University of Lincoln

CMP2804 Team Software Engineering

UNIVERSITY OF
LINCOLN

SCHOOL OF
COMPUTER SCIENCE

Toby Armond: 27050587

Archie Baldry: 26411141

Marco Caçoete: 26492272

Lorna Foster: 26093976

Harry Paton: 26523240

Earl Smalley: 16617508

Oliver Smith: 26357261

May 9, 2024

Contents

Software Engineering and Planning

1.1 Software Engineering Strategy/Approach

The artefact that we were tasked to develop is a feedback tracking tool called "FeedTrac". We have designed FeedTrac for universities so that they can provide their staff and students the ability to post and review feedback allowing everyone's voices to be heard on important issues to facilitate a better-quality university experience.

The methodology we decided to use for this project was Agile Martin 2008, p. 15. Agile is a technique that has grown from within the software development community itself rather than being imposed on developers by upper management. This means it was built and has proven itself through trial and error as a successful methodology that serves the interests of developers and encourages a more flexible and responsive workflow. This success has meant that it now features prominently in the professional world and so understanding how to use Agile will improve our ability to integrate into existing professional development teams once we begin our careers in the industry.

The alternative methodology we could have chosen was the Waterfall approach. Waterfall divides the software development life-cycle into segments that follow a linear progression with predictable or strictly planned timelines. These segments may look like this: Requirements, Analysis, Design, Implementation, Testing, Maintenance.Indeed (2023) Waterfall wouldn't be an appropriate method for this project because it's generally considered to be outdated and not always conducive to successful software development. This is because they emphasise the use of plans such as Gantt charts which are ineffective because it's not known with certainty what tasks the project will have during the entire development cycle, nor how long those tasks take, their costs or priority.

Agile was appropriate because we are students that needed to save time for studying various programming concepts and designs, meaning that we needed to go back and update previous work as we learnt more. This kind of workflow would've been impossible with Waterfall since we wouldn't keep up with the schedule given the strict deadlines and cut-off points for each segment. Additionally, it's difficult to plan segments as we don't have the experience to know what exactly will need doing and in what order, nor do we have the intuition as to how long it should take. Agile on the other hand, allows us the flexibility we need to move around the project when and where necessary.

For dynamic planning strategies we used active text and voice conversations through group chats on a discord server and a TODO.md file that we kept updated on our GitHub repository and documentation was essential to the workflow of the project Thomas & Hunt 2019, p. 23. The TODO.md file was split into sections that detailed work that needed doing for each page of the website and a priority queue as seen in figure 1 was formed from this so that everyone knew which work needed doing first. Optional tasks were added as the lowest priority items and gave ideas on extra functionality beyond the requirements. Any work completed was added to our individual work journals so we could keep track of what had been completed while also having a good way to determine everyone's contribution levels. We tried Kanban as a strategy in the previous assignment, however it didn't seem as effective this time around compared to the new strategies we used. We did not utilise the Scrum strategy in full as it requires time to learn how to use it effectively since we would have needed someone to take time away to learn how to be scrum master. That being said, throughout the development, we had a cycle of

meetings where we planned our next goals, followed by a period of concentrated work towards those goals. This is what is known as a "sprint" in scrum so it could be said that we partially utilised the scrum strategy, it was just in a more decentralised way where we each had our own specialisations and built the artefact together democratically rather than having a centralised hierarchy of leadership. The TODO.md file could be seen as our product backlog which was updated for each sprint iteration Drumond (n.d.).

To aid our sprints we used a priority queue chart available at https://github.com/OSmith132/FeedTrac/blob/Main/Docs/chart2.md to organise our workflow shown in figure 1

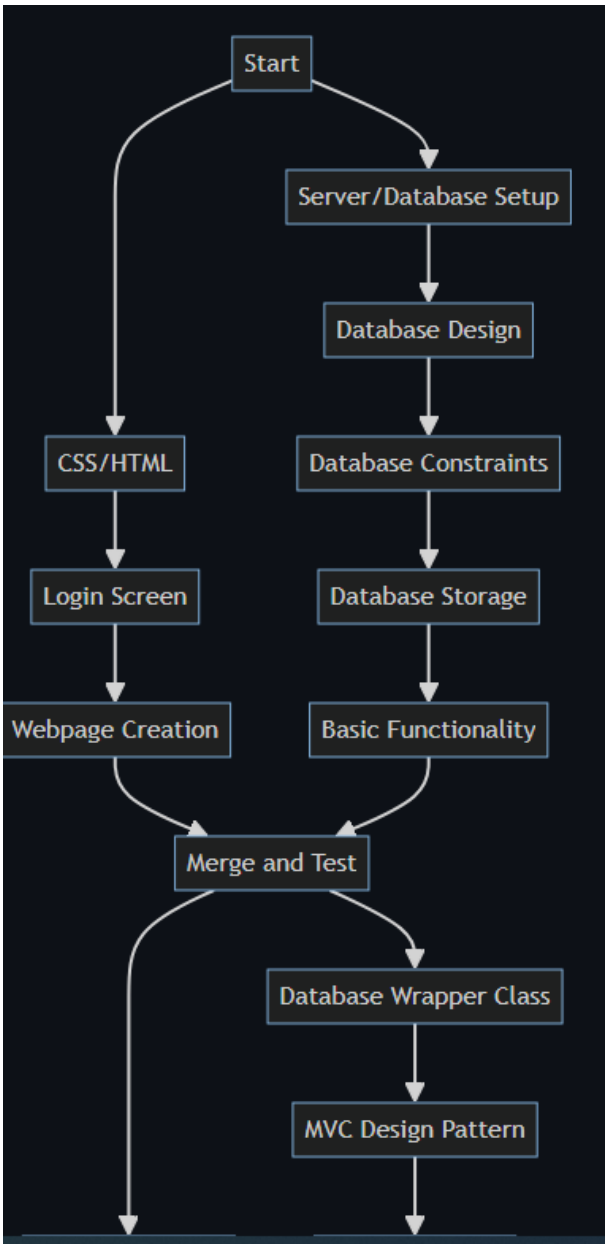

Figure 1: *Priority Queue*

1.2 Version Control

For version control, we decided to use GitHub since it's a free, widely used and popular tool for sharing a codebase and collaborating with a team, especially in the professional software devel-

opment industry. This also served as good work preparation for future careers in the industry as learning how to use tools that interface with GitHub such as GitHub Desktop or Git Bash during a functional development cycle will lead to better employability. Many other methods for version control or collaboration are also free, but only allow one or two members to collaborate at a time; further access for additional members is often locked behind a full payment or subscription fee. The use of version control is essential to a group project Thomas & Hunt 2019, p. 84.

