John Eccles Institute: a system for collecting, demonstrating, and promoting upcoming seminars, talks and lectures at the ANU

24th May 2024

Author: Chenwei Niu u7377070

Chenwei.Niu@anu.edu.au

School of Computing (SOCO), The Australian National University (ANU), Canberra, ACT, Australia

Primary Supervisor:

Prof Hanna Suominen, SOCO and School of Medicine and Psychology, The ANU, Canberra, ACT, Australia

Co-supervisors:

Prof John Watson and Dr Shaam Al Abed, John Curtin School of Medical Research (JCSMR), The ANU, Canberra, ACT, Australia

Dr Chirath Hettiarachchi, SOCO, The ANU, Canberra, ACT, Australia

Project Client:

ANU John Eccles Institute, represented Prof John Watson (Director of JEI); Prof Hanna Suominen (Associate Director (Neuroinformatics) of JEI); Ms Louise Fleck (JEI Manager); and Dr Shaam Al Abed (JEI Advisory Committee Member)

The project report for COMP8755 Individual Computing Project

Unless otherwise stated, this report is my own original work.

Chenwei Niu

24th May 2024

Contents

Abstract

Each public seminar at the Australian National University (ANU) is an invaluable opportunity to learn something new of interest. ANU students and staff often miss valuable seminars due to the time-consuming process of visiting each faculty's event webpage. Particularly for a discipline such as neuroscience, which encompasses multiple disciplines (e.g., medicine, biology, psychology, computing, etc.), the problem of segregation of seminars between different faculties exists. So, this project aims to create a centralized platform that consolidates and displays all upcoming seminars efficiently, enhancing accessibility and participation. This report of my Master of Computing capstone individual project at ANU will explain a system that crawls ANU's public lectures, talks and seminars on neuroscience related topics and presents them centrally on a website with useful capabilities. In addition, the system is able to send seminars weekly to subscribers that may be of interest to them based on natural language processing. The system also provides a graphical, web-based management tool that is easy for administrators to use. The implementation utilized the test-driven development, design thinking, scalable design with Agile development methodology, delivering iterative, usable products and regularly improved by client feedback on routine meetings. Differences between implementation approaches were compared and the justification of technical decisions were demonstrated. The system employed Scrapy for crawling, PostgreSQL for data persistence, ReactJS for the website's front-end, Fast API for the website's back end, NodeJS for the management application's server side, and SpaCy library and built-in trained English language models as tools for NLP processing, mainly applied to seminar recommendations and data extraction. My project helped the John Eccles Institute complete its seminar website, addressing the problem that neuroscience-related seminars are scattered in different departments.

Keywords: Scrapy crawler, Email assembling, Natural Language Processing, SpaCy, John Eccles Institute, Seminars spider

Introduction

As interdisciplinary sciences continue to emerge and develop in today's world, they are attracting increasing interest from policymakers and investment organizations. A growing number of individuals are also engaging in interdisciplinary research and learning. However, this field faces numerous challenges. Interdisciplinary research involves collaboration among researchers from different disciplines to develop new approaches, theories, methodologies, products, projects, and policies (Archibald et al., 2023). Nash noted that one significant issue in interprofessional disciplines is the limitation caused by disciplinary fragmentation, scholars often spend considerable time finding and integrating information from various fields, and accessing detailed information is challenging for those seeking in-depth knowledge (Nash, 2008). Furthermore, participants in interdisciplinary learning and research often find that understanding the specialized terminology of different disciplines is time-consuming, confusing, and frustrating (Nash, 2008). There is a pressing need for a platform that integrates information from multiple related disciplines. Such a platform would not only provide emotional value but also enhance cohesion within interdisciplinary research teams.

The John Eccles Institute of Neuroscience is my project client, as one such interdisciplinary institute related to medicine, biology, psychology, and computing. The ANU existing approach to interdisciplinary sciences is to set up a dedicated institute website to support access to information, and to post research results, awards, and networking events on it. The department website also gives researchers a sense of belonging. However, the information presented on the website may not be in-depth, and more specific information has to be sought on the relevant College, School and Faculty websites. Taking seminars as an example, these are good opportunities to learn new knowledge, but scholars need to spend much time and effort to identify appropriate seminars on the websites of different colleges. Numerous neuroscience-related seminars, lectures, talks, reading groups, journal clubs, and visits from prominent figures are likely to interest many individuals both within and outside the ANU. However, there is currently no central platform to gather and disseminate information about these opportunities. The clients from John Eccles Institute proposed a web platform, which could be conceptualized as a bot, to achieve this goal.

Consequently, the aim of this Master's project was to develop a centralized platform that effectively consolidates and showcases all upcoming seminars, thereby improving accessibility and encouraging participation.

The main outcomes of this project were as follows:

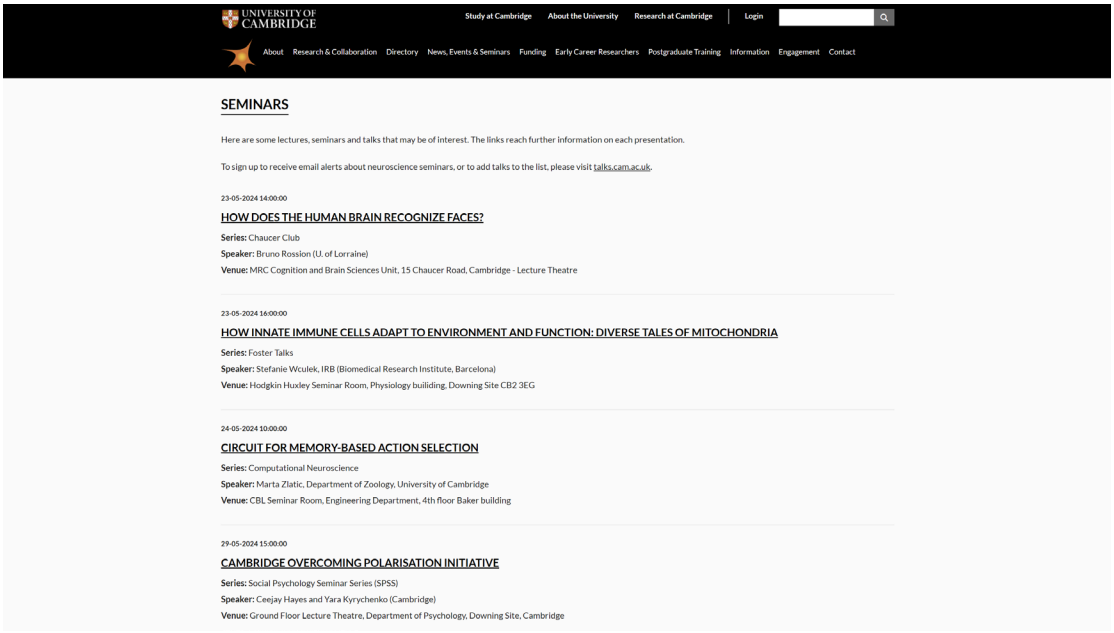
1. A customer-satisfying website that collects all upcoming seminars and public lectures

of a given college, with many useful features including sorting seminars by user preferences, bookmarking specific seminars.

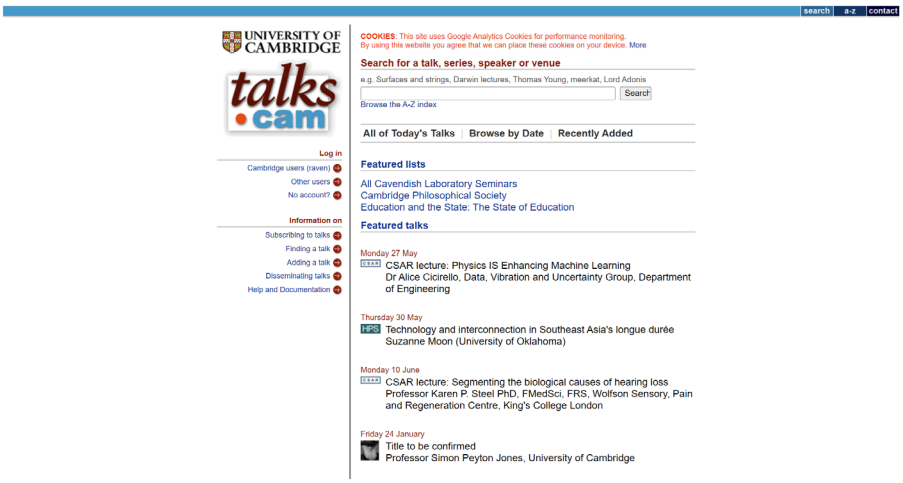
2. A functional information management system capable of managing recipients and seminar information in a database.
3. A promotional email distribution system that filters out and sends subscribers seminars they might like based on their interests.

Background and Related Work

To address the important disciplinary fragmentation issue in transdisciplinary science, in our client's case, neuroscience, there have been other world-renowned universities such as the Cambridge University and Johns Hopkins University have developed similar systems holding neuroscience related seminars. Figure 1 is a concrete example from the Cambridge University neuroscience integrated website. The world's leading universities are visionaries, and their main goal of implementing such sites is to create a supportive environment, mitigate barriers to accessing multidisciplinary information, and enhance multidisciplinary scholars' self-identity and sense of belonging, thereby increasing team effectiveness, motivation, and innovation. Two useful features offered by a separated website (Figure 2) that are worth noting are the ability to search for seminars and subscribe to promotional emails pushing out seminars of interest to users. Although the system obtains these useful features, it is inconvenient as the users need to redirect from neuroscience seminar webpage to talks.cam webpage to use them.

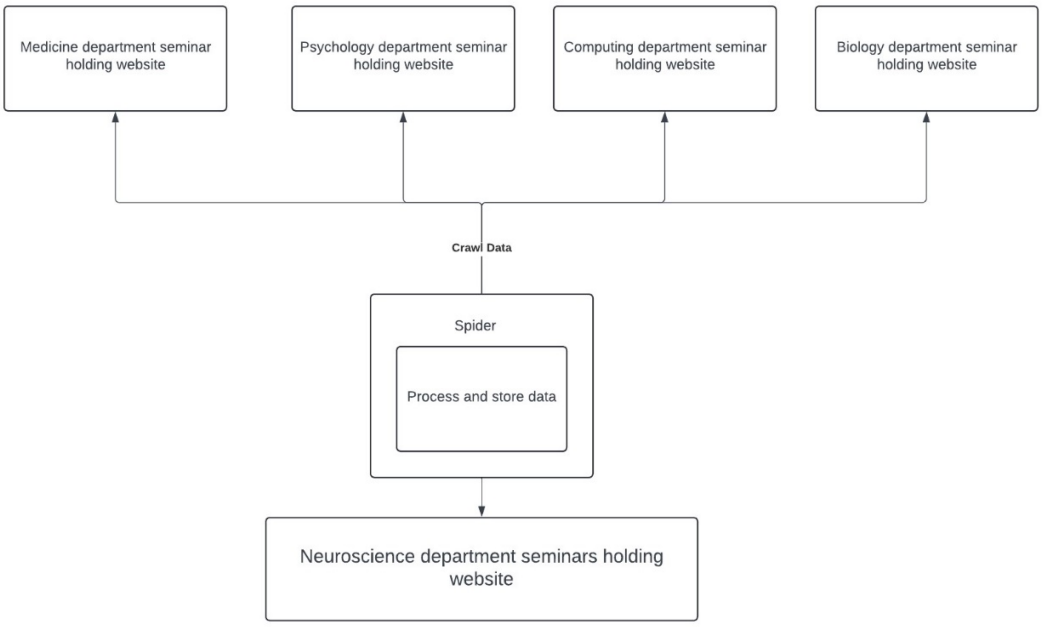


(Figure 1, Screenshot of seminars webpage of neuroscience faculty in the Cambridge University, https://neuroscience.cam.ac.uk/news-events-seminars/seminars/)



(Figure 2, Screenshot of talks.cam, with seminars searching and subscribing functionalities, www.talks.cam.ac.uk)

Although we cannot determine from the website alone whether they used the technical route of crawling other entities' specific seminar pages, the technical route of crawling was chosen for our product in order to reduce human involvement and increase automation. A web crawler, also known as a spider or search engine bot, automatically browses the content of webpages from a given range or over the entire Internet, learning and storing pivotal information of each webpage so that information can be retrieved when needed (Cloudflare, 2024). "Crawling" is the technical term for this automated data access process (Cloudflare, 2024). An abstract flowchart is shown in Figure 3, where a crawler crawls seminar data from different departments, processes and stores it before presenting it to the neuroscience website.



(Figure 3, Flowchart of crawler technical approach)

My product has a greater advantage over their system because of its adequate design and the

use of appropriate technological approaches. My product also integrates useful features described above on the same website instead of separating them on two different websites. To get an overview of the technical routes as well as other valuable information through Figure 4, my project poster.

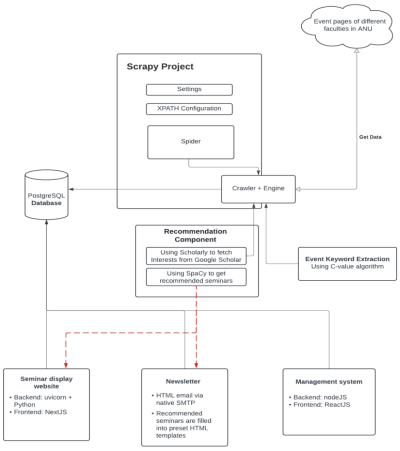
ANU John Eccles Institute Project:
An automated and user-friendly system for collecting, demonstrating, and promoting upcoming seminars, talks, and lectures at ANU

Chenwei Niu

Browsing through each college's event webpages to discover seminars that may be of interest takes a huge time. This system has been developed to give academics, students, and other audiences an easy and effortless access to seminar announcements. It collects upcoming seminars from many ANU colleges and provides a website to display and search them. Subscribers can also receive customised seminars newsletters based on their interests. Finally, administrators are supported by an easy-to-use system to facilitate managing communications and subscriptions.

