Developing a Python-Based Social Networking Web Application

Mustaq Hussain
King's College London Mathematics School

October 2021 - April 2022

Abstract

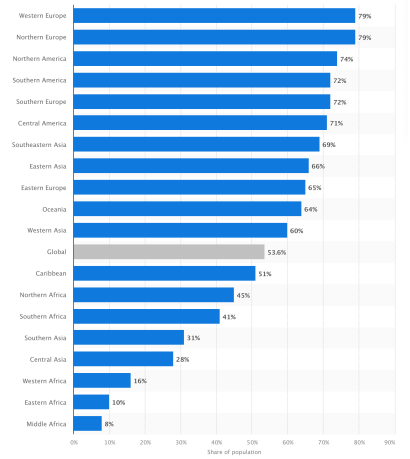
Social media platforms facilitate the creation and sharing of content and, in doing so, have created paradigm shifts in the methods which individuals use to communicate. The use of these interactive technologies is heavily integrated within contemporary society and, thus, an effort should be made to comprehend the mechanisms behind their creation. This paper attempts to do so as it details the software development process undergone while creating 'Ophidia', a social networking web application developed using the Python web framework Django. This includes the comparative analysis of a range of software development life cycle methodologies and an assessment on the Python web frameworks: Flask and Django. This report further outlines the design, testing and eventual deployment of the software alongside the post-deployment maintenance patches made to the programme in response to user feedback.

The resultant artefact can be found available at http://ophidia.herokuapp.com which includes a fully functioning user authentication and profile system along with posting, searching, rating, commenting, following and direct messaging functionality. The source code of the application can be found at: https://github.com/mhus04/DjangoSocialMedia

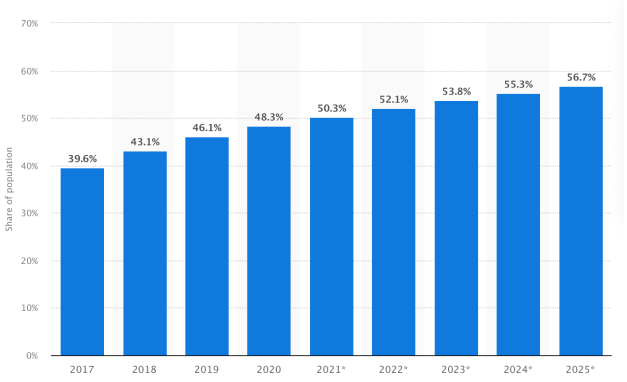
Contents

1 Background

Social networking platforms have undergone a remarkable success story regarding their exponential growth in adoption and usage figures. Various statistics are capable of illustrating how social media has become an integral part of contemporary society.



(a) Global social network penetration rate as of January 2021, by region



(b) Social network penetration worldwide from 2017 to 2020 and extrapolated until 2025

Figure 1: Current and projected future social network penetration rates [1][2]

Figure 1 depicts the social media penetration rate by region and then the global annual penetration rate extrapolated up until the year 2025. In regions with the highest proportion of developed countries, the share of the population contributing to social media traffic can reach up to three quarters [1] and the global share of the population with access to social media has only risen in recent history [2], indicating that the impact of social media will only become more pronounced and widespread.

Social networking applications are so prevalent that they have triggered paradigm shifts on how we express ourselves to each other and even our methods of accessing content and information.

Figure 2 is representative of this where respondents were asked about whether social media had an influence on thirteen total aspects of daily life, resulting in ten out of thirteen attaining a majority vote. These encompassed censorship as well as polarisation and foreign intervention in politics, aspects that underline the extent of social media's influence.

The idea that social media has the influential capacity to govern who will or will not form our legislative bodies is not irrational and has, in fact, happened before. During the peak of the Arab Spring, and in specific, the Egyptian Revolution of 2011, social media facilitated the spread of information which contributed to the eventual toppling of the Mubarak government. Egyptians used Facebook, Twitter, and YouTube as a means to communicate and organise demonstrations and rallies to overthrow the incumbent president. During the height of the revolution, the 23 most popular protest videos garnered an approximate of five million total views and the number of Tweets from Egypt increased from 2,300 to 230,000 per day [4]. Mubarak, ruling for a total of 30 years prior to the revolution, regarded the networking applications to be such a threat to the regime that the government temporarily cut off internet access for a period of time during February 2011 [5].

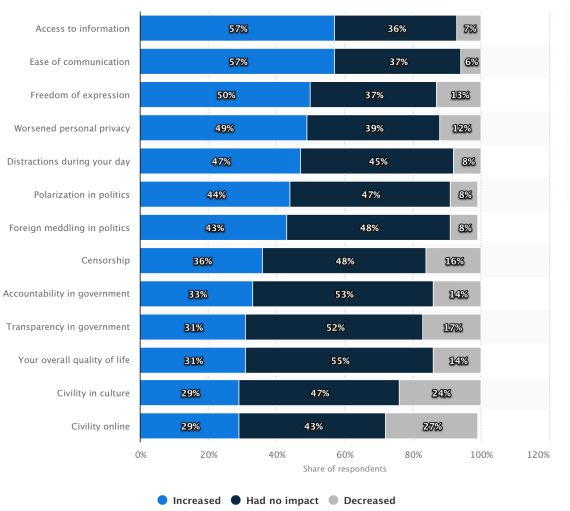


Figure 2: Share of internet users worldwide who believe that social media platforms have had an impact on selected aspects of daily life as of February 2019 [3]

We allow these platforms to have such a large leverage on our day-to-day lives, yet we do not make an effort to understand how they work. This paper attempts to do so by detailing the software development process undergone while creating 'Ophidia', a social networking web application developed using the Python web framework Django. This includes the comparative analysis of a range of software development life cycle methodologies and an assessment on the Python web frameworks: Flask and Django. This report further outlines the design, testing and eventual deployment of the software alongside the post-deployment maintenance patches made to the programme in response to user feedback. In undertaking this project, it has provided myself with an introductory experience in both back-end development and object-oriented programming whilst reinforcing my proficiency in utilising the Python programming language.

Long-term projects require effective project management to ensure a productive workflow and, thus, in order to establish an adequate guideline for the project, I decided to employ an SDLC model.

1.1 Software Development Life Cycle (SDLC)

The software development life cycle (SDLC), alternatively referred to as the software development process, is a standard project management framework [6] that defines the steps involved within the development of software at each phase, providing developers with systematic, step-by-step approach to programming. Employing an SDLC framework is critical to any software development project as it helps developers to clarify their goals, effectively manage software projects, verify that software is sufficiently tested before it goes into production, and increase the likelihood of completing the project within an outlined time period.

There are seven distinct phases within the standard software development life cycle [7], each of which has its own process and end product that that contributes to the following phase. They are as followed:

1. Planning. This is the stage where the developer plans and establishes the scope of the project, defining a rough outline of the final product. In a real-life scenario, this stage would be utilised to secure funding for the remainder of the project. This is not applicable to my EPQ as I intend to create the artefact without any monetary expenditure. The planning phase also entails the creation of a project schedule, defining the time periods within the subsequent phases must be completed within.

2. Requirements. This phase is used to define the objectives of the software to meet the requirements of the end-user and what the end product is intended to achieve. Existing systems can be scrutinised, examining its deficiencies to identify any remediations that can be then implemented into the current project in order to create an improved product. Requirements of previous system can be fed into new projects to establish any fundamental objectives that an application within the same field must meet.

