

Skill Tracking and Recommendations for CodeWOF

Griffin Baxter

*Computer Science and Software Engineering
University of Canterbury
Christchurch, New Zealand
grb96@uclive.ac.nz*

Abstract—CodeWOF is a web application developed by the University of Canterbury Computer Science Education Research Group. The app’s purpose is to help school teachers maintain and revise their programming knowledge with the use of various coding exercises. At the start of this project, all programming questions on CodeWOF were laid out on a single page, which caused issues when users chose a question to attempt. The solution involved modifying CodeWOF to include descriptive tags and filtering for each programming question, tracking the proficiency of users’ programming skills, and using this information to suggest exercises for the user to perform. In this modified version of CodeWOF, two question recommendations are provided to users each time they navigate to the site, and are uniquely generated based on a given user’s tracked skills. As part of this solution, research was conducted on the teaching, refreshing, tracking, and recommendation of programming skills. After the modifications were completed, the improvements to CodeWOF were evaluated through a survey sent to digital technologies teachers around New Zealand. This evaluation showed that while most users did not find the original questions page to be too bad, almost all survey respondents found the new version of CodeWOF to be significantly better. Notably, users liked the filtering panel, the new questions page with question tags and filtering, and appreciated the question recommendations.

Index Terms—Computer science, programming

I. INTRODUCTION

CodeWOF is a web application developed in Python using the Django web framework by the University of Canterbury Computer Science Education Research Group (CSERG). The app’s purpose is to prepare school teachers by maintaining their programming knowledge with the use of various coding exercises, as discussed on the CodeWOF website [1]. Digital technology teachers in New Zealand can often encounter skill fade in regard to programming languages between teaching sessions. As such, CodeWOF provides the ability for users to work on short daily programming exercises, which aims to reduce the aforementioned skill fade. A study of open online courses, similar to CodeWOF, showed that this application of knowledge in exercises increases the memory retention of participants as desired [2]. Additionally, as shown in a study based on computing education [3], assessment of skills is the most powerful tool to promote the learning and teaching of novices in programming, which is what CodeWOF achieves in bite-sized activities.

At the start of this project, all programming questions on CodeWOF were laid out on a single page. This caused issues when it came time for users to pick a question to attempt, due to the large number of questions available on CodeWOF. The

project involves modifying CodeWOF to track the proficiency of users’ programming skills, and suggesting exercises for the user to perform. Its main objectives and milestones include: performing background research into how CodeWOF works to maintain skills; research of skill tracking; implementation of categorisation and tracking into CodeWOF; researching and implementing question recommendations; and conducting a survey, which was sent to digital technology teachers around New Zealand.

II. BACKGROUND AND OBJECTIVES

A. Parties Involved and Organisation

The Computer Science Education Research Group (CSERG), as implied by the name, is a research group for computer science education, known for CS Unplugged [4] and the Computer Science Field Guide [5]. This project is supervised by Tim Bell, the head of CSERG. Additionally, programming assistance is provided by Jack Morgan, the Project and Technical Lead for CSERG.

As CodeWOF was designed for teachers in New Zealand, they are also involved with the development of the app. Additionally, teachers have given CodeWOF to their students to use, further expanding the user base. The Digital Technologies Teachers Aotearoa (DTTA) association is also involved, which contains teachers of the subject from all around New Zealand.

B. Pre-existing results

Rebekah McKinnon, a final-year software engineering student from 2021, researched the categorisation of CodeWOF questions. This research was performed in terms of tracking the skill level of users. As part of this, filtering and achievement badges were added. The study concluded by showing that the usability of finding questions in CodeWOF had improved.

C. Objectives

In CodeWOF, all programming questions are laid out on a single page, with only some colour differentiation for exercise categories. This layout is not ideal, due to the large number of programming questions available. This project involves tracking the proficiency of users’ programming skills. From here, CodeWOF would be able to suggest exercises for users to choose from, based on their history of using the app. An example of this may be recommending questions based on programming concepts such as ‘while loops’ or ‘conditional statements’, where the user has not answered these types of

questions in a predetermined amount of time. The objectives of the project are shown in the following list:

- Perform background research into the teaching of programming skills, and refreshing such skills, including the methods used to make CodeWOF effective.
- Research the tracking of programming skills for use as part of the question recommendations system for CodeWOF.
- Integrate the previously researched and developed question categorisation and skill sets into CodeWOF, including resolving issues that were keeping it from being pushed into the production code, and manually tagging all existing questions with their relevant categories.
- Create a survey for New Zealand teachers on the usability and improvements with the implemented skill tracking and recommendation features, including a pilot study run by the teachers in the CSERG lab and gaining approval from the ethics committee.
- Implement the tracking of programming skills and levels for users of CodeWOF.
- Research into the recommendation of questions for users to choose from, using approaches based on the questions previously completed.
- Implement recommendations of CodeWOF questions to users as they use the app.
- Conduct the aforementioned survey for New Zealand teachers.

Additionally, stretch goal objectives were created, in case spare time was left to work on this project. However, due to time constraints, the extra objectives were not started, as listed:

- Research motivation models for getting users to keep coming back to CodeWOF to continuously sustain and improve their programming skills.
- Implement motivation models as part of the recommendation system for CodeWOF questions.

III. EXISTING SOLUTIONS AND RELATED WORK

A. Teaching Programming Skills

Computer science and the teaching of programming was introduced to New Zealand schools in 2011 [6]. Because of the relatively recent introduction, a majority of teachers did not inherently learn programming skills in school themselves. This makes the teaching of programming skills uniquely challenging.

CodeWOF aims to prepare teachers for digital technology courses; however, the way they use the app inherently changes how they impart programming skills to their students. A recent study shows that mastering programming fundamentals are best done by assessing skills in individual atomic elements [7]. CodeWOF achieves this by presenting questions that involve few atomic programming elements within answers that only require a small method or passage of code to be written. Additionally, CodeWOF includes Parsons problems, which only require the arrangement of pre-made code blocks [8].

As part of a study, it was shown that having more scheduled and automated feedback for programming assignments helped to improve each iteration of answering questions [9]. Another study showcased similar results in terms of overnight feedback for programming projects, of which a delay was used to mitigate procrastination [10]. CodeWOF is similar, in that it provides instantaneous feedback to questions. This helps teachers across every single iteration of question answering in the app.

Additionally, spoken language is an essential aspect of teaching programming. A study showed that even the trivial addition of keyword glossaries for exams taken by those from a non-English speaking background helped significantly in terms of the student's performance [11]. Relating to this, CodeWOF has abstracted the text on the site to allow for multiple languages to be easily implemented. Unfortunately, the languages themselves have not yet been implemented. One of CodeWOF's main goals is supporting the Te Reo Māori language, which would be especially helpful for New Zealand teachers to educate students through the Hangarau Matihiko curriculum [12].