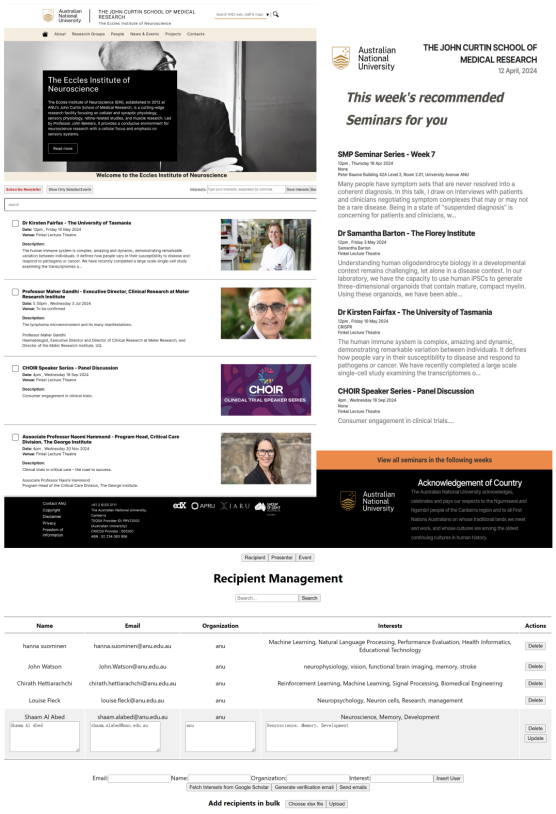
Background:
The project is an individual postgraduate project for The ANU Eccles Institute of Neuroscience. Neuroscience tends to be transdisciplinary, ranging from medicine, psychology, and biology to music, law, and computing. Therefore, seminars related to neuroscience held by other faculties should be shown on the website. The neuroscience departments at other world's leading universities like Cambridge University have similar websites and systems.

Method:
- ***Scrapy*** crawling framework was used to regularly crawl assigned websites, then processing and storing seminars details into a ***Postgres*** database.
- ***Scholarly*** was used to fetch seminars speakers' research interests from Google Scholar.
- ***SpaCy*** and its build-in NLP word-to-vector models were used to calculate similarity between user interests and event keywords, then filtered out recommended seminars.



Result:

Figure. (1) Seminars display website, (2) Weekly recommendations email, (3) Administrative management system.



Evaluation:
- When determining recommended seminars, a threshold value is required. After evaluation by supervisors and clients, 0.725 was confirmed as the final threshold.
- In the meetings with clients, they expressed their endorsement of the effectiveness, usability, and aesthetics of the systems.

Conclusion:
The project addressed the problem of academics interested in neuroscience spending much time looking for seminars of interest. The systems provided a high degree of automation and an effective administrative tool.

Future Work:
- The robustness of information extraction could be enhanced by adding more hierarchical natural language processing algorithms used in the newspaper3k library to the XPATH extraction method.
- Further engineering work could be conducted to avoid service interruptions in fetching subscribers/recipients' research interests from Google Scholar in bulk.

Contact Information: Chenwei.Niu@anu.edu.au

(Figure 4, Project Poster)

The superiority of my system involves search functionality, intelligent recommendation system, good portability, utility, and effectiveness. These advantages are detailed in the sections “Modules and Implemented Methods”, “Results”, and “Advantages of the project product”.

The crawler mechanism is the cornerstone of the whole project, because without the crawled data, then all the functionality is only on paper. At the beginning of the project, an desirable idea for me as well as the team was to create a generalized algorithm that could extract the titles, descriptions, speakers, and dates of seminars from the pages through natural language processing. Additionally, this algorithm should also support the extraction of outgoing URLs from the base webpage like breath first search. This will be convenient for the maintenance staff, whenever they want to add a seminar of a certain department, they just need to add the website that shows the seminar list of that department to the system, and it is powerful and versatile enough to accommodate not only ANU's on-campus seminars, but also other universities' seminar websites.

Like the figure below, this Pseudo code is used to select from a list of URLs that are related to the topic website.

Algorithm 1: Baseline Focused Crawler

```
Input: Seed URLs, numPages, urlScoreThreshold
Output: crawled webpages
Insert URLs in Priority Queue;
Download seed webpages from seed URLs;
topicVec ← Build topic representation from seed webpages;
while pagesCount < numPages do
    URL ← pop(priorityQueue);
    append URL to visited list;
    webpage ← download(URL);
    webpageVector = process(webpage);
    pageScore = calculateScore(webpageVector, topicVector);
    pagesCount +=1;
    webpageOutgoingURLs ← extract outgoing URLs from
    webpage;
    add webpage to saved collection;
    for link in webpageOutgoingURLs do
        validate(URL);
        if URL not in visited list and URL not in priorityQueue
        then
            urlVector ← process(URL text);
            urlScore ← calculateScore(urlVector, topicVector);
            if urlScore >= urlScoreThreshold then
                push(URL,priorityQueue);
            end
        end
    end
end
```

(Figure 5, Baseline Focused Crawler Algorithm) (Farag et al., 2017)

The core of this code is twofold.

1.1 Determination of relevant information

calculateScore(webpage Vector, topic Vector) method at line 9 scores the vectors converted from the webpage html content according to a given topic vector to determine whether the web page is relevant by the returned score. When applied to our crawler system, we can firstly extract the content encapsulated within html tags. Then, we could design several calculate score methods to determine whether a html tag is seminar title or date or description or speaker, etc, and loop through each tag. Once the score exceeds the given threshold, that content should be considered as valid seminar information. A concrete example could be the calculate score function for date, i.e. calculateScore(htmltag Vector, date Vector). It takes two parameters, a vector converted from content in html tag and a valid date format vector.

1.2 Due with outgoing URLs

The code from line 13 to line 22 give the approach of processing outgoing URLs. Firstly, it validates whether it is a legitimate URL, if the URL has not been visited and is not in the priority queue, then the URL will be transformed into a URL vector and calculate the value to see whether it is related to the topic by comparing with threshold. If relevant, the URL can be temporarily pushed into the priority queue, waiting for the while loop to process. If applied to my system, here I give a concrete example, for example, the URL is https://comp.anu.edu.au/events/, since the text of the URL contains the keyword *'events'*, then there is a high probability that its calculated vector score is greater than threshold, it will be pushed to the priority queue. For example, https://www.anu.edu.au/giving/support-us which has no seminar indicating keywords in the text, so it is very likely that this URL's score is less than threshold and will not be processed further.

However, after a 1-week feasibility analysis, we decided not to use this method. The reasons for this are as follows.

2.1 Training a generic information extraction model

To recognize the title, date, speaker, etc., the solution that comes to mind is to use the well labelled dataset to train several models. There are two directions for the models, the first is a LayoutLM-like model that extracts structured data based on the layout of

image, like the one below, which can extract all kinds of information based on the layout (Kalra, 2023). Although it does not contain a calculating score function, the model determines whether to extract that information by the layout of the text, which is itself a calculation of score. However, we need to spend much time labelling these pieces of text accurately and manually, and once the layout is changed, the model cannot handle it. However, each department and faculty at ANU has a different seminar website layout.

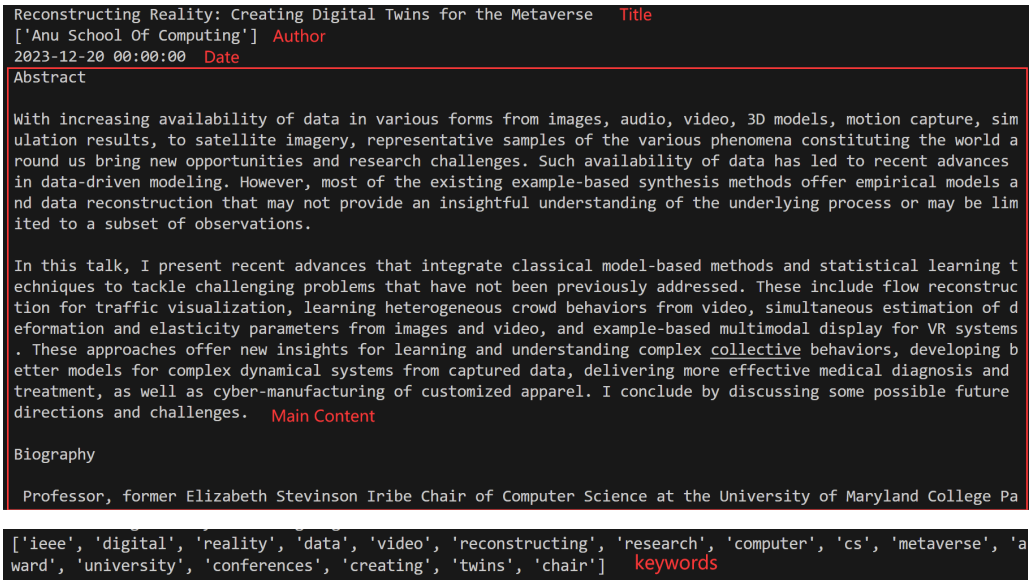
(Figure 6, invoice data extraction) (Kalra, 2023)

The second model is a natural language processing model that firstly converts HTML to plain text and then extracts title, description, date, etc. through text classification. However, after downloading the well trained gpt-2 and Bert models from the well-known website “hugging face” and testing them, we found that the models cannot distinguish between title and description well, and even the extraction of presenters’ names does not perform well. Therefore, using off-the-shelf models is not feasible. If we train the model ourselves, it takes a long time to create the dataset, and we have to go through a long time of model selection, model fine-tuning and model training. Moreover, the accuracy of the trained model cannot be guaranteed. For the requirements of our system, it is important to automate as much as possible so that the maintenance staff can correct as few errors as possible in the crawled data. Then, keeping the crawled data accurate is the priority.

2.2 Using traditional NLP algorithms

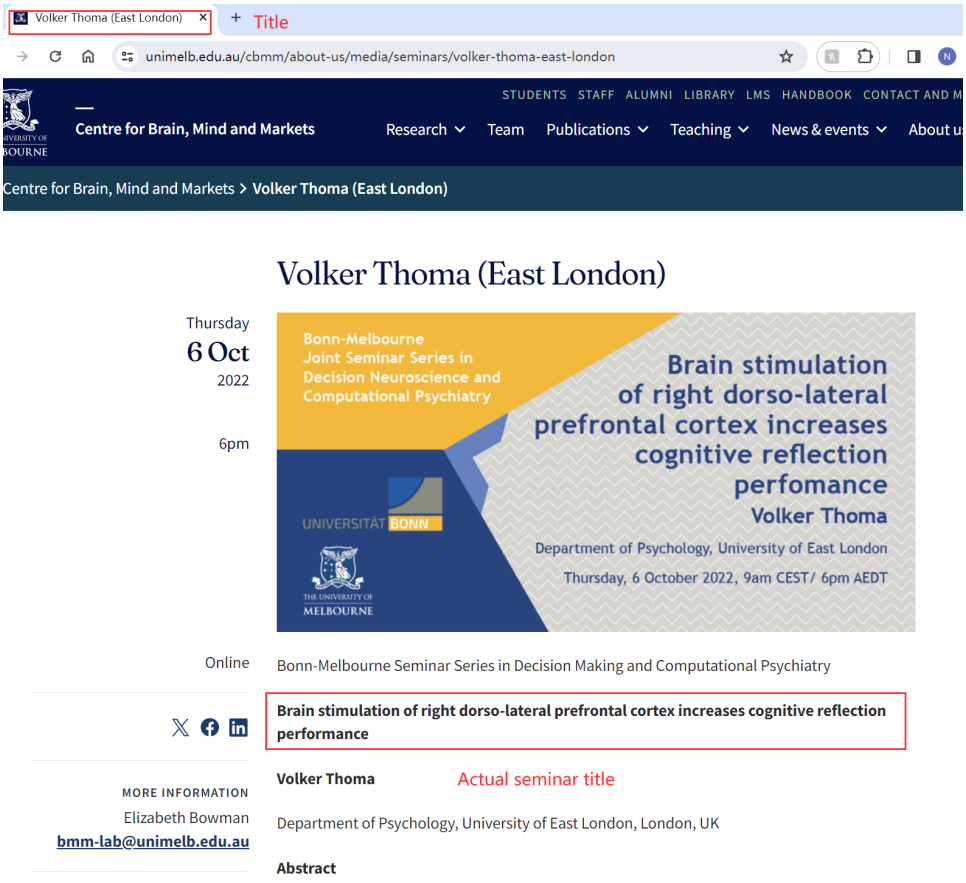
If we do not train models, using traditional NLP algorithms is also a path to design our own algorithm to achieve the optimal result. However, after our initial research, we found that there is a very well-known python information extraction library called newspaper3k (Ou-Yang, 2020), which performs functions very similar to what our system intends to accomplish. It also extracts title, content, author, article keywords

and so on. As you can see from the Figure 7below, it is very good at extracting title and main content for ANU seminars, but not so good at extracting seminar date and speaker.



