitor. Donec laoreet nonummy augue.
Suspendisse dui purus, scelerisque at, vulputate vitae, pretium mattis, nunc. Mauris eget neque at sem venenatis eleifend. Ut nonummy. Fusce aliquet pede non pede.

Project Landing Page

Figure 19: Student PDF File

This file is saved in the cache, and if any updates are made to the submission data and it is created again, it simply overrides the existing file (if still in the cache):



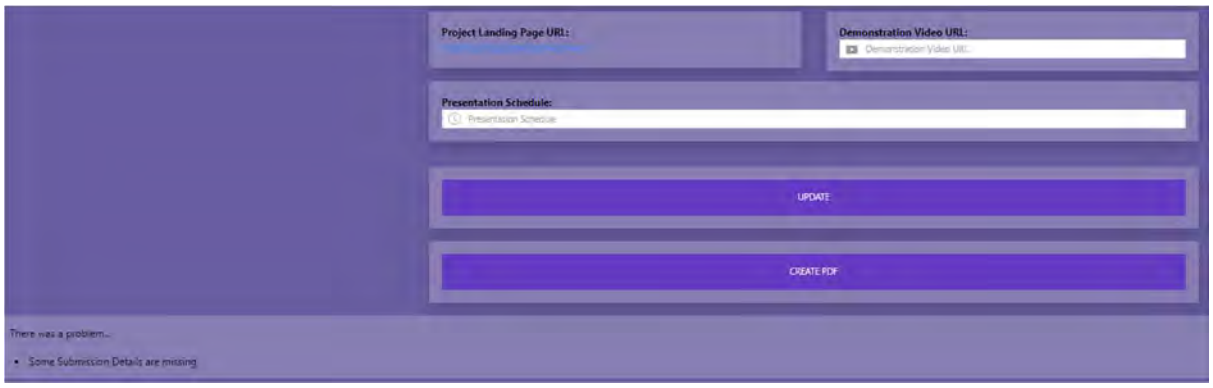
NDA
This project information is withheld under NDA

John Mullane

Figure 20: Student PDF file protected under NDA

The PDF creation is carried out using the jsPDF module (Hall, 2017).

If the project is *not* protected under NDA, and the Demonstration Video URL is not inputted, then because this is an essential part of the PDF file, the ‘CREATE PDF’ button will generate an error:



Project Landing Page URL:

Demonstration Video URL: Demonstration Video URL

Presentation Schedule: Presentation Schedule

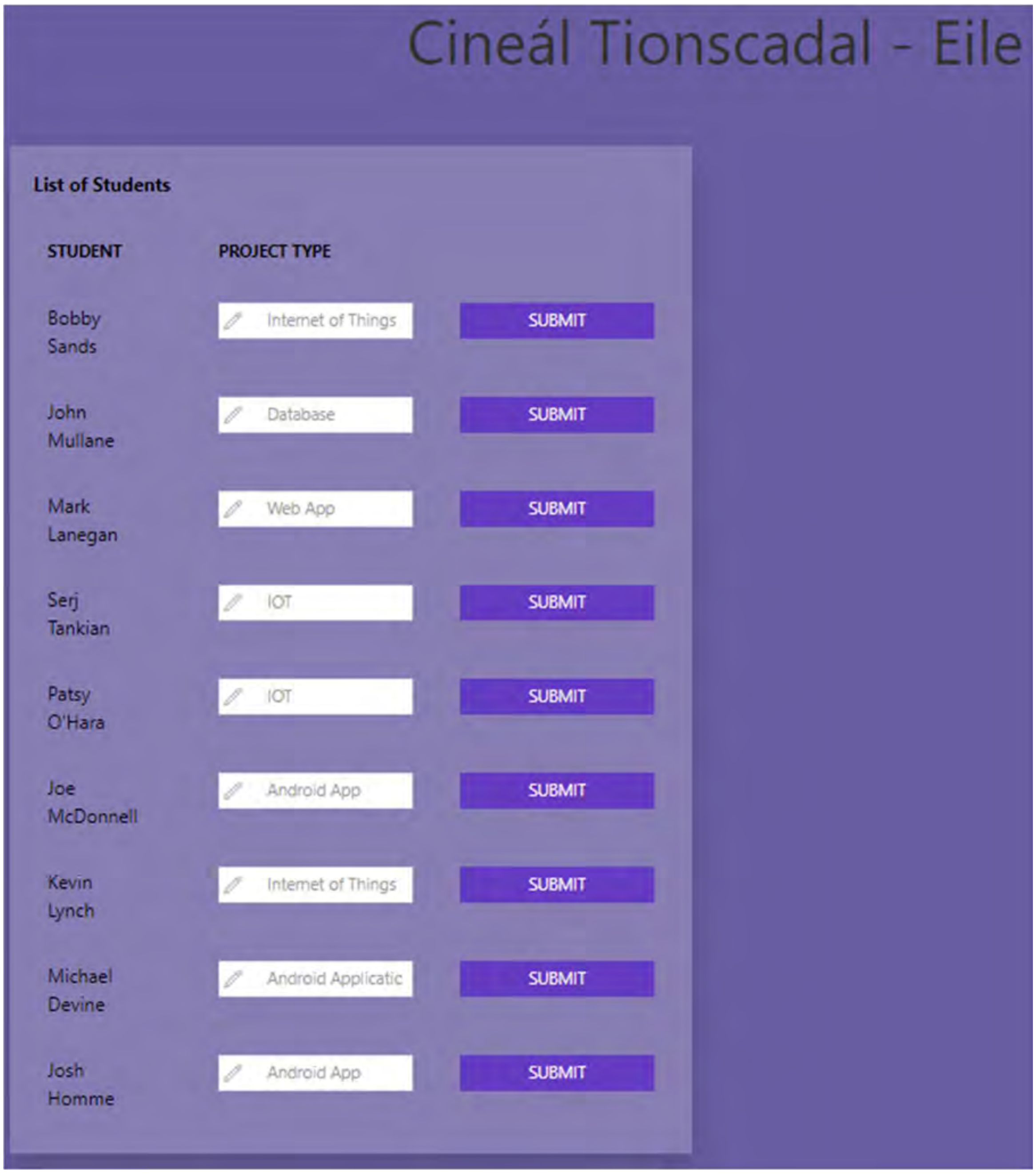
There was a problem...

- Some Submission Details are missing

Figure 21: Error message, generated due to omission of data required for PDF creation

10.2.3.4 Project Type 'Other' Form

The bottom of the admin home screen contains the additional functionality menu. If the user selects the "'OTHER' PROJECT TYPES" screen, they are taken to the Other Project Types selection screen. This is where they can edit all Project Types of submissions, whose student selected 'Other' when choosing their project type at the initial student submission stage, just like in the admin submission form for student view as outlined above. The reason for this additional screen is to easily compare all 'Other' project types in the one go, so that appropriate project types can be selected. I.e., many projects may be categorised as project types that are similar to others, so instead of having multiple categories of essentially the same type displayed on separate pages in the handbook PDF (e.g., having one page to list all 'IOT' projects, and a completely separate page to list all 'Internet of Things' projects, the user can now at a glance see what new project types are going to be included in the handbook, and make any edits or alignments they deem necessary.



