COMP3900 - Project Report

P16 - Successful Outcomes

'); DROP TABLE Students;--

Amanda McGilchrist	z5358641@ad.unsw.edu.au	z5358641	Frontend Developer
Aster Berida	z5421285@ad.unsw.edu.au	z5421285	Backend Developer
Brendan Purdon	z5309111@ad.unsw.edu.au	z5309111	Backend Developer
Eiman Zare	z5417758@ad.unsw.edu.au	z5417758	Frontend Developer
Justin Yang	z5361584@ad.unsw.edu.au	z5361584	API/Backend Developer
Pepsi Sharma	z5418973@ad.unsw.edu.au	z5418973	Scrum Master

Installation Manual

The following ports must be free:

- 3000 (frontend)

Easy Mode (Docker)

Copy the following line into a command prompt:

curl -sSL https://coursemapper.unsw.sh/install | bash

This installer script will automatically generate the secrets required, as well as download and run the latest Docker images required. It assumes that you have docker (or some other container environment available as a docker context) installed.

You can access the web interface at http://localhost:3000 via a web browser.

Medium Mode (Docker)

If the first method doesn't work (usually this occurs only if you're not running an x86_64 Docker context), try the following:

1. Clone the git repository
2. Run docker compose build (Note: this step may take a while)
3. Run docker compose up -d

You can access the web interface at http://localhost:3000 via a web browser.

Hard Mode

Follow the instructions at https://coursemapper.unsw.sh/

You can access the web interface at http://localhost:3000 via a web browser.

Demo Instance

If you are unable to run the code locally for some reason, https://coursemapper-demo.unsw.sh/ is running the latest version of the service.

System Architecture Diagram

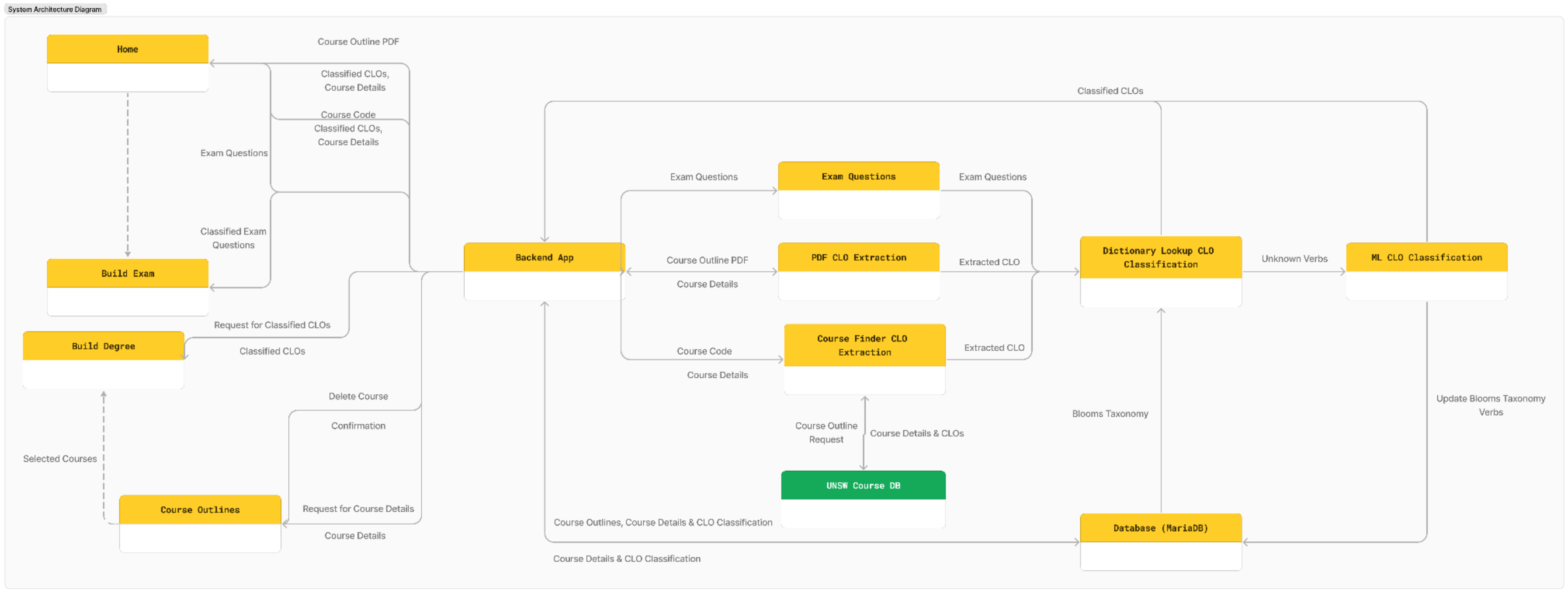


Figure 1: System Architecture Diagram

Backend

The backend revolves around the following functionality, storing and accessing data between the database, extracting CLOs and details from a course outline, and classifying verbs to bloom's taxonomy.

The database uses mariaDB, an open source project based on MySQL. The DB contains three tables, a CLO table, a Bloom's Taxonomy table and a course details table. The CLO table store which stores the text of a CLO, the course it is from and classified verbs. The course details table stores the details of a course such term, delivery mode, e.c.t., and the classified CLOs of the course. The Blooms table stores a verb and its associated blooms level. This includes non verbs, eg. “and ", so the ML node knows not to analyse it.

The backend can extract data using course outline PDFs, course codes and exam questions. Extraction from PDFs extracts the details of a course and the CLOs of a course. Extraction from a course code, requests the course outline of a given course from the UNSW course outline database, from which the course details and course CLOs are taken. Extraction from an exam question takes the text input of the exam questions and prepares the data for Bloom's classification.