As discussed, there are numerous studies on effectively teaching programming. Additionally, automated tools for the purpose of gaining programming skills, such as CodeWOF, are abundant nowadays. Furthermore, there are even tools that exist outside the realm of teaching object-oriented programming, where usage was found within query languages such as SQL. A study showed this by making the query language more engaging to students [13].

B. Refreshing Programming Skills

Teachers of the digital technologies curriculum in New Zealand are not always teaching programming throughout the entire school year. Even those that teach programming throughout the whole school year still have the summer break period, in which they are likely not to be teaching [14]. This means that teachers often need to refresh their programming skills.

Refreshing the knowledge of a particular programming language is also a valid use-case for CodeWOF, especially in cases where multiple languages are taught or used by a digital technologies teacher. A study that inspected Stack Overflow questions found that programmers often make failed attempts at relating a new programming language with what they already know [15]. This means that refreshing these programming skills is necessary, as the base knowledge from prior languages can be detrimental to a teacher's mental model. However, this may be limited in CodeWOF's current implementation, due to only supporting questions in Python.

In terms of skill refreshing outside of programming, a study was performed on combatting skill fade within a remote military environment [16]. The study's experiment concluded that regularly scheduled refresher training was an important tool for maintaining the crew's skills. Another study showcased similar findings, in which scheduled refresher interventions were demonstrably useful tools for counteracting skill decay [17].

Both of these studies correlate to the success of CodeWOF's intended use-case, as a way to maintain skills over regular usage.

C. Tracking Programming Skills

A study showed that tracking students' performance in a basic programming class could predict their performance in a data structures course [18]. This shows that tracking is helpful for showing what the student needs help with in the present and what they will need help with in the future. However, this is not always the case, as the study was unable to reliably predict the students' performance as they enter the advanced-level programming course. Being able to teach programming skills requires a wide range of programming knowledge, which, as shown in the study, can be facilitated by tracking skills to find out areas that could use improvement.

D. Recommending Programming Questions

A study looked into the benefits of Yahoo answers having a recommendation system to match users with relevant questions to answer [19]. The approach organised content and social signals into channels as part of the recommendation process. The content signals relate to the text and categories of questions or answers, whereas the social signals relate to the user interactions regarding asking or answering questions. Good performance was demonstrated from the model developed in the study, however, it was tested on a large data-set, which makes the model not entirely viable if used in CodeWOF. Another study involved using an exam question recommendation system, which uses content-based and knowledge-based recommendation techniques, similar to the content and social signals, respectively [20].

Similar to the Yahoo answers study, a question recommendation engine used modelling to estimate user interests to match users to questions [21]. This recommendation mechanism was able to provide discrete priorities for each question, which was used as part of their question load balancing between users.

Another study used a question recommendation system to personalise job interview questions [22]. While not related to programming, the system was demonstrably efficient and effective for selecting job talent. However, this made use of a complex neural network, which also used data from over 10 billion "click-throughs". As there is not a copious amount of data from CodeWOF to use, such an approach is unfeasible for recommending programming questions.

One study made use of an Intelligent Tutoring System (ITS) to develop a system for helping students learn the C# programming language [23]. A suitable ITS, that does not teach but instead helps users maintain their programming skills, may be appropriate to use in CodeWOF. However, it would also require a significant time investment that is likely to be more than what is possible with this project.

IV. PROPOSED SOLUTION

A. Design and Implementation

CodeWOF is developed in Python using the Django web framework. This includes the use of HTML and JavaScript. Many of the HTML components used Bootstrap, allowing CodeWOF to have a responsive design. Docker is used to run the development proxy using the Windows Subsystem for Linux (WSL).

1) *Question Categorisation and Skill Sets*: Continuing from last year's project, the previously researched and implemented question categorisation and skill sets were incorporated into the latest codebase of CodeWOF. The previously implemented features were new question tags for difficulty levels, programming concepts, and question contexts. As part of this, the questions page features a filtering pane for each question tag. Comments and issues mentioned on the pull request from last year's project were addressed, which primarily involved issues around code quality. One of the review comments mentioned that only a small portion of the questions had been tagged with the new filters, so I went through more than 100 further questions and manually tagged each one, as per the filtering criteria. This did not affect the functionality of the existing questions, as it simply added tags to them. Additionally, unit testing was added regarding the new features and codebase changes, as these were not present in last year's project.

I also simplified the checkbox selection on the filtering panel regarding the hierarchy of tags. Child tags are indented in the filtering panel, as shown in Fig. 1. In the original implementation, selecting a parent tag did not select the subsequent child tags. This was resolved by creating a script to automatically select the child tags when a parent tag was selected.

From here, I showed Tim and Jack the changes made. Tim gave feedback on the default question order, mentioning the randomised ordering being confusing. Because of this, the default order was set to be in terms of difficulty, from easiest to hardest. Jack gave feedback on adding tags to each question block, as there was no way to tell which tags belonged to which question, except with the use of the filtering panel. Using this feedback, and incorporating tag styling such that it blends in with other CSERG sites, such as DTHM for kaiako [24], allowed the tagging to give significantly more detail to the user. An example of the finalised question block can be shown in Fig. 2.

2) *Skill and Level Tracking*: Skill and level tracking was implemented in CodeWOF as part of the question recommendation system. This will ensure that the question recommendations are relevant to each user, in terms of how they have answered questions, and which questions they have answered.

A new set of utility functions were created for tracking the skills and levels of CodeWOF users. The functions ultimately output a dictionary containing the quantity of correctly answered questions, and the number of attempts required to successfully answer the questions, for varying types of question categories. The question categories included as part of this are

Filter Questions

Difficulty level

☐ Easy

☐ Moderate

☐ Difficult

☐ Complex

Question type

☐ Debugging

☐ Function

☐ Parsons

☐ Program

Concepts

☐ Display Text

☐ Functions

☐ Inputs

☐ Conditionals

☐ Single Condition

☐ Multiple Conditions

☐ Advanced Conditionals

☐ Loops

☐ Conditional Loops

☐ Range Loops

☐ String Operations

☐ Lists

Contexts

☐ Real World Applications

☐ Mathematics

☐ Simple Mathematics

☐ Advanced Mathematics

☐ Geometry

☐ Basic Geometry

☐ Advanced Geometry

Reset

Filter questions

Figure 1. The filtering panel used in CodeWOF on the questions page.


