

Investigation Into The Digitalization And Omnichannel Experience Post Pandemic Education Could Present

LM118 – Bachelor of Engineering in Electronic and Computer Engineering

Project Interim Report

Emmett Lawlor

18238831

John Nelson

22/10/21

Declaration

**This interim report is presented in part fulfilment of the requirements for the LM118 Bachelor of Engineering in Electronic and Computer Engineering Bachelors Project.**

**It is entirely my own work and has not been submitted to any other University or Higher Education Institution or for any other academic award within the University of Limerick.**

**Where there has been made use of work of other people it has been fully acknowledged and referenced.**

|  |  |
| --- | --- |
| **Name** | **Emmett Lawlor** |
| **Signature** | **Emmett Lawlor** |
| **Date** | **15/10/2021** |

Table of Contents

Replace this text with a table showing the section/sub-section numbers, titles, and page numbers (such as shown below).

[Abstract ii](#_Toc83116529)

[Declaration ii](#_Toc83116530)

[Table of Contents ii](#_Toc83116531)I

[List of Software, Languages Used i](#_Toc83116533)V

[Chapter 1 introduction 1](#_Toc83116536)

[Chapter 2 Motivation for the project 4](#_Toc83116536)

[Chapter 3 Background Theory 6](#_Toc83116537)

[Chapter 4 Implementation to date 10](#_Toc83116538)

[Chapter 5 Action plan 2](#_Toc83116539)

[Chapter 5 CONCLUSIONS 2](#_Toc83116539)

chapter 7 [References 16](#_Toc83116540)

[Appendix A: Project Gantt chart - 2 -](#_Toc83116542)

[Appendix B: Interim presentation slides - 2 -](#_Toc83116543)

List of Software, Languages Used

**Languages:**

* Python 3.9
  + Frameworks:
    - Pandas [7]
    - Selenium [6]
    - Scrapy [8]
    - SKLearn [9]
    - FastAPI [10]
    - icalendar [11]
* JavaScript
* HTML
* CSS

**Software:**

* Tableau (Data Visualisation)
* Amazon Web Services (EC2 instances)

Introduction

This project is an all-around investigation into the opportunities presented by the digitisation of the entire world due to the Corona Virus pandemic. In the academic year of 2018/19, there was an estimated 237,328 students in Third Level Education in Ireland [1]. The entire country closed and these students had to go online, considering the trends of the CAO statistics of acceptance rates 2020 vs 2019 and the trend in points for 2019-2021 inclusive (figures 1 and 1.1), we can determine this number is still around the 200k range, that’s an interestingly large application of a new learning medium that we cannot ignore, a blended omnichannel experience.

***Omnichannel (in retail)*** *– “[Omnichannel] is a multichannel approach to sales that focus on providing seamless customer experience whether the client is shopping online from a mobile device, a laptop or in a [physical] store.”* [2]

Graphical user interface

Description automatically generated with medium confidence

*Figure 1: CAO Total Acceptances showing an upwards trend of 6.75% for 2019 vs 2020, Level 8’s only, according to cao.ie statistics (2021 Media Pack,* [*http://www.cao.ie/index.php?page=mediapack&bb=mediastats*](http://www.cao.ie/index.php?page=mediapack&bb=mediastats)*)*

Chart, waterfall chart

Description automatically generated

*Figure 1.1: Population receiving Leaving Certificate results. cao.ie statistics, aggregated by Emmett Lawlor from Leaving Cert Points Statistics (*[*http://www.cao.ie/index.php?page=points&bb=mediastats*](http://www.cao.ie/index.php?page=points&bb=mediastats)*)*

For the first time, a lot of people experienced an online learning experience. This prompted a mass online adoption and proposed a lot of new challenges for students and lecturers alike, aspects of this project were already in development at this stage, but further conjugated interest and motivation to investigate further, leading to the formation of the project;

“*Investigation Into The Digitalization And Omnichannel Experience Post Pandemic Education Could Present*”.