Once extraction of one of the three user inputs is done, the CLOs/Exam questions are classified to Bloom's taxonomy. This is done by the same classification node for all of the three user inputs. CLOs/Exam questions are first run through a dictionary lookup, which checks every word against the list of Bloom's taxonomy verbs in the database. If a word is not present in the database, it is then checked using the ML classifier. This utilises the BERT model, and is trained off our initial list of Bloom's taxonomy verbs and non verbs, and will identify a verb as one of the blooms level or a non verb. Once a verb is classified it, and its classification is stored in the database, so if seen again, the dictionary lookup will be able to classify it. Once a CLO is classified, it is stored in the database within its respective course. Exam questions are not stored anywhere, including the database, and are instead sent straight to the front end.

Frontend

The frontend consists of four pages: the Home page, Course Outlines, Build a Degree and Exam Evaluator.

Home Page

- **Upload Course Outlines**: Users can upload a course outline PDF. The backend processes the PDF and displays a preview of the course details once the upload is complete
- **Search by Course Code:** Users can input a course code to retrieve and display the corresponding course outline details and CLOs. This feature simplifies data access by avoiding the need to search for the course outline PDF on the university's website. It works by reverse engineering the UNSW Course Outline Finder and sending the requests to its API for the relevant data.
- **Classify Exam Questions**: Users can submit exam questions to be classified according to Bloom's taxonomy. Once submitted, the frontend sends the questions to the backend for processing, when the frontend has the returned results, users are then redirected to the Exam Evaluation page.

Course Outline Page

- **Display Course Outlines:** This page lists all the uploaded course outlines found in the database. Users also have the option to delete an individual course outline from the list.
- **Filtering and Searching:** Users can filter through the list of course outlines based on various criteria and perform searches to find specific courses quickly.
- **Navigation to Build Degree Page:** A user can select a range of course outlines they would like to have analysed. After clicking the NEXT button they are navigated to the Build a Degree page. This also triggers the API that analyses all the courses selected and returns the results to the frontend for display.

Build a Degree Page

- **Display Classified CLOs:** Shows CLOs for each course, with verbs colour-coded according to Bloom's Taxonomy levels for clear understanding
- **Bar Graph of Bloom's Taxonomy Levels:** Features a bar graph illustrating the combined counts of Bloom's taxonomy levels from all of the selected course outlines.
- **Navigation Options:** Users can navigate back to the Course Outline page by selecting the BACK button. They can also use the navbar to go to the Home or Course Outlines page.

Exam Evaluation Page

- **Displays exam questions:** Users have been redirected from the upload exam page to the Exam evaluation page, the exam questions are listed for the user to see.
- **Classified Exam Questions:** Shows the count of Bloom’s Taxonomy levels from the classified exam questions, with results represented in a bar graph.

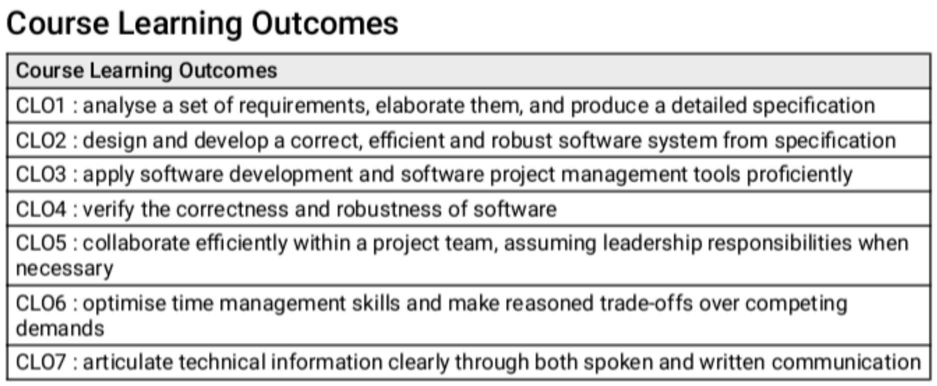
Design Justifications

Course Outline Upload and CLO Extraction

The initial design was to provide the user a way to upload course outlines and to then scrape the course outlines for CLOs. This was to be done through searching of the CLO heading in the PDF.

For sprint 1, 2 and 3 this approach was used for extracting CLOs from uploaded course outlines. This lack of deviation across sprints was due to the consistent formatting of UNSW course outlines leading to predictable and fast extraction of CLOs from an outline.

Here each UNSW outline has a page titled "Course Learning Outcomes", with most lines in said page being a CLO. Once the CLO page is identified, the function filters for lines starting with "CLOX" to extract the contents of each CLO.



Course Learning Outcomes

Course Learning Outcomes
CLO1 : analyse a set of requirements, elaborate them, and produce a detailed specification
CLO2 : design and develop a correct, efficient and robust software system from specification
CLO3 : apply software development and software project management tools proficiently
CLO4 : verify the correctness and robustness of software
CLO5 : collaborate efficiently within a project team, assuming leadership responsibilities when necessary
CLO6 : optimise time management skills and make reasoned trade-offs over competing demands
CLO7 : articulate technical information clearly through both spoken and written communication

Figure 2: Course Learning Outcome Example

However, from sprint 2 and 3 functionality was expanded upon from the initial design requirements by incorporating "search by course code" functionality. This presented users with a faster way to upload course CLOs for existing courses as users no longer had to find and download the PDF of a course. Instead, the website directly requested the course information from the UNSW course outline database.

This works by accessing the same API that UNSW's Course Outline Finder uses. First the course outline id of the latest version of a course is extracted from https://courseoutlines.unsw.edu.au/v1/publicsitecourseoutlines/search?year=2024&searchText= [course code]

With the course outline id, the course CLOs and details are extracted from,

https://courseoutlines.unsw.edu.au/v1/publicsitecourseoutlines/detail?coId= [course_ID]

This approach for extracting CLOs of existing courses is the most effective as it uses existing infrastructure in a seamless manner, to extract the required data quickly and reliably. Despite this the CLO extraction from PDFs is still an effective and valuable feature as it can collect required data for courses not yet existing on the UNSW database.