Cineál Tionscadal - Eile

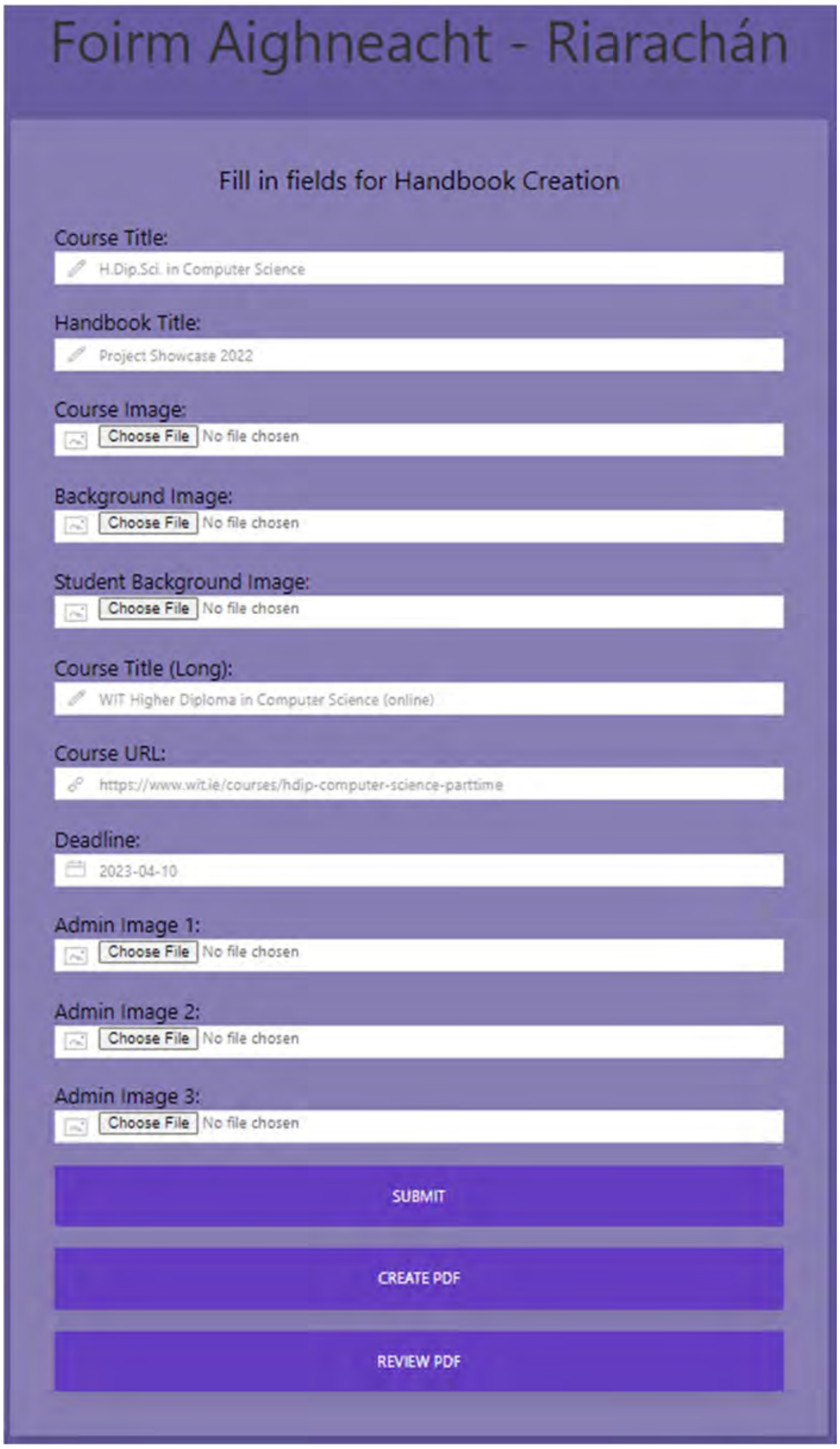
List of Students

STUDENT	PROJECT TYPE	
Bobby Sands	Internet of Things	SUBMIT
John Mullane	Database	SUBMIT
Mark Lanegan	Web App	SUBMIT
Serj Tankian	IOT	SUBMIT
Patsy O'Hara	IOT	SUBMIT
Joe McDonnell	Android App	SUBMIT
Kevin Lynch	Internet of Things	SUBMIT
Michael Devine	Android Applicatic	SUBMIT
Josh Homme	Android App	SUBMIT

Figure 22: Project Type Other Screen

10.2.3.5 Admin PDF Form

The "ADMIN PDF FORM" functions in almost the same way as the student submission for. It is where the administrator submits the project deadline (which is optional), and all data that is required for inclusion in the handbook PDF file. The administrator can also create and then review and download the PDF created from this data. This PDF consists of the first pages, and the last page of the handbook PDF:



Foirm Aighneacht - Riarachán

Fill in fields for Handbook Creation

Course Title: H.Dip.Sci. in Computer Science

Handbook Title: Project Showcase 2022

Course Image: Choose File No file chosen

Background Image: Choose File No file chosen

Student Background Image: Choose File No file chosen

Course Title (Long): WIT Higher Diploma in Computer Science (online)

Course URL: https://www.wit.ie/courses/hdip-computer-science-parttime

Deadline: 2023-04-10

Admin Image 1: Choose File No file chosen

Admin Image 2: Choose File No file chosen

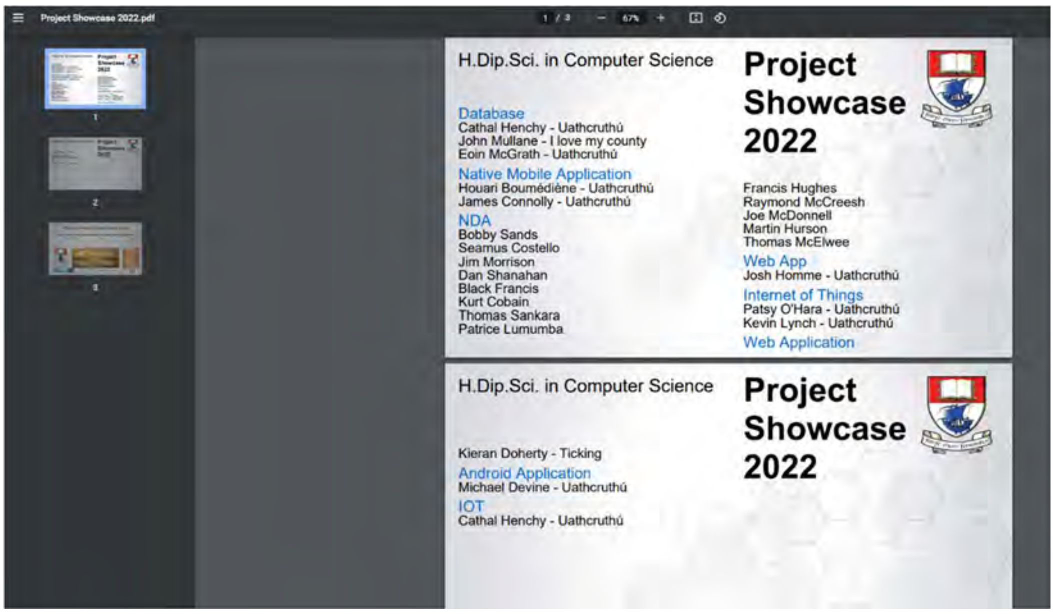
Admin Image 3: Choose File No file chosen