 *Program*
Rectangle Area

Figure 2. An example of a question block in CodeWOF on the questions page.

difficulty levels, programming concepts, and question contexts. Types of question categories include the “Easy” difficulty, and the “Functions” concept. The scores are gathered based on questions answered in the past month, as well as those answered at any time. This is so that data of recently answered questions can be made use of, in which questions answered many months ago may not be relevant due to potential skill fade, as mentioned earlier in this paper. Unit tests were written to validate the created functions, based on various types of inputs in terms of questions that are eventually answered correctly.

It was decided against using time tracking as part of this milestone. As discussed with Tim, while it would be useful to know how long a user spends on particular types of questions or attempts, such as being able to tell whether a user is struggling with a question, this type of tracking is prone to outlier results. The user may have left their CodeWOF tab open whilst doing something else, and many questions inherently take longer than others to complete, both of which would lead to results that are unproven in terms of their usefulness. This is opposed to tracking how many attempts it takes for a user to answer a question, where each attempt

made by the user intentionally shows that they believe their answer is correct, by the fact that they pressed the “Run code” button.

3) *Question Recommendations*: The recommendation of programming questions was implemented in CodeWOF, with help from the skill and level tracking. These tracked skills were used to improve the delivery of appropriate and helpful recommendations to CodeWOF users.

While the studies from the “Recommending Programming Questions” research are helpful in creating a recommendation engine for programming questions, modifications would need to be made to suit CodeWOF. The primary concern regards the aforementioned studies using recommendations to suggest similar items. Whereas with CodeWOF, it is instead desired to recommend a diverse range of questions, not just similar questions to those the user has already completed successfully. As well as this, the proposed methods of the studies can lead certain users into diversity-reducing feedback loops caused by the recommendation algorithm, as shown in a study about news readers [25]. Because of this, I am proposing that the following solution for question recommendations in CodeWOF be suitably referred to as a “reverse recommendation engine”.

The reverse recommendation engine works by making use of the aforementioned skill and level tracking. From these tracked skills, a set of functions have been developed to create a dictionary of scores. These scores intend to denote how well a given CodeWOFF user performs at answering certain question types, which is a crucial metric used as part of the recommendation engine. Each score corresponds to a particular type of question category, each from questions answered in the past month or at any time period, just like the tracked skills and levels.

Each score, which corresponds to a category type such

as the “Easy” difficulty, is only calculated if a respective question exists that has been answered by the user. Otherwise, the category type is not assigned a score. Using the formula shown in eq. (1), each score is assigned a numeric value. The symbol s_t indicates the score for a given category type and time period, t . Additionally, t_c and t_a refer to the number of questions correctly answered and the number of attempts for each completed question, respectively, for the given category type and time.

$$s_t = (-t_c) + (\text{avg}(t_a) - 1) \quad (1)$$

The equation intends to yield a low or negative score when the user is performing well at answering the given type of question. This is influenced by the first part of eq. (1), where each correctly answered question removes a value of one from the score. This is due to the number of correctly answered questions showcasing how many times the user was able to answer a question of a given type, and thus the less of a need for the user to practice that type of question.

Whereas, if a user is struggling with a particular question, the intent is to retrieve a high or positive score. This is portrayed in the latter half of the equation, in which the average number of attempts taken to answer each question is added to the score. Additionally, a value of one is removed from eq. (1), referring to the first question attempt, which is unnecessary to influence the score, as each correctly solved question takes at least one attempt. The more attempts it takes for a user to answer a particular type of question, the more it is likely to portray that the user is finding said question more challenging, as they are generating more and more incorrect answers.

The scores, alongside a list of all unsolved questions, are used to ultimately calculate a user’s recommended questions. Two question recommendations are shown to the user, each with its own meaning and description, as shown in Fig. 3.

The first question recommendation, which will be referred to as “recommendation A”, aims to retrieve a question that is at a difficulty level that the given CodeWOF user is accustomed to. Recommendation A, as shown on the left in Fig. 3, will prefer questions outside the concepts and contexts they are used to. This recommendation’s target audience intends to encapsulate those that want to maintain their programming skills with a question at a comfortable difficulty, with the addition that they may want to branch out in terms of the programming concepts or question contexts they are used to answering.

In contrast, “recommendation B”, as shown on the right in Fig. 3, aims to recommend a question at a slightly higher difficulty level than what the user is likely used to. Recommendation B will also prefer concepts and contexts the user is accustomed to. The users interested in this recommendation would be those looking for a challenge, where they would be attempting a harder question than what they are used to. At the same time, concepts and contexts that the user has provably answered well are incorporated into recommendation B, to ensure that the question is not unreasonably difficult.

The purpose of giving two recommendations to the user is to give the user control in terms of which type of question they are looking for and what they want to get out of the question. Recommendation engines for many social media sites, such as YouTube’s sidebar recommendations on videos [26], often “guess” as to what the user wants to watch, without any explicit choice from the user.

Furthermore, each recommendation’s descriptions were carefully written to not sound harsh, or depict them as flawless recommendations. For example, recommendation A uses the word “likely” to portray the fact that the question is not guaranteed to be easy for the user. If harsher wording were used, indicating that the question should be easy for the user to solve, and they end up finding the question challenging or unsolvable given their current knowledge, this has the potential to needlessly discourage the user from using CodeWOF’s question recommendations. Additionally, the “likely” wording shows that the recommendation may not be entirely correct in terms of the question’s difficulty and the user’s current skills. This is, ideally, showing the user that the recommendations are only there to assist them, and are likely to not be generating the most reliable recommendations possible.

The two questions generated from the recommendations use the scores allocated to each category type. Comfortable and uncomfortable categories are part of the terminology created as part of the logic for finding suitable questions to recommend in CodeWOF. A comfortable category refers to a category that the CodeWOF user is accustomed to. For example, recommendation A would aim to retrieve a question with a comfortable difficulty, while also making use of uncomfortable concepts and contexts. A similar situation applies regarding recommendation B, but with the opposite logic. Because each difficulty level can be ranked, in terms of easiest to hardest, and concepts or contexts cannot be ranked in such a manner, their respective comfortable and uncomfortable properties are calculated in different manners.