Mapping of CLOs to Bloom's Taxonomy

The initial design we had proposed used a dictionary lookup to manually identify Bloom's taxonomy verbs present in a CLO. If no match is found a machine learning model would then be used to identify the Blooms level.

Our sprint 1 iteration of the project solely incorporated the dictionary lookup method. This was where a prewritten list of each blooms level and their associated verbs is stored in the backend. When assessing the record of each Bloom's verb present in a CLO is stored. Due to the majority of course outlines using Bloom's taxonomy verbs, this method worked for the majority of course outlines with only outlier courses not mapping to any verb. This approach was kept for sprints 2 and 3, with the only adaptation being an expansion of Bloom's verbs, as lists vary from resource to resource (Utica University, n.d.) (Shabatura, 2014).

For sprint 2 we used the LLM BERT for the machine learning functionality. The BERT model was trained on the verbs of each of Bloom's levels. When a word in a CLO did not match one of the prewritten verbs, it was passed through the ML model to assess if it was a verb, and if so, which blooms level it belongs to. If assessed as a verb this record of Blooms verbs in a CLO was updated.

After sprint 2 we received feedback from the client about improving the efficiency and load time of the ML model. We did this by incorporating a memorisation method to help computation time throughout operation and by additionally pre defining a set of commonly occurring words as non verbs, e.g. “and”, to help structure output. The memorisation method involved updating the dictionary of verbs and non verbs so that once a word had been assessed through the ML method, the dictionary lookup would be able to classify it for future cases. This helped to reduce computation time and did not impact reliability as ML classification would be the same for the same input word.

By using the combined solution, a fast and reliable initial scan can be performed to assess CLOs, with the machine learning model being used to assess edge cases presented. This approach provides the best of both worlds for this functionality.

Data Representation

The initial design we had proposed was to provide the user with a variety of graphical data to visualise the quantity of each category present in a course.

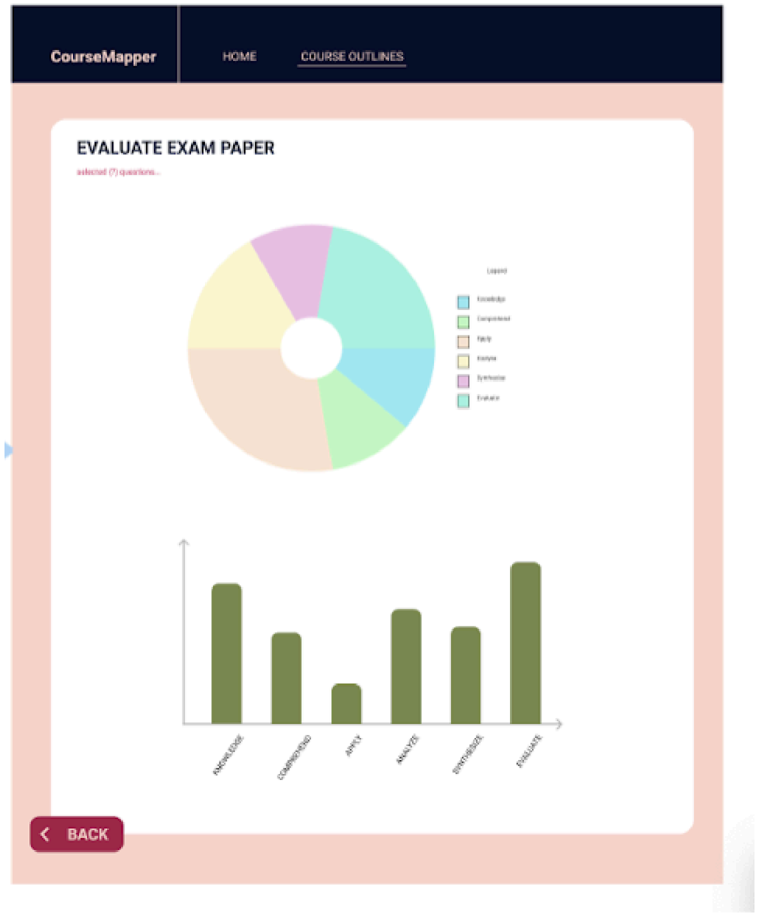


Figure 3: Initial Analytics Design

For sprint 1 this was done through a bar graph which displayed to the user the quantity of verbs from each Bloom's level from a selected list uploaded course outlines.