3. Design. This stage describes the method in which the requirements will be met, converting the requirements that have been gathered in the previous stage into a software design plan by designing the IT infrastructure and system model. This can be done by stating the resources that need to be utilised to meet the outlined objectives and creating flowcharts to illustrate the system's architecture. This is done to avoid inefficiencies or potential crashes/bug that may appear without proper planning.

4. Development. Information from previous stages is fed into this phase to allow developers to create the software, ensuring that it fulfils the desired requirements. At the completion of this phase, the project should have functional software that can then be tested and deployed.

5. Testing. All functionalities of the resultant software is rigorously tested during this stage, checking and patching any errors or bugs that may impact the software ability to meet all the initial requirements and ensuring a quality product is created.

6. Deployment. Subsequently, the system is deployed to an environment where users are able to access the software and begin operating the system.

7. Maintenance. The software is continually monitored after it has been deployed to ensure the patching of any errors that may have been unidentified during the initial testing period that may hinder its ability to meet its defined objectives.

1.1.1 Comparitive Analysis - SDLC Methodologies

There are a multitude of different SDLC models [8] - each approach being its own variant of the standard framework and differing in the way the project progresses through its stages. Implementing an SDLC model that is appropriate for the current project is critical as each methodology has their own respective advantages and disadvantages which can aid or harm the efficacy of the finished artefact in terms of meeting its initial requirements and/or the efficiency of its production. This meant that a comparative analysis of SDLC models had to be made to ensure a suitable methodology was utilised. Due to the large quantity of variants available, a shortlist of the most appropriate models was analysed.

During my research, the Agile model [9] had attracted my attention due to its distinctions from traditional SDLC models as a result of its iterative nature. The application is decomposed into smaller iterations, often referred to as 'Sprints', after which, Agile employs a flexible approach where future tasks are based on what features need to be developed as opposed to the set structure utilised by the majority of alternative SDLC methodologies. This results in feature-driven development which evolves to meet the changing product requirements. There is a focus on regular testing as customer interaction forms the backbone of the Agile framework: constructive criticism provided by the user is fed into the next sprint, each of which vaguely follows the phases of the standard SDLC process as can be seen in the figure below.

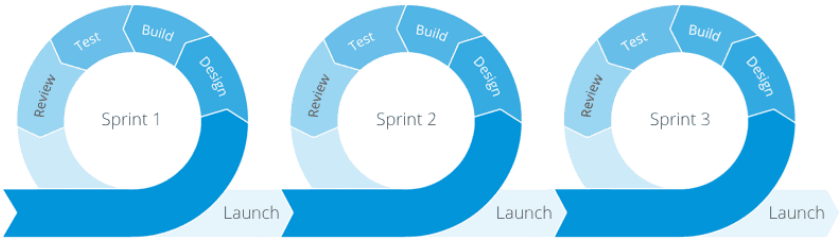


Figure 3: Visual Representation of the 'Sprint' format of the Agile methodology [10]

The feedback is used to propose new features to implement within the existing product to resolve issues from the previous build, from which new tasks are formed and a new iteration begins.

Utilising the Agile framework to manage the project would provide many advantages. The lack of a rigid plan means that time usually occupied by the testing phases of other models can be redistributed within the design phases of each feature iteration. The iterative approach of the Agile framework provides significant flexibility, allowing major changes to be made within an application amidst the development process which cannot be done within the invariable structure of traditional SDLC methods. In a professional development scenario, this framework's strength originates from its ability to deliver a functioning prototype during the development process which can be used to present the progress that an application is making to stakeholders, investors etc. However, for the EPQ, presentable software only needs to be released at the end of the development cycle and so this benefit is redundant for my use-case. The Agile methodology also heavily relies on customer interaction as stated before. Users are consulted within every sprint and the collection of this data

is time-consuming and to repeat the process multiple times is not feasible for a time-constrained project such as the EPQ.

Another methodology being considered was the Big Bang model [11] due to its direct contrast with the Agile model in its simplicity and, thus, minimal time-consumption. It is a process where no specific framework is followed and all possible resources are focused directly on only the development process. Requirements are received and met dynamically, and as a consequence, it possesses the same, if not greater, flexibility of the Agile framework as major changes made to the application do not require an entire revamp of the software.

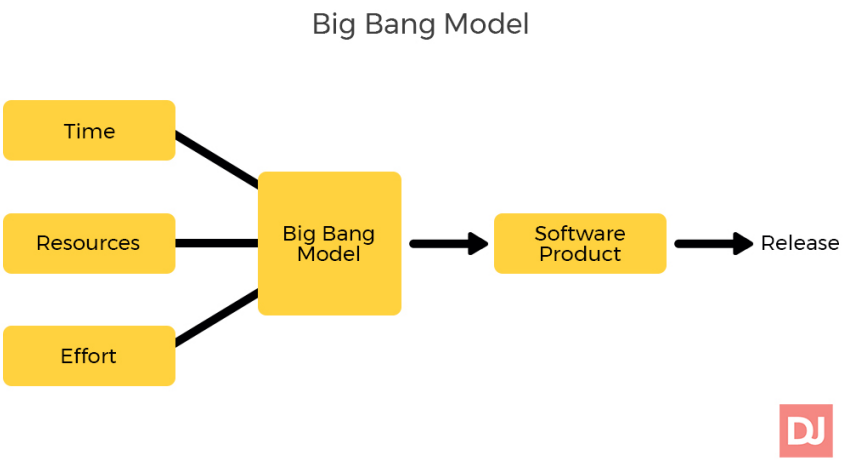


Figure 4: Diagram visually depicting the Big Bang SDLC model [12]

The simplicity of the model amounts to minimised planning with easy management of the project requiring little to no resources. As emphasised before, this leads to a dramatic reduction in time spent as compared to other frameworks but also presents a plethora of implications. This method of project management introduces a large room for error which is made increasingly more probable with larger project duration. Variability in planning and lack of a set structure will provide minimal guidance and can lead to a lack of understanding on what steps to perform next, especially for less experienced developers such as myself. The highly variable nature of model can also lead to a lack of documentation which will lead to inefficiency when creating the write-up of the EPQ.

The last methodology within the shortlist was the Waterfall model [13] which is named as such due to its sequential properties: each phase must be completed before the next phase can begin and there is no overlapping in the phases, executing the software development process in a linear flow. Phases within the development process begin after completion of the previous without overlap. The Waterfall model is the rawest form of the standard SDLC framework as the whole process of software development as can be seen by its division of the development process into separate phases with the outcome of each phase acting as the input for the next.

The Waterfall model begins under the assumption that the feasibility of the project has already been analysed and the initial planning has already concluded and so there are only six steps as opposed to the seven in the standard framework. They are as followed: Requirement Analysis, System Design, Implementation, System Testing, System Deployment and System Maintenance. The purposes of each respective stage closely reflect the functions of their corresponding phases

within standard SDLC. The figure below is an illustration of the Waterfall Model.

