# SENG402 Research Project - Gamification of programming skill maintenance for teachers

Maree Palmer
University of Canterbury
Christchurch, New Zealand
mpa588@uclive.ac.nz

Tim Bell - supervisor
University of Canterbury
Christchurch, New Zealand
tim.bell@canterbury.ac.nz

## ABSTRACT

A new initiative from The Ministry of Education mandates that New Zealand schools provide digital technologies education in primary and secondary school curriculums from next year (2020). It follows that, in order to teach the new curriculum, teachers should possess basic programming skills. As these teachers may not specialize in programming, they would benefit from maintaining their programming skills over the course of the school year in order to teach the required modules to their students.

A solution to this problem is the CodeWOF website which encourages teachers to maintain their programming skills by solving daily problems in Python. In this project, I improved the CodeWOF website by adding elements of gamification. Gamification is where elements of gameplay are applied in an effort to increase user interactions with a product or service, such as a website. Two elements of gamification were added to CodeWOF to motivate teachers to practice their Python programming skills. First I added badges that users can achieve by attempting and completing problems. Additionally I added a points system which awards users points upon completion of a question or earning a badge. These elements are designed to encourage teachers to regularly use the CodeWOF website. Thus, teachers are more likely to maintain their programming skills to effectively teach the digital technologies modules provided by the new curriculum.

## CCS CONCEPTS

• **Applied computing → Education**.

## KEYWORDS

teaching, skill maintenance, gamification, website development

**ACM Reference Format:**
Maree Palmer and Tim Bell - supervisor. 2019. SENG402 Research Project - Gamification of programming skill maintenance for teachers. In *Proceedings of SENG402 - Software Engineering Research Project (SENG402).* ACM, New York, NY, USA, 17 pages.

## 1 INTRODUCTION

The Ministry of Education's new initiative mandates that New Zealand schools must provide digital technologies education in primary and secondary school curriculums from next year (2020)[5]. It follows, that in order to teach the new curriculum, teachers should possess basic programming skills[6]. As these teachers may not specialize in programming, they need to maintain their knowledge of programming over the course of the school year in order to teach the modules required to their students.

Last year, the 'BitFit' prototype was developed by Jessica Robertson. This prototype was a Django-based website that contained multiple Python questions along with simple progress trackers and point systems[10]. The point system and simple badge system were the only points of gamification present in her version of the project. The target audience for this website mainly consists of teachers who would be teaching Python as part of the standards set by the Ministry of Education.

The initial goal of this project was to integrate the prototype website developed last year with the DTHM4Kaiako website, a project by the Computer Science Education Research Group at the University of Canterbury[2]. This website contains digital technologies resources for New Zealand teachers. The project is now known as "CodeWOF", and this project incorporates all elements of last year's BitFit project in an independent website under the same name [1]. This will allow teachers already using the DTHM4Kaiako system to access the CodeWOF system to maintain their programming skills.

This project will be further improved by adding elements of gamification to improve user engagement and motivation. These elements include points for solving questions, and badges for attempting questions, solving questions, and consistently using the CodeWOF website. These elements of gamification will soon be available to users of the CodeWOF website. In addition to using gamification to increase user motivation, the impact of colours used in web design to stimulate a user's engagement was also studied and implemented. In addition to this, research will be performed to add the foundations of an Intelligent Tutoring System to aid users in identifying areas they need to improve on in order to maintain their skills.

In Section 2, I describe the background of this project in regards to teachers using it, and my goals for this project. Section 3 covers related literature with gamification in learning environments, and previous work done on this project by Jessica Robertson. Then, in Section 4 I describe in detail the design, implementation, and management process for this project. Section 5 contains my discussion and evaluation of my solution, as well as limitations in the project and a self-reflection on my contribution to the project. Finally, in Section 6 I conclude what work has been achieved in this project, and what future work could be performed for the CodeWOF system. In Appendix A, I provide evidence of my project management through my Kanban board, and Appendix B contains a user manual


*SENG402, 2019, University of Canterbury,New Zealand*

for CodeWOF. Appendix C shows the test data I used while manually testing my changes to the CodeWOF website, which is from the sampledata.py file in the repository. Appendix D lists currently available websites for teaching programming skills. Appendixes E and F contain the initial BitFit EER diagram and the updated EER diagram for badges respectively. Appendix G contains the currently available list of badges in CodeWOF, and finally Appendix H contains screenshots from the updated CodeWOF website.

## 2 BACKGROUND AND OBJECTIVES

The CodeWOF project is a staff-proposed project spearheaded by my supervisor Tim Bell. In addition, the Computer Science Education Research Group was heavily involved in this project. This group created the DTHM4Kaiako website, which is a website that provides resources to digital technologies teachers in New Zealand. The DTHM4Kaiako website was integrated with the CodeWOF website during the early stages of this project, and so the group provided valuable knowledge and advice for my further development of CodeWOF. In particular, Jack Morgan was my first point of contact for any technical difficulties I encountered. Additionally, the group acted as reviewers to my pull requests to merge my changes into CodeWOF's production branch. Other members of the Computer Science Education Research Group also aided in the project, acting as stakeholders to offer advice and opinions on the features the website should contain and how these features should work. From this, the main stakeholders from the project were defined as teachers using CodeWOF, the Computer Science Education Research Group, and myself as a developer.

This project was initiated in 2018 by Jessica Robertson under the name 'BitFit'. The goal of the project was to improve the Bitfit system used by students in the EDEM665 course (a course provided by the University of Canterbury for teachers learning how to teach computer programming). This system was originally based on CodeRunner quizzes (like those used in COSC121) and Moodle (the base for the UC Learn website), and was perceived by users to be "clunky"[30]. In addition, the site was not flexible or sufficiently extensible to add motivational, monitoring, or feedback features, all of which would be central to the project's final goal of a gamified, intelligent tutoring system. Jessica developed a prototype of the Bitfit system, which featured the base website where teachers could practice their Python programming skills. This website had the basic gamification features of a point system and a badge system.

The objective of this project was to fully gamify the Bitfit (now CodeWOF) system to encourage users to consistently use the website to maintain their Python programming skills. I performed extensive research and discussion with stakeholders about which gamification elements would be best suited for motivating users of the website. A points system and badges were the features that were deemed most suitable to improve CodeWOF, and so the base systems in the original BitFit project were extended and improved upon, both implementation-wise and visually. These elements were then implemented in the modified CodeWOF system for release at the end of the project. In addition, I performed other research into website design to determine which colour schemes and user interface elements would encourage regular use, and implemented these changes in the website.

The other main objective of this project was to integrate the BitFit system with the DTHM4Kaiako website created by the Computer Science Education Research Group. This involved working to integrate the website into the deployment infrastructure provided by DTHM4Kaiako. However, this goal later changed when the group decided that CodeWOF should be a separate website from DTHM4Kaiako, and the goal became developing CodeWOF as a fully independent website from DTHM4Kaiako, albeit with similar infrastructure in regards to deployment.

This project also had initial objectives of researching concepts in intelligent tutoring systems to aid CodeWOF's teaching potential. Utilizing an intelligent tutoring system in CodeWOF would actively provide help to users in maintaining their programming skills, which would make CodeWOF more effective at helping to maintain users' skills overall. These objectives involved researching and implementing "skill areas" for each question offered by CodeWOF. Skill areas would relate to skills such as programs, functions, loops, user input, and so forth. This would lead to research into recommending questions to users that targeted the skill areas the user had the least experience in. Research was also planned into a hint system to prompt users when they were struggling to answer a question. These components would have been the founding of an intelligent tutoring system for CodeWOF to aid users in maintaining their Python skills. However, these objectives were not achieved in the set timeframe, and will be discussed later in this report.

This project was also run in parallel to a study by Lucy Turner, a Computer Science student at the University of Canterbury. Lucy's study involved gathering information on how teachers use the CodeWOF website. Her research involved studying how often the users attempted questions, and the code the users submitted in their attempts. This is in contrast to this project, which focused on the technical side of setting up the CodeWOF website, and researching and implementing gamification features to encourage user motivation. However, the information gained from Lucy's study could be used in future to aid research into which features and questions help teachers maintain their programming skills best.

## 3 RELATED WORK

In the field of digital technologies education, continuous professional development is an effective tool to ensure that a teacher's skills are sufficient in order to teach a changing curriculum[34]. In New Zealand, there are a few efforts that I am aware of to support teachers with their own programming skills. For example, Margot Phillips is currently developing a program to tutor teachers in Python. This program is currently called MuscleGym, and provides in-person tutoring for teachers who teach Python. This course was created to be less isolating than online learning. In addition, it was designed to be an efficient course as teachers are very time-poor, and so require time-efficient methods of learning and retaining skills. In addition to MuscleGym, the Teaching and Learning Research Initiative (TLRI) is sponsoring research into aiding digital technologies teachers in discerning good code style, semantics, and syntax [36]. Their research focuses on quickly and efficiently providing teachers with the skills to identify and give meaningful feedback on common code style and syntax issues. This aspect of skill building and maintenance for teachers, rather than students, is

relatively rare in current research based around digital technologies teachers. CodeWOF is designed to fill this skill maintenance niche.

Gamification is an effective method of motivating users to perform tasks on educational websites, which is one of this project's objectives. In terms of motivation, Ryan and Deci devised three components of self determination theory - autonomy, competence, and relatedness[31]. If a person feels they have control over their actions, they have a satisfactory level of competence, and they can relate their skills to other people, they will display intrinsic motivation. Intrinsic motivation to complete tasks hence leads to learning. Richard Landers applied this theory further to state that greater intrinsic motivation in users could be achieved through gamification[17]. Gamified platforms increased user autonomy in their learning by allowing them to control and customise the content they interacted with, such as which modules to work on or how users ca customise their profile (such as badges or avatars). This motivation that was caused by gamification directly relates to this project's objective in motivating users to engage with CodeWOF, and so gamification should increase user motivation in this way.