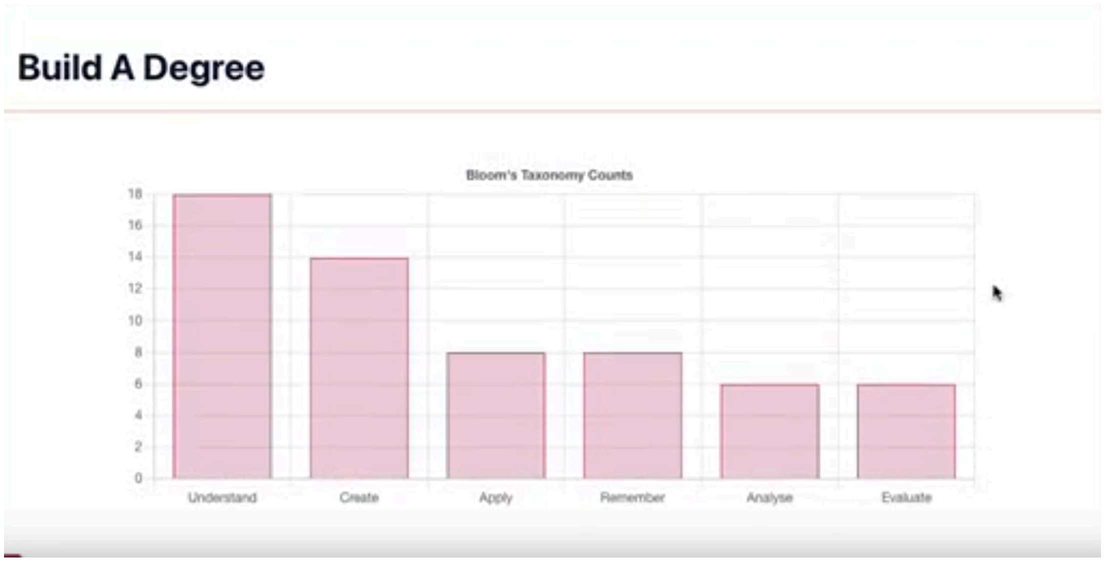
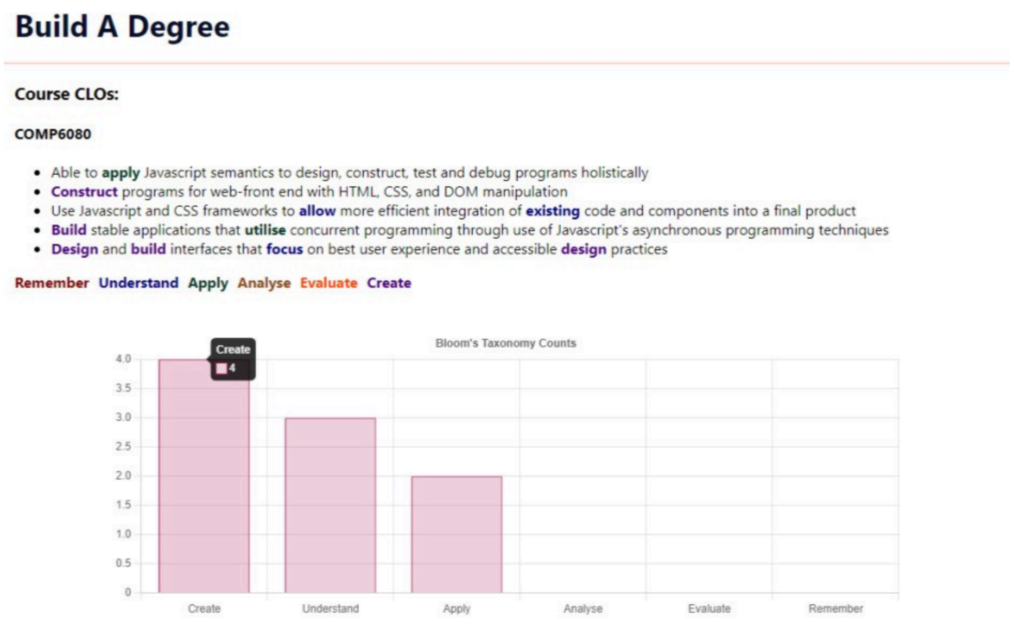


Figure 4: Sprint 1 - Build a Degree

After review with the client, an additional requirement of displaying the CLOs for each course alongside the analytics was provided to us. Improving upon that feedback, we added colour coding to any Blooms Verbs present in the displayed CLOs. This provided the user a clearer understanding of which levels of blooms verbs are present in the CLO of a course for sprint 2.



Build A Degree

Course CLOs:

COMP6080

- Able to **apply** Javascript semantics to design, construct, test and debug programs holistically
- **Construct** programs for web-front end with HTML, CSS, and DOM manipulation
- Use Javascript and CSS frameworks to **allow** more efficient integration of **existing** code and components into a final product
- **Build** stable applications that **utilise** concurrent programming through use of Javascript's asynchronous programming techniques
- **Design** and **build** interfaces that **focus** on best user experience and accessible **design** practices

Remember Understand Apply Analyse Evaluate Create

Figure 5: Sprint 2 - Build a Degree

Also, in sprint 2, we included a filtering system for uploaded course selection. This allowed users to filter by 8 different categories including, Course Name, Term, Delivery Mode, Delivery Format, Location, Faculty, Study Level and Campus. This functionality made it easier for users to assess course combinations, and therefore in assessing different degree makeups for their educational merit.

Again after sprint 2, our client provided feedback for a desire of having a preview page for the uploaded course. This allows the user to see details of the course to be able to confirm the right information was uploaded, which was particularly useful for the search by course code feature, as the user had no way of seeing the details of a course after upload. Finally during sprint 3, the ability to delete course outlines which have been uploaded was added. This provided a better user experience as this software is designed for the development of degrees which inherently sees adaptation of courses and course CLOs.

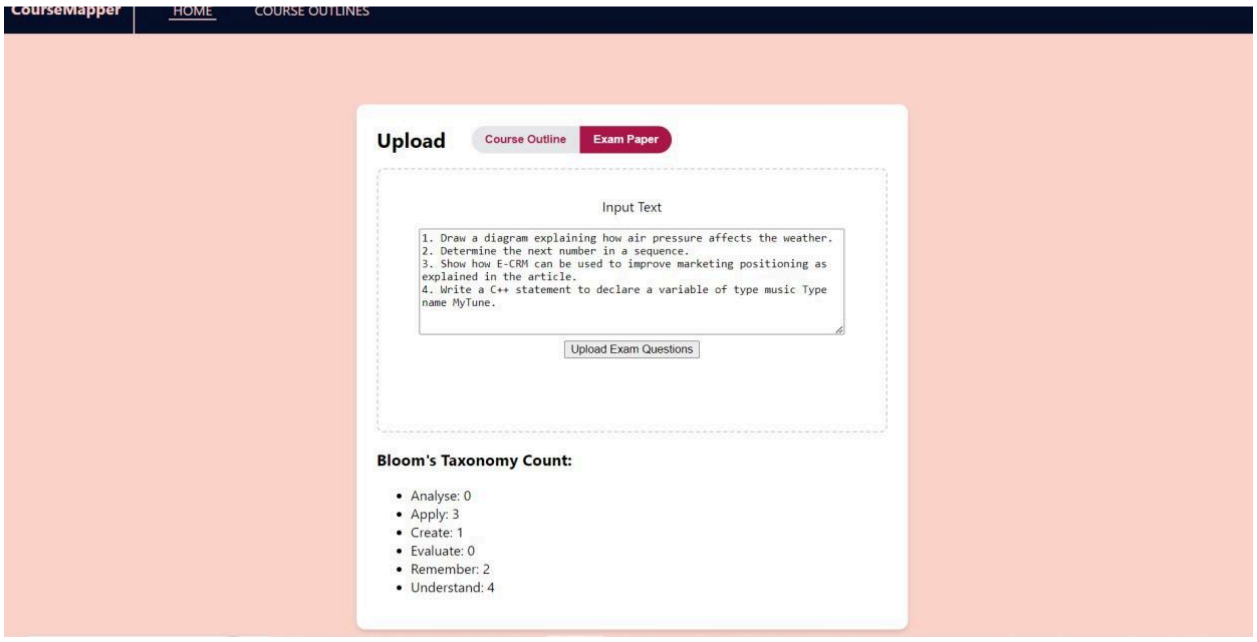
Through an iterative design process, and consultation from the client, consistent improvement was made in improving the depths and effectiveness of analytic features provided to users.