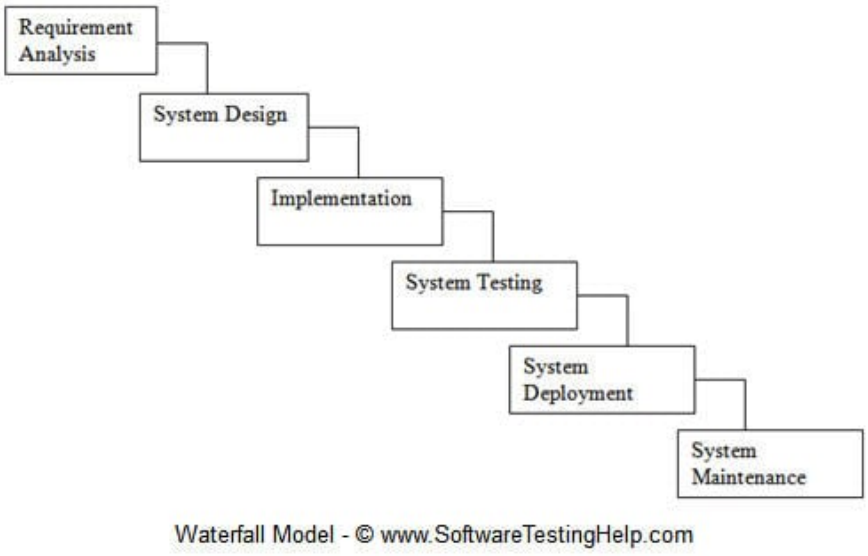


Figure 5: Diagram illustrating the iterative nature of the Waterfall methodology [14]

The Waterfall model performs categorisation on the software development process and thus a schedule can easily be created with set deadlines for each stage of development. Each phase of development proceeds in strict order which results in a project that is easy to manage due to it clearly defined stages and well understood milestones. It is made clear that each phase has its own deliverables leading to the next, giving the model a direct, rigid nature that allows for easy and effective documentation, but also extreme inadaptability. Therefore, to use the Waterfall model, the requirements must be well-defined as the model is unable to accommodate any alterations to ensure the phases are executed sequentially.

With these three models taken under consideration, a decision had to be committed to for the entirety of the project. The Big Bang model [11] seemed to be the most simple to implement and its flexibility would allow complete freedom within the development process to add or remove requirements and features when required. However, the absence of rigidity or any guidance makes the development cycles more error-prone and difficult to document. The Agile framework [9] seemed to be the perfect middle-ground between possessing adaptability, by allowing dynamic development due to its iterative approach, and having a set framework, illustrated by the rigid structure within each sprint. Although, there were issues regarding its suitability for the current project. Due to its reliance on consumer feedback to generate the features and requirements for each sprint, the process proved to be significantly time-consuming and is ill-fitted for time-constrained development. The Waterfall process [12] had contrasted with the two prior models mentioned due to its invariability and inability to accommodate evolving requirements. Consequently, extra effort is needed to ensure that requirements are well-defined. In a real-world scenario, dynamic requirements can pose a major issue when deciding to implement the Waterfall methodology into a project, however, in the EPQ, I will only come across static requirements and would be able to ensure that requirements are well-defined via a thorough requirement analysis process. This set structure also provides its own benefits as it categorises the development life cycle and, in the process, facilitates the creation of concise documentation. The simplicity of the framework ensures that it is easy to follow and manage

but also efficient and for the reasons stated prior, I had decided to commit to using the Waterfall methodology as the process of choice for managing this software development project.

The article is split into chronological sections that reflect the individual phases of the chosen model, verifying that all stages of the production cycle have been thoroughly analysed. As stated before, the methodology assumes that the initial planning has been completed, therefore, the next step to be made was to create a project schedule.

1.2 Planning

1.2.1 Project Schedule

The first step upon beginning the project was to establish an initial timeline. The schedule was translated into a visual context by producing a Gantt chart, as in Figure 6.

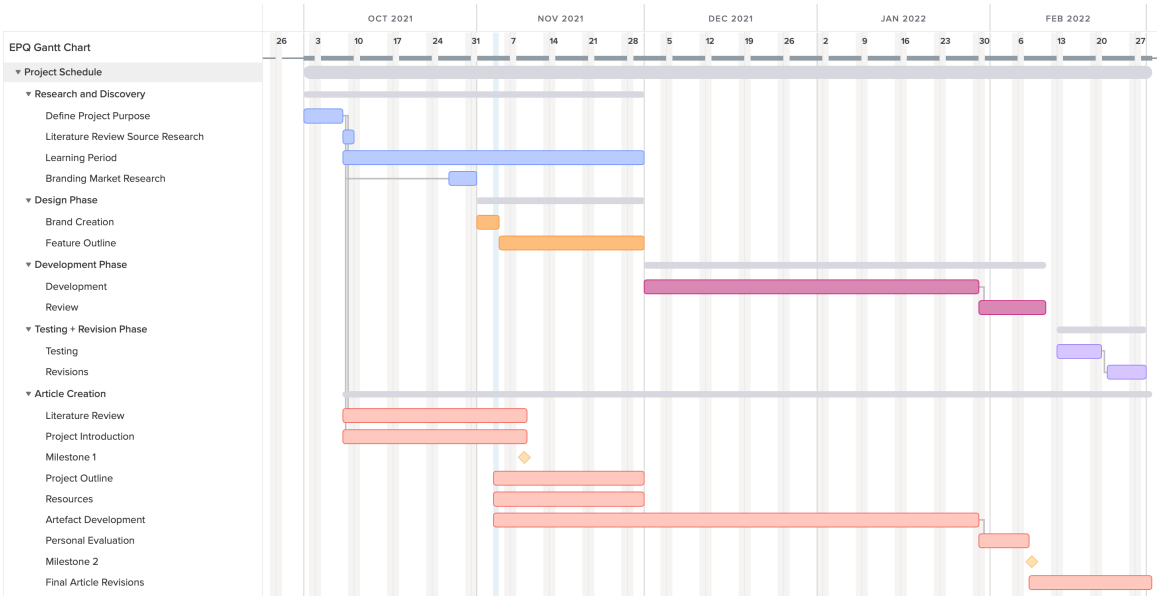


Figure 6: Gantt Chart visualising the schedule of the project

The purpose of specifically using a Gantt chart as the method of schedule visualisation was so that tasks were evenly spread out throughout the project timeline. The chart is capable of effectively portraying the number of concurrent tasks which prevented the creation of a schedule containing an unrealistic amount of workload within a limited timeframe, therefore ensuring the timeline is as accurate as possible. Though it would be inevitable during the earlier stages of the timeline, I attempted to form a chart so that no more than three or four task are done simultaneously, especially during the later stages where other commitments may have more priority.

The chart allowed me to categorise my tasks into the sections which are colour-coded as the following: Research and Discovery (Blue), Design Phase (Orange), Development Phase (Pink), Testing and Revisions phase (Purple) and Article Creation (Red), which can be seen in Figure 6. The figure presents the scheduled time in which each section and task must be completed within using the colour-coded bars.

The Gantt chart also consists of a translucent grey horizontal line adjacent to the section titles which represent the duration of which each sector will take to complete. There are certain lines which anchor off the end of tasks, for example: multiple can be seen branching off the “Define Project Purpose” task in the “Research and Discovery” section. They represent dependencies, indicating that the task connected must be completed before the next task is able to begin. The article creation will occur over then entire duration of the project. Within that section, the two milestones can be seen. They are represented by a rotated orange square, ensuring the article is completed sufficiently within the allotted time ready for feedback.

"Research and Discovery" involves the duration to commit to an artefact idea and performing the background research behind what preliminary techniques and resources are required for such a project. The "Design Phase" incorporates the assigned time to establish an outline for the project which be regularly referred to during the programming done throughout the "Development Phase". Once development has completed, I have allocated time within the "Testing and Revisions Phase" to ensure that all implemented features function as intended.