More specifically, our version control workflow started off with the creation of documentation that helped the team keep track of what needed to be done such as a TODO list. Since this documentation was added to our GitHub repository, it was easy to commit new versions of this as we continued through our development cycle. Once we decided on how we would begin to build the artefact, we were able to commit the initial necessary files to the repository. After this, we created the README file which details the necessary instructions on how to set up the project.

Once the repository was set up, we began working together on the project. We could clone the project to a local repository on our individual systems and make changes there, once we were ready to commit and push the change, we communicated with each other so that we didn't have as many merge conflicts that needed to be resolved. We used separate branches for the larger updates that needed to be more thoroughly tested which could then be brought over to the main through a pull request, whereas smaller updates could just be pushed straight to the main branch. We didn't end up using versioning numbers to keep track of each update. Instead, we used commit messages with added descriptions if necessary.

1.3 Dependencies

- XAMPP https://www.apachefriends.org/
- PHP https://www.php.net/
- Javascript https://developer.oracle.com/languages/javascript.html

Implementation

2.1 Explanation

FeedTrac is a web-based, feedback tracking application for use within an academic institute, e.g. the University of Lincoln. The application's primary function is that any member of the student body can construct and submit a report to their relevant department regarding a specified lecture, workshop, room, lecture hall or lecturer, or even a general item of feedback for their course or the institution as a whole. Each report, known in the FeedTrac program simply as "Feedback", can contain a title, a body of text, urgency and two status tags which record whether the Feedback is open or closed, and unresolved or resolved. Once the report is submitted, it becomes accessible to every relevant user, whether that is a fellow student or a member of staff, who then has the chance to leave comments, upvote, and open a dialogue with the author. This enables them to discuss potential solutions or improve the clarity of the Feedback. From the homepage, users can view a summary of applicable Feedback reports by filtering by course, urgency, status and timeframe. There is also a search bar for finding keywords within a Feedback title, and options for sorting the table.

Only users who are logged in may view individual Feedback items, or even the homepage, so all institutional data is kept private. Admins who decide to host FeedTrac on behalf of their institution may wish to restrict account creation to alumni and staff, to prevent access from the general public. Additionally, student authors and admins have the option to delete individual Feedback reports, Feedback comments, and even their entire account, if they no longer require them. Although students are advised to mark their Feedback as resolved (to later be marked as closed by a member of staff or admin) rather than outright deleting reports, as this allows anyone in the future with the same issue to reference the original report.

We used the XAMPP website server solution stack to create our artefact with two teams, one for the frontend and one for the backend. This approach was deemed the best to create our artefact as it allowed our team members to play to their strengths and focus on a more manageable amount of work. The backend team was responsible for creating and maintaining the MariaDB database, as well as creating the hooks that could be called by the frontend written in PHP. The frontend team was charged with creating the HTML for the pages as well as CSS stylesheets to determine the aesthetics of the artefact. The frontend team also made use of Javascript script blocks to necessitate the function of the client-side interaction with the artefact.

The PHP "hooks" were created using a MVC "model view controller" Freeman & Robson 2021, p. 520 shown in figure 2 design with feedback being reported to individual users using an observer model Freeman & Robson 2021, p. 37.The database was queried using MySQL queries within PHP Suehring & Valade 2013, 519

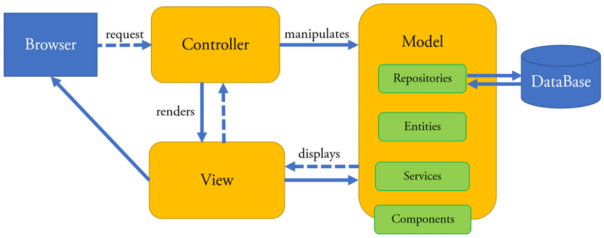

Figure 2: *MVC model structure*

The MVC classes we created functioned as the 'backbone' of the system, providing us with reliable implementations of complex interactions with the database in both user and feedback contexts. As we became more accustomed to using this pattern, it allowed us to define a clear division of labour between the frontend and backend teams, promoting consistency, streamlined collaborations, and enhanced maintainability in our software.

Pages in the site were then created by using base HTML written by members of the frontend team and populating the elements within the page using PHP scripts created by the backend team. Furthermore frontend elements such as styling (e.g. dark and light mode) were creating using Javascript blocks to add client side interaction with the site.

For our alert system we implemented an observer design pattern, we established a subscription mechanism that subscribed users by default to any new event, in our case, a new feedback item, or an update to a feedback item, in the form of a comment, status change or rating change. For convenience an inbox was created to contain updated feedback related items. Every user has the option to unsubscribe from the alert list in the settings menu.Freeman & Robson 2021, p. 37

