University of Glasgow | School of Computing Science

Level 3 Project Case Study Dissertation

Team CS36 - The Institute of Infections, Immunity & Inflammation

Martina Karaskova - 2478724K
Tara Nally - 2439105N
Conor Tucker - 2455252T
Ricky Arthurs - 2465714A

28 March 2022

Abstract

This case study presents a team project developed for the Institute of Infections, Immunity & Inflammation. The project idea points out a major difference in the diagnostic capabilities between the western world and low-income countries. Our project aims to solve this issue by creating an application which will supplement the need for these diagnostic tools. The document elaborates on the process of the project development, design and testing decisions and other software engineering practices the team used throughout the development process. Simultaneously, the team reflects on the experience with these practises and talks about their overall experience working on the project.

1 Introduction

This document presents a case study for the Team Project 3 computer science course developed by team CS36, made up of 4 students. The project was proposed by The Institute of Infection, Immunity & Inflammation at the University of Glasgow for the purpose of developing a diagnostic tool which will help low-income countries with diagnostics of parasitic infections as well as provide better communication between the Glasgow University Institute and other countries.

The customer presented the project as an application for uploading images with different types of users. With this objective in mind, a point-and-click website was developed. This proposal was turned into a website which implemented different user types with the ability to upload and download different files as well as monitor the website content. The website's functionalities are presented within a user-friendly and functional front-end interface.

This case study provides thorough analysis of the software development process. This covers information about the project customer, the motivation and inspiration behind the proposal, goals and objectives for the team to accomplish, the final product, and a description of utilising different software engineering practices throughout the project development. Subsequently, this is followed by the reflection on the continuous product iterations. Further, this document talks about design decisions, different software testing practices and reflects on the challenges the team faced and the knowledge they gained from it.

The document is concluded with a summary of this case study as well as the team's overall reflection on the project and the experience working as a team.

This case study is structured as follows:

Section *Project Background* discusses an overall background for the project. This consists of a description of the customer and their vision for the proposal, aims and objectives and overall requirements regarding functions and features for the end product.

The team's collaboration and engagement throughout the project development is presented in the *Reflection* sections. This consists of the discussions regarding the use of General Software Engineering Practices within the team. Further, a detailed description of project development cycles is presented followed by the discussion of implementing different design technologies as well as challenges this had created. Lastly, the section talks about quality assurance along with the team's decisions for their implementation.

In the *Conclusion* section of the document, the team reflects on the process of developing a product for a real-life customer, the lessons and skills gained from this responsibility and experience and how this can be applied for future projects to improve on software engineering practices.

2 Project Background

2.1 Customer Background and Objectives

The Institute of Infection, Immunity & Inflammation is a research institute in the College of Medical, Veterinary & Life Sciences at the University of Glasgow. The mission statement of the institute is "scientists and clinicians working together to promote and develop research, drug discovery, and improvements in patient care." Our team communicated with Hannah Bialic, a Public Engagement manager for the Wellcome Centre for Integrative Parasitology. The Centre aims to research "parasite biology and host/parasite/vector interactions" [13]. The Centre has recently been collaborating with the College of Medicine in Blantyre, Malawi.

The Wellcome Centre asked us to create a tool for collaboration between clinicians and researchers in Scotland and Malawi. The audience is made up of three groups: the general public that want to learn more about parasites, researchers that want to share their scientific papers, and clinicians that want to keep up to date with the latest discoveries about parasites and collaborate with other clinicians to make quicker diagnosis of parasitic infections. The website should have the ability to post images from microscopes to be used as reference images during diagnosis.

The website should have a clear, usable design to account for the varying backgrounds of users that will have access to it. The team took inspiration from the Wellcome Centre's webpage [15] and the WebMD website [14].

Due to the different needs of the three groups, we created a different portal on the web page for each. This was also beneficial to ensure the data is secure. The clinician section, in particular, contains sensitive medical information so it was important that the data is secured. The team decided to use an account system with three different levels of access permissions. Additionally, only the administrators are able to create an account and can decide the permissions for all other users.

There was also a need to have an admin user type that can populate the public content and moderate the private content posted by users to the site. The team decided to take advantage of Django's built-in admin interface to provide these features to the admin users [5]. As a result, there is a fourth type of user with further access permissions, the admin or superuser, that can access every feature of the website including the admin interface and control everything in the database.