The Gantt Chart illustrates the linear sequential properties of the waterfall SDLC model being utilised for the management of the project [13], exempting the time allocated to learning the chosen Python web framework which will be done in conjunction with other phases due to time constraints. In a real-case scenario, the developer would already have familiarity with the framework to be implemented. Therefore, this segment would ordinarily be omitted and is not formally a part of the requirement analysis stage - commencing after the planning has concluded and the schedule has been established.

With the initial schedule created, it was then time to make a decision regarding which resources to utilise and implement within my project.

2 Requirement Analysis

As stated prior, the requirement analysis phase within the Waterfall model is used to define the objectives of the software to meet the requirements of the end-user and what the end product is intended to achieve. In order to do so, I had decided to conduct a literature review. Examining previously published works would aid in the decision-making regarding which resources to utilise - such as the web framework. It would also facilitate the analysis of existing systems which can feed in fundamental requirements for creating a social networking application.

2.1 Literature Review

This section aims to review five sources, all of which originate from a variety of types to diminish bias. This chapter revolves around three aspects that are integral to laying the foundations of research to create my own web application: Framework Selection, Django Learning Process and Existing Django Applications. Viewing existing applications that utilise the Django framework, in conjunction with the analysis of the provided official documentation, will allow to widen the scope of my knowledge regarding the fundamental features, tools, and libraries required when undertaking a project of this scale.

2.1.1 Framework Selection

The decision to commit to a single Python web framework for the duration of the project formed a vital segment of the research period which led me to come across the graduate thesis: 'Comparative Study on Python Web Frameworks: Flask and Django' authored by Devndra Ghimire [15] which aims to provide a detailed comparison of both frameworks, highlighting features and limitations by creating a social media and eCommerce application with both Flask and Django respectively.

The source is reportedly made via a 'novices' perspective' which highlights that the target audience of the thesis are entry-level programmers who intend to begin to program using a Python web framework for the first time which complemented my prior lack of knowledge. The reliability of the source is accentuated by its nature as a graduate thesis as they are often closely supervised by a committee of scholars. Discretion is often advised when referencing a bachelor's thesis as it is still considered a student's work and is not peer-reviewed. However, all claims made by the paper seem to be supported by valid sources. The usefulness of the paper is primarily to provide complementary information before a commitment to a framework is made however, upon further analysis, the source had formed a larger segment of my preliminary research than previously anticipated. As mentioned prior, the author had coincidentally created a networking application, similar to the intended artefact, and, thus, I intend to emulate the certain elements of the writer's programming process such as the use of the cloud platform Heroku to deploy the program, due to the emphasis brought to its extensibility and "seamless, easy to use deployment process" and logging.

The source informs the reader of Django's ease-of-use due to its extensive features and support for libraries and another reason being its scalability, making the framework suitable for a large-scale applications unlike its Flask alternative which was stated to be more appropriate for "smaller-sized applications" and so I had decided to commit to utilise the Django framework as opposed to Flask as I deemed it more appropriate for the intended scale of my project.

After selecting the web framework, the next step of research involved getting familiar with the features and syntax of the framework myself.

2.1.2 Django Learning Process

A signification portion of the project timeline must be dedicated towards learning the syntax required to utilise the framework to its fullest extent. I decided to refer to the official documentation [16] found readily available on the official Django website which offers concise guidance on preparing a development environment for use of the framework and provides an overview on the syntax required to enable a user to utilise the plethora of features integrated within. The intended audience are Python developers who intend to make database-driven web applications as it is made clear that a basic level of Python proficiency is assumed.

The usefulness of the article, compared to alternative sources, originates from its nature as an open-source project. The public Django GitHub repository [17] has over 2,000 contributors who are able to constantly update the documentation which helps eliminate any errors/biases that may appear and to keep the source relevant as the framework updates and evolves. Limitations of the document lie within its primary strength. In its pursuit to present all features of the framework, the documentation is lengthy as the PDF version of the documentation for the latest release as of the creation of this article, Django 3.2.9, contains 2,086 pages. Due to the time-limited nature of this project, this source will not be fully explored. However, it will continue to form a significant portion of the basis of my research as I will only refer to portions of the article that are relevant to my project. Its format as a book means that it is unable to provide aid if, when following the article, an unexpected error occurs and no guidance is provided for common errors that novice programmers, such as myself, may face. However, this is easily rectified using programming troubleshooting sites such as StackOverflow.

StackOverflow [18] is a question and answer platform whose intended audiences is stated to be for "professional and enthusiast programmers". The site is popular amongst the developer sphere with 12 million plus visits. [19] With a user base of over 16 million, it is more than likely that any issues I come across throughout the duration of the project timeline will have already been posted and replied to. In the unlikely chance that this is not the case, I am able to submit my own question and add to the site's already large archive of more than 22 million posts.

The reliability of this source stems from the active participation of its community as answers to questions can be voted on to indicate the best solution. The limiting factor of StackOverflow is the possibility of being provided with incorrect or unrelated solutions by other members. This is easily avoided as a reputation system is implemented within the site to indicate how trustworthy the user providing the information is. Following advice from those who have amassed a large amount of reputations points and votes on their answer will ensure the validity of any solutions that are implemented within my own programme.

Utilising both StackOverflow and the official Django documentation had helped develop my Django syntax proficiency to a level where I was comfortable enough to view other Django projects to use as reference for my own.

2.1.3 Existing Projects

Before attempting to undertake a highly practical project, an effort must be made to view similar projects, in my case, social media applications, to take inspiration so that an outline can be drawn out on what features to implement.

Instagram [20] is a photo and video sharing social media service with over 1.3 billion active users, making it the fourth most used social networking application at the time of writing. [21] Instagram was chosen as reference rather than other more popular or as feature rich applications as its web client is also programmed with Django [22] and, therefore, serves a representation of the limits of what the Django framework is capable of.

Features of Instagram that I wish to incorporate within my own project include but are not limited to:

- User authentication system
- Social feed
- Comments section
- Follower and rating system
- Direct messaging capabilities

A limitation of using Instagram as a reference point for my own project is that the level of Django proficiency I had developed thus far was still too limited to comprehend applications developed at the commercial scale. Therefore, the main aim became to look solely at what features to implement to make my platform as user-friendly as possible rather than how to implement these features. To understand how to do the latter, I had the look for other projects with less complex source material.

The source used for reference had to be appropriate for the level of knowledge that I had obtained through the project thus far which led me to the online course provided by freeCodeCamp.org named 'Python Backend Web Development Course (with Django)' [23].

The associated YouTube channel has over 4.3 million subscribers and over 240 million total views [24]. Though social media analytics do not necessarily infer a factual and reliable source of information, supplementing this with previous knowledge of freeCodeCamp's notoriety within the web development field due to its free 3,000 hour circular with over four thousand contributors on its public GitHub repository [25] and its nature as a donor-supported non-profit organisation [26] makes the reliability of the source seems to be relatively high.

The primary limitation of the source originated from its target audience. The course was aimed at developers who were new to the Python language and so a large proportion of the video was dedicated to learning basic Python functionalities which I largely had to skip.