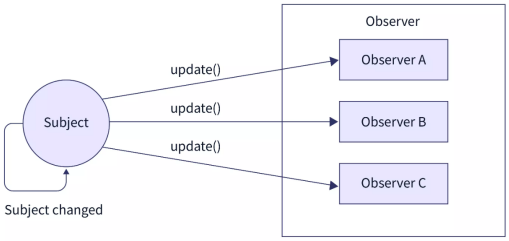

Figure 3: *Observer model structure*

```php
<?php
// Small local function to call the alert when a new comment or any action is taken involving the feedback item only alerts subbed users.
function achtung($users,$Feedback_Controller,$user_data){
    foreach ($users as $user) {
        if ($user['userID'] !== $user_data["userID"] && $user['sub'] == "1") {
            $Feedback_Controller->sub_alert($user['userID']);
        }
    }
}

if (isset($_POST['submit_comment'])) { // Check to submit comments if a comment hs been posted.
    $comment_text = $_POST['comment_text'];
    $Feedback_Controller->new_comment($user, $feedbackID, $comment_text, $ratingPoints_comment); // New comment saved in database.
    date_default_timezone_set('Europe/London'); // Sets date for correct timezone, to date comments and feedback items.
    $newDate = date_create();
    $Feedback_Controller->modify_date($feedbackID,$newDate); // Changes date of last update to feedback item.
    achtung($users,$Feedback_Controller,$user_data);  // Calls alerts again.
```

```
18      header("Location: " . $_SERVER['REQUEST_URI']);  // Refreshes page and ends scripts to
        ↪  show updates.
19      exit();
20  }
```

A constraint to fully being able to implement the observer was our email system. We had a single positive result for a successful email being sent from our server so we know that our code is working, however Mercury, the software that is bundled in Xampp, is very limited in the way that it works since it relies on email relaying through a real email address to send an email message that doesn't look suspicious to the receiving email server shown in figure 4. To prove our concept we used a mail catch website that hosts temporary email addresses, this website doesn't have the same policies in place as real email service providers so it allowed us to conclude that we were successful. The email portion of our code is commented out for now, however it is something we hope to keep working on after the submission of our project, detailed below in the evaluation section of this report.

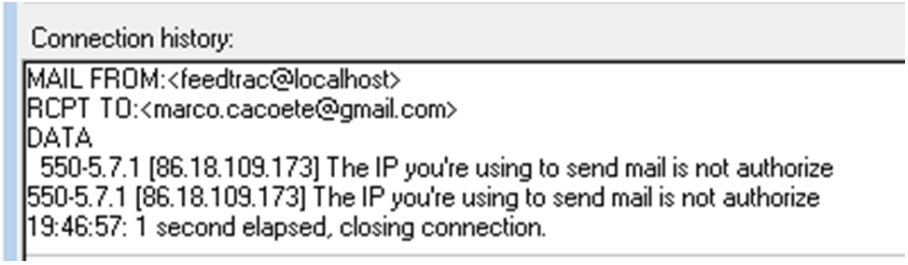

Figure 4: *Denied by service*

Figure 5: *Successful email test*

Though our attempt might be slightly flawed or incomplete when it comes to the results, the actual logic we put in place to make it work is sound. If we look at our alerts as also being considered emails being sent successfully shown in figure 5, we can say we were successful in its implementation.

For User login, passwords were used which were then hashed using PHP built in hashing methods, this meant that only encrypted passwords were stored on the database increasing the

security of the site overall.

Users also could not edit other users profile through the use of session data Suehring & Valade 2013, p. 440 allowing only the user to only edit their own profile.

Images were uploaded to the server through the use of POST endpoints and were uploaded to the server directly, this has some flaws as images are not posted for human review and could be unsavoury in nature.

Where possible, we prioritized using known libraries e.g https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js to maximize efficiency and reliability, and establish clear code boundaries within our project. By relying on well-established libraries, we not only accelerate development by taking advantage of pre-built solutions, but also ensure consistency and compatibility within our codebase. Relying on these functions offers us a robust foundation we can build off, and reduce the risk of having to refactor foundational code later down the line.

2.2 Justification

XAMPP was chosen as the solution stack for the server due to its several advantages. Being free, open-source, and widely used, it offers a cost-effective and well-supported development environment. XAMPP comes bundled with an Apache web server, a MySQL database server, and a PHP interpreter, eliminating the need for separate installations and configurations. This integrated environment streamlines development, particularly for beginners or projects requiring rapid prototyping.

MariaDB served as the database management system within the XAMPP solution stack chosen for our project. It is known for its high compatibility with MySQL. This compatibility allows applications designed for MySQL to run seamlessly on MariaDB with minimal or no code modifications, saving us development time and effort. MariaDB's scalability ensures the database can grow alongside the application without compromising performance, making it suitable for projects with an anticipated increase in user base or data volume.

PHP's widespread adoption, purpose-built design for web development, ease of use, and rich ecosystem of frameworks and libraries made it ideal for the server-side scripting language in this project as part of the XAMPP solution stack.

HTML is the fundamental building block of web pages. It provided us with a standardized way to structure content, including headings, paragraphs, lists, images, and more. Additionally, HTML elements can be easily customized and styled using CSS, allowing for a wide range of UI designs.

CSS handled the visual presentation we were able to modify the appearance of the UI by adjusting CSS styles without affecting the underlying HTML content. This separation also allowed for easier collaboration between content creators and designers our frontend and backend teams.

We used a MVC model to increase the modularity and flexibility of the artefact, making future expansion and refactoring easier for existing or new team members. AJAX was used as allowed dynamic page creation by providing the client browser with only the information required on the page. Substituting as necessary.

2.3 Implementation challenges

At various points in the development of the frontend for our website, we wanted to reposition and group elements in a logical order so that users could more easily interact with the website's features. Historically, frontend developers were limited to positioning elements either inline, block or grid. This was causing us many problems with our initial hand drawn designs as a lot of the functionality we expected from headers, navigation bars and dropdowns would break when the window was resized or if the user was viewing the website on their mobile device. However, with the addition of flex containers in 2009 (widely available across major browsers, caniuse.com (n.d.) we were able to utilise these components to our advantage and allow our page contents to react to any changes in window size without breaking functionality. This also meant that components that were previously too long to fit into the design could now be implemented with flexible wrapping.