Figure 23: Admin Submission Form

The creation of the PDF of the administrator, is slightly more complex than that of the student. Because it's main purpose within the handbook is to provide an introduction to the presentation schedule, and to summarise what's on offer, a list of associated students is required. Since the number of students can vary, the admin PDF has to be flexible in the space that it can dedicate to cater for the student list, and the number of pages required to cover that list – so when the list would reach the end of the page, it would have to continue at the next column in that page, and once it reached the end of the page again, it would have to continue on a new page that would only be added if required. Also, each student is grouped together as per their project type, and each is listed below that project type, which for clarity uses a different font and different line spacing. In order to accommodate this complex requirement, a rather complicated nested loop with multiple if statements were required. This loop has over 100 lines of code, and thus cannot easily be explained or displayed here. By continuing with the use of relative (and not absolute) coordinates for placement of text etc. on the PDF file, the indexing variable is used as a multiplier for the placement of each new line. Developing this proved very challenging and involved a lot of trial and error in order to get the coordinates, and multiples correct. But eventually, by creating a condition whereby a very simple arithmetic formula that is dependent on the number of lines added (i.e., the indexing variable – 'countColumn1' and 'countColumn2'), returns a value that is less than the height of the page (with a buffer area at the bottom), each new line is added sequentially, until the returned value of the formula returns a value that is greater than the page limit. At this point, the indexing of lines continues on the next column within that page, and once this page limit is reached, a new page is added, and the process continues, line to line and column to column on the new page.

```
//loops through all unique projectTypes so as to find each submission that
used that type
var countColumn1 = 0;
var countColumn2 = 0;
while (i < projectTypesUnique.length) {
  let j = 0;
  doc.setFontSize(pageDimensions[0] / 13);
  if (pageDimensions[0] / 7 + countColumn1 * 8 > endOfPage) {
    countColumn2 += 1.5;
    doc.setTextColor(0, 102, 204);
    doc.text(projectTypesUnique[i], pageDimensions[0] / 1.9,
pageDimensions[0] / 4 + countColumn2 * 8);
  } else {
    doc.setTextColor(0, 102, 204);
    doc.text(projectTypesUnique[i], pageDimensions[0] / 40,
pageDimensions[0] / 7 + countColumn1 * 8);
  }
  while (j < submissions.length) {
    if (
      !submissions[j].submissionIncomplete &&
      submissions[j].projectType &&
      submissions[j].projectType === projectTypesUnique[i]
    ) {
      doc.setFontSize(pageDimensions[0] / 15);

      //If list reaches the end of the page, in order to prevent spill
over, it is moved to the next column in the page
      if (pageDimensions[0] / 7 + countColumn1 * 8 > endOfPage) {
        countColumn2++;
      ...
        }
    ...
      }
  ...
    }
```

Figure 24: Extract of loop that creates user submission list in Admin PDF file



H.Dip.Sci. in Computer Science

Project Showcase 2022

Database
Cathal Henchy - Uathcruthú
John Mullane - I love my county
Eoin McGrath - Uathcruthú
Native Mobile Application
Houari Boumédiène - Uathcruthú
James Connolly - Uathcruthú
NDA
Bobby Sands
Seamus Costello
Jim Morrison
Dan Shanahan
Black Francis
Kurt Cobain
Thomas Sankara
Patrice Lumumba

Francis Hughes
Raymond McCreesh
Joe McDonnell
Martin Hurson
Thomas McElwee
Web App
Josh Homme - Uathcruthú
Internet of Things
Patsy O'Hara - Uathcruthú
Kevin Lynch - Uathcruthú
Web Application

H.Dip.Sci. in Computer Science

Project Showcase 2022

Kieran Doherty - Ticking
Android Application
Michael Devine - Uathcruthú
IOT
Cathal Henchy - Uathcruthú

Figure 25: Admin PDF File

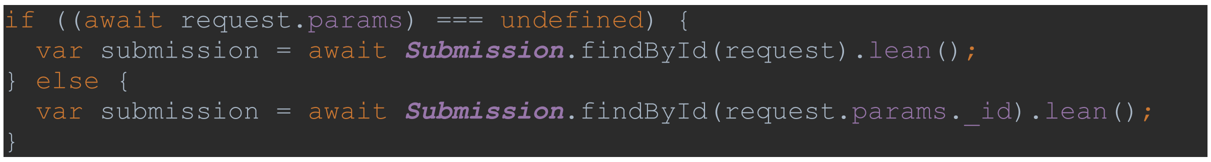
10.2.3.6 Creation of Handbook

The ‘CREATE HANDBOOK’ function is the main feature around which this whole application is based. Its function is two-fold:

- To create introduction/summary pages for each section of the handbook (i.e., the handbook is sectioned into different project types specific groupings)
- To merge all of these PDF files, and the student submission and admin submission PDF files together (using the pdf-merger-js module (nbesli, 2022))

Initially, this was functional only after all external PDF files (student submission and admin submission) had been created. These files would first need to be created, and then this function would fetch them from the cache. Obviously, this was not very feasible for use in a long-term working release, and what was needed was to instead carry out all file creation functions in one go, through this function. So this was then refactored, with the other PDF creation functions written into this function. Although in theory this worked fine, it did make for an extremely long, verbose function, several hundred lines in length. It also went very much against the DRY principle, as these other PDF creation functions were still retained in their original locations (it seemed like an important feature of the application to be able to review all submissions in PDF format on demand, without having to create the entire handbook in order to see how well they rendered). At this stage of the development process, the coding was getting very confusing, with PDF creation functions all over the place, in multiple files. I decided to therefore create a new “pdfs.js” file, which I placed inside the “utils” folder. This file contained *all* pdf creation functions, which were then deleted from all controllers, and replaced with equivalent functions calls.

Making these functions accessible as part of the handbook PDF creation function *and* direct calls for isolated PDF creation required a bit of refactoring. One such change involved the parameter data format. Calling these files directly, involved passing a http request from a <form> html element, with the stipulated URL parameter (_id) being passed into the function as an object. However, when calling these functions from the handbook PDF creation function (i.e., calling them indirectly), the parameter being passed (_id) was passed as a string. Because the image call to Cloudinary needed a URL object file path, the string parameter could not be passed, so this needed to be converted into an object:



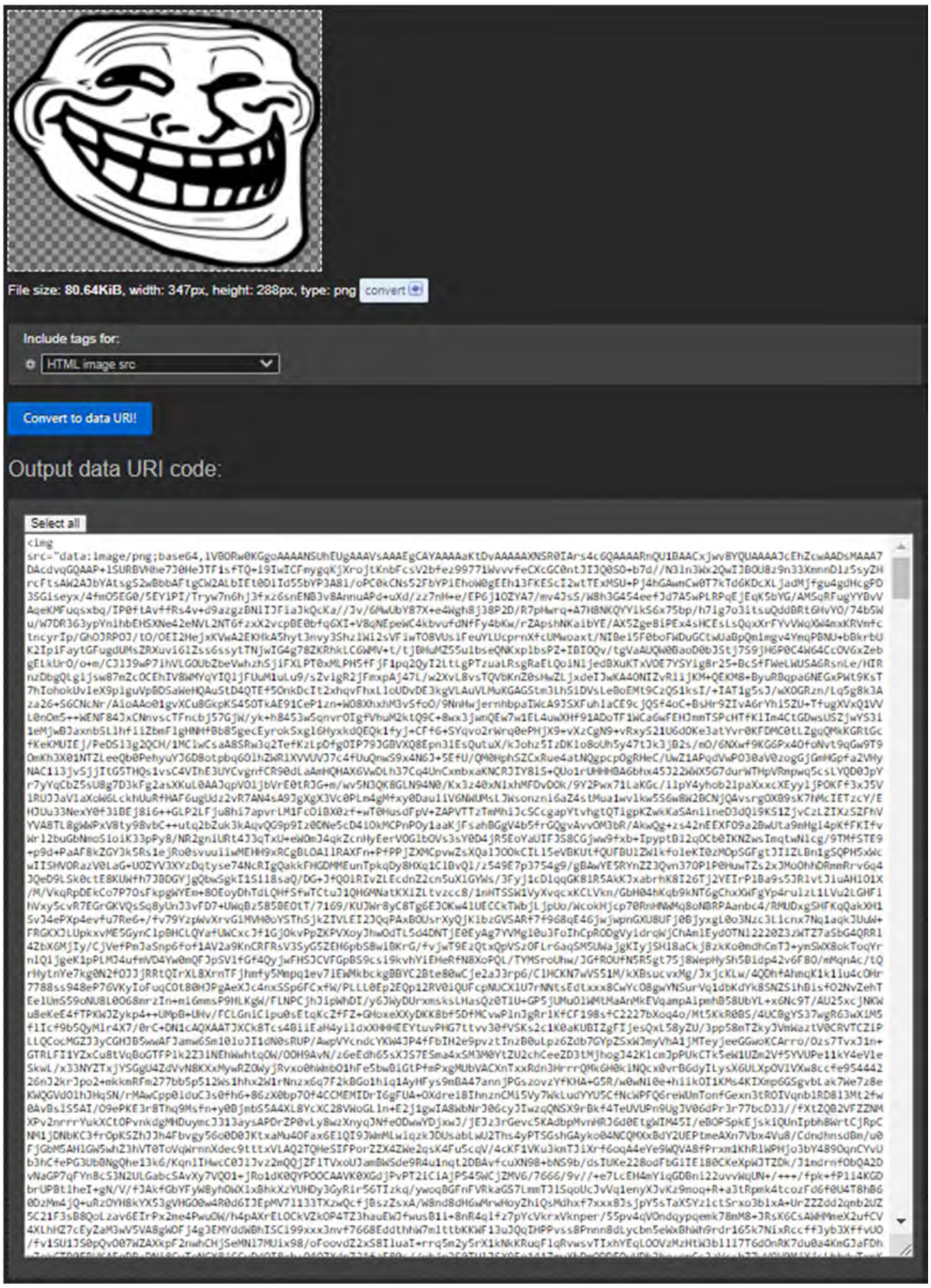
```
if ((await request.params) === undefined) {
  var submission = await Submission.findById(request).lean();
} else {
  var submission = await Submission.findById(request.params._id).lean();
}
```

Figure 26: Changing parameter format to maintain consistency, and compatibility with all functions

10.3 Problems Encountered/Bugs Remaining

10.3.1 Image Manipulation

The jsPDF addImage() function accepts the image data URI in base64 format, Image-HTML Element or Canvas-HTML Element. I spent two whole weeks searching for a solution to this. Basically, I wanted to convert the existing Cloudinary URL, into a useable image data URI format, as the raw format is completely unusable:



File size: 80.64KiB, width: 347px, height: 288px, type: png

```
<img
src="data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAVsAAAEgCAYAAAAaKtDvAAAAAXNSR0IArs4c6QAAAARnQU1BAACxjwv8YQUAAAAJcEhZcwAADsMAAA7
DAcdvqGQAAP+1SURBVHhe7J0HeJTF1sfTQ+19IwICFmygqKjXrojtKnbFcsV2bfez99771Wvvvfe CXcGC0ntJIJQ0S0+b7d//N3ln3Wx2QwIJBOU8z9n33XmnnD1z5syZH
rcFtsAW2AJbYAtsgS2wBbbAFtgCW2ALbIEt0D1Id55bYP3A8i/oPC0kCNs52FbYPiEhoW0gEEh13FKEScI2wtTExMSU+Pj4hGAwmCw0T7kTd6KDcXLjadMjfgu4gdHcgFD
3SGiseyx/4fmO5EG0/5EY1PI/Tryw7n6hj3fxz6snENB3v8AnnuAPd+uXd/zz7nH+e/EP6j1OZYA7/mv4JsS/W8h3G454eefJd7A5wPLRPqEjEqK5bYG/AM5qRFugYYBvV
AqeKMFuqsxbq/IP0ftAvffRs4v+d9azgzBN1IJFiaJkQcKa//Jv/6MwUbY87X+e4Wgh8j38P2D/R7pHwrq+A7H8NKQYY1kS6x75bp/h71g7o31tsuQddBRtGHvYO/74b5W
u/W7DR363ypYn1hbEHSXNe42eNVL2NT6fzxX2vcpBE0bfqGXI+V8qNEpeWC4kbvufdNfFy4bKw/rZApshNKaibYE/AX5ZgeBiPEx4sHCEsLsQqxXrFYVVWqXW4mxKRVmfc
tncyrIp/GhOJRPOJ/tO/OEI2HejxKVwA2EKHkA5hyt3nvy3Shz1W12sVFiwTO8VUsiFeuYLUcprnXfcUMwoaxt/NIBei5F0boFWDuGCtwUaBpQm1mgv4YmqPBNU+bBkrbU
K2IpiFaytGFugdUMsZRXuvi61Zss6ssytTNjwIG4g78ZKRhkLCGWMV+t/tjBHuMZ55ulbseQNKxplbsPZ+IBIOQv/tgVaAUQW0BaoD0bJStj7S9jH6P0C4W64CcOV6xZeb
gELkUrO/o+m/CJ1J9wP7ihVLGOUbZbeVwhzhSj1FXLPT0xMLPH5fFjF1pq2QyI2LtLgPTzuaLRsgRaELQoiN1jedBXuKTxVOE7YSYig8r25+BcSfFWeLWUSA6RsnLe/HIR
nzDbgQLg1jsw87mZcOCEhIV8WMYqYIQ1jFUuM1uLu9/sZvigR2jFmxpAj47L/w2XvL8vsTQVbKnZ0sHwZLjxdeIJwKA4ONIZvRl1jKM+QEKM8+ByuRBqpa6NEGxPWt9KsT
7hIohokUv1eX9pIguVpBDSaWeHQAuStD4QTE+5OnkDcIt2xhqvFhxL1oUDvDE3kgVLAuVLMuKGAGStm3Lh5iDVsLeBoEMt9CzQS1ksI/+IAT1g5sJ/wXOGRzn/Lq5g8k3A
za26+S6CNcNr/A1oAAo01gvXCuBGkpKS45OTkAE91CeP1zn+WO8XhxhM3vSfoO/9NnHwjernhbpaIWcA9JSXFuhlaCE9cjQSf4oC+BsHr9ZIvA6rYhi5ZU+TfugXVxQ1VV
L0nOm5++WENF84JxCNnvscTFncbj57GjW/yk+h8453w5qnvrOIgfVhuM2ktQ9C+8wx3jwnQEw7w1EL4uwXHf91ADoTF1WCaGwFEHJmnTSPcHTfK1Im4CtGDwsUSZjwYS31
1eMjwBJaxnbSL1hfi1ZbmFlgHNHfBb8SgecEyrokSxg16HyxkdQEQk1fyj+CFf6+SYqvo2rWrq0ePHjX9+vXzCgN9+vRxyS21U6dOKe3atYvr0KFDMC0LZgqQMkKGRtGc
fKeKMUIEj/PeDS13g2QCH/1MC1wCsaA8SRw3q2TefKzLpOfg0IP79JGBVXQ8Epn31EsQutuX/kJohz5IzDK1o8oUh5y47tJk3jB2s/mO/6NXwf9KG6Px40foNvt9qGw9T9
OmKh3X01NTZLeeQb0PehyuYJ6D8otpbq6O1hZWR1XVVUVJ7c4fUuQnwS9x4N6J+5EfU/QM0HphSZCxRue4atNQgpcpOgRHeC/UwZ1APqdVwPO30aV0zogGjGmHGpfa2VHy
NAC1i3jvSjjItG5THQs1vsC4VIhE3UYCvgnfCR90dLaAmHQHAX6VwDLh37Cq4UnCxmbxaKNCRJIY81S+QUo1rUHHHBA6bhx45J22WWX5G7durWTHpVRmpwq5csLYQD0JpY
r7yYqCbZ5sU8g7D3kFg2asXKuL0AAJqpVO1jbVrE0tRJG+m/wv5N3QKBGLN94N0/Kx3z40xN1xhMFDvDOk/9Y2Pwx71LaKGc/i1pY4yhob21paXxxcXEyy1jPOKFf3xJ5V
1RUJJaV1aXoW6LckhUuRfHAF6ugUdz2vR7AN4sA9JgXgX3Vc0PLm4gMfxy0Dau1iV6NWUMsLJWsonzni6aZ4stMua1wv1kw5S6w8W2BCNjQAvsrgOXB9sK7hMcIETzcY/E
HJUu33NexY0fJ1BEj8i6++GLP2LFju8hi7apvrLM1FcO1BX0zf+wT0HusdFpV+ZAPVTTzTmMhiJcSCcgapYtvhgtQTigpKZwkKaSAniineD3dQi9KS1ZjvCzLZIXzSZFhV
YVA8TL8gWWPxV8ty98vbC++utq2bZuk3kAqvQG9p9Iz0DNe5cD41OkMCPnPOy1aaKjFsahBGgV4b5frGQgvAvvOM3bR/AkwQg+zs42nEEXFO9a2BwUta9mHg14pKfFKIfv
Wr12buGbNmoSioiK33pPy8/NR2gniURt4J3qTxU+eWOmJ4qkZcnHyEerVOG1bOVs3sY0D4jR5EoYaUIFJS8CGjww9fxb+IpyptB12qOCb0IKNZwsImqtwN1cg/9TMfSTE9
+p9d+PaAF8kZGY3k5Rs1ejRo0svuuiiwMEHH9xRCgBLOA1lRAXFn+PfPPjZXMCpvwZsXQa1JOOkCIL15eVBKUtfQUFBU1ZW1kfoleKI0zMOpSGFgtJI1ZLBn1gSQPH5xWc
wIISHVORazV0LaG+UOZYV3XYzDqtyse74NcRIgQakkFHGDMMEunTpkqDy8HXq1C1BvQ1/z549E7p3754g9/gBAwYE5RYnZZ3Qvn37OP1P0HuwTZs2xJMoOhhDRmmRrv6q4
JQeD9LSk0ctE8KUWfh7JBDGYjgQbwSgkI1S118saQ/DG+JfQ0iRIvZLEcdnZ2cn5uX1GYWs/3FyJ1cD1qqGK81R5AkKJxabrhK8I26Tj2YEIrP1Ba9s5JR1vtJiuAHIO1X
/M/VkqRpDEkCo7P7OsFkpgWYEm+8OEoyDhTdLQHfSfwTCtuJ1QHGMNatKXiZLtvzcc8/1nHTSSW1VyXvqcxKCLVkn/GbH04hKqb9kNT6gChxXWFgYp4rulzL1LVu2LGHFi
hVxy5cvR7EGrGKVQsSq8yUnJ3vFD7+UWqBz585BEOtT/7i69/KUJWr8yC8Tg6EJOKw41UECCkTWbjLjpUo/WcokHjcp70RnHNWMq8oNBRPAanbc4/RMUDxgSHFKqQakXH1
SvJ4ePXp4evfu7Re6+/fv79YzpWvXrvG1MVH0oYSThSjkZIVLEI2JQqPAxBOUsrXyQjKibzGVSARf7f968qE46jwjwpnGXU8UFj0BjyxgL0o3Nzc3L1cnx7Nq1aqkJUuW+
FRGKXJLUpkxvME5GynClpBHCLQYafUWCxcJf1GjOkvPpZKPVXoyJhwOdTL5d4DNTjE0EyAg7YVMg10u3FoIhCpRODgVyidrqWjChAm1EydOTN1222023zWTZ7aSbG4QRR1
4ZbXGMjIy/CjVefPmJaSnp6fof1AV2a9KnCRFRsV3SyG5ZEH6pbS8wiBKrG/fvjwT9EzQtxQpVSzOFLr6aqSM5UWajgKIyjSH18aCkjBzkKo0mdhCmTJ+ynSWXBokToqYr
n1QijgeK1pPLMJ4ufmVD4Yw0mQFJpSV1fGf4QyjwFHSJCVFGpBS9csi9kvhYiEHeRfN8XoPQL/TYMSroUhw/JGfROUfN5R5gt75j8WepHySh5B1dp42v6F8O/mMqnAc/tQ
rHytnYe7kg0N2fOJJjRRtQIrXL8XrnTFjhmfy5Mmpq1ev7iEWMkbckgBBYC2Bte80wCje2aJ3rp6/C1HCKN7wVS51M/kXBsucvxMg/JxjcKLw/4QDhfAhmqK1k1iu4cOHr
7788ss948eP76VKyIoFuqCOt80HJPgAeXJc4nxSSp6FCxfW/PLLL0Ep2EQp12RV0iQUFcpNUCX1U7rNNtsEdtxxx8CwYcO8gwYNSurVq1dbKdYkBSNZSihBisfO2NvZehT
Ee1UmS59oNU8L0O68mrzIn+mi6mmsP9HLKgW/FLNPCjhJ1pWhDI/y6JWyDUrxmsksLHasQz0TIU+GP5jUMuO1WMtMaAnMkEVqampAipmhB58UbYL+x6Nc9T/AU25xcjNKW
uBeKeE4fTPKWJZykp4++UMpB+UHv/FCLGniCipu0sEtqKcZfFZ+GHoxeXXyDKK8bf5DfMCvwP1nJgRr1KfCF198sfC2227bXoq4o/Mt5KkR0BS/4UCBgYS37wgR63wX1M5
f1Icf9b5QyMlr4X7/0rC+DN1cAQXAATJXCk8Tcs4BiiEaH4yildxXHHHEEYtuvPHG7ttvv30fVSKs2c1K0aKUBIZgFIjesQxL58yZU/3pp58mT2kyJVmWaztV0CRVTCZiP
LLQCocMGZJ3yCGHJB5wwAFJamw6Sm101oJI1dN0sRUP/AwpVYcndcYKW4JP4fFbIH2e9pvztInzB0uLpz6Zdb7GYpZSxWJmyVhA1jMTeyjeeGGwoKCArro/Ozs7TvxJ1n+
GTRLFI1YZxCu8tVqBoGTFP1k2Z31NEhWwhtqOW/OOH9AvN/z6eEdh65sXJS7ESma4xSM3M0YtZU2chCeeZD3tMjhogJ42K1cmJpPUkCTk5eW1UZm2Vf5YVUPe11kY4eV1e
SkwL/x33NYZTxjYSGgU4ZdVvN8KXxMywRZOWyjRvxo0hWmbO1hFe5bwBiGtPfmPxgMUbVACXnTxxRdn3HrrrQMk6H0kiNQcx0vrB6dyILysX6ULXpOV1VXw8ccfe954442
26nJ2krJpo2+mkkmRFm277bb5p512Ws1hhx2W1rNnzx6q7F2kBGo1h1q1AyHFys9mBA47annJPGszovzYfKHA+G5R/w0wNi0e+hiikOI1KMs4KIXmp6GSgvbLak7We7z8e
KWQGVdO1hJHqSN/rMAwCpp0iduCJs0fh6+86zX0bp70f4CCMEMIDrI6gFUA+OXdrei8IhnznCMi5Vy7WkLudYYU5CfNcWPFQ6reWUmTonfGexn3tROIVqnb1RD813Mt2fw
0AvBs1S5AI/O9ePKE3r8Thq9Msfn+y0BjmbS5A4XL8YcXC28VWoGL1n+E2j1gwIA8WbNrJ06cyJIwzqQNSX9rBkf4TeUVUPn9UgJV06dPr3r77bcD33//fXtZQB2VFZZNM
XPv2nrrrYukXCtOPvnkdgMHDuymcJJ1JaysAPDrZP0vLy8wzXnyqJNfeODwwYDjxwJ/jEJz3rGevc5KAdbpMvnHRJ6d0EtgWIM45I/eBOPSpkEjskiQUnIpbh8WrtCjRpC
NM1jDNbKC3frOpKSZhJJh4Fbvgy56o0D0JKtxaMu40Fax6E1QI9JWmMLwiqzkJDUsabLwU2Ths4yPTSGshGAyko04NCQMXxBdY2UEPtmeAXn7Vbx4Vu8/Cdndhnsd8m/u0
FjGbM5AH1GW5whZ3hVT0ToVqWrnnXdec9tttxVLAQ2TQHeSIFPorZZX4ZWe2qsK4Fu5cqV/4cKF1VKu3kmTJiXrf6oqA4eYe9WQVA8fPrxm1KhR1WPHjo3bY489OqnCYvU
b3hCfePG3Ub8NgQhe13k6/Kqn1IHwcC0J1Jvz2mQQjZF1TVxoUJamBWSde9R4u1nqt2DBAvfcuXN98+bNS9b/dsIUKe228odFbGiIE180CKeXpWJTZDk/J1mdrnf0bQA2D
vNaGP7qFYn8cS3N2ULGabcSAvXy7VQO1+jRo1dK0QYPOOCAAVK0XGdjPvPT2iC1AjP545WCjZMV6/7666/9v//+e7LcEH4mYiqGDBni22uvvWqUN+/+++/fpk+fP114KGD
brUPBt1heI+gN/V/fJAkfGbYFyWByhOWX1xBhkXzYUHDy3GyRir56TIzkq/ywoqBGFnFVRkaGS7LmmTJ1SqoUcJvVq1enyXJvKz9noq+R+a3tRpmk4tcozFd6f0U4T8hB6
0DzMm4jQ+uRzOYH8kYX53gVHGO0w4R0d6IJEpMV71133TXzwQcfjBszZsxA/W8nd8dH6wMrwHoyZhiQsMdhxf7xxx8JsjpY5sTaX5Yz1ctSrxo3b1xA+UrZZZdd2qmb2UZ
5C21F3sB8QoLzav6EIrPx2me4PwuOW/h4pAXrELOCkVZkOP4TZ3hauEWJfwusB1i+8nR4q1fz7pYcVkrxVknper/55pv4qVOndqypqemk78mM8+JRsK6CsAWHMmeX2ufCV
4XLhHZ7cEyZaM3wV5VA8gWOFj4g3EMYddWBhISCi99xxx3nvf7668EddthhW7m1ttbKKWF13uJQqIHPPvss8PnnnBdLycbn5eWxBhWh9rdr165k7NixRccffJyb3XffvUO
/fv1SU1JS0pQvO07WZAXkpF2nwhCHjSeMN17MUix98/oFoovdZ2xS8IluaI+rrq5m2y5rX1kNkKRuqF1qRvwsvTIxhYEqLOOVzMzHtW3b1117T6dOnRK7du0a4KmGJaFDh
```