(Figure 7, Newspaper3k information extraction result)

From the source code, it is because the programme gives priority to the meta data or specific HTML tags such as <title></title> in the web page to match the title, author and publish date rather than the regular expression matching and filtering. These problems can certainly be solved by modifying the source code of newspaper3k, but there is always a trade-off between versatility and accuracy. an obvious example is the following, due to the structure of the ANU website, the title of a seminar is highly likely to be the title of the page, but the title of the seminar page at the University of Melbourne for example, is the seminar presenter.



(Figure 8, A randomly picked seminar page from the University of Melbourne) (Bowman, 2022)

If the html title tag is given a higher preference as newpaper3k default, the crawler will make massive errors when crawling the University of Melbourne pages, but the accuracy of title extractions will be reduced if regular expressions are used. Since the client clearly stated that the system might crawl lectures of universities other than ANU in the future, I decided to sacrifice some generality for accuracy under the weighing of pros and cons. At the end, scrapy crawler framework was chosen to implement the cornerstone part, and the details are explained in the **Methods** section.

Methodological Framework

1. Overall Methodology

The development of our product followed a comprehensive methodology grounded in principles of human-centered computing, design thinking, and participatory research. Initially, we conducted a thorough needs assessment through user interviews and meetings to understand the specific requirements and challenges faced by our target audience. Additionally, with a concept in Agile development I "played" different stakeholders in the project to explore their needs and ensured that the needs of all users were met (Jama Software, 2022). With well-defined high-level functional requirements, a variety of use cases and scenarios were progressively identified. Utilizing the design thinking framework as outlined by Interaction Design Foundation (2016), in the process of finding requirements, design thinking came in handy as it helped me to diffusely identify the problem and then convergently define the problem, then diffusely try out the solutions, and then to finalise and implement the most feasible solution. We engaged in iterative prototyping and testing, ensuring that each design iteration was informed by direct user and client's feedback. This process was complemented by participatory research techniques (Vaughn & Jacquez, 2020), which involved users as co-creators throughout the development cycle to ensure the final product was both user-friendly and functional. By integrating these methodologies, we aimed to create a platform that not only met the technical specifications but also resonated with and addressed the real-world needs of its users.

2. Evaluation

2.1 Evaluators:

Five clients, including the project supervisor, were involved in the evaluation of the project.

2.2 Approach

Weekly meetings with the supervisor and monthly meetings attended by all clients would evaluate the progress of my project, decide on important parameters and algorithms as well as suggest improvements. The final all-hands meeting before the project deadline will assess the aesthetics, usability, and efficiency of the finished project.

3. Ethics Statement

This project targeted quality improvement of the client's existing workflow related to advertising seminars, building professional networks, and nurturing an inclusive transdisciplinary community. As such, ethical approvals related to research involving human participants were not needed. However, paid staff members from the JEI were involved in this project from its conceptualisation and design to its software development, evaluation, and enhancement. These staff members were the project client, and they conducted this work voluntarily as part of their normal paid work at the ANU.

The project team also collaborated with the ANU CIO and his team in this project to make sure that, crawling activities of this project would not be terminated by their security screens.

Modules and Implemented Methods

To facilitate project aims, an analysis and several meetings were conducted at the beginning to determine the number of modules, the partition of modules, the dependencies among modules and the develop sequence of modules. The analysis illustrates that there should be 7 modules, which are crawling seminar data, processing data, data persistence, the seminars display website, data management application, seminar recommendation system, and email generation and delivery systems.

1. Crawling seminar data

In order to obtain presentations about neuroscience seminars, one first needs to go to the event page of many different ANU departments to crawl the seminar data. After carefully consideration, an open-source web crawling framework, Scrapy, was adapted in this project. Scrapy has many advantages fitting right in with the character of the project.

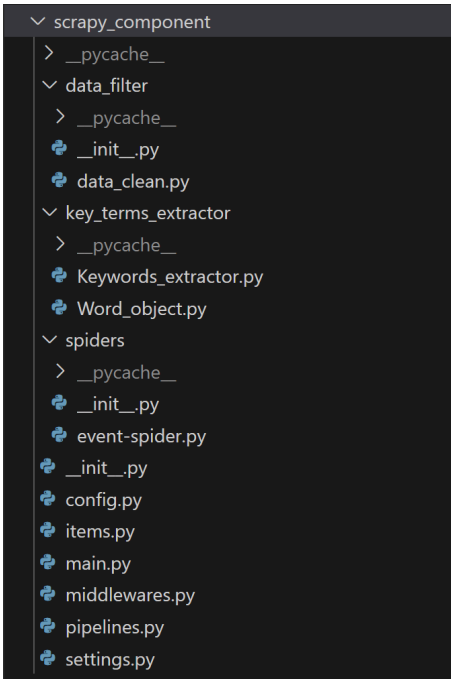
Firstly, scrapy provides user-agent masquerading functionality, which in initial tests allowed data to be scraped when permission to scrap was not yet obtained from the administrator. This feature also prevents the crawler from being banned by the server's protection system by allowing users to write some code used to disguise or invoke proxy services in the given middleware file. Secondly, scrapy provides massive wrapper methods for sending requests and extracting data, such as **`start_request()`** and **`parse()`** methods. Thirdly, scrapy provides pipeline functionality, through the given **`pipeline.py`** file, we can specify the data persistence policy, to persist the crawled data locally or in the database. Fourthly, scrapy supports common HTML selectors and XPATH to help users target the information they need to extract.

Figure 9 is a configuration file at path **`scrapy_component/config.py`**. The maintainer can add more faculties' seminar websites by adding required XPATH information to it. XPATH is a W3C recommended query language using path expressions to navigate through and select elements in XML or HTML documents (W3schools, 2024). Detailed instructions for adding a new URL are in the Readme.md under the root directory under the heading **"How do you add a new target website? "**

(Figure 9, Screenshot of config.py)

The main code related to web scraping and data processing is contained within the `**event-spider**` file. The web crawler retrieves details from each event page on the homepage of every department, based on the information provided in the configuration file. Much key information can be extracted using XPATH or directly by the process, such as title, time, venue, crawl time, image URIs and web page URLs. However, the XPATHs of much information are not fixed and must be processed to obtain, for example, the description of the event, the event keywords, the speaker, the organisation of the speaker and whether the event is a seminar or not.

The crawling time and frequency can be managed by editing the source code in `**schedule_execute.py**` under the root directory. At the same time, this file is also the general entry point of the whole crawler system. There are two modes when executing, default and scholarly modes, that will directly affect the algorithm in processing seminars' speakers. The terminal command to run the default mode is `**python schedule_execute.py**` or `**python schedule_execute.py -d**`. The terminal command to run scholarly mode is `**python schedule_execute.py -s**`. See section 2.3 for more information. I implemented the regularly timed crawling function through the methods provided by the `**schedule**` package and used the `**os**` package invoking the terminal to execute commands, to start the scrapy crawler named event-spider.



(Figure 10, The code structure of scrapy module)

2. Data processing

Event records can only be inserted into the database once the necessary information is obtained and standardized using the functions implemented in `**Data_Cleaner**` class. For directly extracted information use the XPATH extraction methods `**extract()**` and `**get()**` provided by scrapy. But for other information, they are handled differently.

2.1 Event description and event type

Although several paragraphs of the description are often stored in several HTML tags, it is still possible to extract all the textual information and merge it into a single string by extracting the parenting tag, followed by conducting regular expression search to decide whether the event is a seminar. If one or of the predefined keywords appears in the description, then it is categorized as seminar. The keywords are *'seminar', 'talk', 'lecture', 'abstract', 'speaker', 'presenter', 'biography', 'bio'*. Another regular expression filters out event abstract as the description based on keywords such as *'Abstract', 'Biography'* and their synonyms. If the description does not contain a similar keyword, then a string containing the textual information is returned directly.

2.2 Keywords

Key phrases are extracted from event description by the C-value algorithm which assesses term importance in text by considering occurring frequency, distribution across sections, and co-occurrence with other terms, to identify key terms that represent

the main themes or topics discussed in the corpus. It prioritizes frequently occurring terms in specific sections or across documents, aiding tasks like text summarization and information retrieval. Although exhaustive evaluation was conducted to determine the best hyperparameters, if any user is not satisfied with the performance of key phrase extraction, it can be adjusted by altering the following four hyperparameters defined in settings.py (see Figure below for details).

```
# Setting for the key terms extractor
# LINGUISTIC_FILTER: the linguistic filter, can be Noun or AdjNoun or AdjPrepNoun
# MAX_LEN: the expected maximum length of a term
# FREQUENCY_THRESHOLD: the frequency threshold. If the number of occurrences is lower than this threshold,
#                      it will not be considered as a key term.
# C_VALUE_THRESHOLD: the C-value threshold

LINGUISTIC_FILTER = "Noun"
MAX_LEN = 2
FREQUENCY_THRESHOLD = 0
C_VALUE_THRESHOLD = 1
```

(Figure 11, Screenshot of hyperparameters of C-value algorithm)

2.3 Speaker and its organization

2.3.1 Getting the name of the speaker

If the page has a fixed field to display the speaker's name, the speaker is extracted directly by XPATH, but if the speaker's name is not displayed in a fixed position, it needs to be extracted from the description and title. Firstly, we extract the biography paragraph in the description by regular expression and concatenate it with the title. Then it is converted into an object of type Doc via the NLP library spaCy, which has tokenised entities. We loop for each entity to determine whether its lexical nature is a person's name. If it is a person's name appearing for the first time, then we will add it to the dictionary; if it is a person's name that has appeared before, then we will increment all the keys in the dictionary which contain the person's name by 1. When we add a new person's name to the dictionary, the following algorithm will be used, name with two words will be assigned to a high value to outstanding other interfering items. The highest value in the dictionary will be returned as the speaker's name.

2.3.2 Getting the interests of the speaker

There may be times when a seminar's keywords may not be relevant to a subscriber's interests. Depending on the needs explicitly requested by the client, a seminar still needs to be recommended to the subscriber if the seminar's speaker's field of study is relevant to the subscriber's interests. Therefore, the speaker's interests needed to be captured somewhere, and

after discussions with the client, we decided to capture the speaker's interests from Google Scholar, subject to ethical and privacy policies being met.

In scholarly mode, Scholarly package was used, which enables the retrieval of author and publication details from Google Scholar in a convenient manner, eliminating the need to solve CAPTCHAs (Scholarly, 2023). When the program gets the speaker's name it searches for it using the **`search_author()`** function defined in the Scholarly library, and if there are no scholars with duplicate names, then it returns the scholar's interests and their organisation parsed from organizational email. If there are renamed scholars, then the top 15 scholars will be added to the list in the search default order, and if there is an ANU scholar among them, then the scholar will be returned directly. If there are no scholars with ANU then take the top two scholars on that list and compare the similarity between their research areas and the keywords for that talk via spaCy, returning information about the scholar with the higher similarity.

In default mode, or when scholarly throws certain exceptions during the crawl, the following algorithm is applied. Much like the extraction of scholars' names, first we extract the biographical paragraph in the description via regular expressions. This is then converted to an object of type Doc by spaCy. We loop over each entity and if its lexical nature is organisation, and we count how many words from that entity appear in the list of organisational keywords, storing the entity text and the result as a key-value pair in a dictionary. The organisational keywords contain *"University", "College", "Uni", "U", "Institution", "of", "Institute"*. Finally, the function returns the entity text with the largest value in the dictionary as the speaker's organisation.

3. Data persistence

3.1 Database Selection

Because data is not crawled and used at the same time, data persistence is necessary in the system. There are two types of databases, relational, and non-relational databases. Relational databases are good at storing structured data of predictable size and growth rate. Non-relational databases are good at handling unstructured data with complex web of relationships and unpredictable growth rate. Lecture requires time, place, speaker, title and abstract. speaker and recipient require name,

email, interested field and so on. Since the information required for lectures, speakers and recipients is deterministic, i.e., structured data. There is also a relationship between the speaker table and the lecture table, so a relational database is the most suitable for our application.

The reason for choosing PostgreSQL over other relational databases such as Oracle and MySQL, is that it is a free and open-source application compared to Oracle. Compared to MySQL, it has better read operations and concurrency performance and can handle data storage read and write faster.

3.2 Storage mechanism and tools

The mechanism for storing records is implemented in the scrapy framework in a file called pipeline.py. The pipeline is a component used for processing scraped data. Its purpose is to handle, clean, store, or perform other operations on the scraped data for further use or display. Specifically, the pipeline in this project is used for storing scraped data into databases, files, or other data storage mediums for later analysis or presentation.

The specific usage of pipelines is to define one or more pipeline classes in your Scrapy project and enable them in the settings.py file. Each pipeline class needs to implement a set of methods, including the `**process_item()**` method that processes each scraped item. When the spider scrapes data, it passes through the configured pipeline one by one, and each pipeline class processes the data until it is finished or discarded.

The code written in the pipeline class uses SQLAlchemy, which helps me connect to the database as well as assemble SQL statements. SQLAlchemy is a Python library used for managing and interacting with relational databases. It offers a highly flexible Object-Relational Mapping (ORM) tool, allowing developers to represent database tables and rows using Python objects, simplifying database operations. Main functions of SQLAlchemy include ORM, database connection management, query capabilities, and schema management. Overall, it provides powerful and flexible tools for Python developers to simplify relational database operations and build and maintain database applications more easily.