There have been several studies into certain elements of gamification to determine their effectiveness in an educational setting. Barata et al conducted a study over a two year period involving gamifying a computer engineering course[4]. Elements used in this study were points (as "experience points"), a levelling system, challenges, achievements, and leaderboards. The use of gamified elements caused a significant increase in class engagement and participation. This motivational effect in an educational environment relates to this project's objective in encouraging consistent usage of CodeWOF by users, and so the gamified elements present in this study would work in favour of this objective.

In a study by Panagiotis et al., two programming classes learning Python were taught using separate approaches to measure their engagement and motivation [27]. One class used a traditional lecture and lab style, and the other class used a combination of Kahoot, Who Wants to be a Millionaire, and Codecademy to gamify the course. The class that used gamified content were more motivated overall, more engaged in their learning, and scored higher on average in exams than their traditionally taught counterparts who only attended lectures and labs on the subject. While this study covered skill learning rather than skill maintenance (which is CodeWOF's purpose), this increase in engagement and motivation could prove vital for users wanting to continue to use CodeWOF.

This research is supported by other studies into gamification-enhanced learning. A study by Nevin et al. focused on gamification elements incorporated into the learning of graduate medical students [24]. These students were millennials, so it was theorised that this made the students more open to technology based learning. The students that used the gamified techniques had better knowledge retention over time, which in turn supports the idea that gamification could aid skill maintenance. A case study by Ibáñez et al. followed a class learning the C programming language also using gamified elements such as badges, points systems, and leaderboards [15]. The study found that students mastered more concepts than students without gamified coursework, and the students working with the gamified course consistently performed more extra work than students who worked with the regular course. This shows a clear increase in learning engagement and motivation to improve when gamified learning took place.

Badges and achievements are elements of gamification that have been found to be effective in increasing motivation in multiple studies. Gibson et al. found that badges (alongside a point and leaderboard system) increased user engagement with the content, as the tangible reward of a badge for their work motivated users to continue to engage with the website[12]. A study by Santos et al. involved gamifying the SAPO Campus platform, a social media platform based on developing users' personal learning environments, by adding badges for users' achievements[32]. It should be noted that the platform was designed around social interactions, with aspects such as friends, blog posts, and shared content. Users of the gamified SAPO platform reported increased focus and motivation to complete various challenges when badges were awarded on challenge completion. The aspect of the social interactions in this platform also potentially encouraged users to earn badges so they could share their achievements with their friends as a symbol of recognition of their work. From these studies, badges were determined to be a good choice for an effective method to motivate users to regularly use CodeWOF.

Leaderboards and points were another prominent area of features in gamification research. Point systems were found in multiple studies to be a source of motivation for users for both investment in the website and for indicating a user's level of experience and interaction with the gamified material[22]. Leaderboards also were involved in a study by Landers and Landers to research how gamification using leaderboards improved student engagement and focus over the duration of a research project[18]. Students were tasked to research a topic and regularly update a wiki page with their findings. The use of a leaderboard encouraged students' regular participation in the project, indicating higher levels of engagement. However, Landers and Landers also noted that in order for a leaderboard to be an effective tool of motivation for users, all users must have a non-zero chance of placing highly on the leaderboard[18]. This requires an assumed general level of skill. For CodeWOF, we can not assume all teachers have equivalent skill levels as the digital technologies curriculum was only announced recently. Therefore, while points could work as a motivational tool for CodeWOF users, leaderboards are currently not a suitable solution.

While current implementation of gamified educational websites is plentiful, these websites tend to focus on teaching new skills, rather than maintaining previously obtained skills (see Appendix D for a list of websites designed for learning programming). The problem space for this project is based on the assumption that teachers would have some prior knowledge of programming in order to effectively teach their students. Therefore, other websites that promote learning rather than maintenance are not the optimal solution for this problem. CodeWOF was built purely for skill maintenance, and so is a specialised website to solve this problem specifically. Likewise, gamified education also focuses on a skill learning aspect, rather than a maintenance aspect, and so the currently gamified systems could be extremely effective for a learning environment but not meet the needs of CodeWOF's users. These two aspects are what make CodeWOF unique to fulfill the stakeholders' requirements over other existing solutions.

# 4 SOLUTION

## 4.1 Design

The University of Canterbury Education Research Group created the base of the CodeWOF website. This website was separated from the DTHM4Kaiako website that it was integrated with in the early stages of this project, although it still allows users to log in to CodeWOF using their DTHM4Kaiako account. In addition, this is when the name of the website changed from "BitFit" to "CodeWOF". This avoids the potential legal conflict with the company bitFit, which provides a platform for improving technical operational efficiency, as well as the fitness company FitBit. The name "CodeWOF" also gives a familiar sense of maintaining a skill to a certain standard like a warrant of fitness for a car.

CodeWOF was developed using multiple languages and frameworks, which were chosen in the previous iteration of this project. The back end of the website was created using Django, a Python based web development framework [10]. The advantages of using Django are, firstly, that it has an in-built admin website framework, which allowed for the CodeWOF admin features to be easily developed. As admin functionality was an important user story in prior iterations of the project, the admin features of Django would have greatly facilitated development for this functionality. Secondly, the fact that Django is Python based meant that it was a known language to any student who would continue this project in the future, as the University of Canterbury introduces its CSSE students to Python from their first year onwards. This increases learnability of the system, as other back-end web development frameworks such as Microsoft's ASP.net are very powerful, but in an unfamiliar language to most current students. Django also has very intuitive class modelling system with models directly connected to and updated in a Postgres database. This system allows for easily and quickly creating new models and extending existing models, and is simple enough to grasp in a very short timeframe[21]. As each student will only work on this project for one year, learnability is extremely important in this context. Thirdly, Django has several in-built modules that allow developers to easily add features such as user authentication [9], Postgres database usage, and the aforementioned admin interface. These modules facilitate fast development, which is crucial in a time-critical project. Finally, one of the primary stakeholders of this project is Jack Morgan, a member of the Computer Science Education Research Group. He often works with Django applications for his development in the group, and so could offer valuable technical advice for Django development, which is immensely useful considering the time constraints of the project.

The front end of CodeWOF was developed using a combination of HTML, CSS and jQuery. jQuery is a widely used Javascript framework designed for web development. One main advantage of jQuery is that it is cross-browser compatible, so the website would run on all browser types and versions. As teachers do not always have access to the latest technology, it is important that they can use CodeWOF regardless of their software limitations. It is also very easy to use multiple plugins with jQuery (similar to Django's multitude of accessible libraries), which facilitates rapid development. A potential downside of jQuery is that it is updated with new patches and releases very frequently, and is not backwards compatible. If an update of jQuery no longer supports features that CodeWOF relies on, it could require major changes to the website itself to fix[35].

In terms of deployment, the production deployment environment uses Google Cloud Platform to host the CodeWOF website. Requests from the website on the client side are sent through a load balancer, to ensure the server does not get overloaded with client requests. The load balancer then sends the client requests through to a Django instance of the CodeWOF backend running in a docker container within Google's AppEngine. The server always has two docker instances running so that a backup is available if one instance crashes, and the load balancer redirects requests to any free docker container. This process is scalable, so that if the website's user base grows larger, more docker instances can be made to handle more requests. These docker instances query the Google Cloud Database when needed. In addition, any requests from the user for static files (such as images) are sent directly to a Google Cloud Storage bucket. If the request requires more than just static images, the bucket redirects the request to the Django containers.

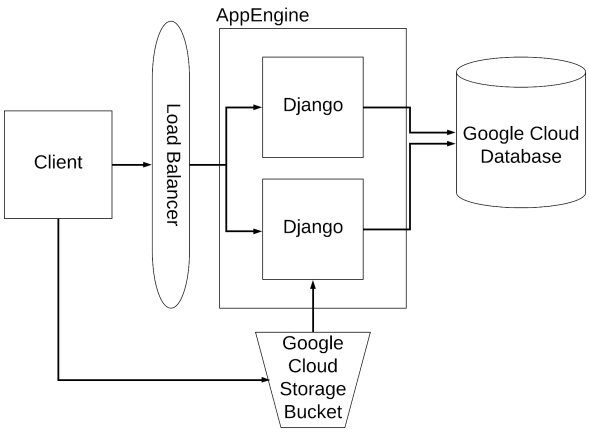


**Figure 1: The production setup of CodeWOF uses Google Platform Services.**

The local deployment setup mirrors production. It has requests go through Nginx , which acts as a load balancer for the setup [25]. Nginx also handles the static file requests, much like the Google Storage bucket. Nginx then communicates with a docker container running the Django server, and the server connects to a PostGres database when required.

Last year, the BitFit project was set up to connect to Sphere Engine to run and test programs entered by users. Programs were sent to Sphere Engine as AJAX requests, and the API returned the test results from the automated tests needed. This service requires a paid subscription to use, and a trial subscription was obtained for a lower price of $100USD per month. However, the subscription was not renewed by the product owners due to pricing ($300 USD per month for an education deal). In addition, questions and solutions did not need to be kept private or secure, which was a main feature of Sphere Engine, as they are accessible on the project's public GitHub repository.