2.2 Final Product

The final product [7] is a web application hosted on PythonAnywhere [11]. The team developed it using Django, a Python web framework [4]. The website is organised into four sections: the public portal, the researcher portal, the clinician portal, and the admin portal. Only admin users are able to register accounts and control access

rights to different pages on the website. However, users are able to change their username, email, profile pictures, and passwords.

Users with accounts and non-registered users can access a page where the admins post public engagement content, information about the parasites that the Wellcome Centre researches, an about us page, and a page that allows them to contact the administrators of the website with any questions or comments.

The researchers can access the public portal and the researcher portal. The main page of the researcher portal is a list of researcher posts that can be filtered by which parasite they relate to using an index on the side of the page. Users can go to pages for each individual post to view and download files attached to the post and create comments on the post. Users can also add research posts through this portal. The clinicians can access the public, researcher, and clinician portals. The clinician portal is almost identical to the researcher portal, except for having more restricted access rights.

Finally, the admin users are able to access all four portals. The admin portal is the Django admin site and allows admins to view and moderate posts, files, comments, and users. Admins are also able to create instances of the aforementioned models, edit the about us page, and create public engagement posts.

3 General Practices

For this project we decided to move along with agile development, which encourages rapid and flexible change by allowing for adaptive planning, progressive development and frequent improvement. This was used in the project to move features and updates from the non-primary branches to being delivered to the customer in the shortest time span possible. Whilst carrying out research into the agile methodology we found that in multiple research papers major corporations were finding that the agile methodology allowed for less critical errors and defects within their applications and allowed for a more hands-on approach with the customer. An example of this can be seen with CISCO(who are the world leaders in networking for the internet), who discovered that within one of their projects, were they began to implement the agile method, defects and critical issues were decreased by 40%, there was no overtime needed and the project was delivered on time [3]. When comparing against more traditional methods such as the waterfall method, we see that within this study carried out over 25 years of investigation into projects that the agile method is 3x more likely to succeed, with a 40% success rate, whereas the waterfall method was 2x as likely to fail with a failure rate of 28% and a success rate of 13% during this study[1].

When selecting which framework we would use within our project development we were prompted by the course coordinators to use Scrum. Although we were prompted to use this framework we still carried out research into others, such as Extreme Programming and Kanban. After carrying out this research we as a team felt that

the Scrum framework was better suited to us as developers. It provided a more hands on approach with the client and allowed them to interact more easily with the team. This allowed the team to meet their required specifications and have a more structured development process with the roles assigned, whereas Kanban has no roles or hierarchy [9]. We see that within Kaban there is no need for timeboxing within the project which may cause a loss of control over the schedule, for when features are needed to be implemented[8]. Although, Kanban would be more useful for situations where multiple small releases are required or where tasks are not dependent on each other. For instance when bugs in software are reported.

The Extreme Programming methodology was not chosen due to the issue that it may not be possible to change any of the priorities the client specifies, which may hinder some of our progress at points during iterations. The Scrum framework allows for change of priority within issues between each sprint, which allows for a more flexible and robust approach to development compared with the Extreme Programming methodology.

Through the use of our agile framework we used the Scrum methodology which defined the roles that we as owners of the application would take when in meetings and outside to help streamline the development process. The roles that we took on were the Scrum Master who contributes and helps those within the team to help further the development of the product, the Scrum team, who implements the software after each series of sprints that are carried out and also the Scrum product owner who is in charge of time management and making sure that the product is delivered on time and to manage any expectations that the client may have and make sure that they are met within the specifications after each sprint or meeting.

During the use of Scrum we as a team decided that a meeting each week within a designated time slot, that was planned by the Scrum Master would allow for an improvement of communications within the team. If issues arose then they would be dealt with separately within a meeting that would be planned for issues to be fixed after the Scrum meeting. The Scrum meeting was kept to a time limit of 20 minutes so that each team member was able to discuss what they had finished within the last week, what they were currently working on and if they had any issues or anything preventing them from finishing the feature they were working on. These short meetings allowed for wide communication within the team as each developer had a sense of what needed to be done for each developer within the team and allowed other developers to set up a separate meeting for help with their feature or code outside of the Scrum [12].