3.3 Database administration and configuration

The drivers, database name, login username and password, port, and host can be changed by editing `**CONNECTION_STRING**` in `**settings.py**` in the scrapy component module.

```
# Set settings whose default value is deprecated to a future-proof value
REQUEST_FINGERPRINTER_IMPLEMENTATION = "2.7"
TWISTED_REACTOR = "twisted.internet.asyncioreactor.AsyncioSelectorReactor"
FEED_EXPORT_ENCODING = "utf-8"

CONNECTION_STRING = "{drivername}://{user}:{passwd}@{host}:{port}/{db_name}".format(
    drivername="postgresql+psycopg2",
    user="postgres",
    passwd="postgres",
    host="localhost",
    port="5432",
    db_name="jei",
)
```

(Figure 12, Screen shot of Database configuration)

3.4 Table design

The figures below show the **Event** table, the **Recipient** table, and the **Scholar** table. In addition to the intuitive columns, it can be noticed that the speaker column in the Event table is a foreign key column, and the stored data is the primary key of the Scholar table. Another worth noting feature is that the date column in the Event table is of type **Text()**, and the standard_datetime column is of type **DateTime**. As Text() type date is not standardized and cannot be compared to utilize sorting and ordering, so the process of standardization is mandatory. The text data in the date column is processed into DateTime by a function in the `**Data_Cleaner**` class for subsequent ordering purposes.

```
class Event(Base):
    __tablename__ = "event"
    id = Column(Integer, primary_key=True, autoincrement=True)
    title = Column(Text())
    description = Column(Text())
    date = Column(Text())
    venue = Column(Text())
    speaker = Column(Integer,ForeignKey("scholar.id", onupdate="CASCADE",ondelete="CASCADE"))
    keywords = Column(Text())
    organization = Column(Text())
    url = Column(Text())
    image_url = Column(Text())
    access_date = Column(DateTime(timezone=True))
    standard_datetime = Column(DateTime(timezone=True))
    is_seminar = Column(Boolean(),default=False)
```

(Figure 13, Event Table)

```
class Recipient(Base):
    __tablename__ = "recipient"
    id = Column(Integer, primary_key=True, autoincrement=True)
    name = Column(Text())
    email = Column(Text(),  unique=True)
    interest = Column(ARRAY(Text()))
    organization = Column(Text())
    is_recipient = Column(Boolean(),default=True)
```

(Figure 14, Recipient Table)

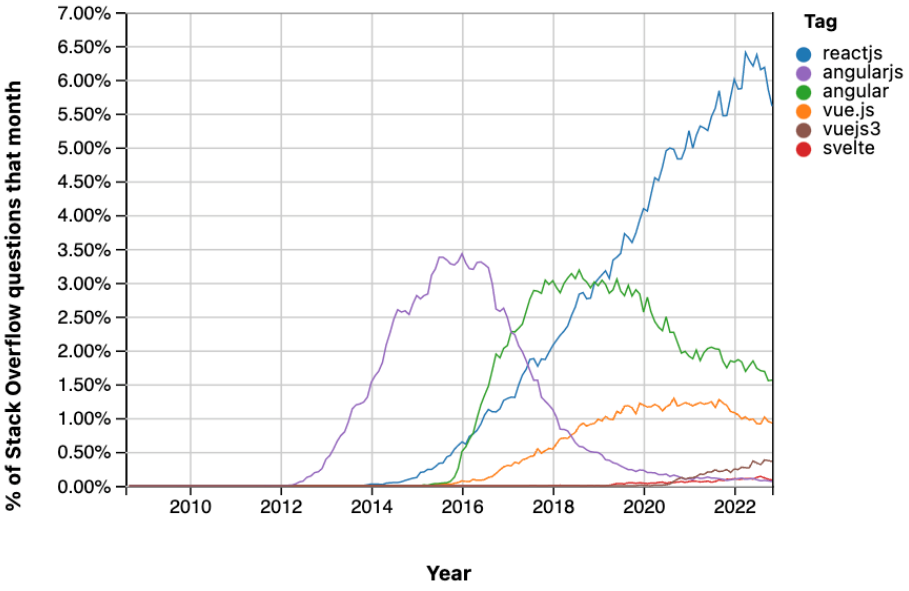
```python
class Scholar(Base):
    __tablename__ = "scholar"
    id = Column(Integer, primary_key=True, autoincrement=True)
    name = Column(Text())
    google_scholar_id = Column(Text())
    interest = Column(ARRAY(Text()))
    organization = Column(Text())
    events = relationship('Event', backref='scholar', lazy=False)
```

(Figure 15, Scholar Table)

4. Website display

4.1 Frontend

Next.JS was used to implement frontend. Next.JS is an officially recommended production-grade React.JS framework that support all the necessary features for deploying and scaling the application in production (Facebook, 2023). The reason for choosing it is that ReactJS is still the most used front-end development framework in the world with a mature community and many libraries and toolset (Krotoff, 2023).



(Figure 16,Front-end frameworks popularity) (Krotoff, 2023)

Besides, I have previous experience in ReactJS development. But the most essential reasons for choosing it are threefold - flexibility, reusability, and performance. Firstly, React JS has good flexibility, a page consists of multiple React components and a change in one of the components will not affect the whole

application. Secondly, React JS has good reusability, developers encountering to develop similar functionality of the module can easily transfer or port the code, and because of the code architecture within component is clear, altering code is not easy to make mistakes. The third reason is good performance, its core provides a virtual DOM program and server-side rendering, part of the components will be server to perform render this time-consuming processing (Prad, 2023). Instead of letting the client perform rendering, server will send a fully rendered HTML page to the client. It ensures fast page display even when the client's browser or computer hardware has limitations. For the front-end CSS, in keeping with the ANU style, I directly downloaded the CSS file used for the ANU College of Engineering, Computing and Cybernetics.

The subscribe to email button was implemented entirely on the front-end, it opens the user's default email software and automatically fills in the title, destination email address and content. All the user needs to do is follow the prompts and fill in their information.

4.2 Backend

For the backend of the site, I chose a combination of python and FastAPI. FastAPI is a contemporary and rapid web framework utilized for constructing APIs in Python, leveraging standard Python type hints (Tiangolo, 2024). The reason is that the main programming language used for the entire data-related functionalities is python, and the syntax and methods of FastAPI are easy to understand for those skilled in python. In addition, it has a good runtime efficiency due to asynchronous features and effective request management (Simplilearn, 2023).

The main functionalities of the backend include:

4.2.1 Fetching upcoming seminars from the database

4.2.2 Searching for upcoming seminars based on keywords

The aforementioned two functionalities were implemented by using SQLAlchemy's API to send SQL queries to the database and retrieve data. For more details on the search mechanism, please refer to Section 3.3 on Elastic Search.

4.2.3 Ability to set and fetch users' custom interests.

4.2.4 Selecting multiple seminars via tick boxes and displaying only the selected seminars.

The implementation of these two functionalities utilizes cookies, which are

small text files stored in the browser. This feature ensures that user-entered interests and selected seminars, stored on the browser, can be retrieved when the website is reopened. If more than one interest is to be filled in, they need to be separated by commas, e.g. "Maths, Astrophysics". It is also important to note that if using localhost in the testing, the cookie will not work because it cannot be set on a domain name of 'localhost'. By convention, the domain name must have at least two dots or the browser will consider it invalid (Buchfelder & Konkov, 2021). When 'localhost' is used as the domain name, the cookie must be completely omitted. **Therefore, we should only conduct local tests on the domain `127.0.0.1`.**

4.2.5 Arranging seminars in desired order

Seminars are returned in ascending order by date, with the nearest ones first. Unless users input specific interests and there are seminars highly relevant to those interests, in which case, these seminars will be prioritized on the page based on relevance. For other seminars, the default order was maintained.

The default display order is set using SQLAlchemy's API functions **`order_by()`** and **`asc()`** to specify the **"ORDER BY"** and **"ASC"** keywords in the SQL query, sorting the data in ascending order based on the **`standard_datetime`** column in the **`Event`** table. After the default sorting, seminars calculated by the recommendation system are moved to the top of the list. For further details, please refer to Section 5.5.

4.3 Elastic search

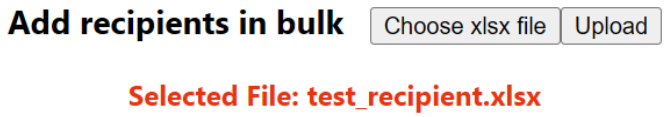
Elastic search functionality was enabled by PostgreSQL database's built-in full text search. Firstly, a "tsvector" column is added in event table after table creation. In PostgreSQL, tsvector is a data type used to store text search vectors. It's specifically designed for full-text search operations and is part of the PostgreSQL's full-text search functionality. Full-text search is used to search for words or phrases within text documents efficiently.

5. Data management application

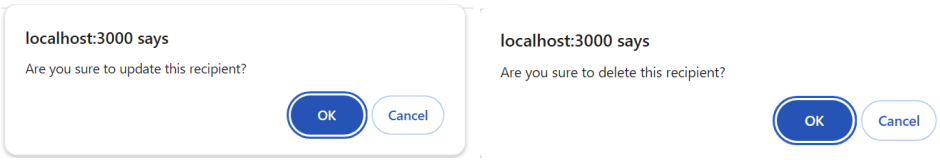
A management program is required to help maintainers to manage information of recipients and seminars, and to generate and send emails. I decided to develop a web-

based software solution because web-based software can be executed cross-platform regardless of the local operating system. It can run based on a supported browser. In addition, web-based software can also reuse the React framework to build pleasant graphical interfaces. the benefits of React are explained in section 4.1. The whole front-end design is based on simplicity and ease of use, basically not set any decorative CSS. But the necessary pop-up prompts and messages (Figure 17-19) were still implemented to warn the users of dangerous operations or inform them of the success or failure of operations, as well as the reason for the failure.

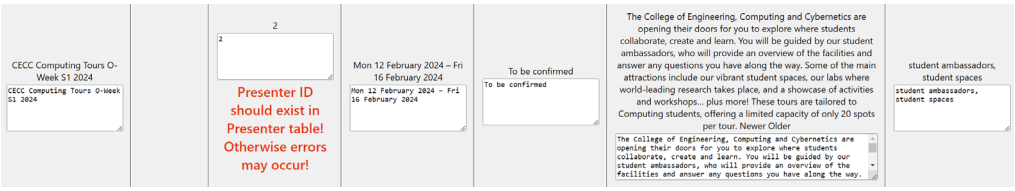
The search function is implemented in the front-end and is not associated with the database, but due to the large number of event columns, if all the columns can be searched, it will affect the efficiency of the front-end's response, so only the title, speaker, event, location, and keywords can be searched.



(Figure 17, Prompt of Selected File)



(Figure 18, Warning Windows of Dangerous Operations)

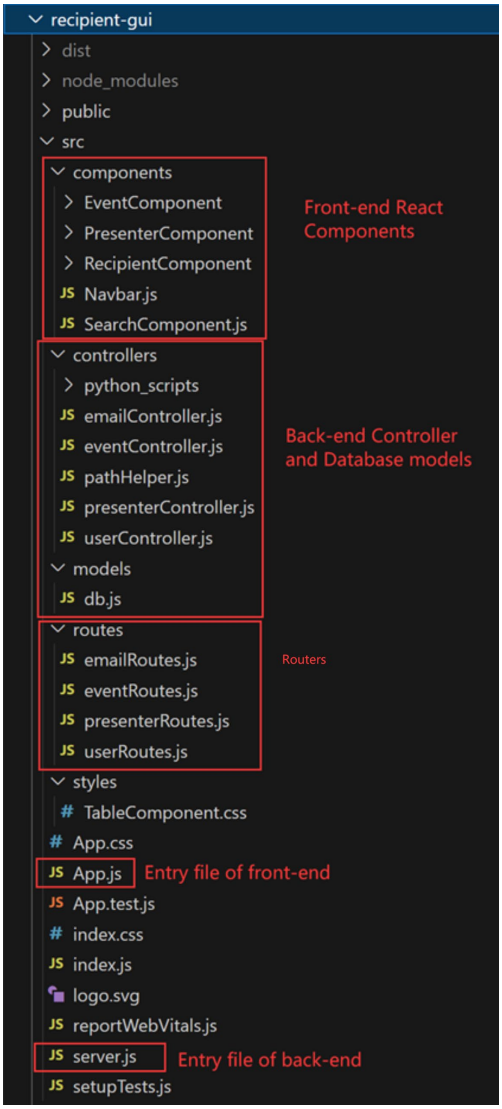


(Figure 19, Warning Message of Dangerous Event Updating Operation)

For the backend, I chose the same programming language as the front-end, JavaScript, for ease of development and maintenance, invoking python scripts only when necessary. For example, some functions require the use of Scholarly, and since Scholarly is a python library, it is more convenient to call python scripts directly to execute them. In order to create server by JavaScript, an environment to run JavaScript was needed, and Node.js and Express framework were chosen without question, because I had much

experience with NodeJS and Express projects. In addition, NodeJS is also the most famous JavaScript runtime environment, the core is derived from Google Chrome browser, with excellent efficiency (Kumar, 2023). Express framework is a minimal and flexible web application framework running on NodeJS, with many robust features (Express, 2024) The detailed structure of the application can be seen in Figure 20.

When the fetch interests button is clicked, the `**interests_loopup.py**` script is run with different parameters to determine whether to fetch the interests of the recipients or the speakers. The script will only fetch objects with null interests.The search is performed using the `**search_author()**` function defined in the Scholarly library, which searches for both names and organisations, and if there are no duplicate scholars with duplicate names, returns the scholar's interests. If there are scholars with duplicate names, the first 15/10 scholars are added to the list in the default order of the search results. The list of scholars is capped at 15 if looking for speaker interests, or 10 if looking for recipients. If there is an ANU scholar in the list, the scholar is returned directly. If there are no ANU scholars in the list, the default top scholar is returned.