A different solution was found in Skulpt [33]. Skulpt is an in-browser implementation of Python that allows scripts to be run

client-side. This removes the need for external AJAX requests as everything can be performed client-side. However, there is an issue that test cases and solution code can be found by inspecting the website on the client-side. I decided that this risk was acceptable based on two assumptions. The first is that most teachers using the CodeWOF site would be beginning to intermediate programmers, and so potentially they would be unaware of how to inspect the source code of the website to find solutions. The second assumption I made is the more important assumption, and that is that users of the website would be looking to improve their skills over time, and cheating by finding the answers would defeat the purpose of the website. Other solutions considered were a system similar to the CSSE Quiz Server (based on CodeRunner in Moodle[8]), where questions and solutions would also be stored securely in the server, due to these questions also being used for university course assessment. This solution was dismissed due to the perceived lack of need for security around questions and solutions, as again cheating using the solutions would defeat the purpose of the website.

## 4.2 Implementation

*4.2.1 Points.* The points system for CodeWOF was designed to motivate users to solve Python questions in CodeWOF to gain points. Points are stored as an integer in a user's Profile model. There are two main ways for users to earn points in CodeWOF - by solving questions, and by earning badges.

Initially, I developed the points system so that users could earn a maximum of 10 points for correctly answering a single Python question. Each time the user attempted a question, the attempt would be checked to determine if it was correct. After answering a question correctly, the user could not earn any more points for that question. If the user's answer was correct on their first attempt, they would be awarded the maximum amount of points. Otherwise, one point would be awarded to the user immediately for making an attempt. The user could earn a maximum of three points for three incorrect attempts. However, for each of the first three incorrect attempts for a question, the user's maximum would decrease by their number of incorrect attempts. For example, a user who unsuccessfully attempted a question three times, they would gain a maximum of seven points for the question as a whole, but would not be penalized further for any more incorrect attempts.

After discussing this approach with the Computer Science Education Research Group, as primary stakeholders of the project, I decided that this approach could potentially produce the opposite effect to the motivation I hoped for. If users decided that losing points for incorrect attempts was not worth it, they could potentially only attempt questions when they were sure to be correct. This defeats the purpose of CodeWOF, which is to practice and maintain skills by attempting questions. Therefore, a new points system was devised which did not deduct any points from a user's unsuccessful attempts. Instead, any points awarded or lost on attempts were removed entirely. In this new system, a user is awarded the base 10 points when solving a question. The user can gain an extra two points if they solve it correctly on their first try. This system is built so it does not discourage users who make mistakes in their attempts, but also rewards users who can determine the solution on their first attempt. The bonus points also act as an incentive for users to maintain their Python skills.

*4.2.2 Badges.* Badges were created using images from Icons8. A user earns badges by achieving certain milestones. Currently implemented badges include creating a profile, attempting questions, solving questions, and solving questions on consecutive days. These badges were designed to entice users into consistently using the website to maintain their programming skills, particularly the badge awarded for consecutive days with questions solved. Different tiers of badges (base, bronze, silver and gold) are awarded based on the number of questions solved, as shown in Figure 2.

**Figure 2: Badges are coloured monochrome, bronze, silver or gold depending on rank.**

The specific implementation of badges, and a user earning badges, were kept very similar to the original BitFit project. However, a few changes were made for the integration of CodeWOF with the DTHM4Kaiako system, adding icons for each badge, and implementing the aforementioned tiering system for badges. The original BitFit system used default Django User objects to represent users of the website. The original EER diagram for the BitFit system can be seen in Appendix E. DTHM4Kaiako uses a custom model that takes an email address as an identifying factor for a user instead of a username like in the default Django User model. The custom model was then used in the final independent CodeWOF website. This allows for a user to change email addresses if desired. The ability to change email addresses is important for teachers, as it was predicted they would be likely to use their work email addresses for the website. If the teacher changed schools, they would also change their work email address to their new school's domain. From this, the model was changed to allow for teachers to change their email addresses if needed.

A new field was added to the Badge model class from the original model. Originally, badges in the BitFit project did not have icons associated with them, and were just displayed as text. The string path to the icon resource is now stored in each Badge object, to link each individual icon with its corresponding badge. In addition, two extra fields were added to the Badge model to implement the tiering system. The first field represents the tier of the badge on an integer scale, with 0 being the lowest tier and incrementing tiers. Tier 0 is a special tier, which only contains the account creation badge, and all other badges have a base of tier 1. This field is used in calculating the number of points earned for achieving the badge. Integers were chosen for this tiering to facilitate the calculation of points for each badge. Enums were another possible option, but as badges have different ranges of tiers (i.e. there are more badges for consecutive days with solved questions than there are for number of attempts) integers were deemed as a much simpler ranking system. The other field added represents the next Badge object in the tiering

(the "parent"). This field is used for determining the highest ranked badge to be shown on the user's profile (with each parent being the next highest tier of badge). The updated EER diagram of the relationship between badges and user profiles is shown in Appendix F.

Currently, there are four types of badges offered in the CodeWOF system: account creation, number of attempts made, number of questions solved, and number of consecutive days with questions answered. These badges are designed to encourage regular use of the website, and encourage users to continue attempting questions to maintain their skills. Out of these four badge types, the creation badge is the simplest to award, as it is awarded on creation of a user account, with no other constraints. The attempts badges are determined by a single SQL query to the Postgres database for the website, retrieving the count of the Attempt objects stored through the user's profile. A loop then iterates through all badges with 'attempts-made' in the badge ID, and checks if the number associated with the badge is less than or equal to the user's number of attempts, awarding the user the badge if this condition is true. Question badges are awarded in the same way, with an SQL query on the number of Attempt objects linked to the user's account, where 'passed_tests' is true. SQL queries were used here as they can quickly and efficiently gather the data required to check if these badges had been achieved.

Consecutive day badges were the most difficult of the badges to implement, and several approaches to implementation were analysed before deciding on the final approach. The first thing that was considered was adding a new field to a user's profile for storing their current "streak" of days with questions answered. This would allow for relative simplicity when determining if a new badge has been earned. However, the addition of another field to the user's profile would over-complicate the profile model, and have it be responsible for too many operations. In addition, the field would need to be updated extremely frequently, which would be less efficient than all other options considered. This approach was hence discarded.

Another option explored was to create a new model called DayWithAttempt, which would store the user profile, the attempt made, and the day the attempt occurred. Days with questions answered would have been unique for each object linked with a user's profile. This model is similar to the LoginDay model from previous iterations of this project, though LoginDay stored days the user logged in to the website rather than days with questions answered. While this option was considered for the final option, I decided that new objects stored could be made redundant if I instead used an SQL query to find the exact information these objects would store. A query to find Attempt objects for the user's profile with unique dates would be much more efficient than storing potentially hundreds of DayWithAttempt objects for every user.

The final approach used was an SQL query to find unique days where users solved questions. This option was chosen as not only would it not require storing multiple new objects for each day with an answer, but it could also be run asynchronously as part of a helper function in an AJAX request. This way, the user's consecutive days with questions answered could be calculated without blocking the user interface, so the user wouldn't need to wait for the check to be made. However, this approach also posed technical challenges. On the Django server for CodeWOF, times and dates are stored in UTC (Coordinated Universal Time, equivalent to Greenwich Mean Time) as opposed to local time. This means that the main timezone used by the majority of our users (who are based in New Zealand) is 12 hours ahead of server time. This was a major issue for calculating consecutive days, as timezones being converted to UTC when saved in the Attempt model meant that users completing questions on consecutive days could not get credited for this due to timezone conversions. This was managed by converting Attempt datetimes to the timezone set in the user's computer settings.

The check for if a user has earned a consecutive day badge is similar to that of the attempt and question badges. Before checking if the user has earned any badges in this type, the helper function for calculating the user's current number of consecutive days with answers is run. The result from this function is stored as a local variable, so it is only calculated once overall, increasing the efficiency of the check. This variable is then checked against each numeric value for each Badge object the user can earn, with the badge being awarded if the variable is greater than or equal to each badge's threshold.

A full list of badges available on the CodeWOF website can be seen in Appendix G.

*4.2.3 User Interface.* When a user submits an answer for a question, an AJAX request is sent containing the user's submitted code, the user's data (their User model), and the question itself. This answer is then checked against the test cases for the question using, and an Attempt object is created containing the test case results. Test results are all checked in browser using Skulpt JS, rather than requests being sent to an external server. From there, the number of points the user has earned is calculated, and a check for new badges earned is performed. These points and badges are then added as objects in a result JSON object, along with the test results. This JSON object is sent back to the client.

As soon as the client receives the results from the sent AJAX request, the user interface updates to reflect three outcomes: test case results, points earned, and badges earned. To show the test case results from the user's inputted answer, each test case displayed on the website will update to be either red or green in colour - red for a failed test, and green for a passed test. This mirrors the common colour themes used in our society (red for "stop" or "bad", green for "go" or "good"). The question is considered solved when all test cases pass and are highlighted in green, and so a user would be able to easily check these results through this use of colour. Points earned are immediately shown on screen, as the user's total points in the website's navigation bar will automatically change to their updated total. Finally, any new badges earned will appear on screen as "toasts". Toasts are a form of notification in Bootstrap[19], a component library used in conjunction with HTML, CSS, and Javascript, all of which are languages used to develop CodeWOF. These notifications currently display the names of all badges earned from the user's attempt. Points and badges instantly updating is a form of instant gratification for users, and this motivates users to continue to seek more rewards for their work[23].