Exam Questions

This feature was proposed as a novel function not commonly found in existing tools and useful for addressing a key need for educators. During the initial design phase, it was planned to be done through a ML function to assess the non-standard nature of exam questions.

Through consultation with the client and the design process, this was revised to an assessment of a parsed set of number questions that would then be categorised through the Blooms taxonomy mapping functionality used for the mapping of CLOs. This was because it standardised the format the exam questions would be in, improving ease of use.

During sprint 2, the first iteration of the exam functionality was implemented. This was as described above, where users were prompted to provide the exam questions, then the categorisation was presented to the user.



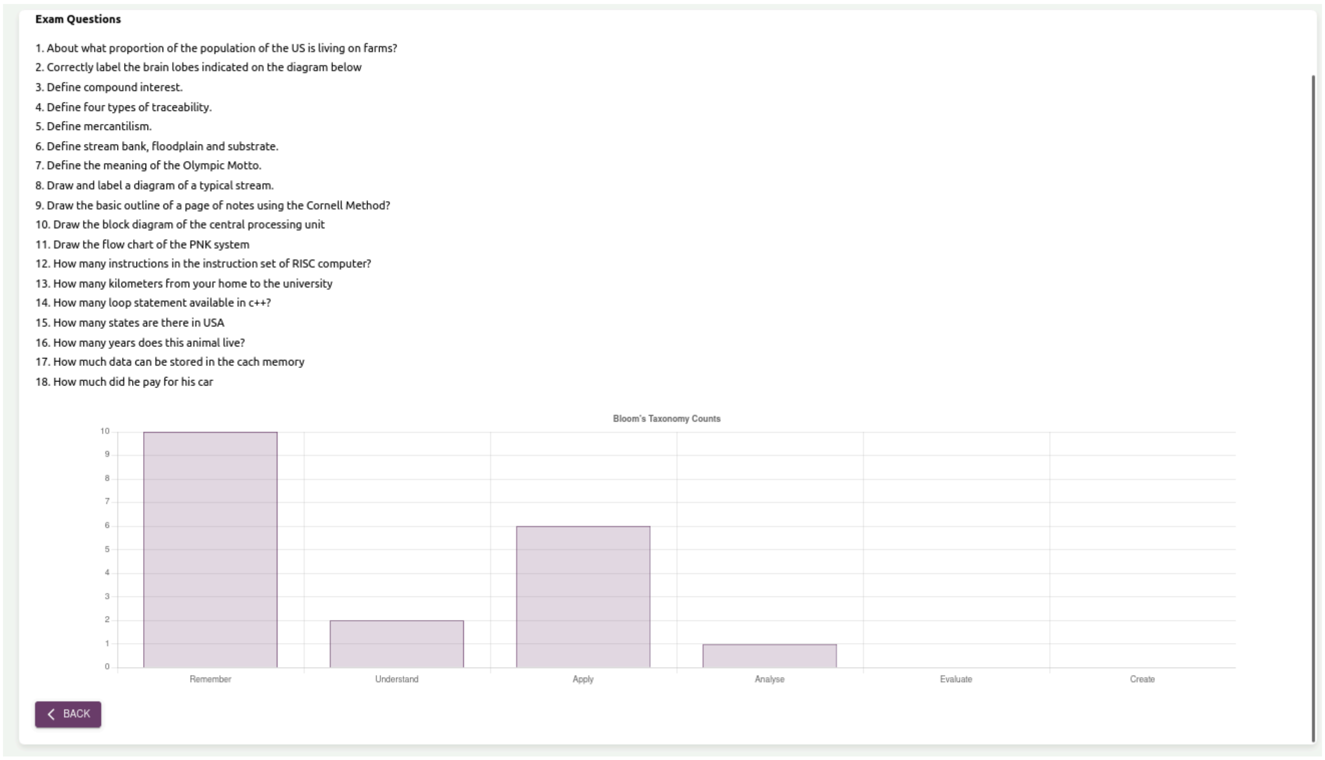
```
1. Draw a diagram explaining how air pressure affects the weather.
2. Determine the next number in a sequence.
3. Show how E-CRM can be used to improve marketing positioning as explained in the article.
4. Write a C++ statement to declare a variable of type music Type name MyTune.
```

Bloom's Taxonomy Count:

- Analyse: 0
- Apply: 3
- Create: 1
- Evaluate: 0
- Remember: 2
- Understand: 4

Figure 6: Sprint 2 - Exam Questions

During sprint 3, the analytics were expanded to be more aligned to what was present for the course outline analytics. This included charts displaying bloom's taxonomy counts and presentation of uploaded exam questions



Exam Questions

1. About what proportion of the population of the US is living on farms?
2. Correctly label the brain lobes indicated on the diagram below
3. Define compound interest.
4. Define four types of traceability.
5. Define mercantilism.
6. Define stream bank, floodplain and substrate.
7. Define the meaning of the Olympic Motto.
8. Draw and label a diagram of a typical stream.
9. Draw the basic outline of a page of notes using the Cornell Method?
10. Draw the block diagram of the central processing unit
11. Draw the flow chart of the PNK system
12. How many instructions in the instruction set of RISC computer?
13. How many kilometers from your home to the university
14. How many loop statement available in c++?
15. How many states are there in USA
16. How many years does this animal live?
17. How much data can be stored in the cach memory
18. How much did he pay for his car