(Figure 20, Code Structure of Data Management Application)

For retrieval, creating, deleting, and updating records in the database, a npm library `**pg**` was used, which, along with properly configured database information, makes it possible to execute written SQL commands in the database, as shown in Figures 21 to 22.

```
const { Pool } = require('pg');
const pool = new Pool({
    user: 'postgres',
    host: 'localhost',
    database: 'jei',
    password: 'postgres',
    port: 5432,
  });

module.exports = {
    pool
}
```

(Figure 21, Screenshot of database settings)

```
const result = await pool.query(
  'INSERT INTO recipient (email, name, organization, interest, is_recipient) VALUES ($1, $2, $3, $4, $5) RETURNING *',
  [email, name, organization, interestArray , true]
);
```

(Figure 22, Typical Pure JavaScript SQL Query in The Application)

Another important feature is the bulk addition of email recipients. A standard structured Excel file is required to be uploaded, after which the file is stored with the filename 'uploaded_recipients_info.xlsx' in the root directory in the 'static/recipient_list' folder in the root directory. After that the program will convert the Excel file to JSON and add it to the database row by row, if any problems occur then it will report error messages, please see 'Recipients, Presenter, and events management system' under Result Section for the specific error reporting image. Figure 23 is a standardised Excel file that should have only one default sheet and the following four columns: name, email, interest, and organisation. Only the Interest column is not mandatory, the rest are required. If more than one interest is to be filled in, they need to be separated by commas, e.g. "Maths, Astrophysics".

name	email	interest	organization
Tom	tom@gmail.com		anu
Kate	kate@gmail.com		unimelb

(Figure 23, Valid Structure of Excel File)

The implementation behind the 'Verify Email' and 'Send Email' buttons is described in Section 7.

6. Seminar recommendation system

Providing tailored seminar recommendations to different recipients is an important module in the system. The implementation of this function is based on spacy. Spacy is an industrial-strength open-source library for performing natural language processing (NLP) tasks in Python (SpaCy, 2024). It provides tools and resources for tokenisation, part-of-speech tagging, named entity recognition, dependency resolution, and more. The program will firstly query all seminars from the joint tables of seminars and scholars. This step will not only get the keywords of a particular seminar, but also the fields of specialisation of the presenter of that seminar, then the program would query all the recipients from the database. After that, the program loops through each seminar and for each seminar loop through its keywords and presenter's fields of specialisation. Spacy provided functions calculate the similarities with each interest of the recipient, as long as any of the calculated similarity value is greater than the threshold preset in the code, then this seminar is added into the recipient's list. The recipients and corresponding lists will be packaged as key-value pairs and stored within a dictionary. The dictionary will be returned as the result. The dictionary will also be used in emails generation and delivery module.

`en_core_web_lg` model was used in calculating similarities and determining recommended seminars. It is one of the core models provided by the SpaCy library. Optimized for CPU, this model is designed for processing web-based text, including blogs, news articles, and comments (SpaCy, 2024). It comprises various components such as tokenization, tagging, parsing, named entity recognition (NER), and a lemmatizer (SpaCy, 2024). Additionally, it includes word vectors with 514,000 unique keys, each represented by a 300-dimensional vector. This model is suitable for a wide range of natural language processing tasks, including part-of-speech tagging, syntactic parsing, named entity recognition, and word vector representations (SpaCy, 2024).

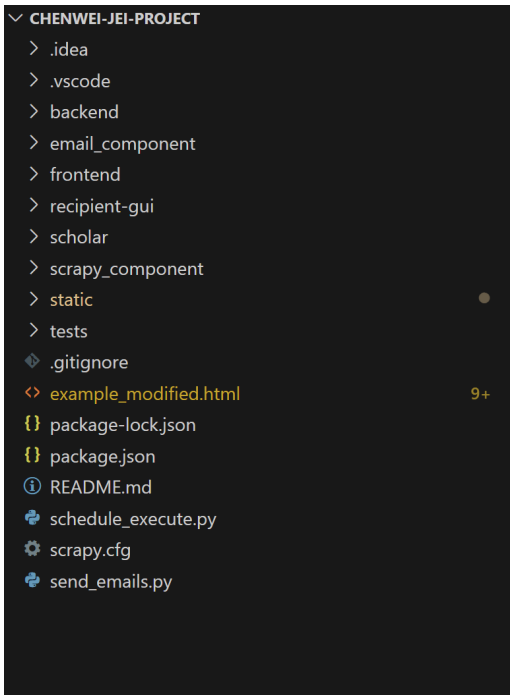
7. Email generation and delivery system

Since most of the companies online that offer platforms for generating and sending out charge for their services, I chose to use a native smtp library in order to save costs, as well as to reduce user privacy disclosure.

After knowing which seminars to be sent to which recipients, the next step is to do the final proofing and send the email. As it is inevitable that some errors will occur during the process of crawling and processing data, it is necessary to generate a verification email and check it before officially sending the email. The functions for generating the checksum email and sending the email can be executed either through the terminal or through the management tool's buttons.

When generating a validation email, all events crawled in the interval before the current time are added to the html template of the email. The default time interval is 7 days. Here is a concrete example, if the current time is 8th January 2024, then the program would add seminars crawled between 1st January and 8th January into the validation email.

After checking the verification email, the administrator can then proceed to send emails by click button offered in management system. The email sender is pre-set in the code to a dedicated Gmail functional account. The logic of sending email is that, for each recipient, replacing the recipient and time tags in HTML template by the current recipient and current time. Then, the seminar objects obtained from the seminar recommendation module are encapsulated and appended to the HTML template. Finally using Gmail SMTP server to send it out.



(Figure 24, Location of Verification Email [example_modified.html])

Results

I have designed and implemented a seminar crawling, displaying, and advertising system for John Eccles Institute of Neuroscience. The main artifacts of this project were as follows:

1. A customer-satisfying website that collects all upcoming seminars and public lectures of a given college, with many useful features including sorting seminars by user preferences, bookmarking specific seminars.
2. A functional information management system capable of managing recipients and seminar information in a database.
3. A promotional email distribution system that filters out and sends subscribers seminars they might like based on their interests.

The code is deployed at https://gitlab.cecs.anu.edu.au/u7377070/john-eccles-institute-seminar-project/-/tree/main?ref_type=heads. By scrolling down on that page, the contents of the readme.md file can be seen, which contains guidelines for the user to install, deploy, use, and continue implementing better products in the future. If problems encountered with the readme.md display, please try opening the readme.md file in the root directory. If you are unauthorised and do not have access to the project source code, please contact me or my primary supervisor.

The participatory research methodology was applied during the development process by discussing with the clients during regular meetings so that they were actively involved in formulating the project envision, parameter values, and implementation strategies. All clients were satisfied with the aesthetics, usability, and efficiency of the project during the final evaluation meeting. Besides, members of John Eccles Institution have successfully adopted and used the tools and that those clients/supervisors from SOCO also examined the code and software system in more detail, repeatedly over the project timeline to support making it robust.

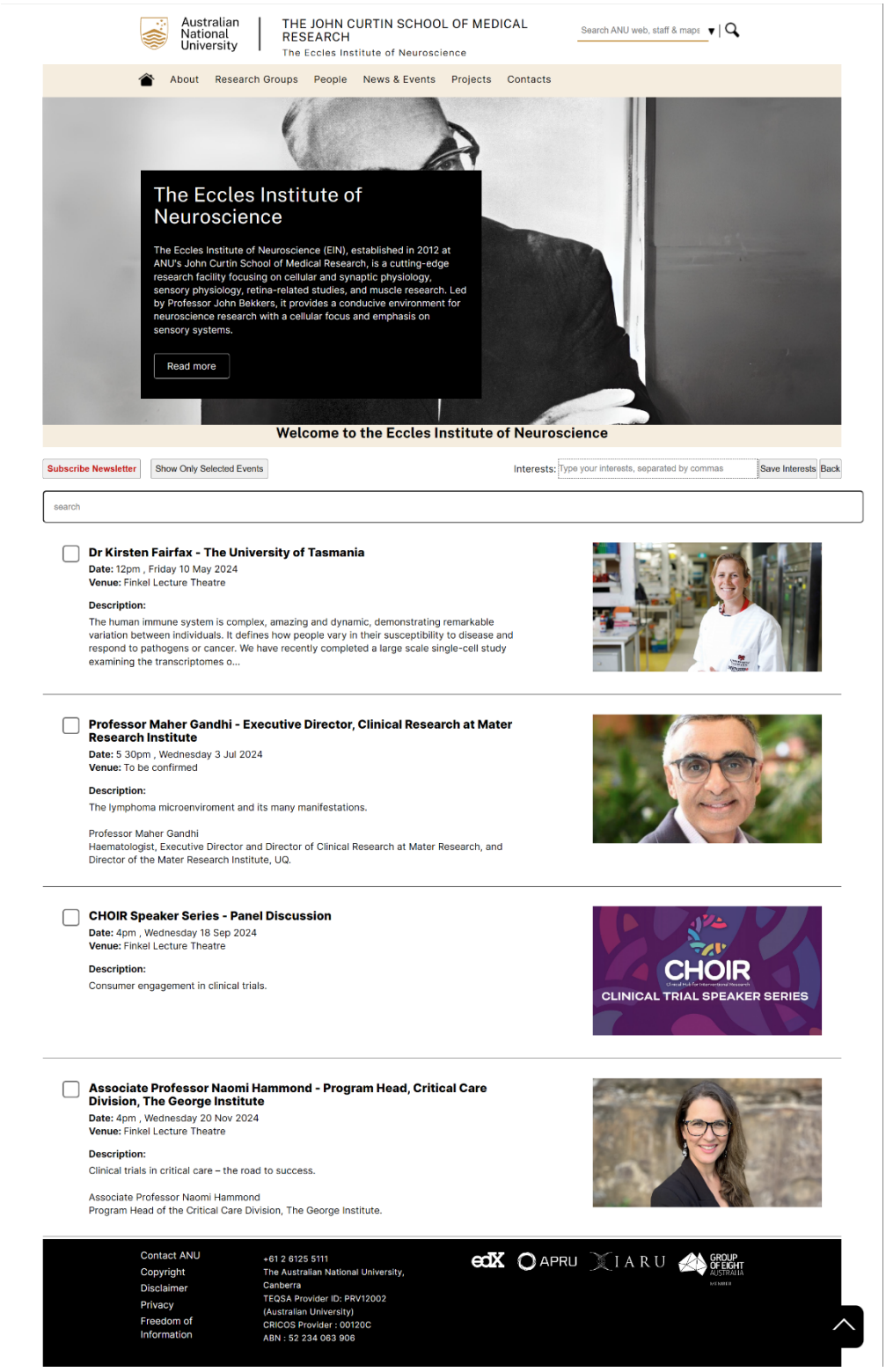
1. Website holding together seminars.

The current system enables the crawling functionality of three faculties at ANU, School of Computing, Medical school of ANU, and John Curtin School of Medical Research. The maintainer could add more crawling faculties by adding required information in the configuration file. Maintainer could also manage the crawling time and frequency by editing the source code in `**schedule_execute.py**`.

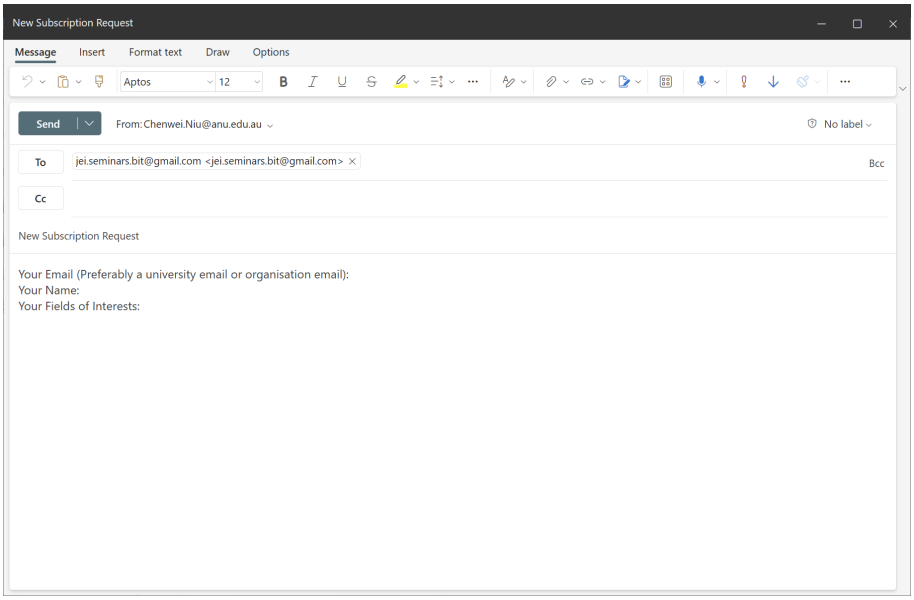
Most of the technical choices for the back end and front end of the site were done independently. But for some aspects, I consulted with 4 clients at two meetings in March and April to determine the available technical approaches for the seminar filtering functionality with users' selection, and whether to add images when displaying seminars instead of plain text.