The AJAX request for answering a question is sent asynchronously from the client. This approach means that the user interface is still

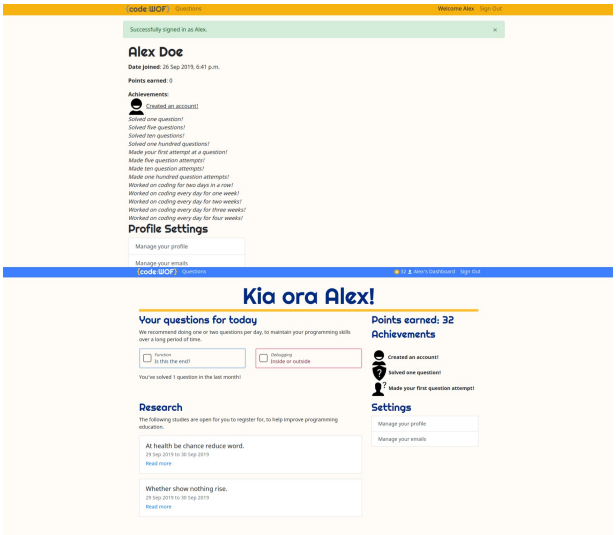


**Figure 3: The design of CodeWOF has changed drastically over the course of this project.**

usable and responsive while awaiting the results of the AJAX request, unlike with a blocking synchronous call. Using asynchronous communication is especially important for scalability. While points only involve checking attempts for a single question the user has answered, badges involved requesting and filtering all user attempts and questions. In particular, calculating the number of consecutive days a user has answered questions on is an O(n) iterative process, so the function will run for a longer time the more questions the user has attempted and solved. As one of the project objectives was to encourage regular use of the CodeWOF website, I expect that over time this function's runtime will increase proportionally to website usage. Therefore, it is important to ensure the user will not be hindered by these calculations, and so an asynchronous request is optimal for this situation.

The CodeWOF website was previously a bold yellow in colour. Research by Zainon et al. states that bright colours in a user interface should be avoided, and softer, darker colours should be utilized[38]. Thus, I decided to change the overall colour scheme of the website to ensure users would have a visual environment designed to aid them in maintaining their skills. The colour chosen for the website was a medium shade of blue (dodger blue). Zainon et al noted that blue colours in the periphery of a website was effective in catching a user's attention [38]. In a separate study by Dzulkifli and Mustafar, the more attention catching the visual stimuli, the better the memory retention [11]. Memory retention is an important factor of skill retention [14], and so blue was considered a good option to attract users' attention to aid this. In addition, Bonnardel et al. performed a study on users' memory retention when browsing a website. The blue themed website was not only liked by users, it also was the colour theme that led to the second highest level of information retention from users reading the content [7]. These studies led me to conclude that blue would be an optimal option for the CodeWOF website theme. The before and after images of the CodeWOF website are shown in Figure 3.

The other colour option that was considered over blue theming was orange theming for CodeWOF. In a study by Bonnardel et al., test subjects retained more information read on websites with an orange theme than any other colour, including blue themed websites[7]. In addition, warmer toned colours, such as orange, are more attention catching than cooler hues and so promote memory retention to a higher degree [11]. As memory retention is a factor in maintaining skills, this would give preference to warmer colours when designing the CodeWOF website. The downside of using orange theming was that this theming is very close with the Vehicle Testing New Zealand (VTNZ) website [37]. As VTNZ is the company in New Zealand that is the closest company linked with obtaining a warrant of fitness (WOF) for vehicles, we decided that the fact that the website was called "CodeWOF" was enough overlap between the two entities. Therefore, using the same colour themes would not be appropriate, and so blue was the best option overall to choose for CodeWOF's predominant colour for improving user engagement with the website.

Screenshots of the current CodeWOF website design can be found in Appendix H.

## 4.3 Project Management (Development Process)

Kanban was chosen as the methodology for this project, with the website KanbanFlow used to keep track of tasks[16]. Kanbanflow also allowed for time tracking on each task, using the inbuilt timing features of the website. The decision to use Kanban instead of Scrum was that once the BitFit prototype was integrated with the DTHM4Kaiako website, any changes made would be incremental improvements to existing systems. As Kanban works best to improve the efficiency of current existing systems, it was deemed most suitable for this situation [20]. Kanban is also suitable for a single developer, while Scrum was intended to be used with a development team. In addition, Kanban allows for flexibility on iteration times, unlike Scrum which has hard deadlines and commitments for each sprint. This flexibility in workload is crucial for the changeable workload of university, as it allows developers to work at their most comfortable speed at each time [28]. Kanban also allows for some expedited task flows. In a time critical project, the ability to expedite tasks to quickly complete them would be essential for if any high priority tasks related to the project arose. I chose the WIP (Work In Progress) limit to be three tasks, to allow for research and implementation tasks to occur simultaneously, as well as allowing for one extra task available for expediting if required. A screenshot of my Kanban board is included in Appendix A.

The setup of the CodeWOF repository changed over time. To begin with, the website was contained in the GitHub repository for the DTHM4Kaiako website, and my changes were added to this repository. Midway through the year, the decision was made to extract CodeWOF into a more independent website. This required the creation of a new GitHub repository for this website, and I then moved my changes for the gamification elements of the website manually from the DTHM4Kaiako repository. For both of these repository setups, I forked the main DTHM4Kaiako or CodeWOF repository, and created a branch titled "issue/gamification" in accordance with the naming conventions of the Computer Science

Education Research Group [26]. At the end of the project, when I was happy with my code, I merged the issue/gamification branch back into the "develop" branch, which was the main branch for the repository. I then created a pull request to merge the development branch from my forked repository back into the original repository. This pull request underwent a full code review from members of the Computer Science Education Research Group. The code review process acted as both a check to ensure my code complied with style guidelines provided by the group, and as an opportunity to receive feedback from the group on how I could improve my solutions, such as only showing the highest tier of a type of badge earned by the user, rather than all badges of that type to reduce needless clutter in the user's dashboard. Once my code was satisfactory to the group, my changes were merged into the main repository.

In terms of actually developing software for this project, I had issues running the project on my Windows machine. This was due to the custom docker build commands used in the DTHM4Kaiako website, and these proved incompatible with my machine. Docker for Windows was a potential solution to this issue, but Windows still did not run the correct commands in the Command Terminal, Powershell, or in an Ubuntu terminal. In addition, my personal laptop was not powerful enough to run a Virtual Machine that would run fast enough to be useful (due to a lack of RAM). This issue proved a blocker for most of Semester One in terms of development. Eventually, a solution was found which was to perform a dual boot system on my laptop, with both Windows 10 and Ubuntu installed. Implementing this dual boot also proved tricky, due to configuration issues, but proved successful in the end. I had a similar issue in Semester Two where my laptop that I had dual-booted with Ubuntu facing a critical system error that rendered the laptop useless. This issue was quickly solved though, as the replacement laptop I obtained had enough RAM to successfully run a Virtual Machine running Ubuntu, which I used for the remainder of the project. The full instructions on how to set up this project are included in Appendix B.

# 5 DISCUSSION, EVALUATION AND LIMITATIONS

## 5.1 Discussion of Implementation

When deciding which elements of gamification should be implemented, multiple options were researched and considered. While points and badges were the only elements implemented in this project, two other options were automated email reminders and leaderboards. These options posed significant technical challenges, or were deemed currently unsuitable for the project. Automated email reminders, while not strictly being specific to gamification, would remind users to complete their daily questions on CodeWOF to encourage users to maintain their skills. In a study by Lucy Turner (run simultaneously with this project), emails were sent out manually to users in a study group as a reminder to attempt questions. The number of user attempts per day since the release of the CodeWOF website is shown in Figure 4.

Emails to users were sent out on days 20, 24, 28, 32, 35, 39, 42, 46, and 49. As seen in the graph, The number of user attempts in a day tended to increase immediately after emails were sent, and therefore email reminders were effective in motivating these users to use CodeWOF. I initially planned to implement this feature in this project, however automating emails posed some technical difficulties. Automation required altering CodeWOF's deployment model by setting up infrastructure for adding a third party library to schedule emails. Two options considered were Celery, a Django package designed for task scheduling[29], and Google Cloud Tasks Service, a Google platform solution for task scheduling[13]. Adding this infrastructure was a task planned by the Computer Science Education Research Group, and the research required would be out of the scope of this project due to the amount of research and time it would take to discover the optimal solution for CodeWOF. Therefore, email reminders were removed from the scope of this project.

Leaderboards serve as an effective way to connect users, and can encourage users to attempt more questions in the hope of achieving better results than their competitors. However, after a discussion with stakeholders in the Computer Science Education Research Group, the inclination was that competition between users could potentially conflict with the intended intrinsic merit of maintaining skills. Therefore, this option was discarded. A similar option that I considered was the option to have "friend groups", where a user could join a group with other users and motivate each other to use the website more often. This would prevent the sense of isolation that often occurs during online activities, while not introducing a competitive element. However, I decided that the research required to determine whether this type of multi-user interaction would truly benefit CodeWOF users would be too time-consuming for this project, and so this is an option to extend CodeWOF in the future.

When beginning development for CodeWOF, I did not expect that the website would be used by a large number of users. However, on release, the website proved to be far more popular than expected, with over 180 users who have collectively made over 8000 question attempts between mid-August and early October (the website's release date and the current date of writing). With the website having been available for less than two months, I estimated there would be 2000 question attempts at most in that time. One of the main reasons that the user base was so much larger than expected was that teachers who used CodeWOF occasionally introduced the website to their students in classes. Students of teachers using the CodeWOF system were not stakeholders I initially considered in this project. It would not be unrealistic to assume that this pattern of class usage could continue in future, which makes the potential user group much larger than the expected group only involving teachers. Therefore, the measures I took to ensure that point and badge calculation would be scalable for a large number of user attempts were justified, as the website could continue to grow in usage.