The Scrum methodology allowed the team to feel more comfortable working closely together. It worked well within the team as it allowed each member to work on an area which they felt they excelled in, this allowed the project to progress more efficiently. Due to the Scrum meeting occurring on a weekly basis this allowed some tasks to slip in the timeline and in some cases were not raised until the next Scrum meeting. To avoid this in future we would try to reduce the time between meetings or have daily standup meetings to identify any issues early so the team can quickly find a solution. In some cases the current sprint was finished before the next sprint was due, this

meant some resources were not used optimally. In this situation the constant flow of Kanban may have been useful. One of the main benefits with the Scrum method was the clear sense of direction within the project, this ensured all developers stayed on course. Therefore, we can see that through the use of the Scrum framework, team morale role as developers were presented with a sense of accountability and structure within the team. This allowed for the completion of specific tasks and challenges that were set out.

4 Iterations

4.1 Iteration 1

In the first sprint the team determined the main objectives for the project. The team was already familiar with developing a web application however, further research on various issues such as uploading large images and creating different roles for private users would have to be conducted. Therefore, the team assigned different tasks to each team member to research the topics needed for the development of the project.

During the iteration, while setting up the Django environment and brainstorming ideas for the development of the website more questions arose regarding the differences between different roles for the private users and the design. It was then agreed that the team would use Django as the main framework for the project and wireframes were created to ensure the team and the customer were on the same page.

4.2 Iteration 2

Since the team researched all the necessary information before this sprint, the communication between the team members and the customer was more effective in the second sprint. The team presented their take on the design by creating wireframes and asked further questions about specific issues that occured while researching the technologies. This included ensuring the team has a correct understanding about the user roles as well as asking the customer for sample data to test large file compression.

During the iteration, the customer sent over their ideas for the wireframes while emphasising that they are giving us full creative control. However, the team decided to prioritise functionality before design and implementing only basic front-end at first. The team then split work among the team members based on their preference and previous experience. A simple website with basic functionality was then developed to present to the customer.

In retrospect, the team should have been less technical during the sprint with the customer. The customer was not familiar with some terms used by the team and this could have created miscommunication. Further, miscommunication also occurred within the team when dividing the tasks. The team immediately improved on this

by utilising wiki and creating issues assigned to a specific member of the team. By tracking tickets and milestones it ensured that a team member could not fall behind and hinder the rate of development for that sprint as these tickets were regularly checked by other team members and discussed at meetings.

4.3 Iteration 3

In this sprint, the team presented a short demonstration of basic features implemented within the application. The customer gave valuable feedback which was then used for further development.

The team made further progress in this iteration. More basic web pages were implemented as well as interfaces for private and public users. More research was then done creating different user roles and the team attempted to implement them. This, however, was met with a lot of bugs and therefore, the team started working on creating tests to ensure the application would be easier and less time-consuming to debug. Later, the customer sent over samples of large files for the website and the team found out the file compression would not be needed since the files did not exceed the Django framework limit.

During this stage, the team should have taken more initiative with implementation of the back-end. The development was not progressing as fast, since the team members were not focusing on one thing at a time. This created issues with the delivery of the product for the next sprint, which was then rushed and not as polished as the team wanted it to be. The team took this into account and decided to assign weekly responsibilities for each team member and assign new ones every Wednesday during the team's meetings.

4.4 Iteration 4

Following the demonstration of the hard-coded prototype in the last sprint, the team created a database and a population script. This allowed for full implementation of user roles and creating posts with files and comments.

In this iteration, the team finished the majority of the implementations within the back-end and started working on the front-end as well as an admin interface. The front-end had to be implemented in a way so that the customer could populate and change the website how they saw fit through the admin functionality. Due to deadlines nearing up, the team decided to utilise the default Django Admin interface. Further testing was also implemented to speed up debugging.

Looking back at this iteration, the team should have started the implementation of the front-end earlier. The team underestimated the amount of work that would go into the front-end and rushed a lot of implementation. However, the team felt the testing made a huge difference with finding errors and saved a lot of time which could then be invested into development of other aspects of the application.

4.5 Iteration 5

This sprint consisted of demonstrating a fully working website while going through features in detail to find any minor issues.The team also agreed to arrange a meeting with the customer to help populate the website for the final demonstration.