As illustrated in Figure 25, the website is an appropriate ANU-style website in yellow and black themes and has the following core features:

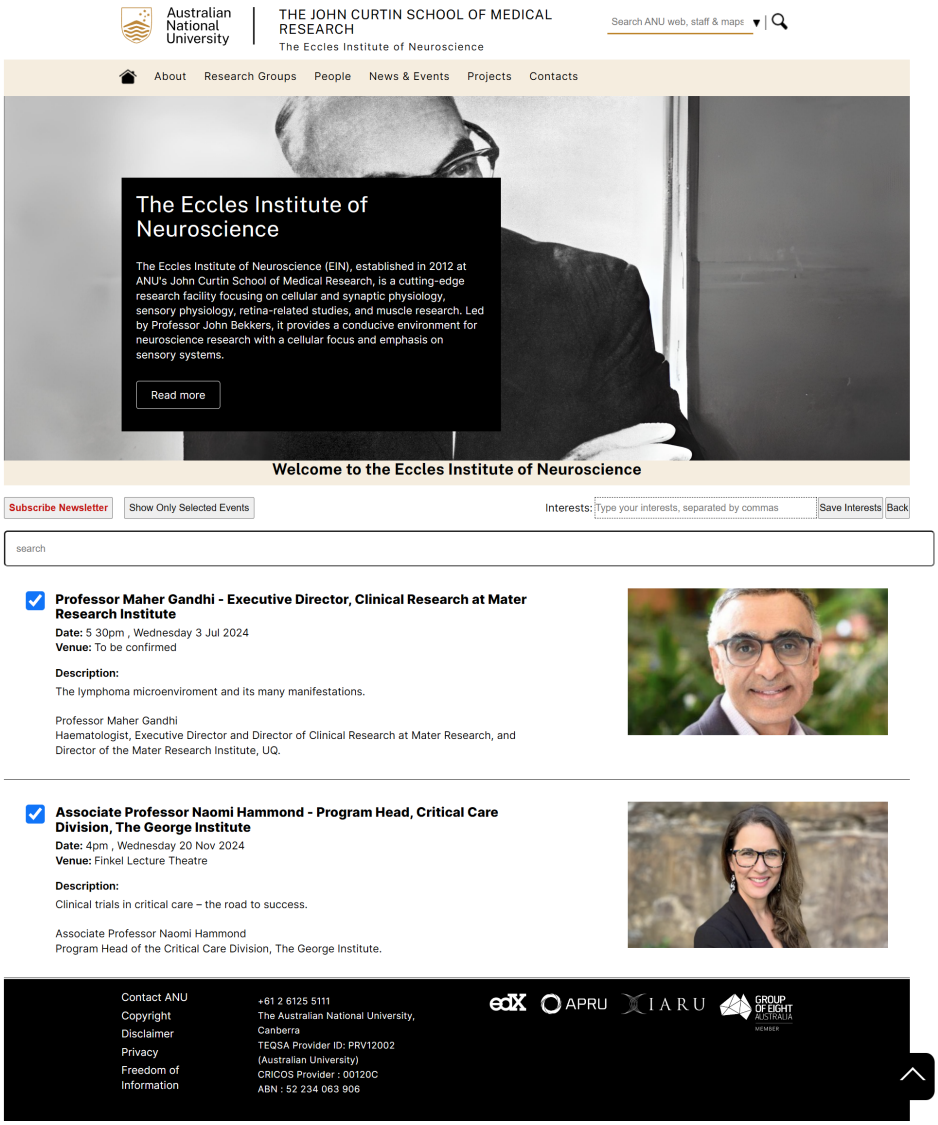
- A display of upcoming seminars ordered by date.
- Subscribe button, used to subscribe new users to the newsletter email.
- Display selected events (The browser remembers the events you have selected).
- Typed in interests, and the order of seminars customised by interests (The browser remembers the interests you have entered).



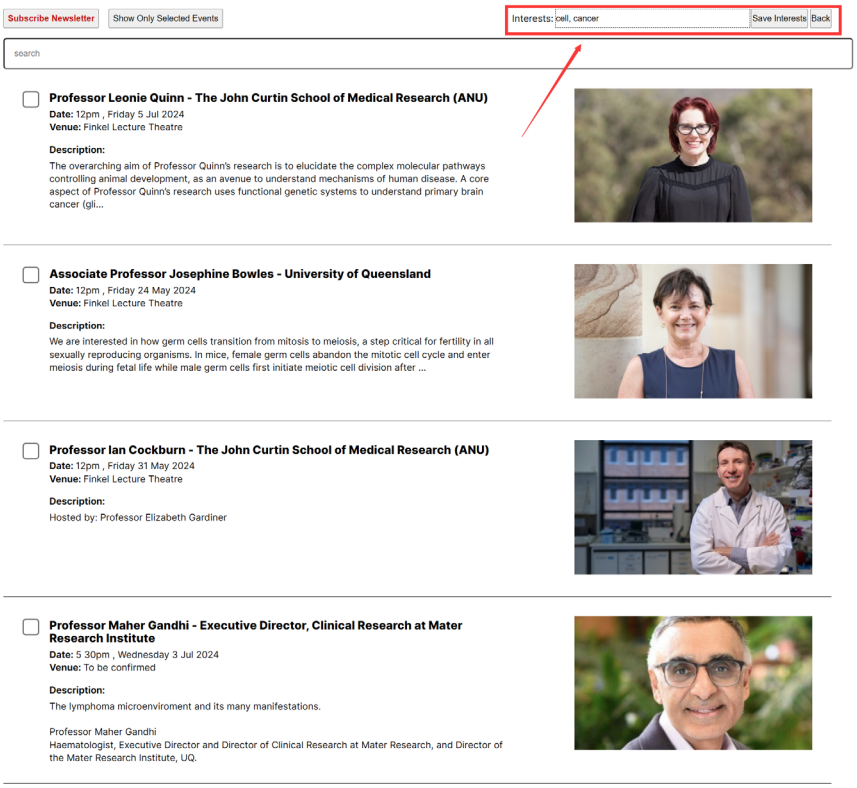
(Figure 25, Website Page)



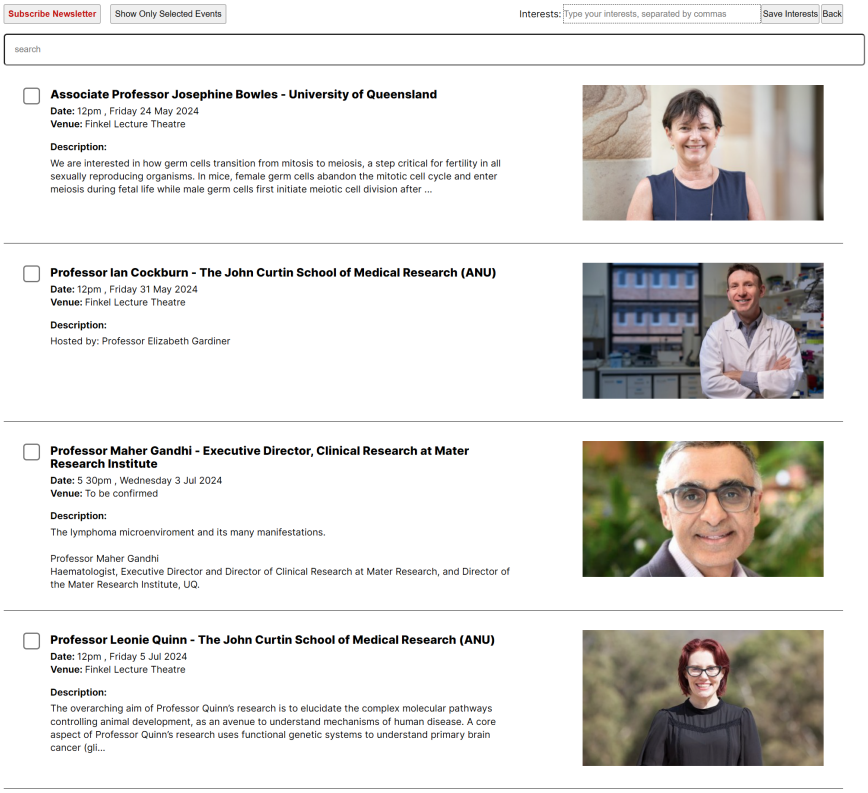
(Figure 26, Automatically generated subscription email)



(Figure 27, Show only selected events)

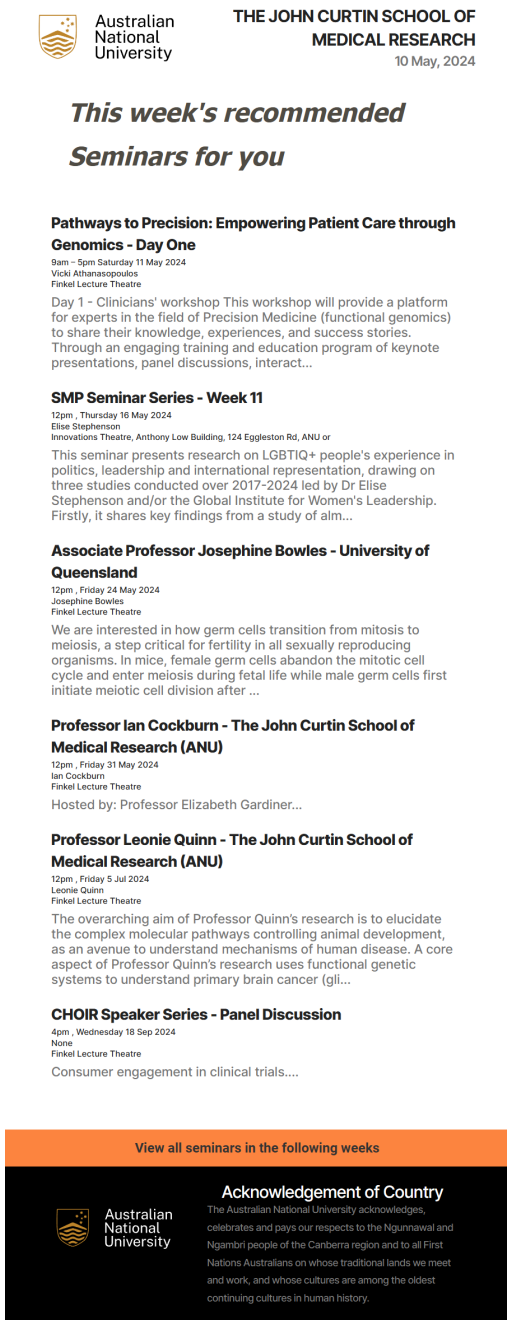


(Figure 28, Seminar order with typed in interests)



(Figure 29, Seminar order with no typed interests)

2. Customized seminar emails generation and distribution system


Australian National University

THE JOHN CURTIN SCHOOL OF MEDICAL RESEARCH

10 May, 2024

This week's recommended Seminars for you

Pathways to Precision: Empowering Patient Care through Genomics - Day One

9am – 5pm Saturday 11 May 2024
Vicki Athanasopoulos
Finkel Lecture Theatre

Day 1 - Clinicians' workshop This workshop will provide a platform for experts in the field of Precision Medicine (functional genomics) to share their knowledge, experiences, and success stories. Through an engaging training and education program of keynote presentations, panel discussions, interact...

SMP Seminar Series - Week 11

12pm , Thursday 16 May 2024
Elise Stephenson
Innovations Theatre, Anthony Low Building, 124 Eggleston Rd, ANU or

This seminar presents research on LGBTIQ+ people's experience in politics, leadership and international representation, drawing on three studies conducted over 2017-2024 led by Dr Elise Stephenson and/or the Global Institute for Women's Leadership. Firstly, it shares key findings from a study of alm...

Associate Professor Josephine Bowles - University of Queensland

12pm , Friday 24 May 2024
Josephine Bowles
Finkel Lecture Theatre

We are interested in how germ cells transition from mitosis to meiosis, a step critical for fertility in all sexually reproducing organisms. In mice, female germ cells abandon the mitotic cell cycle and enter meiosis during fetal life while male germ cells first initiate meiotic cell division after ...

Professor Ian Cockburn - The John Curtin School of Medical Research (ANU)

12pm , Friday 31 May 2024
Ian Cockburn
Finkel Lecture Theatre

Hosted by: Professor Elizabeth Gardiner...

Professor Leonie Quinn - The John Curtin School of Medical Research (ANU)

12pm , Friday 5 Jul 2024
Leonie Quinn
Finkel Lecture Theatre

The overarching aim of Professor Quinn's research is to elucidate the complex molecular pathways controlling animal development, as an avenue to understand mechanisms of human disease. A core aspect of Professor Quinn's research uses functional genetic systems to understand primary brain cancer (gli...

CHOIR Speaker Series - Panel Discussion

4pm , Wednesday 18 Sep 2024
None
Finkel Lecture Theatre

Consumer engagement in clinical trials....

View all seminars in the following weeks

Australian National University

Acknowledgement of Country

The Australian National University acknowledges, celebrates and pays our respects to the Ngunnawal and Ngambri people of the Canberra region and to all First Nations Australians on whose traditional lands we meet and work, and whose cultures are among the oldest continuing cultures in human history.

(Figure 30, Sample advertising email)

After each round of data crawling, the email system automatically generates a verification email containing all the seminars that could be sent out. Once the maintainer checked the correctness of information, they can run `send_email.py` or click the 'send emails' button in the management application to build and send customised recommended seminar emails to subscribers. Figure 30 illustrates a typical

advertising email including date, recommended seminars, and a link redirect to the homepage of the website.

Many important evaluations were made by the client's team during conceptualisation and development. However, one of the most important evaluations was about the threshold value of similarity for recommending seminars. I first determined a suitable range of 70-82.5 through personal experimentations. I consulted 5 client representatives on 4th May 2024 to set an appropriate threshold to filter in just the right the seminars. Each seminar data will generate 5 rankings related to the client representative's interests at different thresholds, 72.5, 75, 77.5, 80, and 82.5, from which the client representative will select the most correct ranking. The used seminar data were crawled on the 4th of March, the 12th of April, and the 4th of May, so each client representative received a total of 15 rankings, with 72.5 being the statistically best value. At this value, the seminars recommended to the recipients are a precise match to at least one of their interests and are neither too broad nor too narrow.