A significant challenge I faced during the development process was the initial project setup. As the infrastructure of the DTHM4Kaiako website, which CodeWOF was part of for a short time, required a Linux environment to run, I was ill-equipped to begin the project. The lab machines at the University of Canterbury did not allow me the correct user privileges to set up the environment either, though they used the required operating system. I also did not have the resources to run an Ubuntu virtual machine powerful enough for development on my laptop, and initial attempts at dual-booting failed. These difficulties meant that I was blocked from

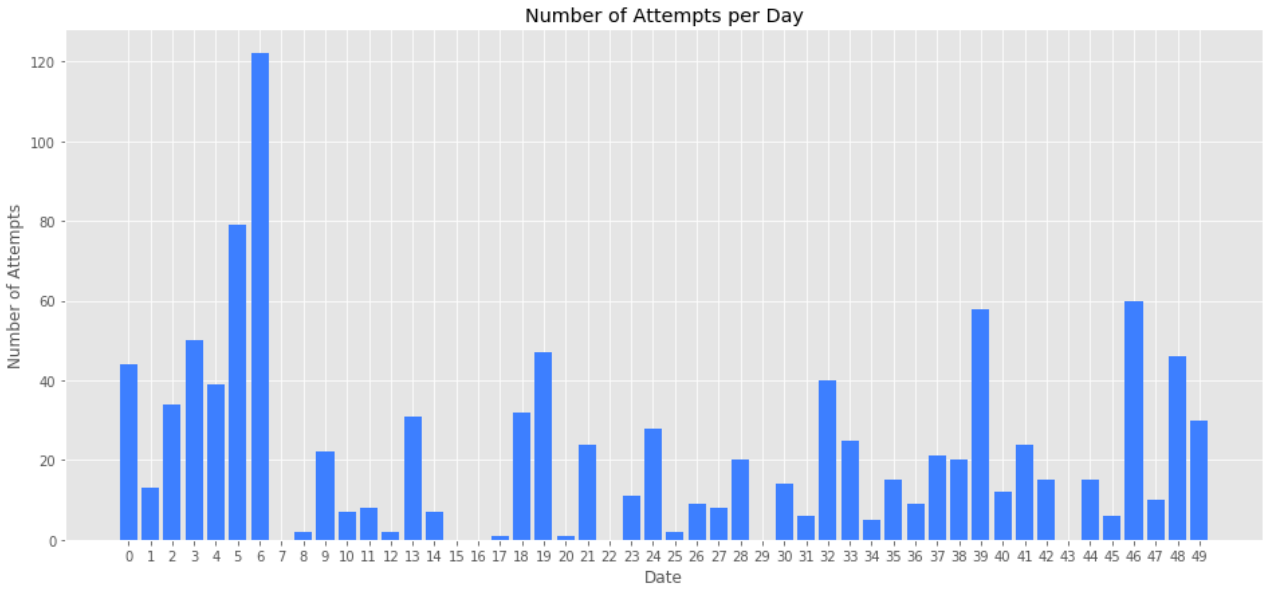


**Figure 4: The number of attempts per day was highest when the website was released, with smaller activity spikes centred around when reminder emails were sent.**

working on development until early May, which was extremely close to the end of the first semester of work and almost two months after the project started. This was a major setback for me, and this late start for development added more time pressure to my project, which eventually resulted in me being unable to complete some of my project objectives such as research into Intelligent Tutoring Systems. Additionally, other factors of gamification such as leaderboards and friend groups could have been researched in this time, so the gamification objective was also impacted by this time delay.

As I developed my solutions, I attempted to test all components as thoroughly as possible. I unit tested the functions that were not integrated with the user interface, and manually tested visual components in the user interface (test data used in manual tests is included in Appendix C). However, nearing the end of the project, a change made to the CodeWOF User model by the Computer Science Research Education group implemented a UserType field, which referenced a UserType model with a "slug" field (a Django field designed to store and generate URLs). For currently unknown reasons, this affected the generation of test data, which rendered my tests unusable until this error is fixed. Therefore, in order to mitigate the loss of tests, I re-wrote my unit tests as feature files, which are files in Gherkin syntax designed for Behaviour Driven testing. I made the decision to use feature files as these files can be converted to automated Cucumber acceptance tests in the next iteration of this project, and so would be useful documentation rather than documentation that could be perceived as unnecessary in future. Cucumber tests can be run in Python using frameworks such as Behave![3], which runs Cucumber tests for Python applications, and so would suit a Django-based website such as CodeWOF.

## 5.2 Limitations

A significant limitation of this project was the lack of user testing performed for the implemented point and badge systems. While these features were implemented and verified to work "correctly", there has been no user validation to determine that this what the target user group wanted. Had I more time, I would have liked to perform at least a survey on user opinions of the points and badges, and how they affected their usage habits. Ideally, an in-depth user study involving one-on-one interviews with users would have provided more valuable insight as to how the system would be used and perceived. This study could be performed as future work for this project.

Another significant limitation of this project was the strict time constraints on the project, with it being timeboxed to less than two full semesters. The time constraints proved a great challenge for me to complete my project objectives in time. Due to the aforementioned setup issues costing me almost two months of development time, I found I was unable to complete several of my objectives in the time remaining.

## 5.3 Self Reflection

Overall, I am proud of what I have implemented in this project. I have learned how to use Django for web development, a framework I had no experience with before this project. In addition, I learned a lot about how gamification is used as a motivational aspect in software development, particularly in how successful badge elements were in motivating users across multiple research studies. I have always been interested in game development, so applying elements of games to an educational website was eye-opening to the possibilities in encouraging application use. Additionally, I found feedback from university staff very useful in steering me in the right

direction in terms of which approaches to take. Tim Bell provided extremely valuable input deciding how to best address the problem space of this project in regards to teachers and their learning habits. Tanja Mitrovic also provided valuable advice on gamification in educational systems, which gave me avenues of research for how to best implement gamification elements to motivate users.

I am less satisfied with my handling of the initial objectives of this project. The main objective was to gamify the CodeWOF website, which I have achieved by implementing the improved points and badges system, so I am proud of that. In my development, almost all of my implemented additions to the CodeWOF website were submitted in a single pull request at the end of the project. Had I released my implemented gamification elements incrementally, I could have gained valuable user feedback in each stage of my development to shape each feature to best please CodeWOF's users. Instead, I chose to "perfect" each feature I implemented with rigorous manual tests and unit tests, which means that I am happy with the quality of my work, but am unsure as to how users will react to it outside of my predictions.

For my other objectives, I did not manage my time well enough throughout this project to complete any of the objectives related to laying foundations for an Intelligent Tutoring system in the future. While this particular objective was admittedly large, I had hoped to introduce at the very least some foundation for a skill tree system for questions. I did not complete anything related to this objective in the end. In addition to losing development time due to setup, I found that I had underestimated time and effort required by other commitments, which further limited my time spent on this project. While I thought I had adequate buffer time in my initial project plan to accommodate these commitments, I was mistaken. These commitments also tended to have time constraints (e.g. other assignments), which led me to give them higher priority than this project at the time. In addition, I had several unexpected events occur during the project which could not have been predicted, and cost me further time to complete my project objectives. To remedy these errors in future, I would provide a larger buffer time for each milestone to accommodate any unexpected events could occur. For other commitments, I would be stricter with my scheduling and ensure I gave this project the amount time I intended to work on it, and compromise less of this project's time for other deadlines.

# 6 CONCLUSION AND FUTURE WORK

## 6.1 Conclusion

CodeWOF was designed to aid teachers in maintaining their programming skills in order to effectively teach their students the new digital technologies curriculum from the Ministry of Education. This project aimed to integrate CodeWOF's predecessor project ("BitFit") into the DTHM4Kaiako website created by the Computer Science Research Education Group. Additionally, the project included gamifiying the CodeWOF website in order to motivate users to engage with the website regularly, and researching how to implement a base for an Intelligent Tutoring system to aid users in maintaining their programming skills.

In this project, I accomplished two out of three objectives. While CodeWOF was successfully integrated with DTHM4Kaiako, it then successfully became an independent website. I added two elements of gamification to the website (points and badges) to encourage users to consistently use the website, while also updating the website's user interface to aid user motivation. However, this objective has not been validated by the target user group, and so the effectiveness of this solution is yet to be determined. Finally, the objective to research the groundwork for an Intelligent Tutoring System was not completed due to time constraint limitations.

## 6.2 Future Work

There are several avenues for future work on CodeWOF. The first would be to perform a user study on the effects of the point and badge systems I created for this project. This study would be critical in determining if my solution was truly successful by determining the level of encouragement and motivation it gave the users. The study could take multiple forms, including but not limited to data analysis of usage before and after point and badge systems were introducted to the website, a survey of all current users as to the effects of my solution, and one on one user interviews to gain deeper insight into the effect gamification had on their usage habits. From there, my solution could be improved from the user feedback gained in the performed study. The study could also endeavor to determine other possible gamification elements that could be added to CodeWOF by asking users what they would wish to use in such a website.

Research into other gamification elements could be performed, such as the viability of cooperative elements for users such as friend groups and shared goals, competitive elements such as leaderboards, or customisation options such as profile avatars, customisation, and collectables. In addition, research into task scheduling in the CodeWOF system could be performed to set up asynchronous tasks such as email reminders. Another goal that could be achieved would be setting up a robust testing system to include a fixed unit test suite and behavioural testing through Cucumber. User interface testing could also be researched here to automate manual tests.