Figure 27: Snippet of Image Data URI for relatively small image (generated by EZGIF.COM (Anon., 2017))

Initially, all I could do was convert the image to the Data URI, which could then be passed as a parameter into the addImage() function. This wholly cumbersome method almost crashed the system on a few attempts and was not in any feasible in the long-term. My aim at this stage was to call some method in the application that would encode the image to the data URL, and store that in a variable that could then be passed as a parameter. Amazingly, finding such a method proved extremely difficult. Predictably, Stack Overflow didn't provide much help, with over-zealous developers providing chaotically complex solutions to what I interpretated as being a straightforward problem, which required a simple solution, not scores of lines of code and external dependencies. I eventually found a module (image-data-uri) that could do what I wanted in one single line of code (ZoracKy, 2022).

```
if (adminSubmission && adminSubmission.studentBackgroundImage !==
undefined) {
  var studentBackgroundImgData = await
imageDataURI.encodeFromURL(adminSubmission.studentBackgroundImage);
}
const personalPhotoImgData = await
imageDataURI.encodeFromURL(submission.personalPhoto);
const projectImageImgData = await
imageDataURI.encodeFromURL(submission.projectImage);
const youtubeImgData = await
imageDataURI.encodeFromFile("public/images/youtube.png");
if (adminSubmission && adminSubmission.studentBackgroundImage !==
undefined) {
  doc.addImage(studentBackgroundImgData, "PNG", 0, 0, pageDimensions[0],
pageDimensions[1]);
}
doc.addImage(
  personalPhotoImgData,
  "JPG",
  pageDimensions[0] / 50,
  pageDimensions[0] / 50,
  pageDimensions[0] / 6,
  pageDimensions[0] / 6
);
doc.addImage(
  projectImageImgData,
  "JPG",
  pageDimensions[0] / 50,
  pageDimensions[0] / 5.3,
  pageDimensions[0] / 2,
  pageDimensions[1] / 1.7
);
```

Figure 28: Addition of three images using the image-data-URI module to first extract the base-64 Data URL, and then the jsPDF addImage() function