During the initial project meeting, an important client emphasized that this specific feature was essential for the project's success. However, the first version of this feature was being developed from October 3rd , 2023, due to the unfinished implementation of the crawler system. In the meeting on October 3rd 2023, while the necessity of sending promotional emails was clear, the design and implementation details were not defined. After careful discussion, it was decided to send ANU-style HTML emails to various recipients. Consultation with two clients revealed that emails should not be sent automatically; instead, administrators should have control over email dispatch to verify content accuracy and prevent errors that could harm JEI's reputation. There is another point worth mentioning, initially, the concept involved matching recipients' areas of interest merely with seminar keywords. However, as development proceeded, client feedback highlighted that seminar recommendations should also consider the speaker's expertise, leading to the requirement of crawling this additional information.

3. Recipients, Presenter, and events management system

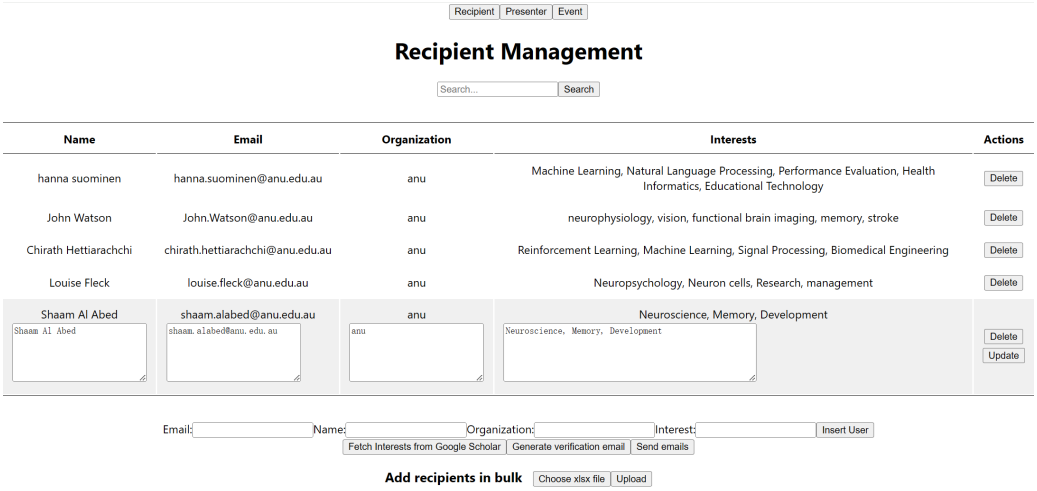
The system provides maintenance staff with a convenient management system that is capable of managing not only recipients and presenters but also events and offers a range of features.

1. A case-insensitive search function.
2. Record updating and deleting functions with helpful prompting windows and alerts.

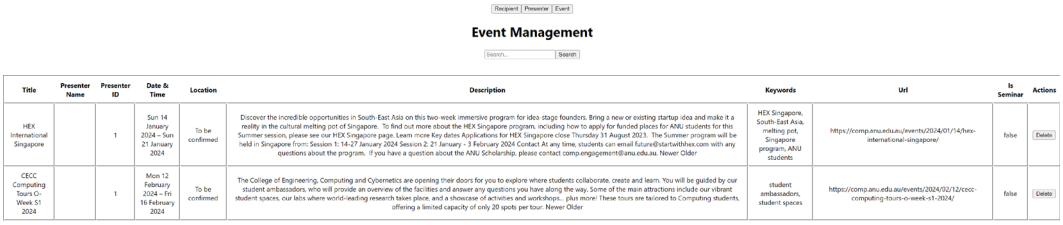
3. New recipients, presenters, and events insertion functions.
4. One-click button to retrieve the fields of specialization and interests for recipients or presenters whose interest data field is null. Data will be fetched from Google Scholar.
5. One-click button for maintainer to generate verification emails under the root directory.
6. One-click button to send personalized newsletters to recipients.
7. One-click button for maintainer to add recipients in bulk. Supported file format is described here.

It hasn't even been added to the requirements list until November 2023. In a meeting with the customer at the end of October 2023, the client had questions about the implementation of managing recipients, and the solution at that time was to use Excel and database table. The system would update the database according to the Excel table regularly. But if any exception arises, the state of the recipients table in database was unknown. So, I proposed to use a database visualisation application `**pgAdmin**` to manage database. However, in the following meeting, the client mentioned that it was complicated to install the `**pgAdmin**` under Linux environment, and the tool was difficult to use before getting familiar with it, and many of the corresponding buttons for operations were not obvious. Therefore, the requirement of graphical management system had emerged.

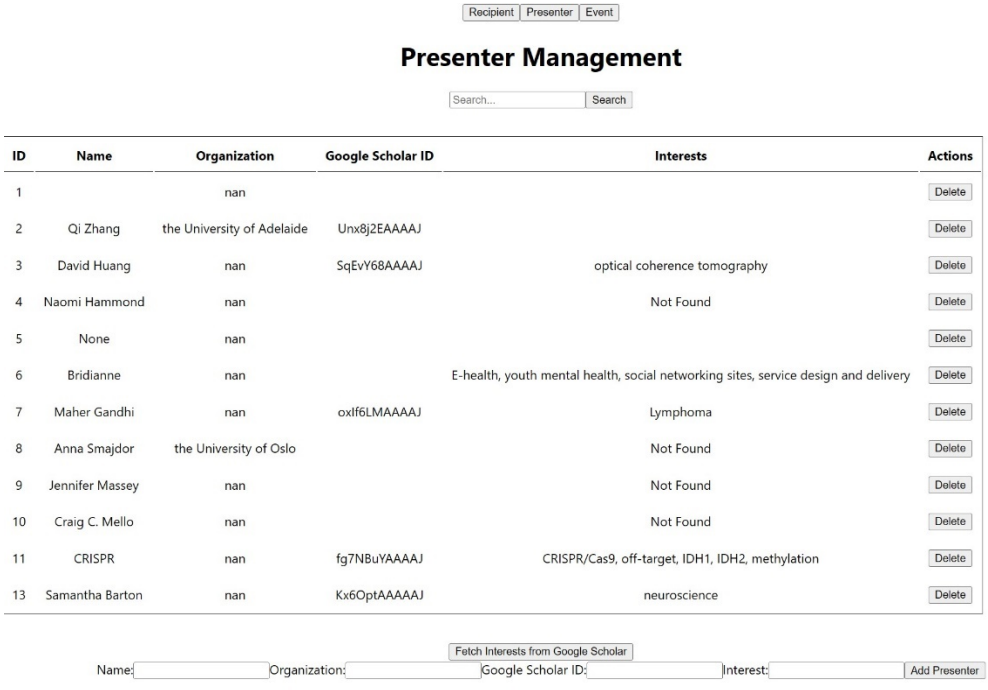
The first version of the management tool was a desktop graphical application using `**Electron**` libraries with separated installers for windows and Linux systems. At the regular meeting in January, the client thought that the installation package was too disk-consuming and complicated to install. After the discussion, we decided to change it to a web application, because web application has the advantage of cross-platform and lightweight. At the February meeting, the functionality was completed, and the client felt that there was still a need for seminar and speaker management, as well as the ability to add recipients in bulk. in March, all the functionality was implemented.



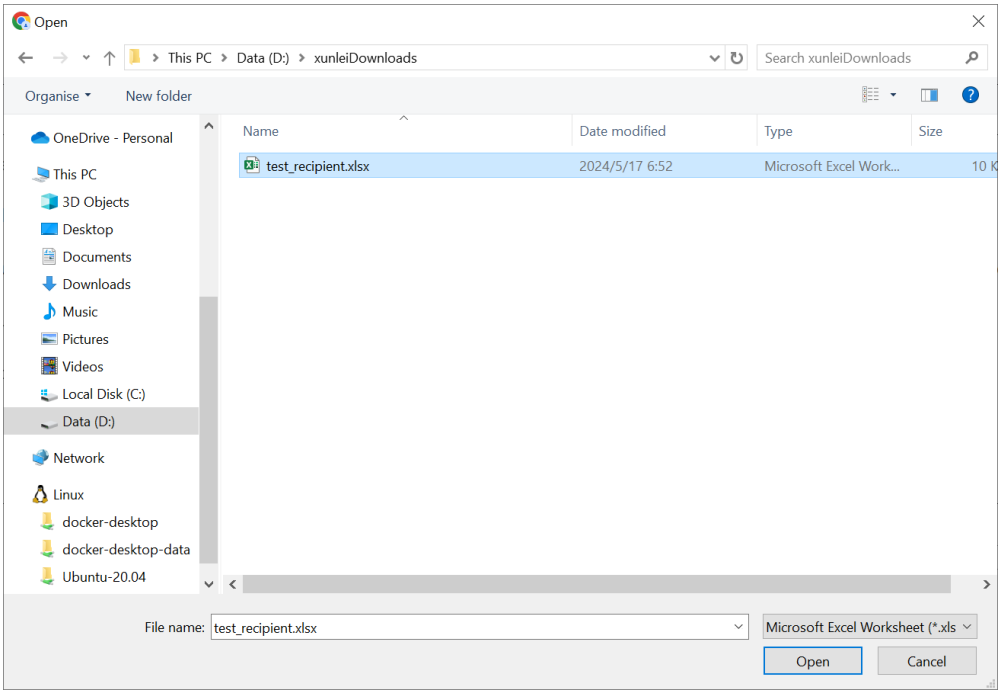
(Figure 31, Recipient Management)



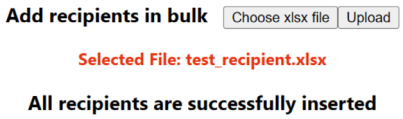
(Figure 32, Event Management)



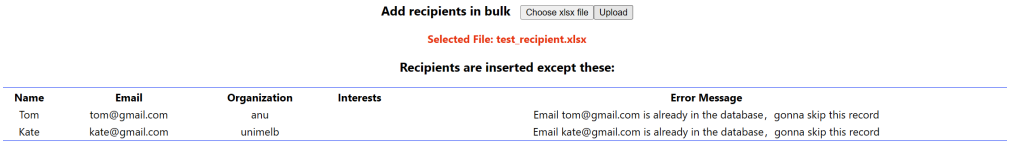
(Figure 33, Speaker Management)



(Figure 34, Choosing Excel file for recipients' bulky insertion)



(Figure 35, Successfully insert all recipients in Excel file)



(Figure 36, Error messages of bulky insertion)

Discussion and Future works

Having presented the product development and evaluation results, we now move on to the discussion and future work section. In this section we will analyse some of the important choices made during the research and development process, the strengths of the product, any limitations encountered during the research process and suggest directions for future work. The discussion aims to combine our good results with potential shortcomings, address any challenges faced, and suggest possible improvements and extensions to allow the ANU and John Eccles Institute to truly derive value from the products.

1. Critical System Design Decisions

Firstly, the research interests of subscribed recipients already exist in the database, but the reason for not retrieving them directly from the database is such that to know which user wants to access their interests would require the implementation of registration and login functionality, which would be an additional burden of use for non-registered users. Additionally storing users' passwords may require higher system security testing, as well as drafting an ethical clause for users to agree to the app storing their passwords and using their information.

Secondly, the elastic search function is implemented by calling PostgreSQL's built-in full-text search. This full-text search has advantages and disadvantages. The advantage is that query efficiency is high and can be searched by the root of the word to the complete word. The disadvantages are that no results are searched if the entered search text is shorter than the shortest word root, and that the search is not sensitive to certain numeric content in the text. Although such disadvantages exist, the efficiency and responsiveness of the search function is something that cannot be compromised, using front-end code for the search function may seriously affect responsiveness.

2. Advantages of the project product

Firstly, let me cover some of the exquisite designs of the website front-end. It's worth noting that the search bar and seminars displays are independent ‘React Components’, while the rest of the current page, including the header, navigation bar, and footer, are all HTML edited from ANU existing page. However, I understand that this HTML page is not up to the standard for direct deployment and will be improved by ANU's professional teams in the future. This is where the benefits of ‘React Components’ come into play, and the React Components could be easily integrated into the front-end page created by the professional team.

Secondly, the user-centred design throughout the project. One of the more prominent points is that the front-end of the webpage only shows the selected seminars, and after the user has selected a number of seminars, they need to go back to the top of the page to click the button, so I designed a Back to Top button to solve this pain point. In addition, the rich and detailed warnings and error messages in the management application can also avoid incorrect operation.

Thirdly, the aesthetics, the promotional emails and the front-end of the web page were carefully designed to ensure that users felt respected and valued.

3. Ethics and Policies

Having detailed advantages of the application and some critical system design decisions, we now turn our attention to discussing the limitations encountered during this process and outlining potential directions for future work. This will help identify areas for improvement and further innovation.

This project aims to collect events information through web scraping technology, providing a convenient resource retrieval tool for the academic community. The project primarily involves scraping the university's internal websites and Google Scholar while implementing stringent privacy protection measures for non-subscribing users. As the national university of Australia, we understand the importance of adhering to ethics and relevant federal policies, as this not only ensures the legality and morality of our actions but also directly impacts the reputation and future government-approved research funding.