Finally, the largest area for future work would be developing an Intelligent Tutoring system to aid teachers in maintaining their programming skills. A skill tree concerning question content could be devised (with skills such as functions, loops, input, dictionaries, lists etc.). Users could track their progress by the number of questions successfully answered in each skill. From this, a recommendation system could be implemented to recommend questions to users from their weaker skill areas, to ensure a user keeps all areas of their programming skills maintained. A hint system could also be implemented to aid users when they struggle to answer a question by providing either feedback on their solution, or external resources to read. It is likely this intelligent tutoring system will not be completed in a single iteration of this project

# REFERENCES

[1] [n. d.]. codeWOF. https://www.codewof.co.nz/
[2] [n. d.]. dthm4kaiako.ac.nz. https://www.dthm4kaiako.ac.nz/
[3] 2019. behave/behave. https://github.com/behave/behave original-date: 2011-10-25T11:02:35Z.
[4] Gabriel Barata, Sandra Gama, Joaquim Jorge, and Daniel Gonçalves. 2013. Engaging engineering students with gamification. In *2013 5th International Conference on Games and Virtual Worlds for Serious Applications (VS-GAMES)*. IEEE, 1–8.
[5] Beehive. [n. d.]. New digital technologies for schools and kura. http://www.beehive.govt.nz/release/new-digital-technologies-schools-and-kura

[6] Tim Bell. 2014. Establishing a nationwide CS curriculum in New Zealand high schools. *Commun. ACM* 57, 2 (2014), 28–30.
[7] Nathalie Bonnardel, Annie Piolat, and Ludovic Le Bigot. 2011. The impact of colour on Website appeal and users' cognitive processes. *Displays* 32, 2 (2011), 69–80.
[8] CodeRunner. [n. d.]. CodeRunner. https://coderunner.org.nz/
[9] Django. [n. d.]. django.contrib.auth | Django documentation | Django. https://docs.djangoproject.com/en/2.2/ref/contrib/auth/
[10] Django. [n. d.]. The Web framework for perfectionists with deadlines | Django. https://www.djangoproject.com/
[11] Mariam Adawiah Dzulkifli and Muhammad Faiz Mustafar. 2013. The influence of colour on memory performance: A review. *The Malaysian journal of medical sciences: MJMS* 20, 2 (2013), 3.
[12] David Gibson, Nathaniel Ostashewski, Kim Flintoff, Sheryl Grant, and Erin Knight. 2015. Digital badges in education. *Education and Information Technologies* 20, 2 (2015), 403–410.
[13] Google. [n. d.]. Cloud Tasks. https://cloud.google.com/tasks/
[14] Alice F Healy and Lyle E Bourne. 1995. *Learning and memory of knowledge and skills*. Sage.
[15] Maria-Blanca Ibanez, Angela Di-Serio, and Carlos Delgado-Kloos. 2014. Gamification for engaging computer science students in learning activities: A case study. *IEEE Transactions on learning technologies* 7, 3 (2014), 291–301.
[16] KanbanFlow. [n. d.]. SENG402 - BitFit - KanbanFlow. https://kanbanflow.com/board/R6et8a
[17] Richard N Landers, Michael B Armstrong, and Andrew B Collmus. 2017. How to use game elements to enhance learning: Applications of the theory of gamified learning. In *Serious games and edutainment applications*. Springer, 457–483.
[18] Richard N Landers and Amy K Landers. 2014. An empirical test of the theory of gamified learning: The effect of leaderboards on time-on-task and academic performance. *Simulation & Gaming* 45, 6 (2014), 769–785.
[19] Mark Otto, Jacob Thornton, and Bootstrap contributors. [n. d.]. Toasts. https://getbootstrap.com/docs/4.2/components/toasts/
[20] ~ Matthias Marschall. 2015. Kanban vs Scrum vs Agile. https://agileweboperations.com/2015/07/27/scrum-vs-kanban/
[21] Youssef Nader. 2018. What is Django? Advantages and Disadvantages of using Django. https://hackr.io/blog/what-is-django-advantages-and-disadvantages-of-using-django
[22] Fiona Fui-Hoon Nah, Qing Zeng, Venkata Rajasekhar Telaprolu, Abhishek Padmanabhuni Ayyappa, and Brenda Eschenbrenner. 2014. Gamification of education: a review of literature. In *International conference on hci in business*. Springer, 401–409.
[23] Lasse Natvig, Steinar Line, and Asbjørn Djupdal. 2004. " Age of computers"; an innovative combination of history and computer game elements for teaching computer fundamentals. In *34th Annual Frontiers in Education, 2004. FIE 2004.* IEEE, S2F–1.
[24] Christa R Nevin, Andrew O Westfall, J Martin Rodriguez, Donald M Dempsey, Andrea Cherrington, Brita Roy, Mukesh Patel, and James H Willig. 2014. Gamification as a tool for enhancing graduate medical education. *Postgraduate medical journal* 90, 1070 (2014), 685–693.
[25] NGINX. [n. d.]. NGINX | High Performance Load Balancer, Web Server, & Reverse Proxy. https://www.nginx.com/
[26] Maree Palmer. 2019. mpa588/codewof. https://github.com/mpa588/codewof original-date: 2019-06-29T12:13:36Z.
[27] Fotaris Panagiotis, Mastoras Theodoros, Richard Leinfellner, and Rosunally Yasmine. 2016. Climbing up the leaderboard: An empirical study of applying gamification techniques to a computer programming class. *Electronic Journal of e-learning* 14, 2 (2016), 94–110.
[28] ~ Yatin Pawar. 2016. 7 key differences between Scrum and Kanban. https://upraise.io/blog/scrum-kanban-project-management/
[29] Real Python. [n. d.]. Asynchronous Tasks With Django and Celery – Real Python. https://realpython.com/asynchronous-tasks-with-django-and-celery/
[30] Jessica Robertson. 2018. BitFit exercises (Python). (2018).
[31] Richard M Ryan and Edward L Deci. 2000. Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being. *American psychologist* 55, 1 (2000), 68.
[32] Carlos Santos, Sara Almeida, Luís Pedro, Mónica Aresta, and Tim Koch-Grunberg. 2013. Students' perspectives on badges in educational social media platforms: the case of SAPO campus tutorial badges. In *2013 IEEE 13th International Conference on Advanced Learning Technologies*. IEEE, 351–353.
[33] Skulpt. [n. d.]. Welcome! -http://skulpt.org
[34] Neil Smith, Yasemin Allsop, Helen Caldwell, David Hill, Yota Dimitriadi, and Andrew Paul Csizmadia. 2015. Master teachers in computing: What have we achieved? (2015).
[35] Tekslate. [n. d.]. 11 Benefits Of JQuery That Every Web Designers Should Know Of. https://tekslate.com/11-benefits-jquery-every-web-designers-know
[36] TLRI (Teaching & Learning Research Initiative. [n. d.]. Supporting teachers and learners of programming by understanding feedback on syntax, semantics and style | Teaching & Learning Research Initiative (TLRI). http://www.tlri.org.nz/tlri-research/research-progress/school-sector/supporting-teachers-and-learners-programming
[37] VTNZ (Vehicle Testing New Zealand). [n. d.]. VTNZ - Your Vehicle Safety Experts. https://vtnz.co.nz/
[38] Wan Mohd Nazmee Wan Zainon, Wong Seok Yee, Choo Seah Ling, and Chee Kit Yee. 2012. Exploring the Use of Cognitive Psychology Theory in Designing Effective GUI. *IJEI: International Journal of Engineering and Industries* 3, 3 (2012), 66–74.

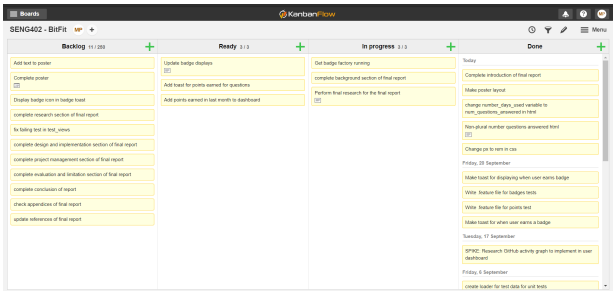


**Figure 5: KanbanFlow allowed me to track tasks with deadlines and time logged to each task.**

# 7 APPENDIX

## A PROJECT MANAGEMENT EVIDENCE

I tracked my tasks for this project using KanbanFlow, as seen in Figure 5.

## B USER MANUAL

In order to use the point and badge systems in CodeWOF, a user must first have a CodeWOF account. From there, a user's points and badges can be viewed on their dashboard. Points can be earned by solving questions - a user earns 10 points for a question answered correctly, with two bonus points for answering correctly on their first try. Badges are also earned through solving questions. Users earn badges by

To set up this project in a development environment, clone this project's GitHub repo [26] into a folder in a Linux environment. The environment should also have Docker and Python installed. Navigate to the project's root directory in a terminal and run the following commands:

```
docker-compose up
docker-compose down
./dev build
./dev start
./dev update
./dev sampledata
```

This will successfully set up and build the project locally. Open a browser to `http://localhost:83/` and the website will run with sample data loaded. A sample user account can be used to log in to the website, with the username `user@codewof.co.nz` and the password `password`.

To stop the running of the project, run the command

```
./dev end
```

This command will stop all Docker containers running.