This project develops on the investigation and demonstrates practical applications with modern, open-source, free technologies with the idea of staying free at a consumer (students, in this case) level. The investigation will consider points of views and an analytical report, where we will try to look at the pros and cons of the digitisation and omnichannel experience of Third Level Education, we will be composing an evaluation on a Student and University level, considering many aspects such as fees, attendance, housing, and others.

The applications will entail of a Machine Learning recommendation model for Higher Level CAO applicable courses (named: The Academy) and a Web-App for students to streamline the start of academic semesters.

**The Academy:**

Named after Plato’s Academy in the fifth century, where philosophers and scholars from mainly Athens and all over Greece, would come to discuss current topics. Plato seen it as a revolution in education, where people can study and learn as they please. [3]

The Academy is to serve as an entry point for people from all over Ireland to come and find courses based on interests. Potential higher level students can enter a body of text or browse courses to receive a list of recommended courses.

This application was chosen from a tactical standpoint of having large amounts of text data from our web scraping sources, this application will be a proof of concept of a browsing system orientating around the students preferences, instead of the usual call to a career guidance counsellor or the ‘select, filter, search’ system implemented in many browsing systems we see today. Students will be able to input a body of text about themselves or their interests and be recommended courses.

This model serves a bigger picture to the applications of Machine Learning models in academic establishments, other areas of interest were “automatic correction and proof checks of essays”, “a module level chatbot, for questions and tutoring”, which we will touch on later in this report.

**The Web-App:**

The Web-App will be a student orientated web application. Students can enter *course code* and *course year*, the web-app returns the list of corresponding modules to which students can then select, upon confirming selection of modules a student can then download an integrated calendar for any system, as well as a list of purchasable PDF books recommended by their lecturers. We will expand later on other possible integrations such as online material, events, digital orientation days, providing books/notes, introductions to lecturers.

Motivation for the project

**Shifts in the industry resulting in a large market for Educative tools:**

The motivation for this project was that of the mass adoption and acceleration of online and virtual environments for students, as well as a strong belief in a designed and structured education for everyone which is as affordable and accessible for students from all and any backgrounds, with or without an official and reputable certification or degree.

Considering there are already options for students to learn online through platforms such as Coursera or Udemy, there are no platforms for an approach to an omnichannel experience, where students can seamlessly drop in and out of online and on campus learning, being able to complete and attend courses in which ever medium preferred, when preferred.

We are now in an early stage adaption of omnichannel learning, Third Level Education institutions have underwent some sort of digitisation while trying to return to campus, the thought that some may prefer this learning method will be the driving force for an omnichannel experience in coming years, as technology improves and as coming generations (current students as parents, or advising younger siblings), have a choice of Online, Omnichannel or On Campus learning.

**Personal motivation from experiencing both sides of financial scales:**

A motivation personal to myself is to make education free, or the price of a laptop. There is a lot of information on the internet, that is a good and a chaotic thing, people with a general interest in a subject often don’t know where to start. I want to add course structure to simple information you can find online where people can access it and have books, Wiki-notes, YouTube videos, be connected with people taking the same course. I also hope to offer corrections and grading using a Machine Learning model. This can lead similar situations to what we have seen on platforms like GitHub, Stack Overflow you can be hired for how active you are, your contributions can be viewed, and in this case, their overall QCA/GPA.

I resented University for a while as it was simply bleeding my parents dry of money, we could not make payments for accommodation or fees, my email inbox full of reminders of fees for late payments. I would work night shifts in Scholars till 2am (making pennies), and still not be able to get a full week worth of groceries. Fortunately, I was introduced to code in 1st year, and in 3rd year I got my first job as a programmer (making dollars), as well as my family having a very positive outcome from the pandemic with career changes. I realised I do not resent University itself, but I was simply under stress of something out of my control.

A lot of people can’t afford college but already have laptops lying around their house, the idea here is that we can then digitize an entire university, and the university can then be flexible in it’s delivery, blended or online with complete synchronicity between them. I don’t want to erase universities, college is great, but there are a lot of people out there that do not have access to the funds for college, who could, without $10K/year, prove worth of a scholarship or for job knowledge, or for general interest.