The most comfortable difficulty is assigned as shown in the flow chart in Fig. 4. As shown by the flow chart logic, the easiest difficulty is set as the most comfortable when scores are not assigned to the difficulties, in cases where there are no questions answered by a given CodeWOF user. This is to ensure that no assumptions are made in terms of how well a user performs at answering CodeWOF questions, and to start them with a question difficulty that is expected to be comfortable to the user. Otherwise, if possible, the most comfortable difficulty is set to the hardest difficulty with a score less than or equal to zero. This is done to ensure that questions are recommended at a difficulty that the user has been demonstrably able to answer effectively, as shown by the low score. In the case where only higher scores are assigned to difficulties, one difficulty level less than the easiest with the said type of score is assigned as the most comfortable difficulty. Additionally, as shown in Fig. 4, this is only assigned with one difficulty level reduced in cases where it is not already the easiest difficulty. The reasoning for this is that while the CodeWOF user has proven themselves to be

Your recommended questions

We recommend doing one or two questions per day, to maintain your programming skills over a long period of time.

Want to maintain your skills?

This question is likely to be at a difficulty you are accustomed to, while making use of a wide range of programming concepts and question contexts.

☐ *Debugging*
Ticket Calculator

Easy Functions Multiple Conditions

Real World Applications

Are you up for a challenge?

This question is at a higher difficulty level than what you are likely to be comfortable with, while utilising concepts and contexts you are well versed in.

☐ *Function*
Double Evens

Moderate Functions Single Condition

Simple Mathematics

Figure 3. An example of question recommendations in CodeWOF on the dashboard page.

able to answer such a question, due to the scoring having been set, the high score indicates that the user is struggling with the given type of question. Therefore, setting the comfortable difficulty to a level lower, intends to remediate the impact of the higher score as part of the question recommendation.

Calculation of the most uncomfortable difficulty is generated using the output of the comfortable difficulty logic. Simply, this uncomfortable difficulty is set to one harder difficulty level than the most comfortable difficulty level. Otherwise, if the most comfortable difficulty is already the hardest difficulty level, this is set to equal said difficulty. The reasoning for this was to apply a challenge that is adaptive to the user's most comfortable difficulty, and to ensure that an outrageously hard question is not delivered to the user, where only a single difficulty level lift is applied.

The most comfortable concepts or contexts are assigned in sets. For example, the calculation of the most comfortable concepts would generate a set of one or more concept types. This is because, as mentioned earlier in this section, the calculation of comfortable and uncomfortable concepts or contexts are performed in a different manner to difficulties, due to not being comparable in terms of their ordering. Because of this, the most comfortable concepts or contexts are assigned by gathering the set of concepts or contexts with the lowest score. Of course, if there are no scores assigned, a set of all concepts or contexts are assigned as the most comfortable. This logic is simply reversed for uncomfortable concepts or contexts, where categories with unassigned scores are assigned as the most uncomfortable. This was done to consider a mix of concepts or contexts, while also prioritising the scoring

assigned. Additionally, the aforementioned concepts or contexts with unassigned scores are especially useful, where these indicate question types that a given user has not attempted.

Both the comfortable and uncomfortable categories are calculated in a prioritised order. The same logic applies for finding said category types, and progression through the ordered list presents each next-most comfortable or uncomfortable category type. The reasoning for this is that questions conforming to a set of given category types may not be available, due to the questions of a given type having already been answered by the user, or the fact that such question does not exist with all attributes applied.

Extending this reasoning, comfortable and uncomfortable categories are calculated using tracked information from the past month, in order to receive the most up-to-date information on how a given CodeWOF user is performing. However, suppose there is no information provided in the tracked data from the past month. In that case, this is substituted for the tracked data from any time period, and the relevant categories are generated using this new information.

Finally, using the comfortable and uncomfortable categories as discussed, question recommendations are generated by searching through questions that the user has not answered. Iterating through the ordered list of category types to find a suitable recommendation ultimately generates a list of suitable questions for each recommendation. From each of these lists, a random question is picked as the recommended question. This is because each question in the list is equally sufficient as a recommendation in terms of its attributes, and so randomly recommending a question from each list allows for some

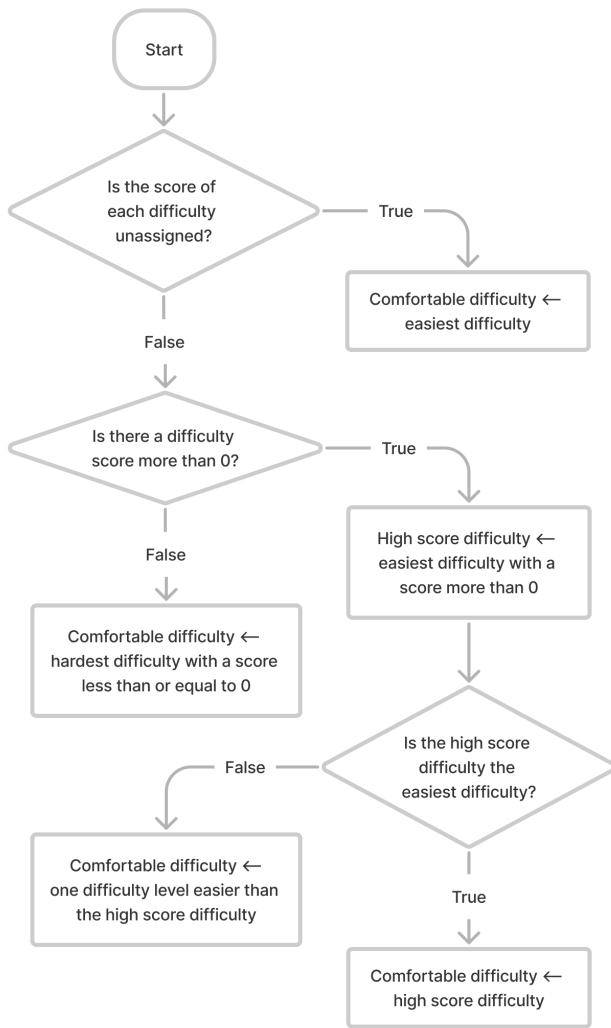


Figure 4. A flow chart depicting the assignment of the most comfortable difficulty, as part of the CodeWOF question recommendation engine.

variety in the questions, without being a detriment to the provided recommendations.

The question recommendations are displayed on both the dashboard, as shown in Fig. 3, and at the top of the questions page. The dashboard previously showcased the daily questions, which were entirely randomly chosen questions for a day, and thus replacing this with the recommended questions was a good fit. Displaying the recommendations at the top of the questions page was done to give an easy way for users to pick a question, further adding to the proposed enhancements brought by the question tags and filtering.

As part of the creation of this recommendation system, unit tests were written to validate the functionality, based on multiple types of inputs in terms of potential tracked data that can be generated and used as part of the recommendation engine.