The population of the website was based on files provided by the customer as well as information from the official website for The Institute of Infections, Immunity & Inflammation. Populating the application through the admin interface instead of hard coding the data within the HTML created some issues with formatting. Therefore, the team decided to implement a rich text editor called CKeditor within the admin interface. This addition would make it very easy for the customer to make changes, as this editor is a more appropiate interface for a non-programmer.

This iteration also saw the addition of a'contact us' feature which the team connected to a gmail account. This created some issues which took some time to debug due to Google Authentication. The login information was then given to the customer.

4.6 Final Demonstration

During the iteration, the team debugged some remaining issues and fully implemented the front-end of the application using JavaScript. The team decided on using PythonAnywhere for the deployment of the final product. However, several issues occured while trying to migrate the project. The main issue was with Gitlab being set to private without the possibility of changing it. This meant the PythonAnywhere could not access the repository and upload the files onto their website. Therefore, the team decided to migrate the project to a public Github repository. This, however, created another issue. The size of the Gitlab repository was too large and exceeded the Github migration limit. Resolving this issue became very time consuming since the team had to purge files from repository history to manually upload into Github.

In hindsight, the reason why the Gitlab repository became large is due to accidentally committing large files used to test the website upload function. Even after deleting these files, they were still stored within the commit history and purging them became very difficult. Unfortunately, none of the team members were aware of this Gitlab functionality. Therefore, this could have probably not been avoided, however, the application should have been migrated to Github sooner and updated regularly.

Before the final demonstration, the customer was given a user guide created by the team which included all the necessary information about the project such as step-by-step deployment of the application, licensing, detailed explanation of all the implemented features, and credentials for PythonAnywhere, populated users and a Gmail account used in the 'contact us' feature.

5 Design and Usability

Whilst originally designing the skeleton of the webpage, we decided to create a nav bar feature that would sit at the top of the pages in a fixed position. This was done to allow for the users to navigate through each of the web pages without the nav bar scrolling down with them in order to allow for less clutter within the screen and for ease of navigation to other webpages when needed. We also had another design that would be a drop down menu that would also display each of the pages that could be navigated to, but after consulting with our client on the design of this, we finally decided on sticking with the nav bar instead of the drop down menu due to lack of content and making the top of the webpage feel empty. The nav bar was also decided upon due to its flexibility of being able to change within the interface according to user permissions. For example, the nav bar would allow for the clinician portal to display for clinicians, but not for researchers when logged in. Another feature that was designed to be added was an A-Z link list inside the parasite page, but due to time constraints and the issue being shown as a low priority, the team decided to drop this feature and instead move forward with a search bar that allowed for the type of parasite to be searched and displayed.

For the use of the images within the website, the team discussed with the client if they had any images of parasites that were available or if they wanted to be used in each portion of the website. The client agreed to sending over images of parasites from their own research to allow for us to populate portions of the website to start, such as the parasites pages with their respective images and also pulled images from the Institute of Infection, Immunity & Inflammation, with permission from the customer to populate the about us page. We also discussed with our client about a way of allowing them to have complete control over the images within the website as this was requested by them within the sprint meetings. We then implemented this by creating an admin portal that allowed the client to delete, replace or add new images into different sections of the website allowing them to have complete control over the design of the website and allowing for more updated images to be posted if necessary or deleting any unnecessary ones. Looking back on the project, we as a group feel as if the admin feature would have been more beneficial to appear on a separate link rather than within the nav bar, as it would have provided more fluidity to the design of the webpage. From this experince of implementing this, we can see that giving the admin full control over the images was beneficial for our website, as it met the goals that were specified by the customer.Although in another project it may be less beneficial if the developers wanted to allow all users to interact with aspects of the website and therefore they would not wish to use this approach that we carried out.