Another challenge we faced in the development of the frontend for our website was a working toggle for our dark/light themes. Our original implementation of dark/light mode was very novel, and relied on duplicated CSS classes for each element, one for light, one for dark. This quickly became problematic when the size of the project grew, and we realised we couldn't easily prototype new features without writing these two classes each time. This approach also did not work nicely with a toggle button, as it took a while for the page to update. To solve this, we redesigned every element to rely upon a single point, :root variables, which we could then modify using a single JavaScript function at the click of a button. For example, the entire colour palette for the website resides in the :root tag in main.css, as seen below:

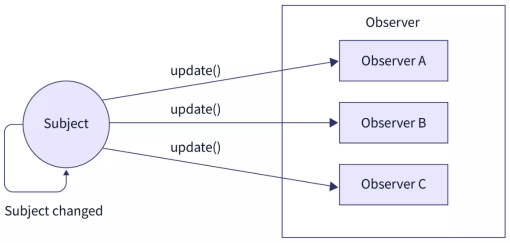

Figure 6: *Observer model structure*

```
:root {
    --a: #111;
    --b: #222;
    --c: #333;
    --d: #444;
    --e: #666;
    --f: #aaa;
    --g: #eee;
    --accent-bright: deeppink;
    --accent-pale: #ff80c3;
}
```

These variables, –a through to –g, are modified in our setColourMode function which defines dark colours for text and borders, and light colours for backgrounds when a light theme is desired (or vice-versa in dark mode). The user's choice of theme is also stored in the website's

local storage, which allows the website to remember the user's theme even after the page has been closed and reopened. Alternatively, we could have implemented it so that the user's choice of theme was tied to their account in the database, which would allow the website to remember the theme across devices. However, this may have been undesirable for users who prefer a light mode on their phone, but dark on their computer screens, and so we decided to stick with our current implementation of per device storage, rather than per user.

As we were all relatively new to website development, particularly backend development, our team encountered challenges in adopting and implementing the MVC (Model-View-Controller) architecture. A crucial factor that greatly facilitated this transition was the series of team meetings specifically organized for the backend team to assist the frontend team in effectively utilizing this design pattern. During these meetings, we discussed the principles of the MVC pattern and held group programming sessions where the backend team guided the implementation and debugging of new frontend features using the MVC classes. Without these meetings, it would have been much more difficult to integrate this practice into our existing codebase as successfully as we did.

2.4 MVC Examples

2.4.1 View Class

```php
<?php
public function search_feedback($searchTerm){
    // Check if any input is empty
    if ($this->empty_input_check($searchTerm)){
        return false;
    }
// Update the rating points of a feedback item
return $this->search($searchTerm);
}
```

2.4.2 Model Class

```
protected function search($searchTerm)
    {
<?php
        // Update the rating points of a feedback item
        $stmt = $this->connect()->prepare(" SELECT
                                                feedback.feedbackID,
                                                resolved,
                                                urgency,
                                                closed,
                                                title,
                                                feedback.text,
                                                feedback.date,
                                                feedback.ratingPoints,
                                                COUNT(commentID) as number_of_comments
                                            FROM
```

```
                                            feedback
                                        LEFT JOIN
                                            comment
                                        ON
                                            feedback.feedbackID = comment.feedbackID
                                        WHERE
                                            title
                                        LIKE
                                            CONCAT('%', ?, '%')
                                        OR
                                            feedback.text
                                        LIKE
                                            CONCAT('%', ?, '%')
                                        GROUP BY
                                            feedback.feedbackID
                                        ORDER BY
                                            CASE WHEN title LIKE CONCAT(?, '%') THEN 0
↪  ELSE 1 END,
                                            title,
                                            CASE WHEN feedback.text LIKE CONCAT(?, '%')
↪  THEN 0 ELSE 1 END,
                                            feedback.text;
                                        ");

        // Check if the SQL query is valid and execute
        if (!$stmt->execute([$searchTerm, $searchTerm, $searchTerm, $searchTerm])) {
            header("location: feedback.php?error=BadSQLQuery");
            exit();
        }

        $results = $stmt->get_result()->fetch_all(MYSQLI_ASSOC);
        return $results;
    }
```

2.4.3 Database Class

```
<?php
protected function empty_input_check(...$inputs){
    // Check if any input is empty
    foreach ($inputs as $input){
        if (empty($input) and $input != 0 and $input != "0" and $input != false){
                return true;
            }
        }
        // If no input is empty, return false
        return false;
    }
```

Testing Strategy

3.1 Strategy

Throughout the module, comprehensive and thorough testing has played an essential role during the development of FeedTrac, guaranteeing correct functionality, consistency, and reliability of our final product. Our testing strategy was comprised of peer reviewing merges, unit, integration, and system testing as well as ad-hoc testing Thomas & Hunt 2019, p. 215 as a group when developing new features.

To streamline the debugging process, we chose to apply the MVC design pattern to allow easier implementation of unit tests and simplicity of method design. These MVC classes were used for interacting with the database in both user and feedback contexts, as well as data continuity and formatting in the case of View classes. This very quickly became the backbone of our program, providing us with reliable implementations of complex database queries and controlling logic. This standardized approach not only improved code readability, but also greatly enhanced our development workflow, resulting in a more robust and maintainable codebase.

When integrating unit testing into our program, we decided as a team that it would be best to encapsulate each set of tests into individual classes depending on which class they are modelled on. These classes can be found at the 'Website/classes/unit-testing/' directory on GitHub. We aimed to get as much test coverage as possible, but due to some methods being unsuitable or too trivial for unit testing, and some inexperience with MVC, we ended up only choosing the most essential functions to be tested. Even with this approach, we feel there is still a very effective amount of coverage through the application which will allow us to effectively maintain future versions and circumvent inevitable code rot Todd (2024).

Due to the nature of the MVC pattern, when creating new features, it was essential that we were reliant on these methods that were already in place. By pairing MVC with unit testing, it allowed us to take on a form of contract oriented programming, where all essential functions are required to pass the tests. This approach guarantees that the imported code will work as expected.

For methods that required integration via other class methods or querying the database to perform their tasks, we decided it would be best to require integration testing. We usually performed this testing individually during the feature's development. We tested the updated version as a cohesive unit to ensure that all interactions with other aspects of the program functioned as expected. Once the development was complete and individual testing was finished, we published the branch with the appended changes for it to be reviewed and tested by other team members. Any discrepancies or inconsistencies with the rest of the codebase discovered during this review process were addressed promptly, either by refining individual methods or adjusting the integration between them.