4) *Evaluation:* As part of the process, a study was designed in order to evaluate the question categorisation and skill sets,

skill tracking, and recommendation features of CodeWOF. The questions involve basic background information, such as how often the teachers teach programming, and questions based on interacting with the current and updated versions of CodeWOF, including the filtering panel, tags, and question recommendations. This study was piloted with teachers in the CSERG lab. I received feedback about some questions being unnecessary, which I cut down in order to reduce the time commitment required to complete the survey. I also received feedback about making sure to explain what tags are in regard to CodeWOF questions. To resolve this, I added a sentence before the question in reference to an explanatory screenshot, as to avoid any possible confusion. The questions from the finalised survey can be found in Appendix E.

From here, an approval form to conduct the survey was sent to the University of Canterbury Human Research Ethics Committee. After this was approved by the ethics committee, the final survey was created using Qualtrics. The survey was sent out to those in the DTTA with eight respondents, including responses from teachers in the CSERG lab.

As part of the survey's ethics committee approval, the raw survey response data was required to be destroyed upon completion of the project. Because of this, the survey data is not available in this report or its accompanying appendices. However, the survey evaluation conducted as part of section V of this report was validated by Tim, prior to destroying the survey data. This validation was done alongside the raw survey data, to ensure that the analysis was an accurate and correct portrayal of the survey participants' responses.

B. Method and Project Management

Background research, using the literature review method [27], was primarily conducted using searching keywords using tools such as Scopus and Google Scholar. I also looked for recent papers from computer science education conferences, such as SIGCSE, ITICSE, WiPSCE, and ACE. Additionally, Tim sent through any papers or articles that they considered relevant.

As mentioned in the proposed solution section of this report, an evaluation of the CodeWOF additions and changes was conducted in the form of a survey with New Zealand digital technologies teachers from the DTTA mailing list and amongst those in the CSERG lab. The mailing list contains over 1000 teachers from around New Zealand. Once approval was attained from the University of Canterbury ethics committee, and the CodeWOF development work for the project was completed, the survey was sent out to the DTTA mailing list in multiple posts. The first post outlined the project and the survey, and the second post was sent at a later time to remind people to fill out the survey.

Weekly meetings with Tim were conducted in order to keep the project focus on track, and to ensure that the milestones set had been met. Any questions that I wanted to discuss with Tim over a voice conversation were also made here. Notes were made of these meetings using Google Docs, as shown

in Section A of Appendix A. Additionally, I had access to contact Tim via Slack.

As well as frequent contact with Tim, I was often in contact with Jack for any queries throughout my development work on CodeWOF. Jack, alongside Tim and the teachers in the CSERG lab, were there to validate any development changes I made to CodeWOF in relation to the set milestones. More specifically, Jack was able to validate the code-specific changes in terms of its quality and efficiency.

Clockify was used for logging and tracking the time I spent on this research project throughout the weeks, as shown in Section B of Appendix A. This was used to ensure that I am spending the required number of hours for a full-year 30-point course with approximately 10 to 15 hours per week.

Trello was used to track the tasks I needed to complete as part of milestones, as shown in Section C of Appendix A. While I had a milestone table created for a similar purpose, using Trello allowed for a Kanban-style [28] board showing my current tasks in columns representing “To Do”, “In Progress”, “In Review” and “Done” tasks. Tasks are moved to the “In Progress” section when they are being worked on, and eventually moved to “Done” when completed and verified by the relevant members of the CSERG lab including Tim and Jack.

The CodeWOF changes were tracked and created using the Git version control system. A fork of the official CodeWOF repository on GitHub was made for development. Once the development was completed at the end of the project, a pull request was created to merge the changes into the official repository.

As part of the assessment of code quality, unit tests have been created with each code implementation. This was done using Django’s built-in automated testing. Additionally, continuous manual testing of the application was done throughout the development process. The test data created can be found in Appendix C. As well as this, the CodeWOF user manual was updated for adding question tags, as shown in Appendix B.

Feedback has been given in meetings from both Tim and Jack. While validating my work for the milestone of question categorisation and skill sets, Tim gave feedback on the default question ordering. Additionally, Jack gave feedback on adding visible tags to each question block on the questions page.

V. DISCUSSION AND EVALUATION

The eight survey respondents indicated that they teach programming in some capacity, with approximations ranging from 20% to 80% of their teaching including programming, over the course of a year. This included teaching over a wide range of year groups, ranging all the way from years four to 13. Additionally, all but two respondents indicated that they teach the Python programming language, which CodeWOF uses for its programming questions, as opposed to other languages.

The first CodeWOF question in the survey asked whether users found the questions page easy to use for finding questions, using the current version of CodeWOF. The respondents

indicated that most users did not find CodeWOF’s existing questions page to be too bad. As part of this, a couple of respondents mentioned that they liked the colour coding. Amongst the criticisms of this page, one survey participant mentioned that the question order was frustrating, and that it sometimes took four to five minutes to find a question they are able to complete. Similarly, someone else critiqued the fact that there is no organisation to help users find questions that practice specific programming skills. Additionally, another respondent mentioned that it would be helpful if the questions were divided into categories. While it was intriguing that users did not find the original questions page to be too bad, the original page was not scalable as more questions were to be added, which would inherently make the problems associated with finding questions on the original page worse.

The next part of the survey asked users to try the modified version of CodeWOF. As part of this, participants were asked whether they found the newly implemented filtering panel helpful. Every survey participant answered by indicating that they found the filtering panel helpful. Users enjoyed the filtering panel for a wide variety of use cases, including using it to filter for Parsons questions, difficulty levels, types of questions, and areas of mathematics. Notably, one user said, “I can pick out the questions which will assist, remind or challenge me”. However, one participant mentioned that it was not obvious that they could click on the filter questions bar to expand its details.

When asked again about the questions page, using the modified version of CodeWOF, almost all users found it easy to use or an improvement over the original page. A similar sentiment was expressed when asked about the tags under each question block. One respondent said “Colour tags help to instantly recognise and distinguish the ones I want”, and praised the Human-Computer Interaction (HCI) regarding the colours portraying the tag’s type. However, in opposition to this, one user found that the multi-coloured tags were overwhelming, and that there were too many for each question. Additionally, another survey participant suggested that clicking on a tag should go to the page for all questions with that tag.

The next part of the survey asked about the question recommendations, and almost all survey respondents found them helpful. One respondent said that it was good to have suggestions that were not simply randomised, which was the case in the original version of CodeWOF. However, some respondents mentioned they would need more time with the modified version of CodeWOF to evaluate the recommendations in-depth. Additionally, a couple of participants mentioned that they prefer to choose their own questions, and thus would likely not make use of the question recommendations.