Figure 6: Sprint 3 - Exam Questions

User Driven Evaluation of Solution

Main Features of Product

The main features of our website are:

- Users can instantaneously start assessing courses upon opening the website
- Users can quickly upload any existing UNSW course outline into the analytics through “search by course code”
- The website allows for potential and unreleased course outlines to be uploaded via PDF
- The website provides comprehensive filtering and searching of uploaded courses for dynamic assessment of different course combinations
- Detailed analytics giving a colour coded representation of Bloom’s levels present in a CLO and course.
- Users can analyse Exam questions to the same Bloom’s taxonomy levels as whole course outlines

Evaluation Objectives

The evaluation metrics that we used to assess our website were:

- Mapping to Bloom’s Taxonomy
- Constructive Analytics
- Reliable and Accurate
- Ease of Use

Target User

Our target user is a university educator tasked with developing university courses or structuring degrees. They most likely would have been introduced to our product through a colleague and told a brief overview of how it works. They are using our website as a way to help visualise bloom’s taxonomy across the courses they are working on.

Mapping to Bloom’s Taxonomy

As the main function of our project, we provide users the ability to upload courses and exam questions, which is then classified by Bloom’s Taxonomy.

Reliable and Accurate

For our target audience, the majority of them would not take the time to verify The accuracy of our classification. As such it was important to ensure accuracy and reliability from the design of the product, therefore several considerations were made. The two step classification of CLOs to Bloom's taxonomy meant that the default case was a dictionary lookup to a predefined set of verbs and non verbs. This operation is ideal for accuracy as it is a known set of outputs.

For the ML component, by utilising a pre-trained model the accuracy and reliability of our product is dependent on the model. While the BERT is accurate for most cases, it still classifies some words incorrectly. To alleviate this we have done extensive training to expand our set of pre categorised verbs and non verbs.

To improve reliability we made sure to standardise input cases to align to the UNSW structure for course outlines. While this does limit assessment to UNSW courses, it makes operation of the website more predictable and reliable for users.

Ease of Use

Users of the website may have no to little introduction of how the website operates, as such it should be self explanatory on how to use it. This is seen with our target user who may have only received a link from a colleague. Our final product achieves this through clear communication of how to operate it. Upon opening the website, the user is presented the option to upload course outlines or to upload exam papers. This makes it easier for the user as it instantly makes it clear what their action should be.

This ease of use is extended by having the option to search by course code, which makes it simpler to add multiple courses, and filtering of upload courses makes it easier to work with large sets of courses. Additionally the incorporation of loading screens, success messages, error messages, pagination, and other similar features, make a clear and concise product where the user understands what the website is done or has done.

While the team has made deliberate effort to create a streamlined user experience, something as qualitative as ease of use will always have slight improvements that can be made. By having to manually go to the course outlines page after uploading course outlines, creates a degree of separation between the user uploading a course and assessing a course. While it would be ideal to directly take the user to the course outline page or even the analytics page, similar to exam questions, this is a trade off. That being it was more important to us for a user to continue to upload multiple courses on the home page, then to be directed straight to the analytics page.

Finally consideration for a range of devices that our users may use was not fully addressed. Due to educators working in a range of settings, it is very likely that some of our users may use devices like tables. As such our product failed to meet this demographic of our target users.

Constructive analytics

As a part of our product we provide users constructive and useful analytics to assess their courses and exams. This is presented through graphs and colour coded visualisation of CLOs and exam questions. This allows users to identify which levels of a Bloom's taxonomy are present. This is useful as different levels correlate roughly to the difficulty experienced and expertise gained by fulfilling a CLO as said level. So if our target user was assessing a 1000 level course our product will make clear to them if there is a proper distribution of “remember” verbs present for such a course.

Additionally the ease of mixing and matching different course combinations for a degree allows for users to experiment and assess the educational merit across whole degrees. In this case course admin would want to see each of the taxonomy levels present so they can be confident that a degree will target to teach the basics for “remember” tasks up towards the more thorough “create” tasks, across the range of courses in the degree.

This is a more open objective with avenues to expand within and beyond the requirements set out by the client. As such further statistical representations could be provided to users to improve their experience.

Limitations and Future Work

Limitations of Work

In developing our product there were many features that we would have liked to include but could not due to limitations imposed on us.

As mentioned in the User Driven Evaluation Section, there are improvements to be made for the ML component of the product. Specifically using a more accurate and powerful ML model. An example of such a model is H20-Danube3 which could provide better performance over BERT. However a limitation in using this, or many other improved models, is cost. Especially with larger scales, there are significant costs which we cannot cover at the prototyping stage.

Expansion into assessment of courses from universities other than UNSW. Due to educational requirements and the prevalence of Bloom's taxonomy, other universities also use CLOs in creating courses. We could incorporate functionality into search by course code, to be able to search from a range of universities. Additionally, assessment of PDFs could accommodate other universities course outline structure. This would expand our product to a wider audience, helping more educators to create better courses and degrees. To achieve this, we would need to expand our CLO extraction method beyond the UNSW course outline structure. While possible to overcome it presents issues of scalability. Additionally, we may require access to universities' databases of courses to implement a widespread search by course code. Finally we would need to provide information on the website to make it clear how different universities use different course code structure.

Avenues for Future Work

While not a part of the requirements laid out by the client for this project cycle, there are a variety of improvements and expansions that could be made to the product. By including accessibility to users for other devices, such as tablets, we increase the number of people that could or would use our product. This however would require extra consideration into the design. This includes considerations such as mobile responsiveness, touch screen usability, navigation design, and others.