10.3.2 Handbook Creation Time Out

Naturally, the handbook creation function is a very large one that a long time to process, with the process time varying depending on the number of submissions being created. When running this app from the IDE (Webstorm), this posed no issues. The only real issue was aesthetics – i.e., the lack of a loading spinner GIF. After deploying to Heroku however, this became a major issue because Heroku has a non-configurable clause whereby it times out after 30 seconds of inactivity – i.e., from the moment the request has been fully sent from the router. Whereas Heroku times out, and an error page appears, the application is unaware of the time out, and continues to work in the background (Anon., 2022). So from a functional perspective, the application still actually works on Heroku, albeit with that severe handicap (i.e., the user is redirected to the error screen, and has to then navigate to the directory listing by manually typing it into the browser address bar, and then has to continuously refresh the page to monitor progress of the PDF file creation). An obvious solution to this would be to revert to the original method outlined in Section 10.2.3.6, whereby the files would already be created, and it would just be a matter of merging them together, which would not take much processing time. As I didn't want to be over-reliant on the of the cache, I thought the best course of action would be to either write up a background job, or refactor the function to periodically send a http request to the server (as part of the existing pdf creation loop), which was all that was needed to prevent Heroku from timing out (Anon., 2022). However, attempts at this never reached fruition, and as the submission deadline was approaching, this bug was not resolved and now remains a part of the application that will need to be addressed in future releases.

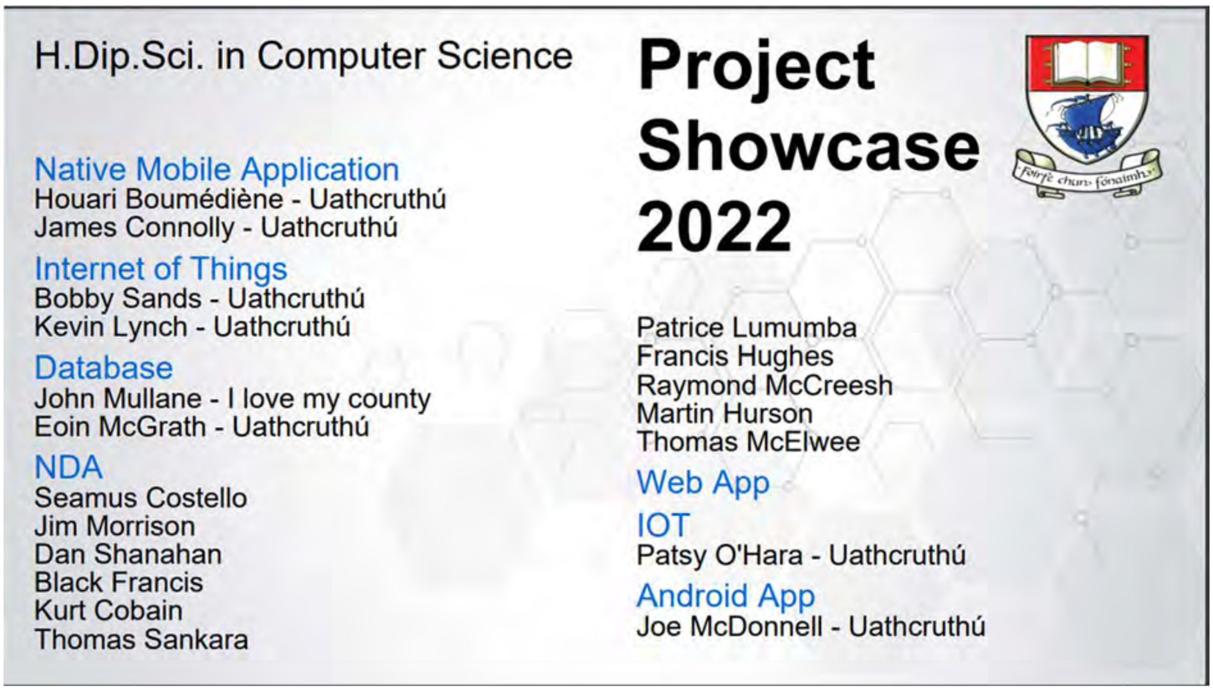
10.3.3 Access to PDF Files

When the user creates any PDF files (be they the user submissions, the admin submission or the entire handbook), they will obviously want to access and view that file. This has proven problematic. I never got the chance to fully explore how they would do this. I initially wanted to an automatic download to be triggered when the 'CREATE PDF' button was selected but didn't get the chance to do any research or development on this. My short-term solution was to rely on the cache, and just follow a link to the cache to view and download it from there. The problem with this however (even in the short-term) was that the button containing the embedded link would return a 404 error if selected, because the target file would obviously not yet exist. The submission list on the admin home page contains a links to each individual PDF file for each user (currently disabled), and there is also a link in the 'REVIEW HANDBOOK' button on the same page, and a link for reviewing the admin submission on the adminSubmission page. One solution that I attempted was to conditionally disable these links (reenabling them if the target file-path existed). Or alternatively, generation of some sort of notification or alert if the file didn't exist. There was some information available online about how to send requests to the server that would return a Boolean if a target file-path existed or not, which seemed like the ideal solution to this problem, but ultimately, I ran out of time and this is another bug that will need to be addressed in future releases.

10.3.4 Miscellaneous

Due to time constraints, a number of issues remain, some are bugs that were encountered, and others are areas for improvement or additional functionality:

- Favicon icon does not appear on certain pages.
- If a user submission is not finished, and they're the only one of that project type, the project type is still submitted to the admin pdf. This could have been fixed quite easily, but I intentionally decided not to get too bogged down with the specifics of the PDF creation and layout, as there were more pressing issues at hand, and getting the PDF creation perfectly aligned with the needs of the customer was well beyond the scope of this project (which is an academic work, not commercial).



H.Dip.Sci. in Computer Science

Native Mobile Application
Houari Boumédiène - Uathcruthú
James Connolly - Uathcruthú

Internet of Things
Bobby Sands - Uathcruthú
Kevin Lynch - Uathcruthú

Database
John Mullane - I love my county
Eoin McGrath - Uathcruthú

NDA
Seamus Costello
Jim Morrison
Dan Shanahan
Black Francis
Kurt Cobain
Thomas Sankara

Project Showcase 2022

Patrice Lumumba
Francis Hughes
Raymond McCreesh
Martin Hurson
Thomas McElwee

Web App

IOT
Patsy O'Hara - Uathcruthú

Android App
Joe McDonnell - Uathcruthú

Figure 29: Example of the inclusion of a project type (Web App) for which there are no submissions to list

- Security, Sessions and Cookies: This was never fully developed, and at present the application is quite vulnerable, with only very rudimentary security in place (sanitize HTML etc.). Also, the student has full access to the administrator's pages, and vice versa (provided the administrator knows the user's id.
- Scalability: There are a lot of inherent problems with scalability. For example, although the relative coordinate scheme for PDF creation makes for great flexibility, this has all been written in terms of returning quotients and not multiples, which are far more difficult for the human brain to process.

```
doc.text(submission.projectTitle, pageDimensions[0] / 5, pageDimensions[0]
/ 20);
doc.text(submission.descriptiveTitle, pageDimensions[0] / 5,
pageDimensions[0] / 12, {
  maxWidth: pageDimensions[0] / 1.3,
});pageDimensions[0] / 5, pageDimensions[0] / 5.7);

doc.text(submission.summary, pageDimensions[0] / 1.85, pageDimensions[0] /
4.8, {
  maxWidth: pageDimensions[0] / 2.23,
});
doc.textWithLink("Project Landing Page", pageDimensions[0] / 50,
pageDimensions[1] / 1.04, {
  url: submission.projectUrl,
});
doc.textWithLink("Video", pageDimensions[0] / 1.09, pageDimensions[1] /
1.08, { url: submission.videoUrl });
```