Overall, when asked to compare the new version of CodeWOF to the original, almost all survey respondents found the new version to be significantly better. One user that appreciated the upgrades said, “I can practice what I’m learning and go back on what I’ve already learnt”. Another participant said, “With the addition of the categories and the filtering, I now feel confident that it will be easy to use in this way”,

regarding using the new CodeWOF with their students. The negatives stated about the new version were, for the most part, repeated from the earlier questions, including a comment about the multi-coloured tags being overwhelming.

The final question asked whether they will use CodeWOF. All respondents indicated that they would use it themselves, and five of the eight participants also stated that they would get their students to use it.

Due to the positive feedback from Tim, Jack, and the survey, this project's code was pushed into the main Git branch and deployed for everyone to use on the main CodeWOF site [1]. This was done after Jack approved the changes made as part of a complete code review, as shown in the GitHub pull request in Appendix D.

A. Limitations

The main limitation of this project is the fact that a long-term evaluation of the recommendation system was unable to be conducted due to time constraints. While the evaluation was able to retrieve initial impressions and thoughts of the new version of CodeWOF, a complete study that spans over months would be required to appropriately evaluate the question recommendations, and how they evolve over time.

Also, due to time constraints, the stretch goal objectives were unable to be met. While all of the concretely set objectives were met, it would have certainly been beneficial to go beyond the planned objectives in order to include motivational models as part of CodeWOF's new question recommendation system.

B. Lessons Learned

Through working on CodeWOF and amongst the CSERG team, this project allowed me to learn a substantial amount about computer science education in New Zealand. As a student that participated in numerous digital technologies subjects throughout high school, and used resources such as the CSERG Computer Science Field Guide [29], it was especially enjoyable to learn about how such resources were created for teachers in New Zealand.

VI. CONCLUSION AND FUTURE WORK

CodeWOF is a web application for preparing school teachers by maintaining their programming knowledge. Research into the teaching, refreshing, tracking, and recommendation of programming skills has shown to be effective. Using this, in tandem with the partially developed question categorisation and skill sets, allowed for a suitable solution to be developed, as outlined by the sentiment of the evaluation participants. As part of this, all of the planned concrete objectives were met. However, this excludes the stretch goals, which were unable to be met due to time constraints.

The solution includes a modified version of CodeWOF with descriptive tags and filtering for each programming question, tracking users' proficiency in terms of their programming skills, and using this information to suggest exercises for the user to perform. The survey, completed by digital technologies

teachers, showed that almost all survey respondents found the new version of CodeWOF to be significantly better. Notably, users liked the filtering panel, the new questions page with question tags and filtering, and appreciated the question recommendations.

As part of the positive feedback from Tim, Jack, and the survey, this project's code was pushed into the main Git branch and deployed for everyone to use on the main CodeWOF site [1].

Future work on CodeWOF, generated from the outcomes of this project, includes improving the generation of questions from the recommendation engine. Due to time constraints, a long-term evaluation was unable to be conducted into how the recommendations change over time. And as such, using data from such an evaluation would allow for both the tracking and recommendations of CodeWOF to be tuned, as the questions recommended to users change over time.

Additionally, as more questions are added to CodeWOF, tags should be dynamically applied to questions using the contents of their answers, instead of the manual approach which is currently required. For example, if a solution was automatically detected to require a "for loop", CodeWOF should be able to append the relevant "loops" programming concept tag to the question.

As suggested by a survey participant, clicking on a tag should go to the page for all questions with that tag. This suggestion makes sense, and should also be included as part of the future work for CodeWOF.

REFERENCES

- [1] "CodeWOF." [Online]. Available: <https://www.codewof.co.nz/>
- [2] R. Teusner, C. Matthies, and T. Staubit, "What Stays in Mind? - Retention Rates in Programming MOOCs," in *2018 IEEE Frontiers in Education Conference (FIE)*, 2018, pp. 1–9.
- [3] A. V. Robins, *Novice Programmers and Introductory Programming*, ser. Cambridge Handbooks in Psychology. Cambridge University Press, 2019, p. 327–376.
- [4] "CS Unplugged." [Online]. Available: <https://www.csunplugged.org/en/>
- [5] "Computer Science Field Guide." [Online]. Available: <https://www.csfieldguide.org.nz/en/>
- [6] D. Thompson, T. Bell, P. Andreae, and A. Robins, "The role of teachers in implementing curriculum changes," in *SIGCSE 2013 - Proceedings of the 44th ACM Technical Symposium on Computer Science Education*, 2013, pp. 245–250.
- [7] A. Luxton-Reilly, B. A. Becker, Y. Cao, R. McDermott, C. Mirolo, A. Mühling, A. Petersen, K. Sanders, Simon, and J. Whalley, "Developing assessments to determine mastery of programming fundamentals," in *Proceedings of the 2017 ITiCSE Conference on Working Group Reports*, ser. ITiCSE-WGR '17. New York, NY, USA: Association for Computing Machinery, 2018, p. 47–69. [Online]. Available: <https://doi.org/10.1145/3174781.3174784>
- [8] Y. Du, A. Luxton-Reilly, and P. Denny, "A Review of Research on Parsons Problems," in *Proceedings of the Twenty-Second Australasian Computing Education Conference*, ser. ACE'20. New York, NY, USA: Association for Computing Machinery, 2020, p. 195–202. [Online]. Available: <https://doi.org.ezproxy.canterbury.ac.nz/10.1145/3373165.3373187>
- [9] P. Denny, J. Whalley, and J. Leinonen, "Promoting early engagement with programming assignments using scheduled automated feedback," in *Australasian Computing Education Conference*, ser. ACE '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 88–95. [Online]. Available: <https://doi.org.ezproxy.canterbury.ac.nz/10.1145/3441636.3442309>