Another function that could be implemented as future work would be assessment of exams directly. Due to exams coming in nonstandard formats, in both layout and format, ie. PDF of text, PDF of images, Moodle Quiz, this is more difficult to implement. To implement this an algorithm which can process the text of an exam and store just the raw questions of the exam would need to be created. Additionally, this would need a node to utilise existing CV libraries to extract the exam text from an image of the exam. However, if done, this would save users from having to manually copy or type questions into a text box and would create a more streamlined process.

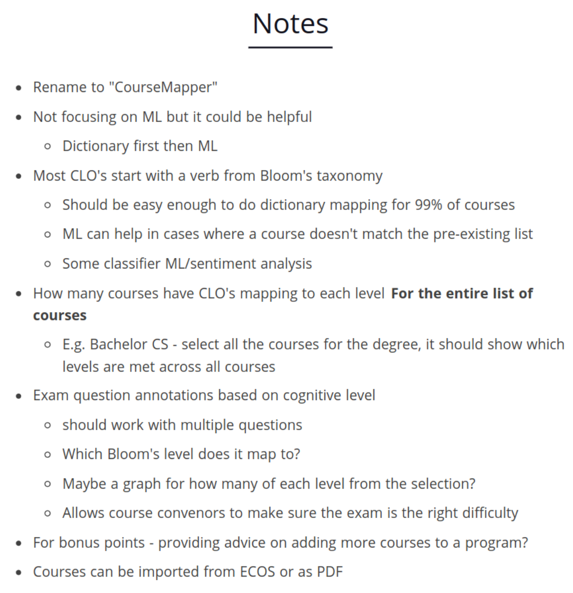
Finally, the website could provide recommendations of improvements to a course or degree based upon the Bloom's categories a set of courses covers. This would require a deeper understanding of the educational merits of different courses at different stages of a students education by the team. This would therefore require additional research to be done or an incorporating people from different specialisations into the team. With such understanding the website could then assess based upon educational guidelines (*1 Guidelines for Bloom's Taxonomy Usage For Instructional Design Background: Bloom's Taxonomy Is a Widely-Recognized Tool Fo*, n.d.) and a criteria provided by the user, eg. Is this a full degree, level 1000 course, level 2000 course etc., provide recommendations. This would enhance the effectiveness of the website as a resource for educators in constructing better courses and degrees.

Engineering Practices

As a part of ensuring the product we produced for the client met their expectations, several software design practices were incorporated into our design process.

To ensure that the intended output of the code was clear, a test-driven development process. This meant that test cases were written for functionality before the functionality was created. This helped to maintain robust and reliable code by preventing bias from writing tests around existing functionality.

Additionally, as a part of our design process we had regular fortnightly meetings with our client, Dr. Sonit Singh. In these meetings we would present new developments made to the product, talk about rational behind forward facing and backend design decisions, and discuss future improvements he would like to see to the product. By doing so this gave the team clear objectives to work towards and allowed for an iterative design process with consistent client feedback at different stages of development.

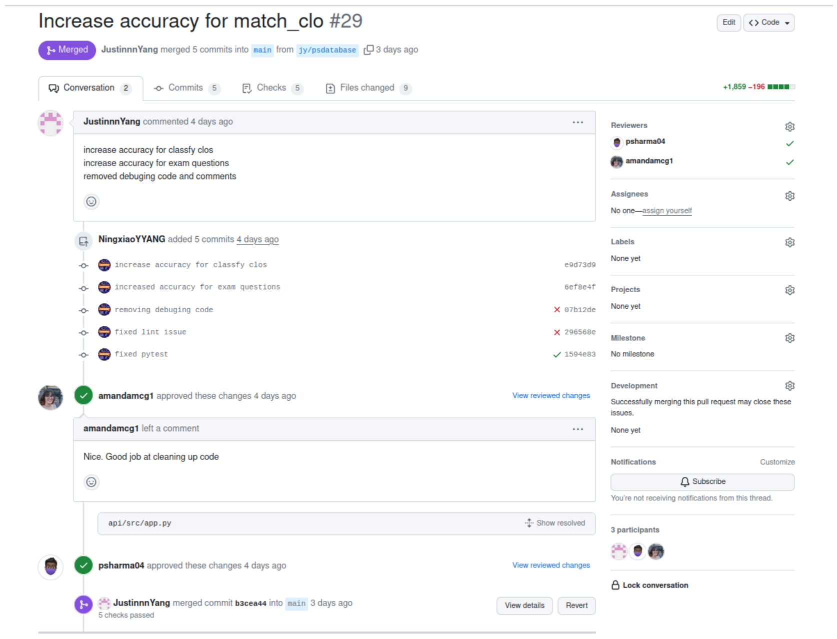


Notes

- Rename to "CourseMapper"
- Not focusing on ML but it could be helpful
    - Dictionary first then ML
- Most CLO's start with a verb from Bloom's taxonomy
    - Should be easy enough to do dictionary mapping for 99% of courses
    - ML can help in cases where a course doesn't match the pre-existing list
    - Some classifier ML/sentiment analysis
- How many courses have CLO's mapping to each level **For the entire list of courses**
    - E.g. Bachelor CS - select all the courses for the degree, it should show which levels are met across all courses
- Exam question annotations based on cognitive level
    - should work with multiple questions
    - Which Bloom's level does it map to?
    - Maybe a graph for how many of each level from the selection?
    - Allows course convenors to make sure the exam is the right difficulty
- For bonus points - providing advice on adding more courses to a program?
- Courses can be imported from ECOS or as PDF

Figure 7: Meeting Minutes from Meeting with Client (12/06/24)

Our group utilised Pull Requests and Code Reviews to enhance code quality. Each code change was submitted through a pull request, which included a description of the modifications. A team member then reviewed the pull request, and either accepted the changes or provided feedback on changes that needed to be made to the modifications. This process ensured that at least a third of the team understood what was being changed, and that it adhered to the standard of code being produced. By incorporating these reviews before merging any code into the main branch, we maintained a high level of code quality and consistency, ultimately leading to more reliable and maintainable software.