**Bringing the internet back to its original purpose:**

It’s also about taking a slice of the social media’s grip on peoples realities. “*Globally, the* ***average time*** *a person spends on social media a day is* ***2 hours 24 minutes****; if someone signed up at 16 and lived to 70, they would spend* ***5.7 years*** *of their life on it*”[4] We need to orientate the internet to creativity, facts and information, make it relevant and up to date.

Background Theory

**Introduction to Web Scrapers:**

Web scraping “is data scraping used for extracting data from webpages”[5]. Web Scrapers traditionally use the files transferred to build the webpage (HTML documents). HTML is *“HyperText Markdown Language”*, web scrapers can use HTML to navigate a webpage autonomously, using a predefined *“path”* to the data of interest. We can use web scrapers to extract large amounts of data in minimal time compared to how long it would take a human to do.

Graphical user interface

Description automatically generated with low confidence

*Figure 3.1: A website to extract data from.*

In Figure 3.1, we can see the correlation of the HTML element to text on the actual website (available in all browsers for all websites, right click on website and click “inspect” to view the HTML.) There are multiple ways to access this element using “x*paths”* and “css *selectors”*, we can use an xpath (right click on a highlighted element, copy, copy xpath),

**Certain issues you can face when using Web Scrapers:**

Captchas:

* CAPTCHA stands for “Completely Automated Public Turing test to tell Computers and Humans Apart”
* Yahoo! Had a talk at Carnegie Mellon University for a software in which would disrupt people who were using bots to register millions of email accounts. A student solved the issue in 2000.
* “Type the characters you see” became the solution until years later OCR and ML became too advanced
* Nowadays we are solving self-driving car’s problems. A majority of human inputs are taken as the ground truth, this helps train the cars as well as separating the bots from the humans.
* reCAPTCHA V3, will have no human to captcha interaction. The captcha will be watching your mouse and how you interact with the website, evaluating how programmatic you are.

Captchas cannot be solved without human interaction, at least not with a decent success rate. Most captchas do not redirect you if you get the captcha wrong, so bots can stay there randomly guessing until a pattern is correct. I used to scrape webpages, and I would have the program wait and alert me if a captcha has been presented, me being at my computer 24/7, would solve it and program would re-continue. I would have about 500 webpages only being asked to do a captcha 4 or 5 times.

DOM stands for Document Object Model, it is a HTML with JavaScript design, wherein the website generates and displays content upon request. The most popular framework used for Web Scraping with Python would be scrapy, although scrapy does not render JavaScript, it will only read the response to the request, which would usually be a HTML document saying, “*JavaScript needs to be enabled*”.

JavaScript in general is usually used as a defence against Web Scraping, a possible integration would be token checking upon clicking of a button which goes to a webpage of interest, if the token is correct to the one sent to the server and is expected with the request, the page is displayed. Frameworks like scrapy do not “*click*”, they would read the HTML document, our *response*, check the elements they are programmed to check and find the attributes they are told to find, in this case a *“href”* attribute would house the URL to get the information of interest, the scraper will simply send a request to that URL, and so no real *“click”* would trigger the generation of the token, and so a “404” is returned.

Graphical user interface, website

Description automatically generatedGraphical user interface

Description automatically generated

Figure 3.2: YouTube is the example of a DOM, a website which does not function without JavaScript.

Most websites, not all, have this level of defence. In the cases in which you need to have JavaScript enabled there are Frameworks available for this. One such framework I worked with for this project was Selenium [6].

Selenium is a web-browser automation software. It’s advertised as a unit test program; if you’re developing a website and you are testing the login flow, you would automate the flow and integrate it to run at compile/testing/deployment/rebuild, and watch it happen, if the bot (script which automates the web-browser) cannot complete the flow it will throw an error telling you where.

I have used Selenium to automate a social media account, such as posting, commenting and liking pictures. I coded a self-sufficient Instagram account which would post a picture found on the web every day.