- [10] D. Bouvier, E. Lovellette, and J. Matta, "Overnight feedback reduces late submissions on programming projects in cs1," in *Australasian Computing Education Conference*, ser. ACE '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 176–180. [Online]. Available: <https://doi-org.ezproxy.canterbury.ac.nz/10.1145/3441636.3442319>
- [11] R. Mason and C. Seton, "Leveling the playing field for international students in it courses," in *Australasian Computing Education Conference*, ser. ACE '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 138–146. [Online]. Available: <https://doi-org.ezproxy.canterbury.ac.nz/10.1145/3441636.3442316>
- [12] "Digital technologies and hangarau matihiko learning," Jul 2021. [Online]. Available: <https://www.education.govt.nz/our-work/changes-in-education/digital-technologies-and-hangarau-matihiko-learning/>
- [13] A. Kleerekoper and A. Schofield, "Sql tester: An online sql assessment tool and its impact," in *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*, ser. ITICSE 2018. New York, NY, USA: Association for Computing Machinery, 2018, p. 87–92. [Online]. Available: <https://doi.org/10.1145/3197091.3197124>
- [14] "Secondary Schools in NZ," Jul 2021. [Online]. Available: <https://parents.education.govt.nz/secondary-school/secondary-schooling-in-nz/secondary-schools-in-nz/>
- [15] N. Shrestha, C. Botta, T. Barik, and C. Parnin, "Here we go again: Why is it difficult for developers to learn another programming language?" in *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*, 2020, pp. 691–701.
- [16] D. Inglis, "Combating skill fade: basic life support refresher training in the remote military environment," *Resuscitation*, vol. 142, pp. e53–e54, 2019, RESUSCITATION 2019 - Controversies in Resuscitation: Abstracts. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0300957219303417>
- [17] A. Kluge and B. Frank, "Counteracting skill decay: four refresher interventions and their effect on skill and knowledge retention in a simulated process control task," *Ergonomics*, vol. 57, no. 2, pp. 175–190, 2014, pMID: 24382262. [Online]. Available: <https://doi.org/10.1080/00140139.2013.869357>
- [18] D. H. Wei and A. N. Burrows, "Tracking students' performance to assess correlations among computer science programming series courses," *J. Comput. Sci. Coll.*, vol. 32, no. 1, p. 9–16, oct 2016.
- [19] G. Dror, Y. Koren, Y. Maarek, and I. Szpektor, "I want to answer; who has a question? yahoo! answers recommender system," in *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 1109–1117. [Online]. Available: <https://doi-org.ezproxy.canterbury.ac.nz/10.1145/2020408.2020582>
- [20] H. Hage and E. Aïmeur, "Exam question recommender system," in *AIED*, 2005, pp. 249–257.
- [21] D. Hu, S. GU, S. Wang, L. Wenyin, and E. Chen, "Question recommendation for user-interactive question answering systems," in *Proceedings of the 2nd International Conference on Ubiquitous Information Management and Communication*, ser. ICUIMC '08. New York, NY, USA: Association for Computing Machinery, 2008, p. 39–44. [Online]. Available: <https://doi-org.ezproxy.canterbury.ac.nz/10.1145/1352793.1352803>
- [22] C. Qin, H. Zhu, C. Zhu, T. Xu, F. Zhuang, C. Ma, J. Zhang, and H. Xiong, "Duerquiz: A personalized question recommender system for intelligent job interview," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 2165–2173. [Online]. Available: <https://doi-org.ezproxy.canterbury.ac.nz/10.1145/3292500.3330706>
- [23] B. G. Al-Bastami and S. S. A. Naser, "Design and development of an intelligent tutoring system for c# language," *European Academic Research*, vol. 4, no. 10, 2017.
- [24] "DTHM for kaiako." [Online]. Available: <https://www.dthm4kaiako.ac.nz/>
- [25] B. Bodó, N. Helberger, S. Eskens, and J. Möller, "Interested in diversity," *Digital Journalism*, vol. 7, no. 2, pp. 206–229, 2019. [Online]. Available: <https://doi.org/10.1080/21670811.2018.1521292>
- [26] "YouTube." [Online]. Available: <https://www.youtube.com/>
- [27] H. Snyder, "Literature review as a research methodology: An overview and guidelines," *Journal of Business Research*, vol. 104, pp. 333–339, 2019. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0148296319304564>
- [28] D. Anderson, *Kanban: Successful Evolutionary Change for Your Technology Business*. Blue Hole Press, 2010.
- [29] "Computer Science Field Guide." [Online]. Available: <https://www.csfieldguide.org.nz/en/>

APPENDIX A PROJECT MANAGEMENT EVIDENCE

A. Meeting Notes

A screenshot of the meeting notes is shown in Fig. 5.

Griffin Baxter
SENG402 2022
Skill tracking and recommendations for codeWOF
Report: <https://www.overleaf.com/project/6243b364191594e8ae1c22a6>

19 May 2022

- Still working on draft interim report, milestone this week, then one week for adjusting (due last week of term)
- Aim for pilot study for next meeting, send first to Tim, then Tracy
- Changed ordering of query results - seems like randomised ordering could be useful

10 May 2022

- Went over Fabian's feedback
- Interface now uses colour/layout from DTHM site; consider being able to deselect individual filters, give some order to the list
- Ethics approval info: <https://www.canterbury.ac.nz/study/ethics/>

2 May 2022

- Fabian sent through some feedback on the project plan, and so next time we meet (probably next week?) I'd like to briefly discuss this with you (just minor things).
- I finished the milestone for "Conduct research into tracking programming skills", this ended up only being a couple of paragraphs in the report but I think the amount of information portrayed makes sense (I wanted to have room for the case where there was a lot to talk about).
- I have almost finished the milestone for "Implement the previously researched question categorisation and skill sets into CodeWOF".
- I have added the code changes that Rebecca made last year (but weren't added to CodeWOF officially) that I wanted to use for this project, and have put them in the fork/branch I am using which I posted above.
- I have fixed most of the problems outlined by Jackie Qiu on Rebecca's pull request (bugs, documentation, code smells), the remaining work left is to add unit testing.
- I have added difficulty, concept, and context tags to every question on CodeWOF (this took a while, only ~15 questions had these added, and I added these to 100+ further questions through manually evaluating each question).
- I'd like to add some further fixes to complete this milestone, including simplifying the checkboxes for the filtering (when an overarching filter is selected, all sub-filters are selected, or something similar).

Figure 5. A screenshot of the meeting notes.

B. Time Tracking

A screenshot of the Clockify project is shown in Fig. 6

C. Kanban Board

A screenshot of the Kanban-style Trello board is shown in Fig. 7.