However, the source was extremely beneficial as it provided guidance for three separate Django projects: a blog, weather app and a real-time chat application. As previously mentioned, direct messaging is major feature that I aspire to implement within my program and social feeds within networking applications are often displayed in a blog-like format. Therefore, these tutorials will regularly referred to during the development period of the project. The course also helped reinforce my knowledge of Django syntax as the video provided an overview the capabilities of the framework.

2.1.4 Summary

The sources critiqued in this literature review helped establish the foundation upon which my artefact will be developed. It has allowed me to make an informed decision regarding the choice of web framework and provided information about the extent of my capabilities with front-end programming and Django, taking into consideration the limited time frame of the project, which has facilitated the creation of an accurate outline of the application.

2.2 Resources

Throughout the development process, a plethora of frameworks, packages and applications will be utilised. As a consequence of the imminent deadline of the project, extra deliberation had be made to list resources which help reduce production time of the artefact.

Creating a list of resources also helps to ensure that each resource is readily available and free of charge as I had intended to complete the project without any expenditure. Certain libraries and frameworks have both free and premium variants, with the free version having limited functionality compared to its premium counterpart. Creating a list of resources allowed me to verify which functionalities were and were not available to me as a free user before commencing the project.

The resources listed below are specific to the development phase of the project therefore packages and applications intended for other purposes such as hosting will not be mentioned within this section of the report.

2.2.1 Libraries and Frameworks

As mentioned extensively throughout the report thus far, I intended to utilise the Python web framework Django. The desire to utilise a Python web framework within this project is rooted in my previous experience in using the object-oriented language whilst obtaining my AS-Level and GCSE Computer Science qualifications.

Another framework I wished to implement within the application is the CSS framework Bootstrap. Bootstrap has a collection of ready-made templates and components consisting of HTML, CSS and JavaScript code and, thus, incorporating Bootstrap within my code will result in a more responsive website. Due to the time-constrained nature of this project, one of the most important affects of utilising Bootstrap is the reduction in time spent developing CSS code which I could now redistribute to other functionalities. As it is free and open-source, I am able to utilise its full feature-set without any concern.

I also wanted to incorporate the FontAwesome library within my artefact. This resource contains a large catalogue of high quality icons, thus saving the time needed and waiving the requirement to create my own icon graphics. However, I am only able to use the free variant of FontAwesome which had restricted me to only utilising icons from FontAwesome version 5.15.4 and below. Thus, extra care had to be made to ensure that icons from later versions of FontAwesome were not implemented within the application.

2.2.2 Django Packages and Applications

There are also a variety of Django packages and applications I intend to implement. These consist of:

- django-allauth: an integrated set of Django applications which addresses user authentication, registration and account management
- django-crispyforms: a Django application which allows developers to adjust the properties of input forms
- crispy-bootstrap5: updates the forms rendered by django-crispyforms to ensure it utilises the latest version of Bootstrap (version 5)

The use of these Django applications will result in the reduction of time required to develop and implement a robust and aesthetically pleasing user authentication and account management system within my artefact.

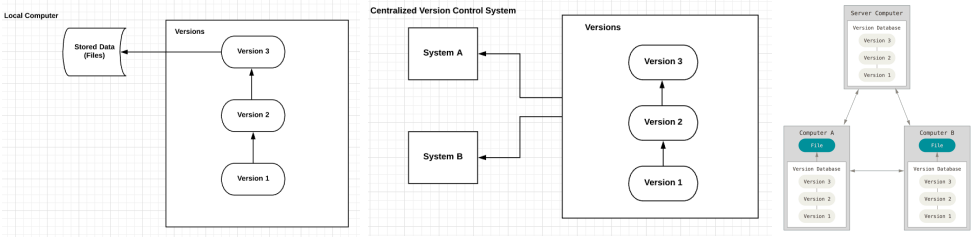
2.2.3 Version Control System (VCS)

During the development of the artefact, I may come across a bug within my source code which is made increasingly likely due to my lack of previous experience with Django. During such an event, I may want to be able to be able recall a past version of my software so that I am able to return to a state where there are no longer any errors which is what a version control system (VCS) sets out to do [27]. It can also act as back-up in the unlikely event of data corruption and/or loss of data.

For this project, I have chosen the Distributed Version Control System (DVCS) Git rather than other alternative VCS types such as Local or Centralised Version Control Systems (LVCS/CVCS). An LVCS, as the name suggests, keeps all versions of the data locally in a version database [28]. This is inconvenient for most developers as it does not allow for collaboration between developers. However, due to the independent nature of this project, the main concern that arises from the usage of VCS is that I am unable to access the database from any alternative devices.

A CVCS would solve this issue as they work by utilising a single server that contains all versions of the files and then any number of clients are able to access the server, allowing me to retrieve files from any devices [29]. Nevertheless, in a long-term project like this, file security must remain a very high priority. The introduction of a centralised server presents a single point of failure and, thus, such a system is not reliable enough for my use case.

DVCSs circumvent both these issues by using a server, similar to CVCS, but instead forcing the client to mirror the entire version history rather than the latest snapshot, unlike a CVCS, see Figure 6 (overleaf). This essentially means each client retains a full backup of the data capable restoring the server thus protecting the files from any potential future issues.



(a) LVCS [30] (b) CVCS [30] (c) DVCS [27]

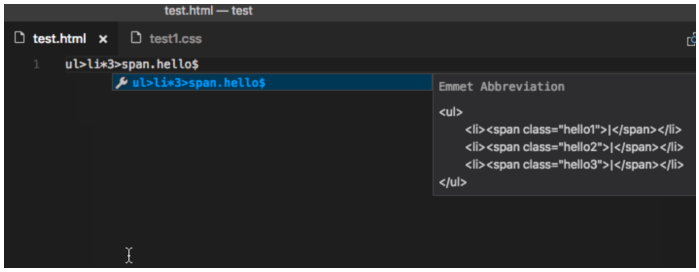
Figure 7: Diagrams displaying the functionalities of various version control systems

The decision to use Git was due my previous familiarity with the system as well as its integration with GitHub: a Git repository internet hosting service. This allows myself to utilise the relatively reliable GitHub servers for version storage but also provides the convenience to be able to access the full version history of code from any device and at any time necessary.

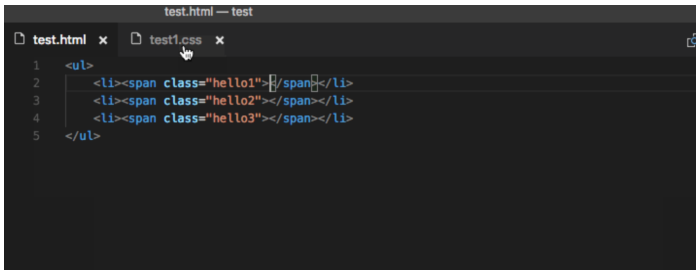
2.2.4 Integrated Development Environment (IDE)

To ensure that workflow is as fluid as possible, it is integral to the project to ensure that the different resources utilised are able to complement and synergise with each other in a cohesive manner. For this reason, the choice to use Git as the system of choice for version control facilitated the decision to use the Visual Studio Code IDE for my project. VS Code, made by Microsoft Corporation and also holding and parent company of GitHub [31], has integrated Git support as well as a natively integrated command line interface, unlike alternative IDEs such as Atom and Sublime [32], allowing

me to commit files to Git, push files to GitHub and access my code repository without leaving the editor.