Increase accuracy for match_clo #29

Merged JustinnnYang merged 5 commits into `main` from `jy/psdatabase` 3 days ago

JustinnnYang commented 4 days ago

increase accuracy for classfy clos
increase accuracy for exam questions
removed debuging code and comments

NingxiaoYYANG added 5 commits 4 days ago

```
increase accuracy for classfy clos        e9d73d9
increased accuracy for exam questions     6ef8e4f
removing debuging code                  ✗ 07b12de
fixed lint issue                        ✗ 296568e
fixed pytest                            ✓ 1594e83
```

amandamcg1 approved these changes 4 days ago — View reviewed changes

amandamcg1 left a comment

Nice. Good job at cleaning up code

`api/src/app.py` — Show resolved

psharma04 approved these changes 4 days ago — View reviewed changes

JustinnnYang merged commit **b3cea44** into `main` 3 days ago
5 checks passed

Reviewers: psharma04 ✓, amandamcg1 ✓

Assignees: No one—assign yourself

Labels: None yet

Projects: None yet

Milestone: No milestone

Development: Successfully merging this pull request may close these issues. None yet

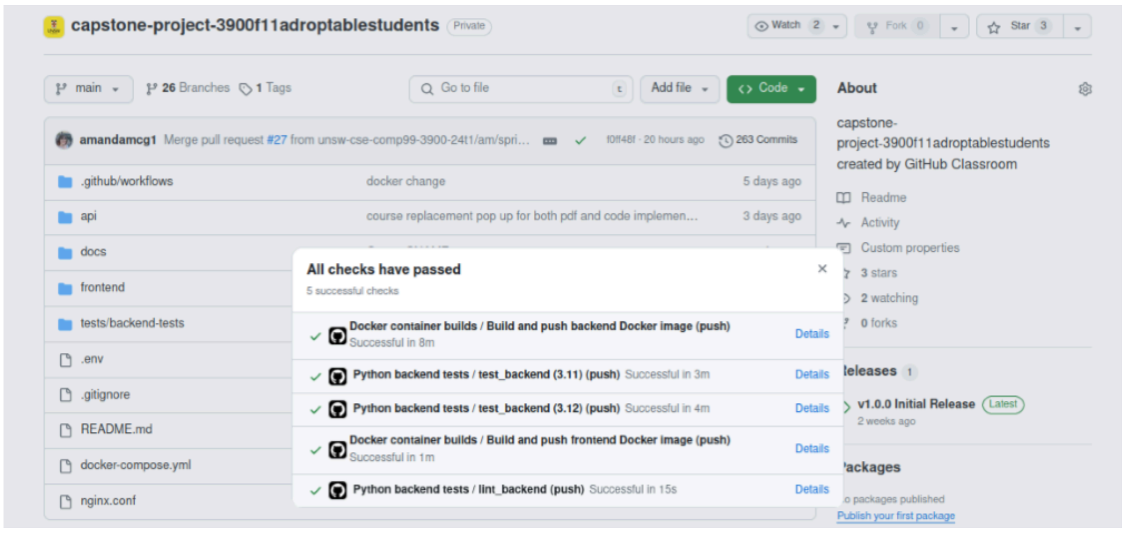
Notifications — Customize
Subscribe
You're not receiving notifications from this thread.

3 participants

Lock conversation

Figure 8: Pull Request With Review

Finally, our group employed CI/CD pipelines ensuring reliable delivery of high quality code. We set up CI/CD pipelines using GitHub Actions which automatically runs our tests whenever code changes are pushed to the repository. These pipelines provided immediate feedback on the integration of new code, enabling us to detect and address issues early.


capstone-project-3900f11adroptablestudents Private

amandamcg1 Merge pull request #27 from unsw-cse-comp99-3900-24t1/am/spri... ✓ f0f48f · 20 hours ago — 263 Commits

.github/workflows	docker change	5 days ago
api	course replacement pop up for both pdf and code implemen...	3 days ago
docs		
frontend		
tests/backend-tests		
.env		
.gitignore		
README.md		
docker-compose.yml		
nginx.conf		

All checks have passed
5 successful checks

✓ **Docker container builds / Build and push backend Docker image (push)** Successful in 8m — Details
✓ **Python backend tests / test_backend (3.11) (push)** Successful in 3m — Details
✓ **Python backend tests / test_backend (3.12) (push)** Successful in 4m — Details
✓ **Docker container builds / Build and push frontend Docker image (push)** Successful in 1m — Details
✓ **Python backend tests / lint_backend (push)** Successful in 15s — Details

About

capstone-project-3900f11adroptablestudents created by GitHub Classroom

Releases 1
v1.0.0 Initial Release Latest
2 weeks ago

Packages
No packages published
Publish your first package

Figure 9: Github Tests Passing

References

1 Guidelines for Bloom's Taxonomy Usage For Instructional Design Background: Bloom's Taxonomy is a widely-recognized tool for. (n.d.). Oklahoma Baptist University. Retrieved August 4, 2024, from https://www.okbu.edu/institutional-effectiveness/documents/guidelines-for-blooms-taxonomy.pdf

Shabatura, J. (2014, September 18). *Bloom's Taxonomy Verb Chart | Teaching Innovation and Pedagogical Support*. Teaching Innovation and Pedagogical Support |. Retrieved August 4, 2024, from https://tips.uark.edu/blooms-taxonomy-verb-chart/

UNSW Course Outline Finder. (n.d.). UNSW Sydney. Retrieved July 5, 2024, from https://www.unsw.edu.au/course-outlines#search=&filters=year%3A2024&sort=relevance&startRank=1&numRanks=10

Utica University. (n.d.). *Bloom's Taxonomy of Measurable Verbs*. Bloom's Taxonomy of Measurable Verbs. Retrieved August 4, 2024, from https://www.utica.edu/academic/Assessment/new/Blooms%20Taxonomy%20-%20Best.pdf