Following this design cycle not only enables us to uphold the coding standards and conventions outlined in our style guide but also ensures adherence to our chosen design patterns, promoting consistency, streamlining collaborations, and enhancing the maintainability of our software. This form of static analysis provided us with a comprehensive insight into the codebase, minimizing the necessity for future refactoring and debugging. Moreover, by critiquing each other's code before merging with our main branch, we could consciously avoid the use of inefficient code and design antipatterns, such as boat anchor or dead code Wikipedia (2024).

3.2 Merging into Version Control

3.2.1 Team Reviewing new features before merge

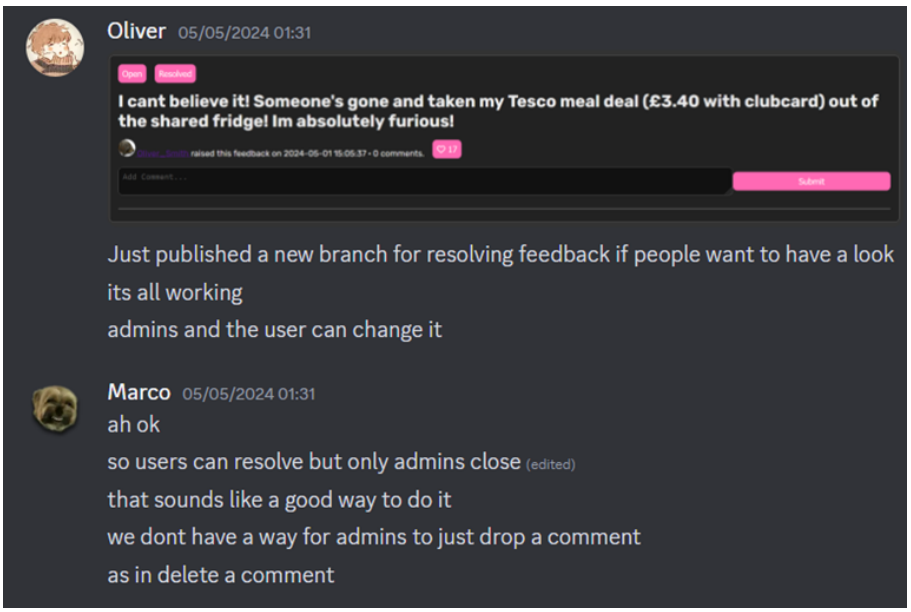

Figure 7: *Discussion before merge*

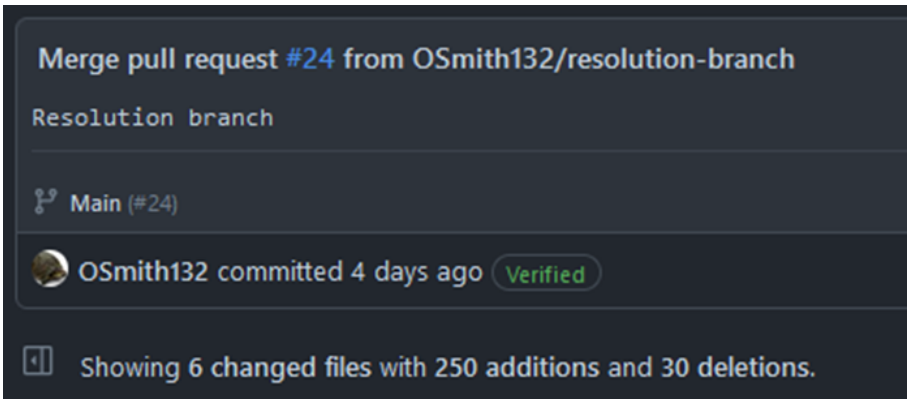

Figure 8: *Github Merge*

3.3 Unit Tests

3.3.1 Test Code

```php
<?php
// Include the unit testing class
include("classes/unit-testing/UnitTesting.FeedbackView.class.php");

// Instantiate unit testing class using existing view class
$FeedbackView_UnitTesting = new FeedbackView_UnitTesting(0);

// Echo either assertion error or 'passed!'
echo $FeedbackView_UnitTesting->test_row_passes_filters();
```

```
echo "Passed!"
?>
```

3.3.2 Pass Example

```php
<?php
public function test_row_passes_filters()
    {
        // Get the current timestamp
        date_default_timezone_set('UTC');
        $date = time();
        // Get the timestamp for 30 mins ago
        $date = date('Y-m-d H:i:s', $date - (30 * 60));
        // make row data
        $feedbackRow = array(
            'feedbackID' => 1,
            'userID' => 1,
            'feedback' => 'This is a test feedback',
            'rating' => 5,
            'resolved' => 1,
            'closed' => 0,
            'urgency' => 1,
            'date' => $date
        );
        $resolved = 0;  // Resolved
        $closed = 0;    // Open
        $urgency = 1;   // Low
        $timeframe = 0; // 1 hour
        // Invoke the method
        assert($this->feedbackView->row_passes_filters($feedbackRow, $resolved, $closed,
        ↪  $urgency, $timeframe));
    }
```

Result

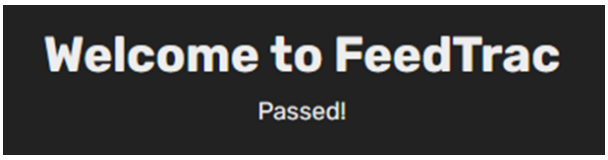

Figure 9: *Unit test Pass*

3.3.3 Fail Example

```php
<?php
public function test_row_passes_filters()
    {
        // Get the current timestamp
```

```
5         date_default_timezone_set('UTC');
6         $date = time();
7
8         // Get the timestamp for 30 mins ago
9         $date = date('Y-m-d H:i:s', $date - (30 * 60));
10
11        // make row data
12        $feedbackRow = array(
13            'feedbackID' => 1,
14            'userID' => 1,
15            'feedback' => 'This is a test feedback',
16            'rating' => 5,
17            'resolved' => 1,
18            'closed' => 0,
19            'urgency' => 3,  // Urgency has been changed to Critical
20            'date' => $date
21        );
22        $resolved = 0;  // Resolved
23        $closed = 0;    // Open
24        $urgency = 1;   // Low
25        $timeframe = 0; // 1 hour
26        // Invoke the method
27        assert($this->row_passes_filters($feedbackRow, $resolved, $closed, $urgency,
          ↪  $timeframe));
28    }
```