(a) Emmett Shortcut



(b) Resultant Code

Figure 8: Demonstration of Emmet's shortcut functionality [34]

VS Code has variety integrated features such as IntelliSense, an code auto-completion tool which also provides parameter information and member lists [33]. A more relevant feature for my specific use case is the built-in support for Emmet snippets and expansion [34]. Emmet is an add-on that allows a user to input a shortcut capable of expanding to full pieces of HTML or CSS code, as in Figure 8 above, allowing for more efficient front-end development.

Further functionality can be implemented within the editor, in addition to the already large array of features available out-of-the-box. This is done via the sizeable extension marketplace for VS Code containing approximately 9,000 extensions [32] allowing for abundant customisability.

2.2.5 Graphics Software

To ensure that I was not infringing any copyright or trademark regulations, I had to create my own logo for the branding of my application. I did not intend to spend a considerable amount of time in this sector of the project as the development phase of the project was of higher priority and required a large duration of time to complete. Therefore, I had to find an application that was both time-efficient but create a brand that was to an adequate standard and fit the intended aesthetic. During my search, I found NameCheap, a domain name registrar. Its subsidiary, Visual, provides free brand design by allowing you to enter the name of your organisation and to select from an array of fonts and icons that fit your planned theme. The site then randomises the selected fonts, icons, and colours to provide you a large collection of options to choose your logo from.

To use the branding in my application, I had to be able to create graphics using a photo-editing software. I intend to complete this project with little to no expenditure and so, rather than using

expensive feature-heavy graphics design applications such as Adobe Photoshop, I would have to instead utilise a simpler but free-to-use software. In an attempt to achieve a balancing point between functionality and price, I came across the free site Pixlr and, in specific, Pixlr E, its more advanced, feature-rich variant. I chose Pixlr E due to the convenience of its nature as a web application, allowing me to upload and edit images straight from the browser rather than requiring me to install software locally to my workstation such as other alternatives such as GIMP.

2.3 Requirements Checklist

As mentioned prior, due to the invariability of the Waterfall SDLC model, the requirements of a project utilising the methodology must be well-defined. Thus, I elected to create a checklist of requirements to summarise the requirements analysis process in a easily referable format. The items of within this checklist will be referred back to within the 'Evaluation' section of the report to determine the extent of which the final product met its initial requirements. The requirements for the artefact is as follows:

- The application must have a robust, secure user authentication system
- The application must have concise and easy-to-use posting and commenting functionality
- The application must allow users to edit or delete their posts
- The application must allow users to like or dislike posts
- The application must allow users to have and customise their own profile profile pages
- The application must allow users to follow each other
- The application must allow users to search for other users
- The application must allow users to directly message other users
- The application must have a user-friendly interface
- The application must be remotely accessible via a domain

3 System Design

With the requirements of the artefact now established, it was time to create a blueprint outlining the general aesthetic ideas for the product which was to be regularly refereed to during the programming process

3.1 Blueprints

Using the requirement checklist mentioned prior, I was able to create a list of the fundamental pages that were going to be implemented within the site. I subsequently categorised these pages depending on the similarity of their design format. The eight resultant groups were: User Authentication pages, Comment Pages, Delete pages, Edit pages, Profile pages, Search and Inbox pages, Messaging pages and Social Feed pages. In doing so, I was able to limit the amount of design outlines required which can be seen in the figure below.

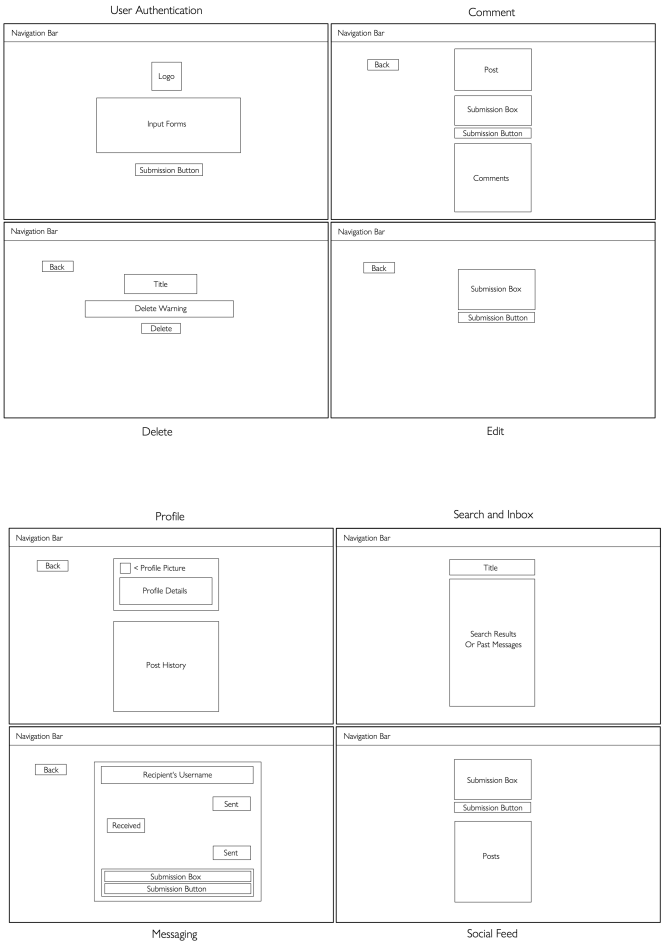


Figure 9: Page design blueprints

The majority of the content within the blueprints can be seen centered with the edges of the

pages often blank. This is to emulate existing social networking sites such as Instagram and Twitter and to make the application more user-friendly for mobile devices.

I had intended for the submission boxes, which can be seen within the social feed, comment and messaging pages, to enable users to input text. This is due to limitations introduced as a consequence of utilising the hosting platform Heroku to deploy the application.

Heroku provides a free cloud service to enabling users to host non-commercial applications. The free variant of Heroku also, however, has a limited feature-set and an ephemeral file storage system [35]. This means that files uploaded to the Heroku server will not persist and will be wiped after the application is restarted which occurs every 24 hours or when a new version of the application is deployed to Heroku.

Consequently, the content of the website would primarily be text-based, similar to social networking apps such as Twitter, rather than image based to limit the affect of the ephemeral file storage on the application. I had to still plan for the users to be able to upload their own profile pictures in order to showcase the capacity of the website to be able to receive, store and display external files and highlight the potential of the site to move to image-based content if external cloud storage were to be incorporated within the application.

3.2 Branding

Before the development of the application, it was necessary to create the branding of the application so that it can be directly implemented during the programming process.

3.2.1 Market Research

I decided to perform exploratory research on existing social networking platforms, emulating common characteristics to form a realistic brand for my own site. I began my market research by viewing the logos of the most popular social media sites, a recurring feature of which was a minimalistic icon ahead of a solid colour.

Figure 10: Existing social media logos [36][37][38]

As visible in figure 8, Facebook has a solid blue background behind a white “F”, SnapChat has their iconic ghost ahead of a bright yellow backdrop and WhatsApp has their green speech bubble overlayed by white phone symbol.

During this period, I had also noticed that a considerable proportion of the logos I had come across had a blue colour scheme. Examples include Twitter, Facebook, LinkedIn, Tumblr, Vimeo etc. Upon further research, it was made clear that this symmetry across rival platforms had to do with notion of colour psychology [39], the perception that colours have psychological and emotional connotations attached to them. The colour psychology of blue promotes peace, calm and tranquillity [40], endorsing communication and interaction [39] and so I made the decision to implement the blue colour scheme within my brand.