3.1 Compliance with Ethics and Policies for Internal Website Scraping

When scraping the university's internal websites, we strictly adhered to university policies and obtained formal permission from the relevant administrators. This process ensures that our actions are legally and ethically sound and comply with internal regulations. We ensure that the data scraped is used solely for academic research purposes and does not infringe on any personal privacy or disclose sensitive information.

3.2 Ethical and Legal Considerations for External Website Scraping

If we plan to scrape other universities' websites in the future, we will face more complex ethical and legal issues. Specific measures include:

1. **Obtaining Formal Authorization**: Before scraping any external websites, we must obtain formal authorization from the target

website administrators. This is not only a sign of respect for the target website but also a necessary step to ensure the legality of the project.

2. **Transparency and Disclosure**: When requesting authorization, we should clearly inform the target website of our scraping purpose, data usage, and potential impacts. This helps build trust and ensures that the target website can provide informed consent.
3. **Privacy Protection**: Strictly comply with the Australian Privacy Act and other relevant laws and regulations, ensuring that no sensitive information or personal data is collected or stored. If personal data collection is involved, ensure data anonymization, and take necessary security measures to protect the data.
4. **Ethics Committee Review**: All scraping activities must be reviewed by the university's ethics committee to ensure that the project is ethically acceptable. This process helps identify potential ethical issues and take corresponding measures to address them.

1.3 Ethical and Legal Risks of Scraping Google Scholar

We currently use the external Python library Scholarly to scrape Google Scholar. While this library provides great convenience for academic research, its compliance is unclear, posing potential legal and ethical risks. To ensure the legality and morality of the project, we recommend the following measures:

1. **Code Review and Rewrite**: Conduct a comprehensive review of the Scholarly library to ensure it complies with Google Scholar's terms of use. If compliance cannot be confirmed, consider rewriting this part of the code using scraping methods that meet ethical and legal requirements.
2. **User Agreement and Disclaimer**: When using Google Scholar data, ensure that the data source is clearly marked, and include relevant user agreements and disclaimers to prevent potential legal disputes.
3. **Adhering to Terms of Use**: Strictly comply with Google Scholar's terms of use, avoiding violations of its prohibition on automated access. Consider using Google's provided APIs for data access to ensure compliance.

1.4 Strengths in Handling Non-Subscribing User Data

The project excels in handling non-subscribing user data. Specific measures include:

1. **Local Data Storage**: Interests and selected seminar data of non-subscribing users are stored only in their own browsers, not in our

project database. This measure effectively protects user privacy and reduces the risk of data breaches.

2. **No User Consent Required**: Since no non-subscribing user information is stored, we do not need to draft complex ethical terms or obtain user consent. This not only simplifies the project's operational process but also further ensures the protection of user privacy.

Overall, this project has taken many positive steps in adhering to ethics and policies, but there is still room for improvement. In the future, we need to pay more attention to the legality and ethics of external website scraping, ensuring that all actions comply with relevant laws and regulations. At the same time, by continuously reviewing and improving technical means, we ensure the project's compliance and morality, providing safe and reliable support for academic research.

4. Project limitations and the future potential work

4.1 Scholarly

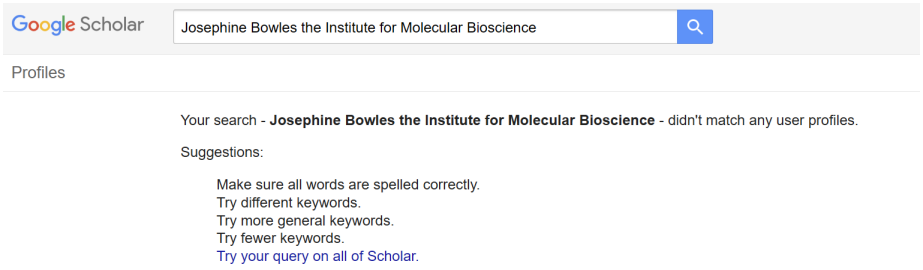
Due to the use of an external dependency, the package Scholarly, sometimes the IP address is briefly banned by Google Scholar. I have made an experimental attempt to add a rest interval in the middle of each crawl, which significantly reduces the speed of crawling Google Scholar's information but ensures the stability of the system operation. However, it is not guaranteed that there will be no banning in the operation, so it will probably be better to pay for renting an IP proxy pool dedicated to data crawling to ensure that the system operation will not be stopped because of the disabling of Google Scholar.

There is another limitation that in the default mode, the organisation of the seminar speaker is extracted from Biography, and there is no guarantee that this organisation is the most appropriate, due to the fact that the Fetch Interests button in the admin app when clicked to search is searching for both the person's name and their organisation. However, after my personal testing, I have found that overly precise organisation names can easily lead to a failure to search for a particular scholar, whereas using a plain university name makes it easy to successfully search for the correct scholar. In the case of Josephine Bowles, for example, if you search for the organisation extracted by the program, "The Institute for Molecular Bioscience", there are no results, but if you search for the University of Queensland, the search is successful. This part of the NLP algorithm may have to be designed

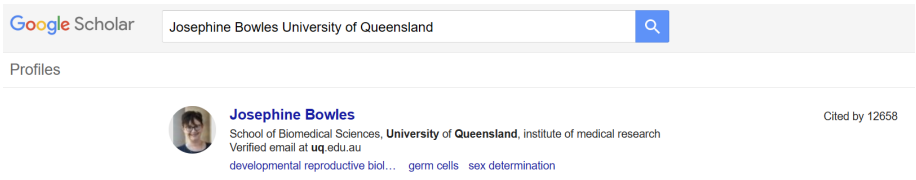
and implemented more sophistically to ensure that it is best suited for Google Scholar searches.

ID	Name	Organization	Google Scholar ID	Interests	Actions
1	Elise Stephenson	Oxford University Press			Delete
2		ANU			Delete
3	Naomi Hammond	nan			Delete
4	Leonie Quinn	the University of Melbourne			Delete
5	Maher Gandhi	nan			Delete
6	Marcus Hinchcliffe	nan			Delete
7	Ian Cockburn	the University of Edinburgh			Delete
8	Vicki Athanasopoulos	ANU College of Health and Medicine			Delete
9	None	nan			Delete
10	Josephine Bowles	the Institute for Molecular Bioscience			Delete
11	Kirsten Fairfax	Monash University			Delete

(Figure 37, Example of Overly precise organization of information)



(Figure 38, Google Scholar search result when using organization as "the Institute for Molecular Bioscience")



(Figure 39, Google Scholar search result when using organization as "University of Queensland ")

4.2 System tests

Although I was following a strict test-driven development strategy during the development process and performs many white-box and black-box tests, I didn't write system test cases due to the sheer size of the system and the fact that there is only one programmer who was multi-tasking. System testing, also called system-level or system integration testing, always involves a quality assurance team evaluating how different components of an application interact within the fully

integrated system (Yasar & Black, 2023). System testing can be classified as a type of black box testing ensuring the application performs tasks as designed, verifying that every kind of user input produces the intended output across the application (Yasar & Black, 2023). So, there might be some bugs in the system, and I hope to do more testing work in the future to make the project completely bug-free.

4.3 Versatility

Regarding the extraction of various information, although the correct rate has been able to reach about 90%, there are still some errors appearing that need to be manually corrected by the administrator, this is because most of the NLP algorithms in my program have only one, but after I have taken a closer look at the code of another content extraction library, newspaper3k, I have found that a combination of multiple laddering NLP algorithms together can greatly increase the information extraction correctness (Ou-Yang, 2020). Future programmers who take on this project should design multiple NLP algorithms and run them all the way from the highest-priority NLP algorithm all the way down to the lowest-priority NLP algorithm to see if their results are consistent; if they are mostly consistent the results are proven to be correct, and if they are very inconsistent then they should return a notification.

Conclusion

In conclusion, the project is a Master's capstone project in crawling, web development, and natural language processing. It provides an automated crawler, a handy information management application, and a promotional email distribution system. In this report, we discussed the background of the project, feasibility analysis, implementation strategy and techniques of different modules, results, evaluation, discussion about technical and ethical aspects of the project and future work. This project has provided significant value to the John Eccles Institute (JEI). Firstly, it has helped address the main demand for an events page that include seminars from other departments related to neuroscience. Secondly, the highly automated crawler system and data management tools allow JEI to minimise the human cost of managing the seminars centralizing product. Thirdly, the project was designed to enhance accessibility and participation of neuroscience events across multiple disciplines. During the final evaluation meeting, all clients expressed satisfaction with the project's aesthetics, usability, and efficiency, while members of the John Eccles Institute successfully integrated and utilized the tools, with supervisors from SOCO consistently reviewing the code and software system to ensure its robustness throughout the project timeline. For more precise information on project dependencies, project deployments, project licences and other technology-related topics see Appendix.

Bibliography:

Archibald, M.M. *et al.* (2023) 'How transdisciplinary research teams learn to do knowledge translation (kt), and how KT in turn impacts transdisciplinary research: A realist evaluation and longitudinal case study', *Health Research Policy and Systems*, 21(1). doi:10.1186/s12961-023-00967-x.

Bowman, E. (2022) *Volker Thoma (East London)*, *Centre for Brain, Mind and Markets*. Available at: https://www.unimelb.edu.au/cbmm/about-us/media/seminars/volker-thoma-east-london (Accessed: 15 April 2024).

Buchfelder, R. and Konkov, E. (2021) *Cookies on localhost with explicit domain*, *Stack Overflow*. Available at: https://stackoverflow.com/questions/1134290/cookies-on-localhost-with-explicit-domain (Accessed: 17 April 2024).

Cloudflare (2024) *What is a web crawler? | how web spiders work | cloudflare*. Available at: https://www.cloudflare.com/learning/bots/what-is-a-web-crawler/ (Accessed: 23 April 2024).

Express (2024) *Node.js web application framework*, *Express*. Available at: https://expressjs.com/ (Accessed: 17 April 2024).

Facebook (2023) *Start a new react project*, *React*. Available at: https://react.dev/learn/start-a-new-react-project (Accessed: 15 April 2024).

Farag, M.M., Lee, S. and Fox, E.A. (2017) 'Focused crawler for events', *International Journal on Digital Libraries*, 19(1), pp. 3–19. doi:10.1007/s00799-016-0207-1.

Interaction Design Foundation (2016) *What is design thinking? - updated 2024*, *The Interaction Design Foundation*. Available at: https://www.interaction-design.org/literature/topics/design-thinking (Accessed: 23 February 2024).

Jama Software (2022) *Requirements gathering techniques for agile product teams*, *Jama Software*. Available at: https://www.jamasoftware.com/requirements-management-guide/requirements-gathering-and-management-processes/11-requirements-gathering-techniques-for-agile-product-teams (Accessed: 23 March 2024).

Kalra, K. (2023) *Layoutlm explained*, *Nanonets Intelligent Automation, and Business Process AI Blog*. Available at: https://nanonets.com/blog/layoutlm-explained/ (Accessed: 21 March 2024).

Krotoff, T. (2023) *Front-end frameworks popularity (react, Vue, angular and Svelte), GitHub*. Available at: https://gist.github.com/tkrotoff/b1caa4c3a185629299ec234d2314e190 (Accessed: 28 January 2024).

Kumar, S. (2023) *Why Nodejs, Medium*. Available at: https://sunilrana123.medium.com/why-nodejs-bdf2c2e47403 (Accessed: 17 February 2024).

Nash, J.M. (2008) 'Transdisciplinary training', *American Journal of Preventive Medicine*, 35(2). doi:10.1016/j.amepre.2008.05.004.

Ou-Yang, L. (2020) *Codelucas/newspaper: Newspaper3k is a news, full-text, and article metadata extraction in python 3. advanced docs:*, *GitHub*. Available at: https://github.com/codelucas/newspaper (Accessed: 17 May 2024).

Prad, R. (2023) *9 advantages of react.js: Why choose it for your web project, Sayone Tech*. Available at: https://www.sayonetech.com/blog/advantages-of-react-js/ (Accessed: 29 January 2024).

Scholarly (2023) *Scholarly-python-package/scholarly: Retrieve author and publication information from google scholar in a friendly, pythonic way without having to worry about captchas!*, *GitHub*. Available at: https://github.com/scholarly-python-package/scholarly (Accessed: 16 May 2024).

Simplilearn (2023) *What is FastAPI: The future of modern web development, Simplilearn.com*. Available at: https://www.simplilearn.com/what-is-fastapi-article#benefits_and_drawbacks_of_fastapi (Accessed: 29 January 2024).

SpaCy (2024a) *Models & languages · spacy usage documentation*, *Models & Languages*. Available at: https://spacy.io/usage/models (Accessed: 17 May 2024).

SpaCy (2024b) *Spacy · industrial-strength natural language processing in python*, *· Industrial-strength Natural Language Processing in Python*. Available at: https://spacy.io/ (Accessed: 21 March 2024).

Tiangolo (2024) *Tiangolo/FASTAPI: FASTAPI framework, high performance, easy to learn, fast to code, ready for production*, *GitHub*. Available at: https://github.com/tiangolo/fastapi (Accessed: 04 May 2024).

Vaughn, L.M. and Jacquez, F. (2020) 'Participatory research methods – choice points in the research process', *Journal of Participatory Research Methods*, 1(1). doi:10.35844/001c.13244.

W3schools (2024) *XML and XPath*, *W3schools*. Available at: https://www.w3schools.com/xml/xml_xpath.asp (Accessed: 21 March 2024).

Appendix

Project Technical Manual:

https://gitlab.cecs.anu.edu.au/u7377070/john-eccles-institute-seminar-project/-/blob/main/README.md?ref_type=heads

If you do not have access to the Technical Manual, please contact me.