Result

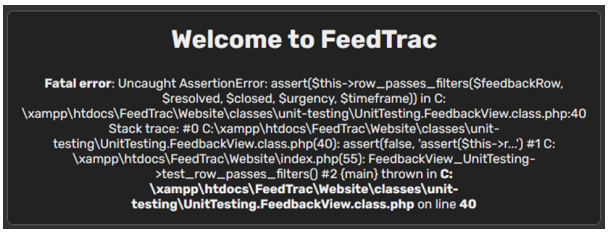

Figure 10: *Unit test fail*

3.4 Error Reports

Error reports were managed through our team Discord server, facilitating efficient collaboration, and enabling team members to promptly identify issues and propose tailored solutions whenever assistance or advice was needed. We found this was a more effective solution then some alternatives, such as GitHub issues, as it is an application we are all familiar with and have no difficulty navigating.

3.4.1 Example of error reports

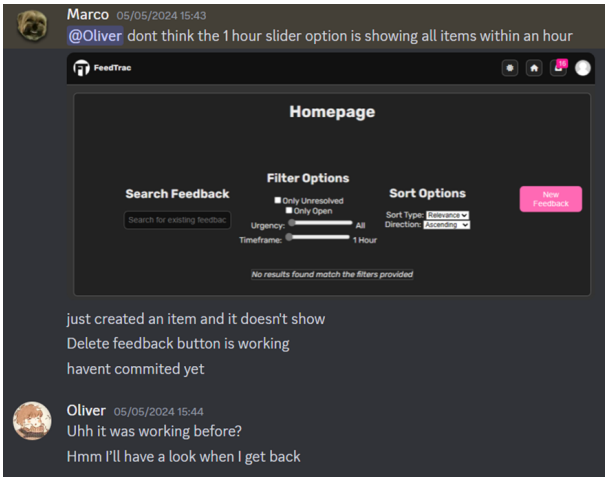

Figure 11: *Initial error report*

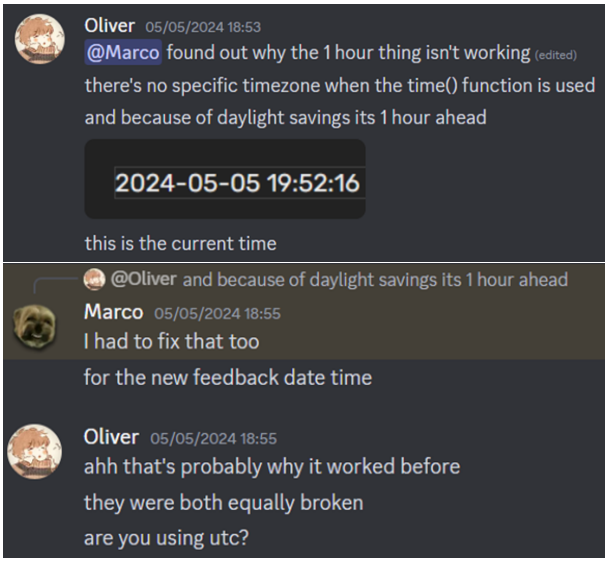

Figure 12: *Working as a team to resolve issue*

3.5 Merge Review

3.5.1 Merge Requested by Oliver, reviewed by Marco

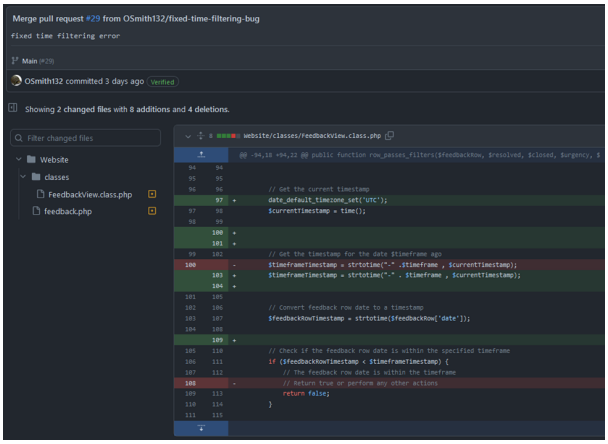

Figure 13: *Merge Request*

3.6 Testing conclusion

In the initial stages of FeedTrac development, we made a deliberate choice to refrain from adopting test-driven development. Given the constraints of a tight timeframe and the fast-changing nature of our ideas and objectives, we prioritized rapid prototyping and iterative development over the process of formally defining the behaviour of our end product. While TDD can provide robust test coverage and code reliability in the long run, it often requires a significant upfront investment in writing tests before implementing functionality, which did not suit our dynamic development environment. For this, we needed a more flexible approach, allowing us to quickly iterate on features and adjust for moving goalposts and functionality changes.

Evaluation

4.1 Functional Requirements

- User account creation with password encryption.
- The ability for a user to submit a feedback item that can read by peers and members of staff.
- The user can compose a feedback item by adding a title, the text detailing the issue and tags like urgency or a specific room.
- A comment system that allows a dialog between students and staff to discuss the issue.
- Rating system to mark relevancy of feedback items posted.
- Robust sorting and filtering system.
- Moderation tools that allow to open/close and set items as resolved/unresolved.

4.2 Internal Evaluation

After carefully looking at our completed artefact, it is clear that we hit the targets that we set out for our project. Our user account creation system allows not only for the basic information to be attached to the user but also for a bit of flair, our implementation of elements like profile pictures was one of the stretch goals that we managed to hit, it also allows for those looking through the feedback items to identify at a glance who is behind a post. The password encryption was one of the earliest things we implemented, its implementation was fairly straight forward as php has built in methods that trivialise how much work is needed to put it in place.