From previous knowledge, I am aware that Python-based software, such as the Python Distribution “Anaconda”, often use their title as a method to pay homage to the language used to create it.

For similar reasons, I decided to name my application “Ophidia”, the most recent common ancestor of modern snakes and chose to implement a snake icon within my logo.

3.2.2 Graphics Design

Bringing these ideas together would allow me to create industry-standard branding, and to do so, as mentioned within the “Resources” chapter, I used Visual’s logo creation tool. The resultant logo is visible in the figure shown below.

Figure 11: Logo to be implemented within the web application

4 Implementation

The prerequisite planning performed in the prior stages of the Waterfall model will now be utilised to create the artefact for this project.

4.1 Programming

The programming phase began by creating a public GitHub repository for version control. The repository was made public to allow users to clone the repository to use the application in the event that an issue arises when during deployment, preventing the testers from accessing the site remotely. To provide aid for local testing, I added a README.md file to the repository which is visible in the figure below.

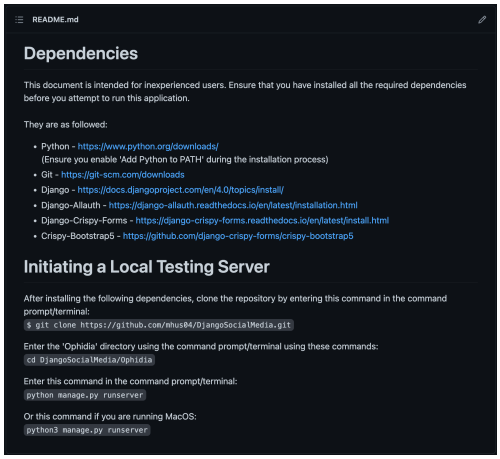


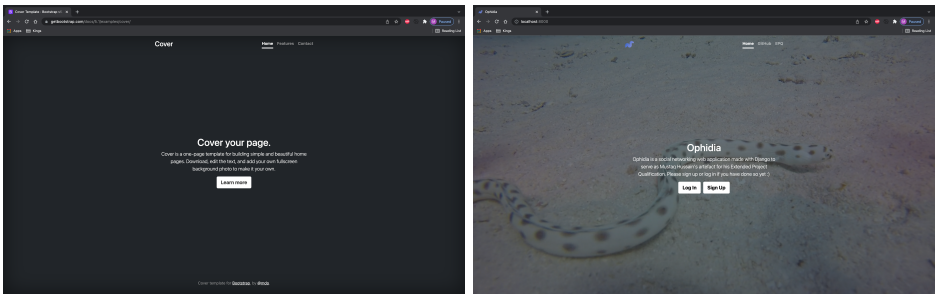
Figure 12: README.md file visible upon opening the GitHub repository

The README.md file uses the Markdown language which is similar in syntax to HTML, a language that I have previous experience in with and will also used extensively throughout this project. The file outlines the perquisites to install enable to application to run alongside with instructions to clone the repository and deploy the application on to a local testing server.

4.1.1 Homepage

Once a GitHub repository was created, it was now time to commence the programming of the application itself. As mentioned in the 'Resources' chapter of this report, I utilised the Bootstrap CSS framework within this application and this enabled me to choose from a variety of templates available on their main website. This allowed me to reduce the development time required to develop the homepage, which was critical due to the time-constrained nature of the project, while also maintaining a high aesthetic standard.

I elected to use the 'Cover' template which you can see in the figure below alongside the final homepage with all the implemented edits.



(a) Cover bootstrap template (b) Final homepage design

Figure 13: Ophidia homepage: before and after

The changes made to the template include:

- Adding the Ophidia logo as a favicon
- Changing the existing template text
- Changing the items within the existing navigation bar
- Editing and adding to the existing button with the intent to link the buttons to the URL path of the user authentication system to allow users to sign up/log in once such a system is created
- Adding an auto-playing snake video in the background to match the site-wide serpentine aesthetic

4.1.2 User Authentication

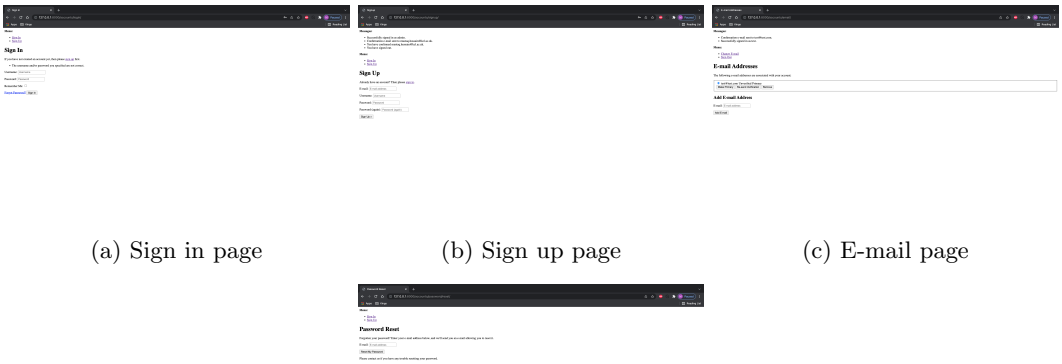
Once a homepage was developed, the next step was to create a user authentication system, enabling users to create and log in to their respective Ophidia accounts. Developing a safe-to-use and reliable user authentication system is notoriously difficult and time-consuming to do [41] and, therefore, I decided to implement the Django application 'django-allauth' within the Ophidia source code. Within the django-allauth public GitHub repository are templates which I intended to utilise within the Ophidia site which are visible within Figure 14.

I also decided to utilise the 'django-crispyforms' Django application, which allows developers to adjust properties of the forms used within their applications on the back-end without having to re-write them in the template. Utilising django-crispyforms allowed me to improve the visuals of the templates provided by django-allauth using Bootstrap so to ensure there is continuity in the styles used throughout the application.

There are many examples of Bootstrap implementation in the end product of the user authentication system, one of which is the change in style of buttons.

As mentioned within the market research section of the report, the Ophidia application was to have a blue colour scheme. The Bootstrap documentation shows four classes of blue buttons available to choose from. The button classes 'btn-primary' and 'btn-info' alongside their outline variants, 'btn-outline-primary' and 'btn-outline-info'. The primary and info buttons can be seen in Figure 15.

The outline variant of the buttons are transparent unless the user interacts with the button by hovering their cursor over it. This functionality can be seen in Figure 16.