Within the website we also discussed ways of allowing for the use of forms, for uses such as the contact us page and the edit profile page. We discussed with the client on any designs they wished to be used to create the forms within our meetings, and the feedback that we received was to create the forms in any we saw fit, as long as they allowed for ease of use and avoided confusion when filling them out.To implement these forms into the website we made sure to follow steps set out by ourselves as a team to create the most user friendly forms. The first rule we set out and implemented

was to not allow for huge amounts of white space and not allow it to take up the entire screen, as it may annoy the users and make the form seem more formidable and discourage the user from using it. We did this by setting a height cap and width on the forms to begin with, and allowing them to expand as the user added more information into textboxes, allowing for a more flexible design. We also allowed for placeholders within text fields to allow for the users to understand what needs to be placed within the fields and also for buttons we decided to follow the rules of placing buttons where they are expected to be, properly size our buttons according to priority within the screen and labelled the buttons with what they do. [2] Looking back on this we as a group felt that these designs for forms worked well within our application as it allowed for easy use and navigation throughout, the feel of the form through the use of the placeholders and button sizing would also work well within other team projects as it would allow for uniformity within their forms, that would allow for ease of use for users and increase their design strength.

In conclusion we see that the main goals that were set out for the website with the customer, was to have a flexible website that could allow for dynamic change throughout the website from the admin portal when prompted and allow for clarity and ease of navigation throughout the website for users. Looking back on this we a team felt as if these goals were met to a high standard as we were able to learn how to implement these goals through learning as a team how to design forms , and creating an admin portal with superior control. Although there were some features that were chosen to be dropped, the website still maintains a high level of design and usability. In the case of the A-Z list and other features that were dropped, we as a team feel if more research went into these topics and time-constraints were less of an issue, then these features being implemented could have left us as a team with a more advanced and flexible solution for the client.

6 Quality Assurance

The team carried out early testing of the back-end in the form of unit testing. Unit testing was selected due to its easy implementation and ability to test individual parts of the software. Specifically, Django's unit tests were used for its use of a Python library module named 'unittest' which uses a class-based approach.[6] The implementation of these tests happened in the earlier stages of the development to detect early flaws and ensure the software worked as intended. The team implemented as many automated unit tests as possible due to their easy and quick implementation. This helped greatly in the early stages of the project development by saving time and in identifying small bugs while making frequent changes to the code.

Further into the development process, the team started experiencing more frequent bugs. Debugging of the broken parts of the software would consume a lot of time and therefore, it was decided integration testing would be implemented. This type of testing tests the application as a whole by combining different software modules and testing them as a group. The team greatly benefited from this testing despite the implementation being more difficult and time consuming compared to the unit

testing. Integration testing made finding bugs easier and quicker and oftentimes found errors the team would not immediately notice.

Toward the end of the project development process, the team implemented some end-to-end testing as well. This type of testing tests the functionality of the software by replicating a real user. Since these tests only work with full working features, it was implemented towards the very end of the development process. The team tried to initially implement these tests more thoroughly but found it costly in terms of time. Therefore, end-to-end testing was only implemented by performing different user actions and checking whether the software response matches the expected one. The team found even minor testing like this to be helpful in ensuring the application is ready for the handover.

In order to fully ensure the software functions smoothly, the team integrated a Gitlab CI/CD Pipeline. It ensured correct and errorless migration and population of the database. Furthermore, the pipeline would run the above-mentioned tests to indicate whether the current state of the application functions correctly. This functionality would run every time a commit was made or a branch was merged.

According to various software testing practices as well as specialists in software development like Martin Fowler[10], continuous integration is important to address bugs quicker and improve software quality. Therefore, the team tried to integrate all types of suitable testing which was then run within the pipeline to help solve any issues within the software. The team felt the implementation of testing within the pipeline made a huge difference. Oftentimes, commits made by the team members created code bugs and issues with migrations. After this implementation, the team could rest assured that errors in production and in progress were detected early on. Looking back, this saved a great amount of time which was then invested into further development of the project. The team would definitely benefit from further testing however, due to time restraints, implementation of other types of testing would not be manageable.

7 Conclusion

The Institute of Infection, Immunity and Inflammation was the first agile project that the team has carried out and allowed for the team to expand on our communication and practical experiences. Whilst carrying out the project the team came face to face with issues that would present a challenge for them to overcome. This allowed the team to become more knowledgeable within this field and learn new lessons within different aspects of the project.