The feedback submission system in the final version closely follows our initial concept. It allows the user to detail the urgency of the issue as well as a specific room if a location needs to be specified. We considered providing more options to tag an item however we felt that most of the information related to the item could just be derived from the actual text post itself. To compensate for this design shift, we later implemented a search by text function. We also made sure the navigational flow from post source to post index to feedback item was a smooth and intuitive transition.

Our commenting system works well to facilitate a discussion between those interested in the item. A username is present which when clicked will take the current user to the commenters profile page, that user's profile picture and date of post are also displayed.

We included a button for feedback and comment deletion with some restrictions in place, a user of admin class, has full moderation powers and can delete anything, while a student can only delete their own submissions, to mitigate any possible accidental posts. Similarly, the change of open/closed and resolved/unresolved status follows the same rules.

We included an option to upvote relevant posts or posts that resonate with someone. We debated implementing an upvote/downvote system, popularized by reddit, but in the end felt that it could introduce some un-needed negativity in the way that users discuss the issues, in the end the choice to only include positive rating means that the worst reaction a user's posts could get is indifference, or a deletion by moderation team in more extreme cases. These rating

votes are also kept anonymous to prevent further friction.

The index page features a very robust and dynamic filtering and sorting system and is probably the most impressive implementation of a system we have. At one point we discussed having extra pages that contained specific feedback items separated by categories, but in an effort to show feedback items that could be relevant for a student, unknowingly to them, and also to streamline navigation of the website, the choice to approach these issues by implementing this system was the right one. This page would also act as the central hub of the website, where users are more likely going to notice things like inbox alerts and updates.

Some features that weren't part of the original requirements include a light and dark mode, which is present as a toggleable option on the header throughout all pages.

A password recovery system, that creates a unique token that is sent to that users email address.

As a result of the discussion of the observer design pattern in the TSE lectures, we thought it would be a good idea to try to implement it in our design, this led to the creation of an inbox and alert system, which is subscription based.

4.3 External Evaluation

To get a better understanding of where we were at in the final stages of development, each team member took our latest prototype and presented it to a potential user of the system. After outlining the concept of what our website was trying to achieve, we gave each user some basic tasks to complete as a measure of our system's usability.

- Task 1 – Create your own account and log-in
- Task 2 – Add a comment to any existing unresolved Feedback
- Task 3 – Create your own Feedback, mark it as closed and then delete it
- Task 4 – Change your profile picture from the default avatar
- Task 5 – Reset your password using the password recovery system
- Task 6 – Delete your account

We have compiled a list of pros and cons we received as feedback from these users below:
Pros:

- Responsive Design – The website is responsive and performs well across various devices, including full mobile support, ensuring a consistent user experience
- Clear User Interface – The interface design is clean and well-organised, making it easy for users to understand and navigate
- Multi-theme Support – The availability of both light and dark modes caters to users with different preferences and visual needs
- Colour Blind Friendly – Both themes have been carefully designed to support users who are colour blind by ensuring that no functionality relies purely on colour
- Personalisable - Users can upload their own profile picture, and there is even an option to display a short bio on their profile page

- Input Sanitation – All elements of the website that accept user input have been appropriately santised using prepared statements before access is granted to the underlying database, preventing malpractices such as SQL injections

Cons:

- Limited Membership Flexibility - The restriction of users to a single course membership could be limiting for those who require access to multiple courses, especially staff and admins
- Filtering Constraints - In addition to the previous point, users would benefit from the option to filter existing feedback items displayed on the homepage by course
- Notification Constraints - The current notification system's limited options for customisation may not fully meet the diverse preferences and needs of users who require more granular control over alerts
- Accessibility Enhancements - While accommodating color blindness, additional accessibility features such as screen reader compatibility or keyboard navigation improvements could further enhance the website's inclusivity

From these points of feedback, we can safely say that our final product is successful in what it set out to achieve, and we are confident that it's in its final stages of development.

4.4 Future Development

The future of Feedtrac has been a topic of discussion throughout the development, one of the worst things about still being a student is that we often work hard on projects but must leave them behind as another module needs our focus. So in discussion we thought that Feedtrac would be a great opportunity to add a valuable project to our portfolio, the consensus among our group members is for us to fork the project when it is complete and keep working on it little by little as a way of keeping it alive while at the same time, using it as an excuse or as an exercise to improve as developers.

Due to our inexperience with many of the concepts we tried to implement in the development it is only natural for us not to have done everything the right way, despite our best efforts some things are just "good enough" to make it work.

Because we were unfamiliar with both the MVC design pattern and Observer we made some mistakes with the implementation, so there is an opportunity to revisit these concepts and refine them in the future. Some other details like having email verification for newly created accounts, or looking into actually making the website safe from external attacks are valuable or designing elements in a way to make them interchangeable or more modular, all of these are things that we can look at in the future to improve.

There are many ways in which we could continue the development of FeedTrac. We could start by building a broader community of software developers that want to join the project beyond our initial team. This would really make use of the fact that our project is open-source which is a decision that we made early on and is what led us to make our GitHub repository public and include a GPL-3.0 *The GNU General Public License v3.0* (2007) license there which is a legal framework that encourages open development through code sharing and transparency.

We should consider what features people may want that were left out from the release version. Given that the project is open source, we may receive issues and comments on GitHub not just from developers but also from users that request updates and bug fixes. Continued development and maintenance are vital to keeping the project alive since as time goes on, there is a phenomenon in programing called “code rot”Todd (2024) whereby code tends to stop working if it’s not maintained an updated with the newest programming language frameworks and dependencies.

On the whole, we are very happy with our released artefact. It meets nearly all of our original project requirements defined in our design brief, and has even expanded in areas we deemed too basic.