## C TEST DATA

Test data used was taken from the sampledata command, which used the following code to set up test data in the development environment.

```
        User = get_user_model()
        # Create admin account
        admin = User.objects.create_superuser(
            'admin',
            'admin@codewof.co.nz',
            password=settings.SAMPLE_DATA_ADMIN_PASSWORD,
            first_name='Admin',
            last_name='Account',
            user_type=UserType.objects.get(slug='teacher'
    )
        )
        EmailAddress.objects.create(
            user=admin,
            email=admin.email,
            primary=True,
            verified=True
        )
        print('Admin created.')

        # Create user account
        user = User.objects.create_user(
            'user',
            'user@codewof.co.nz',
            password=settings.SAMPLE_DATA_USER_PASSWORD,
            first_name='Alex',
            last_name='Doe',
            user_type=UserType.objects.get(slug='student'
    )
        )
        EmailAddress.objects.create(
            user=user,
            email=user.email,
            primary=True,
            verified=True
        )

        user2 = User.objects.create_user(
            'user2',
            'user2@codewof.co.nz',
            password="password",
            first_name='Alex',
            last_name='Doe',
            user_type=UserType.objects.get(slug='student'
    )
        )
        EmailAddress.objects.create(
            user=user2,
            email=user2.email,
            primary=True,
            verified=True
        )

        UserFactory.create_batch(size=100)
        print('Users created.')

        # Codewof
        management.call_command('load_questions')
        print('Programming questions loaded.')

        # Research
        StudyFactory.create_batch(size=5)
        StudyGroupFactory.create_batch(size=15)
        print('Research studies loaded.')

        Badge.objects.create(
            id_name='create-account',
            display_name='Created an account!',
            description='Created your very own account',
            icon_name='img/icons/badges/icons8-badge-
    create-account-48.png',
            badge_tier=0
        )

        Badge.objects.create(
            id_name='questions-solved-1',
            display_name='Solved one question!',
            description='Solved your very first question'
    ,
            icon_name='img/icons/badges/icons8-question-
    solved-black-50.png',
            badge_tier=1
        )

        Badge.objects.create(
            id_name='questions-solved-5',
            display_name='Solved five questions!',
            description='Solved five questions',
            icon_name='img/icons/badges/icons8-question-
    solved-bronze-50.png',
            badge_tier=2
        )

        Badge.objects.create(
            id_name='questions-solved-10',
            display_name='Solved ten questions!',
            description='Solved ten questions',
            icon_name='img/icons/badges/icons8-question-
    solved-silver-50.png',
            badge_tier=3
        )

        Badge.objects.create(
            id_name='questions-solved-100',
            display_name='Solved one hundred questions!',
            description='Solved one hundred questions',
            icon_name='img/icons/badges/icons8-question-
    solved-silver-50.png',
            badge_tier=4
        )

        Badge.objects.create(
            id_name='attempts-made-1',
            display_name='Made your first attempt at a
    question!',
            description='Attempted one question',
            icon_name='img/icons/badges/icons8-attempt-
    made-black-50.png',
            badge_tier=1
        )

        Badge.objects.create(
            id_name='attempts-made-5',
            display_name='Made five question attempts!',
            description='Attempted five questions',
            icon_name='img/icons/badges/icons8-attempt-
    made-bronze-50.png',
            badge_tier=2
        )

        Badge.objects.create(
            id_name='attempts-made-10',
            display_name='Made ten question attempts!',
            description='Attempted ten questions',
```

```
        icon_name='img/icons/badges/icons8-attempt-
made-silver-50.png',
        badge_tier=3
    )

    Badge.objects.create(
        id_name='attempts-made-100',
        display_name='Made one hundred question
attempts!',
        description='Attempted one hundred questions'
,
        icon_name='img/icons/badges/icons8-attempt-
made-gold-50.png',
        badge_tier=4
    )

    Badge.objects.create(
        id_name='consecutive-days-2',
        display_name='Worked on coding for two days
in a row!',
        description='Attempted at least one question
two days in a row',
        icon_name='img/icons/badges/icons8-calendar
-2-50.png',
        badge_tier=1
    )

    Badge.objects.create(
        id_name='consecutive-days-7',
        display_name='Worked on coding every day for
one week!',
        description='Attempted at least one question
every day for one week',
        icon_name='img/icons/badges/icons8-calendar
-7-50.png',
        badge_tier=2
    )

    Badge.objects.create(
        id_name='consecutive-days-14',
        display_name='Worked on coding every day for
two weeks!',
        description='Attempted at least one question
every day for two weeks',
        icon_name='img/icons/badges/icons8-calendar
-14-50.png',
        badge_tier=3
    )

    Badge.objects.create(
        id_name='consecutive-days-21',
        display_name='Worked on coding every day for
three weeks!',
        description='Attempted at least one question
every day for three weeks',
        icon_name='img/icons/badges/icons8-calendar
-21-50.png',
        badge_tier=4
    )

    Badge.objects.create(
        id_name='consecutive-days-28',
        display_name='Worked on coding every day for
four weeks!',
        description='Attempted at least one question
every day for four weeks',
        icon_name='img/icons/badges/icons8-calendar
-28-50.png',
        badge_tier=5
    )


print("Badges added.")
```

**Listing 1: sampledata.py**

## D GAMIFIED PYTHON PROGRAMMING EDUCATION WEBSITES

The following list details some currently available websites designed for learning to program in Python. These websites focus on learning new skills, rather than maintaining current skills, which means they do not fully solve the problems addressed by this project.

- Grok Learning
- Code Avengers
- Codingbat
- Hackerrank
- Exercism
- Project Euler
- Code wars
- Codecademy
- Advent of code
- Codingame
- Practice Python

## E INITIAL BITFIT EER DIAGRAM

Figure 6 displays the initital BitFit EER diagram before improvements to badges and points were made.

## F UPDATED BADGES EER DIAGRAM

Figure 7 is of an EER diagram modelling the relationships between User, Profile, Earned, and Badge models in CodeWOF.

## G LIST OF CODEWOF BADGES

The current list of badges that can be earned in CodeWOF is as follows:

- Create an account
- Attempt one question
- Attempt five questions
- Attempt ten questions
- Attempt one hundred questions
- Solve one question
- Solve five questions
- Solve ten questions
- Solve one hundred questions
- Solve questions on two consecutive days
- Solve questions on seven consecutive days
- Solve questions on fourteen consecutive days
- Solve questions on twenty-one consecutive days
- Solve questions on twenty-eight consecutive days

This list can be expanded upon in future.

## H CURRENT CODEWOF SCREENSHOTS

The following figures (Figure 8 though Figure 11) depict the main screens of CodeWOF after I completed this project.