APPENDIX B USER MANUAL

The user manual for adding questions, most notably the section for "Adding question tags", can be found here: <https://github.com/uccser/codewof/blob/develop/docs/adding-questions.md>

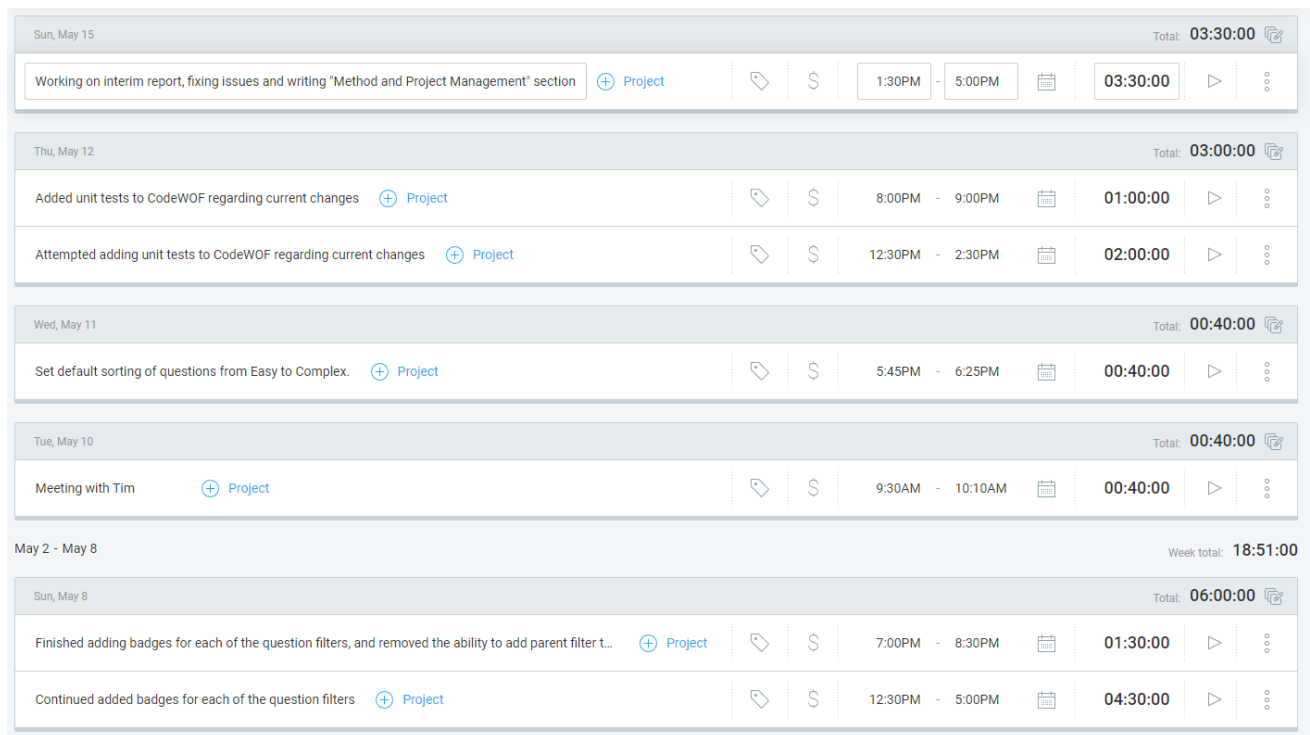


Figure 6. A screenshot of the Clockify project.

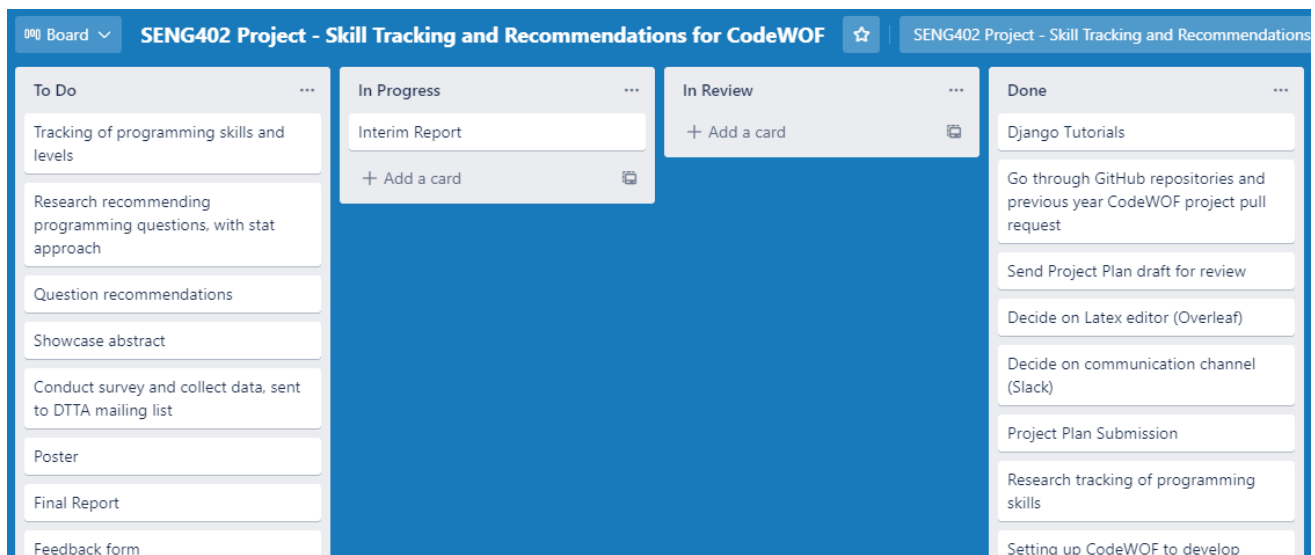


Figure 7. A screenshot of the Kanban-style Trello board.

APPENDIX C TEST DATA

The question data, with their respective difficulty, concept, and context tags, used for the skill tracking and question recommendations in CodeWOF can be viewed in YAML format here: <https://github.com/uccser/codewof/blob/develop/codewof/programming/content/structure/questions.yaml>

APPENDIX D PROJECT PULL REQUEST

The project's merged pull request, containing all of the code implemented as part of this project, can be viewed on GitHub: <https://github.com/uccser/codewof/pull/557>

APPENDIX E

SURVEY QUESTIONS

The following shows the questions present in the survey for this project. Note that the screenshots used as part of the Qualtrics survey are not included.

- Approximately, what proportion of your teaching involves teaching programming, over the course of a year?
- Do you teach programming in Python (as opposed to another programming language)?
- Which year levels do you teach programming to (if any)?
- Do you find the questions page easy to use for finding CodeWOF questions?
- *Note: Click the “Filter Questions” bar on the questions page to access the filtering panel, as shown in the screenshot below.* Do you find the filtering panel on the questions page helpful?
- Do you find the questions page easy to use for finding CodeWOF questions (*using the new version of CodeWOF, including the filtering panel*)?
- *Note: tags are shown in the coloured boxes at the bottom of each question block, e.g. “Moderate” or “Display Text”.* Do you find the tags within each question block helpful for picking questions in CodeWOF?
- *Note: recommendations can be found on the dashboard or the questions page, and are based on questions you have previously completed, how long ago they were completed, and how many attempts it took.* Would you find the question recommendations helpful in CodeWOF, and did you find them helpful during your time using the new version of CodeWOF?
- What are your overall thoughts on the proposed new version of CodeWOF, compared to the existing version (positives and/or negatives)?
- Will you use CodeWOF and/or get your students to use CodeWOF?