We can use Selenium as a web scraper too, one implementation I have come to view as the future of web scraping is the use of OCR (Optical Character Recognition) with Selenium. Character recognition can be used if the HTML document has randomly generated element names and classes. A good example is Facebook, randomly generating the HTML attributes can help stop bots which rely heavily on HTML, so we could use Selenium to open the webpage, save it as an image and use an OCR engine that would be used to read the information in the webpage and translate it into data.

Logins can also be an effective way of dis-encouraging basic scrapers. We can see with Instagram, it does not work with Scrapy (without complicated middleware), and with Selenium we can scrape approx. 500 profiles without logging-in, after this we are restricted from even viewing a profile as our IP address, Mac address, or fingerprint may all be compromised and flagged on Instagram’s systems, unless we log-in and act as a simple account. To solve this, we can create an account using a disposable email account. The newly created account is bound to be blocked to where we would simply generate another email and re-create an account.

Thankfully, none of the websites targeted in my investigation did not have much complexity.

This is all the data that is sent to users of the Web App, times 5 timeslots per 5 modules or so based on continuity of the active weeks (“Wks: 1-3, 4-6, 9”, would result in three subsequent events for one cell in the timetable). It’s important to keep ics files low kB otherwise some programs may not accept them.

**Text Processing and The Academy (AI Aspect):**

Our large bodies of text data need to go *“under the microscope”* in order to be classified and tokenized into machine distinguishable data. To do this we could have used many frameworks for Python; Keras, TensorFlow, NLTK (Natural Language Toolkit)

I chose SKLearn, also known as Sci-Kit Learn for Python. I chose Sci-kit as it is a simple out of the box for my purpose, to demonstrate the capability of Machine Learning in an educational environment.

The idea to include a ML model in this project was to solve certain issues, among them were

* Correction and grading of essays. This was abandoned due to fore-seen data issues in the event a lecturer was to share all digital copies submitted for an assignment on SULIS).
* A chatbot who could operate as a helpful link between students and their questions, to assist in a student who may be poor at self-learning. I personally always prefer to ask questions rather than google them, having a chatbot trained multiple times on one subject could offer multiple correct ways of explaining in the event one answer does not solve a question. I abandoned this idea as it would have turned into more Web Scraping and data gathering, which time cannot allow for.
* Smart course recommendations to help people undecisive of which course to do, whether they be 6th year Secondary students or just people looking for another option in life. I chose this project as I saw the most immediate value in it for the data I have already acquired for my analytical report.

Smart course recommendation is to be implemented so students would have better results through searching courses and or describing the course/field they want. To implement this system we will use K-Means Clustering.

I have used K-Means Clustering before using a car database, to create a recommendation system dependent on input of certain parameters (model year, cylinders, door numbers, etc.). The idea was to input parameters of a car you like and it would recommend cars with non-identical, but cars inside of the cluster.

I will comment more on this aspect in later reports, in order to keep information consistent as majority of the completed work currently falls on a Proof of Concept, not the final work.

Implementation to date

Here we will touch on the current implementations of the Web-App.

**Web Scrapers**We gather our large data sets using xpath selectors in Scrapy for Python.Xpath representation of element in figure 3.1:

***//\*[@id="HeaderContent\_Repeater2\_SemesterLabel\_0"]***

or

***/html/body/form/div[4]/h2/span***

Similar to a file path in format, search using forward slashes, but imagine you have 4 desktops and you need to index one, you would use *desktop[x]*, where x is in the range [1] to length of the array. We can write within our code;

1. **“<scrapy.object> = < *html.object >.findElementByXpath(‘/html/body/form/div[4]/h2/span’)*”**

Here we set a variable name to a scrapy object, we could have searched by *findElmementByID(‘HeaderContent\_Repeater2\_SemesterLabel\_0’)* . We can access the ID with the variable set at (1), but we will extract the text instead as the ID has no real value like an SKU (Stock-Keeping-Unit, unique identifier to all products that “track the price, product details, and the manufacturer”[5]( https://www.investopedia.com/terms/s/stock-keeping-unit-sku.asp)) would if you were doing this on an e-commerce site.