Figure 30: Example of the relative coordinate scheme, using quotients instead of multiples for return values

- User Interface: There are many UI discrepancies that were never addressed, the colour coordination of the menu links and the single column view of the admin home screen being just two examples.
- Lack of any Unit testing – whereas there is some very basic validation of all user inputs, there is no unit tests.
- Inconsistency in partial views allocation

11 Conclusion

11.1 Summary

The development of this application has had mixed results. It has failed to deliver a lot of the fundamental, basic features – namely the inability to create the handbook PDF without timing out, and the exclusion of essential security features.

However, the key success of this application is its ability to deliver some nice results based on the adoption of minimal use of technology. There are only four external modules employed for the PDF creation (jsPDF, pdf-merger-js, Cloudinary and image-data-uri). Apart from that, the delivery of this product has been achieved through the adoption of very basic javascript methods. Not only does this make the application less vulnerable to the maintenance of external libraries, but it also makes for better written code, whose independence makes it far more scalable for future product releases.

11.2 Key Learnings

The main takeaway that I can take from this project is proper care should be taken to prioritise work. A lot of time was spent trying to implement features, trying to perfect already implemented features, and dwelling too long on bug fixes. The definition of a bug should also be addressed early on in the development of any project. The interdependencies between objects and attributes makes for a very complex application framework, that is very susceptible to bugs. I found myself refusing to move on until all possible eventualities were addressed. This led to a very robust, error proof application, but at the expense of fundamental basics that were never addressed. In real life (as well as in this academic instance), the fundamentals should be addressed first. A working release can be presented to the customer, complete with all (minor) bugs, which can be addressed in future iterations. If too much time is spent addressing some of the more difficult to encounter or innocuous bugs – as part of the pursuit of perfection – the critical features can fall between the cracks.

12 Appendices

Appendix A: External Project Links

GitHub Repository	https://github.com/cathalohinse/Uathcruthu
Heroku Application	https://uathcruthu.herokuapp.com/
Project Landing Page	https://cathalohinse.github.io/Uathcruthu/
Demonstration Video	https://www.youtube.com/watch?v=Nly0TXLTJAA

Appendix B: Application Models

- User

firstName	String
lastName	String
email	String
password	String

- Submission

firstName	String
lastName	String
projectTitle	String
descriptiveTitle	String
projectType	String
personalPhoto	String
projectImage	String
summary	String
projectUrl	String
videoUrl	String
submitter	Object (User)
nda	Boolean
projectTypeOther	Boolean
presentationTime	String
submissionIncomplete	Boolean

- Admin

firstName	String
lastName	String
email	String
password	String

- AdminSubmission

courseTitle	String
handbookTitle	String
courseImage	String
backgroundImage	String
studentBackgroundImage	String
courseTitleLong	String
couresUrl	String
deadline	String
adminImage1	String
adminImage2	String
adminImage3	String
submitter	Object (Admin)

Appendix C: Controllers/Utilities & Functions

- Accounts

Index	Loads home page
showSignup	Loads Sign-up page
signup	Registers a new user
showLogin	Loads Login page
login	Logs a user in to the application
showSubmissionForm	Loads the student submission form page
showAdminHome	Loads the Admin Home page
logout	Logs a user out of the application

- Admins

showSubmissionAdmin	Loads the admin student submission page
submitByAdmin	Submits the student's details
createPdfUser	Creates the student PDF file
showOther	Loads the 'Project Type Other' page
submitOther	Submits updated submission Project Type
showHandbookForm	Loads the admin submission form page
adminSubmit	Submits the administerial details
createPdfAdmin	Creates the admin PDF file
createHandbook	Creates the handbook PDF file

- Submissions

showSubmission	Loads the student submission page
submit	Submits the student's details

- ImageStore

configure	Configures users credentials and db
getAllImages	Returns all images from db
uploadImage	Uploads image to Cloudinary db
deleteImage	Deletes image from db

- Pdfs

refresh	Ostensibly sends http requests to server
createFullUserPdf	Creates the Student PDF file if it *is not* under an NDA
createNdaUserPdf	Creates the Student PDF file if it *is* under an NDA
createAdminPdf	Creates the Admin PDF file

13 Bibliography

Anon., 2017. *Image to Data URI converter.* [Online]
Available at: https://ezgif.com/image-to-datauri
[Accessed 14 April 2022].

Anon., 2022. *Request Timeout.* [Online]
Available at: https://devcenter.heroku.com/articles/request-timeout
[Accessed 15 April 2022].

Brewster, C., 2020. *6 Examples of JavaScript: Where and When to Use - Trio.* [Online]
Available at: https://trio.dev/blog/examples-javascript
[Accessed 13 February 2022].

Dhaduk, H., 2021. *Express vs. Hapi: The Battle For Being Best Node.js Framework.* [Online]
Available at: https://www.simform.com/blog/express-vs-hapi/
[Accessed 14 February 2022].

Dziuba, A., 2020. *Why and When to Use Node.js in 2021 [Complete Guide].* [Online]
Available at: https://relevant.software/blog/why-and-when-to-use-node-js/
[Accessed 13 February 2022].

Hall, J., 2017. *jsPDF.* [Online]
Available at: https://artskydj.github.io/jsPDF/docs/
[Accessed 14 April 2022].

Hamedani, M., 2018. *How Node.js Works | Mosh.* [Online]
Available at: https://www.youtube.com/watch?v=jOupHNvDIq8
[Accessed 13 February 2022].

Johansson, A., 2020. *5 Reasons JavaScript is Still Better Than Python.* [Online]
Available at: https://www.computer.org/publications/tech-news/build-your-career/5-reasons-javascript-is-still-better-than-python
[Accessed 13 February 2022].

nbesli, 2022. *pdf-merger-js.* [Online]
Available at: https://github.com/nbesli/pdf-merger-js
[Accessed 14 April 2022].

Rose, S., 2019. *Java vs JavaScript: Everything You Need To Know!.* [Online]
Available at: https://medium.com/@scarlett8285/java-vs-javascript-everything-you-need-to-know-3362b5fa4128
[Accessed 13 February 2022].

Schwaber, K. & Sutherland, J., 2020. *The 2020 Scrum Guide.* [Online]
Available at: https://scrumguides.org/scrum-guide.html
[Accessed 11 April 2022].

Swersky, D., 2018. *Hapi vs. Express in 2019: Node.js framework comparison.* [Online]
Available at: https://raygun.com/blog/hapi-vs-express/
[Accessed 14 February 2022].

ZoracKy, D., 2022. *Image Data URI.* [Online]
Available at: https://github.com/DiegoZoracKy/image-data-uri
[Accessed 15 April 2022].