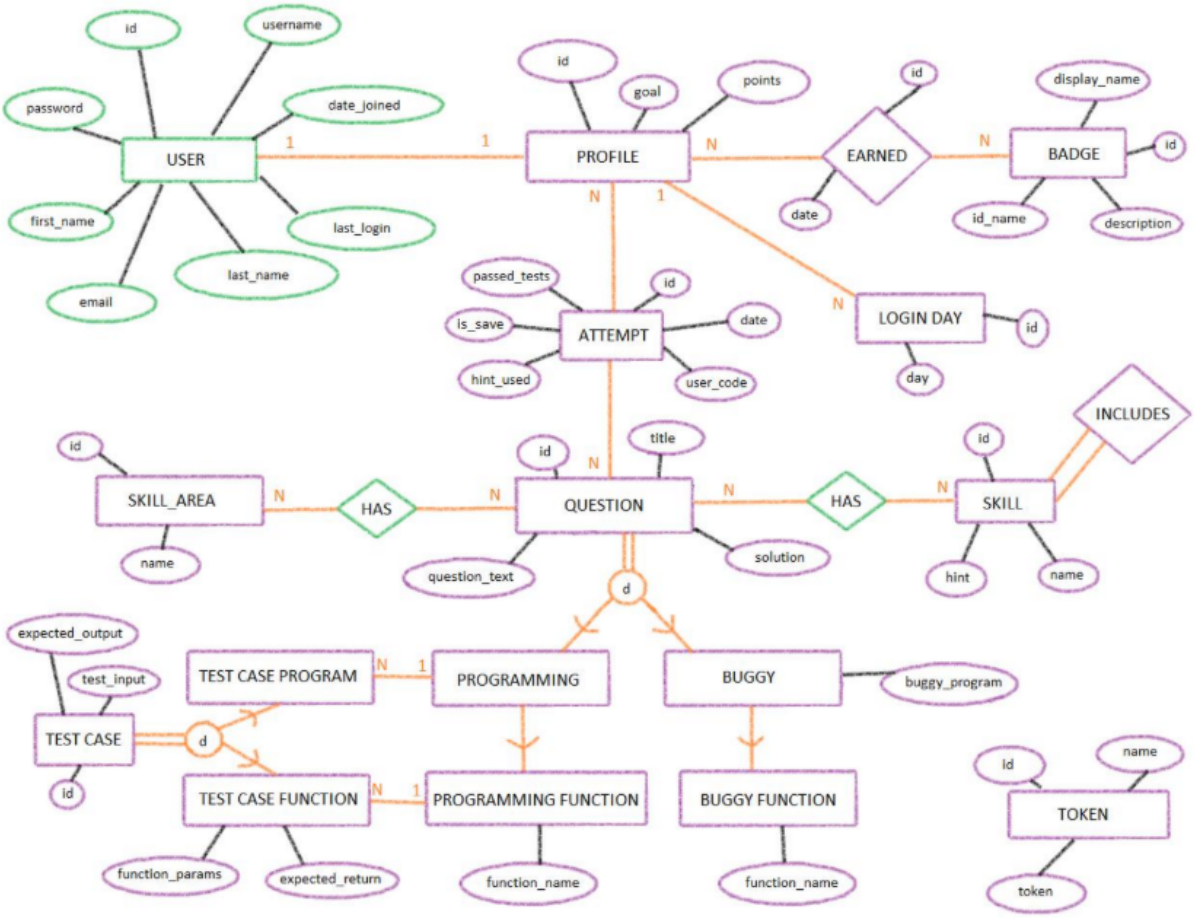


**Figure 6: The initial EER diagram for BitFit was created by Jessica Robertson in 2018.**

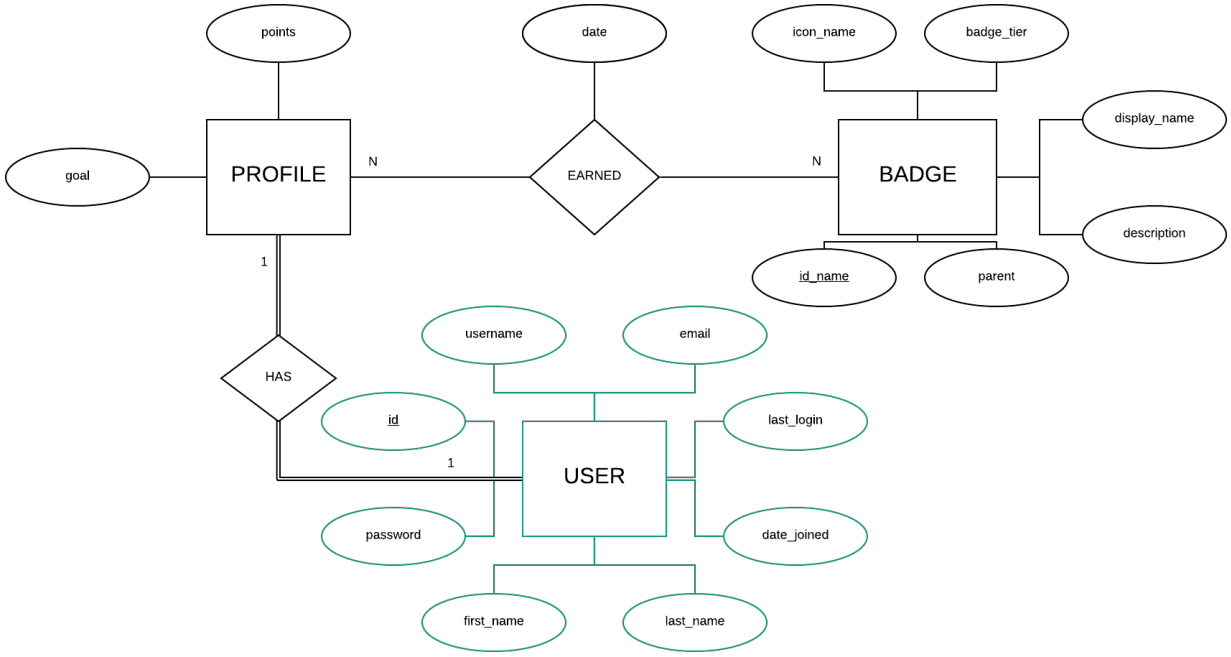


**Figure 7: In the EER diagram of the badge system, the User model in green is a default Django model.**


**Figure 8: The home screen introduces users to CodeWOF. The updated colour theming can be seen in the navigation bar.**

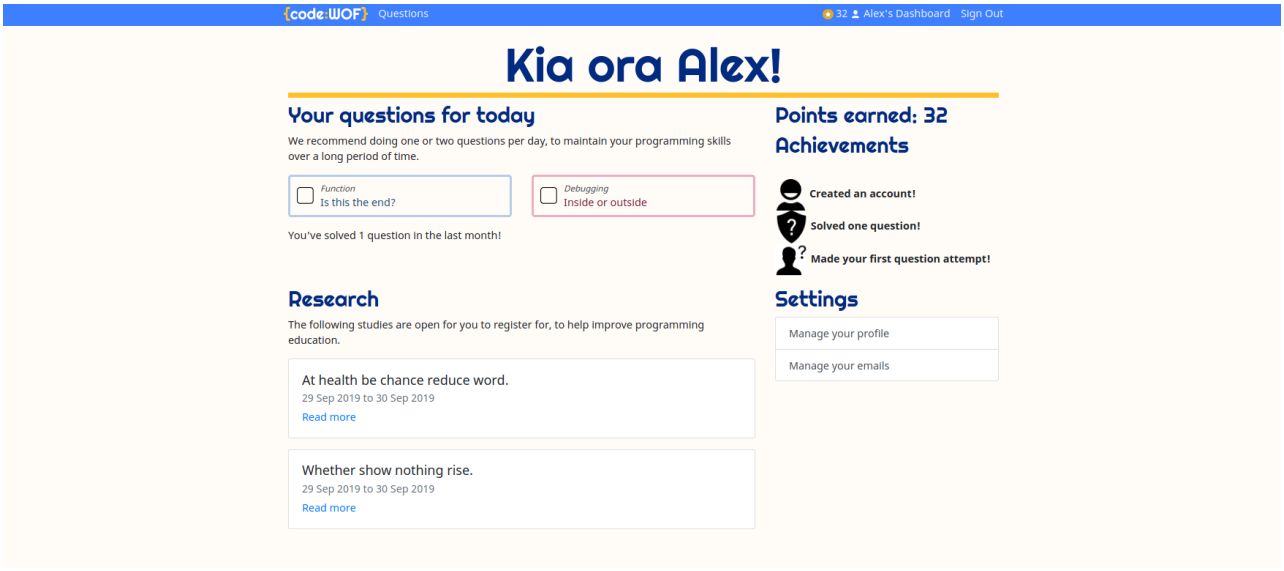


**Figure 9: A user's dashboard displays two suggested questions, their points, and badges earned.**


**Figure 10: Questions are displayed in boxes outlined in colours corresponding to question types.**

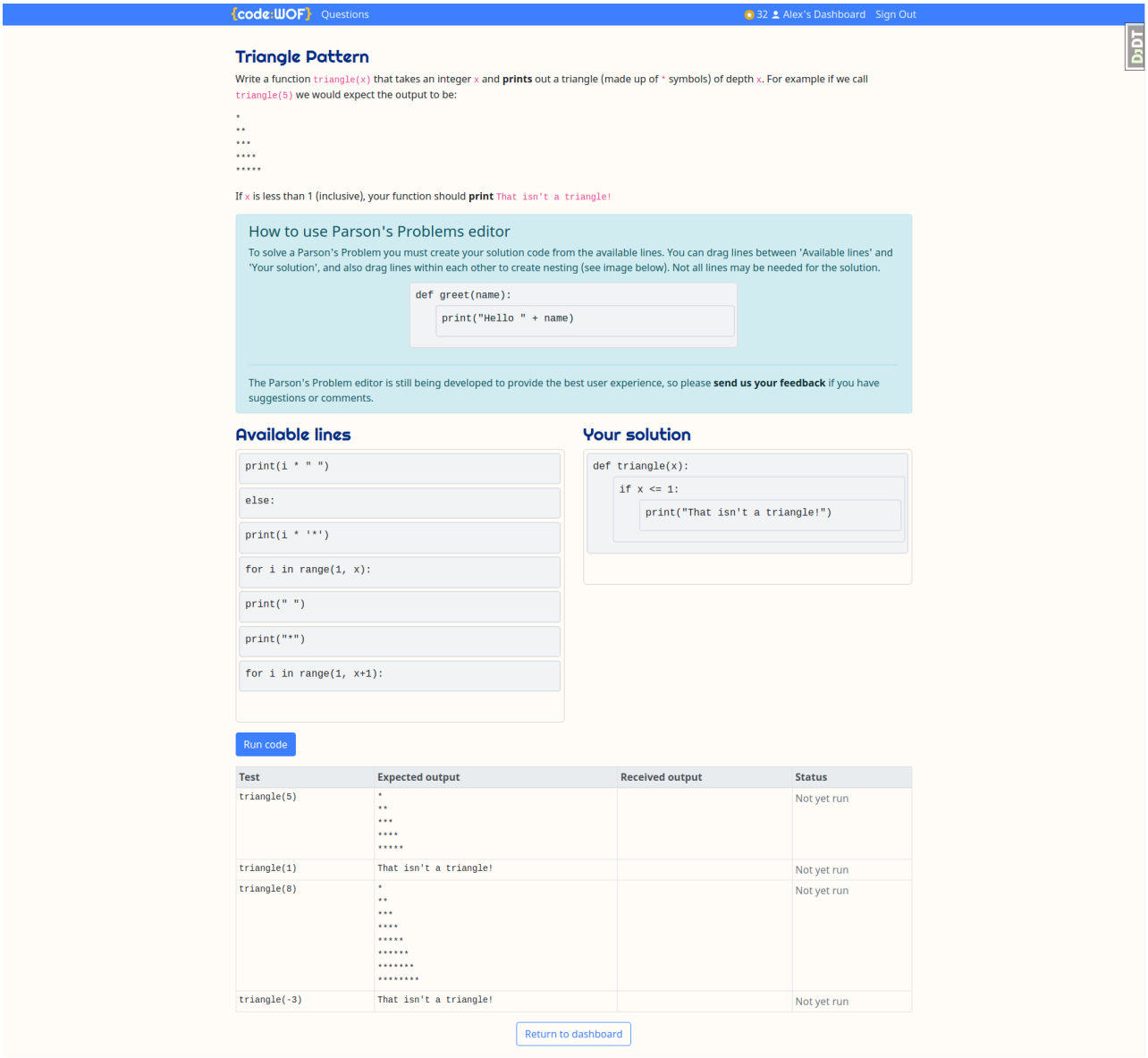


**Figure 11: Parsons problems are a type of question in CodeWOF. Users can view test cases for a question before submitting their code.**