To access attributes we can use:

**“<string> = <scrapy.object>.get\_attribute(‘innerHTML’)”**

or

**“<string> = < *html.object >.findElementByXpath(‘/html/body/form/div[4]/h2/span/text()’)*”**

We can assign the string variable to a JSON object, which we then save into a file object to be read later. A JSON object is generally data stored in a transferable form, it is typically used to transfer data from server to client across networks, it’s used in API’s (Automated Programming Interface, a body of pre-defined code for open use), where JavaScript would be the receiver, where it can then access the values stored.

*Text

Description automatically generated*

*Assigning a key and value in to a json type variable, and we see how to access the values.*

**

*We appended our json type data to a list, which is would usually be saved to memory and sent to client side at request.*

*Text

Description automatically generated*

*Here we see a simple for loop, for object in list of objects, read their key values. We detail the potential error and how to handle it, in this case we don’t care so we pass.*

**Qualifax Scraper:**

Selenium was used for qualifax.ie, the website’s search method was not available without JavaScript, so using selenium I could, view all courses and flick through each to gather initial information (figure 3.3). Here our real valuable information is the course link, which we will run another selenium scraper to open all these links and extract information on the courses (figure 3.4).

A picture containing timeline

Description automatically generated

Figure 3.3: Columns (our keys in sense of JSON), and the information (values), as well as the link leading to the course qualifax and university webpage. <https://rb.gy/k2wjyw> (qualifax search result page link shortened by <https://free-url-shortener.rb.gy/> )

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Figure 3.4: Columns (our keys in sense of JSON) that we can extract from the qualifax course webpage. <https://rb.gy/t1pvyw> (qualifax course page link shortened by <https://free-url-shortener.rb.gy/> )

Here we see our data laid out for us, keys and values, we will be creating some more keys as we can extract additional information from the links as well as dropping uninteresting columns, some courses have more columns than others, and that’s expected, where other courses have no value for a specific key (upon aggregation of all the data into a dataset) we will use a “NULL” type object.

This data was extracted purely from within our script, and written as csv (Comma Separated Values, wide compatibility with programs like Excel).

This data will be used in our AI aspect, I chose this data over the UL data as both contain a large amount of informational text data with other classifications (one line tags to distinguish courses; course name, course type, institution, county, etc.) but this data spans the entire country.

**UL Web Scraper:**

UL web scraper was created using a hybrid example of Selenium + Scrapy within Python, selenium scraper would scroll <https://timetable.ul.ie/UA/CourseTimetable.aspx> saving a .html file of each year for each course. Upon completion we use scrapy to iterate through all the .html files to translate relevant information into JSON data, as this will be the data implemented into the Web App, so we needed it easily accessible and transferable.

How we translate this data into something useful like an .ics file (Internet Calendar Scheduling, exchanging calendaring and scheduling information such as events, to-dos, journal entries, and free/busy information [6] <https://datatracker.ietf.org/doc/html/rfc5545> ) is to use a package for Python simply called “*icalendar*”. We used icalendar to translate time-slot information into RFC-5545 standard’s markdown language and correct data format (timestamps, etc.).

**Web App Tech Stack:**A “*stack*” is often defined as how we describe the webapp and it’s layers that make up a website and it’s services, from server to client. The tech stack consists of backend (servers), frontend (website, client interface) and the middle layer, which facilitates the communication between frontend and backend, often a JavaScript framework.