Group Work Conclusion

5.1 Co-authored Reflection and Team Evaluation

Upon reflection, we generally worked well together as a team. Since we were using Discord to communicate, we were always able to ask each other questions and had separate chat rooms for frontend and backend developers to collaborate separately. However, there were group members who did not contribute to the GitHub repository or the final report which increased the workload for everyone else. If we had all seven team members working together, we would have made much more progress in a shorter amount of time. It is unfortunate to think that we didn't get full cooperation with all team members, since the different perspectives that those missing group members could have provided may have led to unique ideas and innovations that could have taken the project to an even greater level of creativity.

5.2 Individual self-reflection

5.2.1 Toby Armond

Initially I added some documents to the project, however, didn't use them enough during the development of the artefact. This led to code-styles outlined within our code-style document not being used properly and therefore the artefact being harder to develop with more technical debt. I enjoyed adding backend elements to the project and working with the documentation on the project.

5.2.2 Archie Baldry

I believe my prior frontend experience certainly helped accelerate the project's development, as those working on the backend were able to directly interact with their contributions. Additionally, I felt that I was good at going through and cleaning up the project wherever necessary, such as refactoring code, removing deprecated features and making sure each page conformed to our style guide. I am very thankful to my team for creating resources such as work journals and to-do lists, as well as organising regular meetings, as this helped me to properly spread my workload. However, the majority of my contributions to the project came towards the end of the development cycle, and so going forwards I would like to try and manage my time more effectively.

5.2.3 Marco Caçoete

I sometimes prevent myself from trying to do some tasks because they seem too outside of what I am able to do, to combat this for this project, I decided to just throw myself in head first against a challenge by claiming something as my responsibility before even considering if I could do it. I tried to be a driving force and motivate the other group members and keep everyone involved in the conversation. Regarding points I need to improve, I would say that I try to get things to work not caring too much at how ugly or inefficient my solutions are, this sort of thinking is self-sabotaging and doesn't help to fight my bad habits.

5.2.4 Earl Smalley

My specialisation on this project was my organisational skills. I set up a TODO list which focused everyone's efforts on the high priority tasks. I also created optional goals that would improve the website functionality. I created the shared document that we used to be able to collaborate on this report which included many pages of notes that I wrote from the lectures and

extracted keywords. I was also able to cover for the team by increasing my contribution to the codebase when they needed to take a break away from the project during the Easter holiday. I should've made notes during lectures instead of afterwards as it's time-consuming and reduced my progress on the production of the artefact. I also wasn't as good at communication as I found it difficult to keep up with the group chat sometimes. I also never used a microphone to communicate via voice when the rest of the team were in an online group voice call opting to use text chat instead. The way I can resolve these issues in the future is by ensuring that I improve my communication skills so that I can keep up to date with what everyone is working on and contribute more at the meetings.

5.2.5 Oliver Smith

During development, I approached the project with enthusiasm right from the start, maintaining consistent contributions while prioritizing transparency with development ideas and design choices. While I think I made some good contributions, I recognise my eagerness to get started may have been off putting for other team members, resulting in me making many foundational design choices for the system independently. While this approach ultimately worked out, I feel like many of the group members felt excluded from early designs of the database and choice of programming structures. I now acknowledge that a more collaborative environment could have provided diverse viewpoints and enhanced the overall quality of our product. I will aim to develop greater inclusivity in future collaborations.

5.3 Individual Contributions

Name	**Contribution** (%)
Toby Armond	20
Archie Baldry	20
Marco Caçoete	20
Lorna Foster	0
Harry Paton	0
Earl Smalley	20
Oliver Smith	20

Table 1: *Contribution breakdown - Assessment 3*

Media, and Artefact Links

We have produced an unlisted YouTube™ video highlighting the main features of our artefact available here (closed captioning available): https://youtu.be/y7DZdYNA8cM

Our artefact can be accessed on GitHub. Instructions on how to set up the project for marking can be found in the README.md file. https://github.com/OSmith132/FeedTrac

References

caniuse.com (n.d.), 'Css flexible box layout module | can i use... support tables for html5, css3', https://caniuse.com/flexbox. [Accessed 08-05-2024].
URL: *https://caniuse.com/flexbox*

Drumond, C. (n.d.), 'What is scrum? + how to start', https://www.atlassian.com/agile/scrum. [Accessed 08-05-2024].
URL: *https://www.atlassian.com/agile/scrum*

Freeman, E. & Robson, E. (2021), *Head First Design Patterns*, A brain-friendly guide, O'Reilly, CA,USA.
URL: *https://books.google.co.uk/books?id=3GqAzQEACAAJ*

Indeed (2023), 'A Complete Guide to the Waterfall Methodology — indeed.com', https://www.indeed.com/career-advice/career-development/waterfall-methodology. [Accessed 07-05-2024].
URL: *https://www.indeed.com/career-advice/career-development/waterfall-methodology*

Martin, R. (2008), *Clean Code: A Handbook of Agile Software Craftsmanship*, Robert C. Martin Series, Pearson Education.
URL: *https://books.google.co.uk/books?id=i6bDeoCQzsC*

Suehring, S. & Valade, J. (2013), *PHP, MySQL, JavaScript & HTML5 All-in-One For Dummies*, –For dummies, Wiley.
URL: *https://books.google.co.uk/books?id=p9BuBgAAQBAJ*

The GNU General Public License v3.0 (2007), https://www.gnu.org/licenses/gpl-3.0.en.html. [Accessed 08-05-2024].
URL: *https://www.gnu.org/licenses/gpl-3.0.en.html*

Thomas, D. & Hunt, A. (2019), *The Pragmatic Programmer: Your journey to mastery, 20th Anniversary Edition*, Pearson Education.
URL: *https://books.google.co.uk/books?id=LhOlDwAAQBAJ*

Todd, S. (2024), 'What is code rot?', https://www.linkedin.com/pulse/what-code-rot-stuart-todd-pnuve. [Accessed 08-05-2024].
URL: *https://www.linkedin.com/pulse/what-code-rot-stuart-todd-pnuve*

Wikipedia (2024), 'Dead code - wikipedia', https://en.wikipedia.org/wiki/Dead_code. [Accessed 08-05-2024].
URL: *https://en.wikipedia.org/wiki/Dead$_c$ode*