Whilst carrying out our development process as a team we used the Scrum framework to divide roles and tasks between us. For each of our iterations we managed to develop them by using three characteristics: Speculate, Collaborate and Learn. This was done to encourage transparency between the customer and the developers and allow for an early delivery of product. However as a team we encountered issues at

the start, when specifying what was needed from the customer through the use of broad questions, this affected our progress through the development as we did not have all the information to carry on with development. From this we were able to learn and adapt within our next sprint to fix this issue, by pre planning and having more targeted questions at different aspects of the project for the customer. The team managed to gain experience of using the agile framework that was implemented, by learning how to quickly respond and fix the issues at hand.

One of the biggest technical challenges that the team faced was choosing which platform we wished to deploy the application on. For our platform we chose PythonAnywhere, which did present us with small issues. For example PythonAnywhere does not allow for storage of large files within the application when deployed. This caused the efficiency of the team through development to drop and become hindered. Looking back at this choice as a team we felt these issues were caused due to lack of research into other platforms and familiarity with PythonAnywhere. This could have been avoided, we feel with more research into other platforms, to give us as a team a better understanding of which platform would suit our application best.

We also learnt that some issues arose from lack of clarity when communicating with the customer on specific details or features they required. This impacted the team as a whole as it slowed down development as another meeting would be scheduled with the customer to sort these issues. From these issues we managed to learn to target our questions or set up pre plans before our meeting to increase the clarity of communication with our customer.

Throughout the development of the project the team undertook multiple tasks using multiple software technologies, such as Django , JQuery, Javascript and Bootstrap that they might previously have encountered or used. This allowed the team to gain new experience within these technologies through means of research, hands on problem solving. Although we see that due to some team members not being as fluid in some of these languages as others, it slowed down the development process to allow for them to catch up with the levels of other team members. This also came with a positive note, as it allowed for us as developers to hone our research and study skills, when learning these new technologies.

Overall we can see that throughout the project and the development stage the team has overcome many challenges and learned from their mistakes, to gain experience in the different areas of technology that were used throughout the project. Throughout this project the team has learned the importance of teamwork and integration of all team members during iterations. These expertise will allow the team to take these new found skills and experiences into future teams to possibly prevent any issues happening that they have already dealt with or to take lead on a certain version of software that no one has experience with. Therefore we as a team we feel the invaluable knowledge we have acquired about methods of agile development will not only be beneficial to future endeavours but will serve as an aid in each team member's professional career.

References

[1] Why Agile is Better than Waterfall. https://vitalitychicago.com/blog/agile-projects-are-more-successful-traditional-projects/. Accessed: 2022-03-30.

[2] 7 Basic Rules for Button Design. https://babich.biz/7-rules-of-buttons/. Accessed: 2022-03-30.

[3] Case Study: Cisco. https://www.scaledagileframework.com/cisco-case-study/. Accessed: 2022-03-30.

[4] Django. https://www.djangoproject.com/. Accessed: 2022-03-30.

[5] Django Admin Site. https://docs.djangoproject.com/en/4.0/ref/contrib/admin/. Accessed: 2022-03-30.

[6] Django on Writing and running tests. https://docs.djangoproject.com/en/4.0/topics/testing/overview/. Accessed: 2022-03-30.

[7] Final Product. http://parasitecs36.pythonanywhere.com/home/. Accessed: 2022-03-30.

[8] Demystifying Kanban. https://www.scrum-institute.org/Scrum_Roles_The_Scrum_Team.phpv. Accessed: 2022-03-30.

[9] Mattias Kniberg, Henrik & Skarin. *Kanban and Scrum: making the most of both*. C4Media.inc, 2010. Accessed: 2022-03-30.

[10] Martin Fowler on Continuous Integration. https://martinfowler.com/articles/continuousIntegration.html/. Accessed: 2022-03-30.

[11] Python Anywhere. https://www.pythonanywhere.com/. Accessed: 2022-03-30.

[12] . https://www.scrum-institute.org/Scrum_Roles_The_Scrum_Team.phpv. Accessed: 2022-03-30.

[13] University of Glasgow Institute of Infection, Immunity & Inflammation. https://www.gla.ac.uk/researchinstitutes/iii/. Accessed: 2022-03-30.

[14] WebMD. https://www.webmd.com/. Accessed: 2022-03-30.

[15] Wellcome Centre for Integrative Parasitology. https://www.gla.ac.uk/researchinstitutes/iii/wcip/. Accessed: 2022-03-30.