1. **Backend:**
   1. **FastAPI for Python:** FastAPI is a framework for easy building of an API. An API is used to transfer data from one point to another, here it would be from my server to the client. This where the entire backbone lays for the Web-App.
   2. **icalendar for Python:** icalendar is used to parse files into an Internet Calendar Scheduling file, compatible with all systems and native/third party calendar software.
   3. **Scrapy + Selenium for Python:** Selenium and Scrapy are used for the initial collection of data through the web scraping process, and the incremental refresh as timetables change and get updated.
   4. **Pandas for Python:** Pandas is a powerful data analytics framework, it saves us a lot of hassle in reading from databases and filtering/handling of data.
   5. **Heroku:** Heroku is a free and easy hosting service for API’s. The webapp can be reached at a certain URL, but only verified URLs/IP’s (defined within FastAPI) will receive a successful response.
2. **Frontend:**
   1. **HTML:** HTML is used as the base structure of the website to which is presented to users.
   2. **CSS:** CSS is the appearance of the website and works with the HTML to define a unique appearance.
   3. **JavaScript:** As the website is a DOM model, we can use JavaScript to change the HTML in order to display error messages, results, and handle actions taken on the web-app itself, such as pressing of buttons. JavaScript has very strong capabilities, being able to manipulate the CSS and other Attributes of the HTML.
   4. **Amazon Web Services, Linux Apache Server:** This is used as a hosting service, a virtual computer/server runs 24/7 in the cloud, which hosts just the frontend and handles requests. From here, the machine can request and receive responses from the API hosted on Heroku (2.e).
3. **Middle Layer:**
   1. **JavaScript:** The tool in which we use to make requests, using *fetch*, a native function to JavaScript, where we can send and receive data from endpoints on servers such as API’s. This would also be hosted with the frontend to connect Heroku and our AWS virtual machine for the transfer of data.

Action plan

* By end of Christmas Break:
  + Have AI aspect; recommendation system, finished and ready for live demonstration.
  + Start analytical report.
    - Basic investigations into data accumulated so far, its’s consistency and reliability. Some things worth touching on:
      * Errors in CAO statistics.
      * Correlation of stated module data and allocated data in UL.
      * How many optional pathways there are beyond your 1st year entry course.
* Semester 2:
  + Clear the Web App to be shared on student mail lists within UL.
  + Gathering more data for analytical report.
    - More CAO statistics
    - UL FOI request
    - Housing data
  + Start investigation for hypothesis’:
    - Remote learning can solve the housing crisis
    - Institution level education can be 100% free
    - There’s a positive correlation between acceptance rates and housing prices/scarcity
  + Further prospects for this project.

Conclusions

I feel quite comfortable with this project. The only thing stopping the process right now is an “overfit” of our text data for the recommendation model, this will be the fuel behind the text processing, finding the best way to token and classify educational data (course type, university, similar courses from multiple universities, by county).

The Web App today has accumulated 500 unique events (submission of course information; course year and course code) in the month of September alone. I am not satisfied with this amount of data, but without any advertising or publication I hope to change this next semester. The most effective approach was to view the most popular lecture rooms overall on timetables, and put QR codes outside of them. Using the data from their submitted information and the locations I put the QR codes, I plan to go back and find the most effective areas, what courses adopted the tool and what years, to retarget the audience.

I feel the analytical report will be the hardest part, I will try keep this very data based and leave very little up to interpretation.

References

[1] <https://data.oireachtas.ie/ie/oireachtas/libraryResearch/2020/2020-04-03_l-rs-infographic-education-in-ireland-a-statistical-snapshot_en.pdf>

[2] <https://www.bloomreach.com/en/blog/2019/07/omnichannel-commerce-for-business.html#what-is-omnichannel-commerce>

[3] <https://iep.utm.edu/academy/#H3>

[4] <https://backlinko.com/social-media-users#social-media-usage-stats>

[5] <https://en.wikipedia.org/wiki/Web_scraping>

[6] <https://www.selenium.dev/>

[7] <https://pandas.pydata.org/>

[8] <https://scrapy.org/>

[9] <https://scikit-learn.org/stable/>

[10] <https://fastapi.tiangolo.com/>

[11] <https://pypi.org/project/icalendar/>

Appendix A: Project Gantt chart

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

Appendix B: Interim presentation slides

Graphical user interface, text, application

Description automatically generatedText, letter

Description automatically generatedTimeline

Description automatically generatedTimeline

Description automatically generatedText, letter

Description automatically generated