(a) Sign in page (b) Sign up page (c) E-mail page

(d) Password reset page

Figure 14: Screenshots of django-allauth template

(a) Primary button (b) Info button

Figure 15: Bootstrap primary and info button colours

(a) No cursor hover (b) Cursor hover

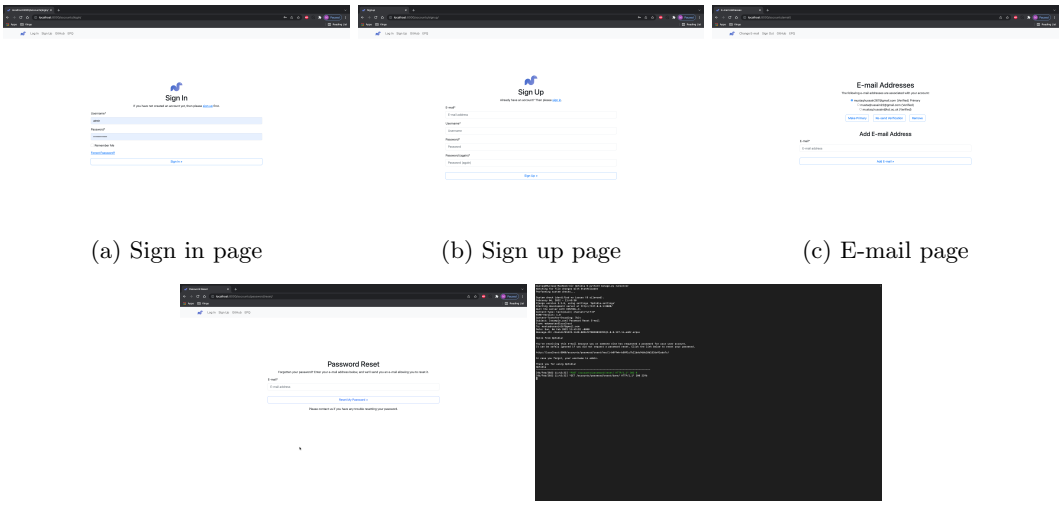
Figure 16: Bootstrap outline button functionality illustrated using a Bootstrap info outline button

I decided to implement an outline button and eliminated two buttons from the shortlist as I wanted the user to experience a sense of interactivity when using the Ophidia website. I decided to choose 'btn-outline-primary' out of the remaining two options as the dark blue of the primary button contrasted with the bright white background, unlike the bright cyan of the info button. This was an especially important factor to consider when referring to outline buttons as the background is visible when the button is not interacted with. Using two bright colours in an outline button would likely overwhelm the user and lead to an uncomfortable experience.

A summary of all the changes made to the django-allauth template is as follows:

- Centering text and forms
- Implementing a navigation bar to allow easy switch-over from sign in to sign up
- Implementing Ophidia branding on sign in and sign up pages
- Updating button aesthetics
- Programming the Ophidia site to send e-mails to the command line interface for efficient access during testing

The end result is visible in the figure below:



(a) Sign in page (b) Sign up page (c) E-mail page

(d) Password reset page (e) CLI password reset email

Figure 17: Final Ophidia user authentication system

4.1.3 Social Feed and Posting Capabilities

The next fundamental feature that was required to be implemented within the Ophidia website was the social feed, allowing users to post and access the posts that others have submitted.

Unlike the other features incorporated within the application thus far, the social feed was to be developed without the use of a template.

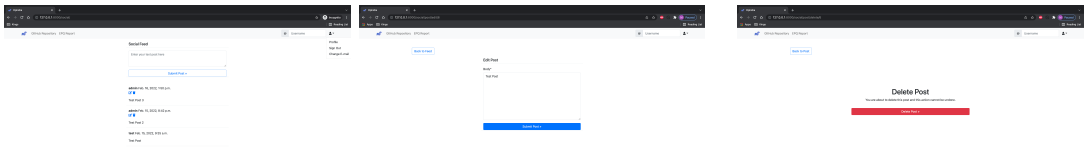
The following steps were taken to create the social feed for the Ophidia application and to allow for users to create, edit and remove posts:

- All content was placed in a Bootstrap container class of reduced width to emulate the experience of a mobile application
- The navigation bar developed when creating the user authentication system was re-implemented in the social feed and throughout the rest of the Ophidia application after the user has signed

in with the items within the navigation bar changed to direct users to the Ophidia GitHub repository and EPQ report

- A FontAwesome 'fas fa-user' icon alongside a Bootstrap drop-down menu was implemented to the navigation bar which when clicked provides users with the option to: sign out, change their email or view their profile (yet to be implemented thus far in the development process)
- A user profile search bar was implemented to the navigation bar with no functionality at that point in the development process as the user profile system had not been created yet
- A text box was created for users to input their content alongside a button programmed to submit their post using django-crispyforms and Bootstrap
- Previous posts programmed to be displayed below the submission form with the username of the user who submitted the post and the time of submission, sorted with the most recent appearing first
- Under the condition that the user viewing a specific post is its creator, FontAwesome 'fas fa-edit' and 'fas fa-trash' icons with the same HTML hex colour code of the primary Bootstrap buttons appear below the post which redirect users to post edit and post removal pages
- Post edit and post removal pages created with a uniform aesthetic, using the same button and text form styles as the previously created user authentication system
- Implemented validation checks to ensure that users must be logged in to access the social feed and comment URLs
- Implemented further validation checks to ensure that only the user which created a specific post can access its edit URL, otherwise, the user returned a HTTP 403 Forbidden error and is refused access
- Users were now redirected to the social feed upon signing into Ophidia

The resultant pages can be viewed in the figure below



(a) Feed and drop-down menu (b) Post edit page (c) Post removal page

Figure 18: Ophidia social feed

As visible within the post remove page sub-figure, the post deletion page utilises a different Bootstrap button class with the colour red: 'btn-outline-danger'. This is done intentionally as it strongly contrasts with the blue 'btn-outline-primary' buttons used widely throughout the Ophidia website. This emphasises that the deletion of a post is irreversible and reduces the chance that a user may delete a post unintentionally.

4.1.4 Comment Section

As the posting functionality was already implemented within the Ophidia website, it made the development of a commenting system much easier as the majority of the source code allowing for posting functionality could be re-utilised to allow for commenting.

To allow users to comment:

- For all users, a FontAwesome 'fas fa-comment' icon with the same HTML hex colour code of the primary Bootstrap buttons appear below the post which redirects the user to a comment page
- Comment page created with for each post with a text form and submission button similar to the social feed created prior and 'Back to Feed' button allowing users to return to the social feed
- Previous comments programmed to be displayed below the submission form with the username of the user who submitted the comment and the time of submission, sorted with the most recent appearing first
- For users that created a comment, a FontAwesome 'fas fa-trash' icon appears below their comment which redirects the user to a comment deletion page
- Comment deletion page created to verify that the user wishes to delete their comment
- Implemented validation checks to ensure that only logged in users can view comment pages

The figure below illustrates the end product as a result of these changes and additions.

(a) Social feed with comments (b) Comments page (c) Comment removal page

Figure 19: Ophidia commenting system

4.1.5 User Profiles

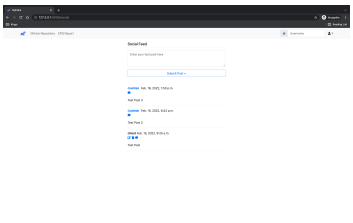
The profile button within the navigation bar drop-down menu had yet to have any functionality up until this point and, thus, the next step was to create a user profile system for the Ophidia application.

The following steps were taken to allow for users to have their own individual profile pages with the ability to edit their pages:

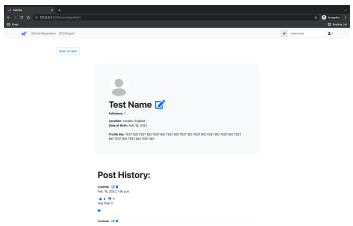
- A profile update page was created to allow users to input their name, date of birth, location, profile bio and add a profile picture if they wish to do so

- Users who have signed up to Ophidia for the first time were now redirected to the profile update page until they had at least added their name to their profile
- A profile page was created for each user which displayed their profile picture, name, date of birth, location, profile bio and post history
- The 'Profile' button within the navigation bar drop-down menu now redirected to the user's profile page
- Added a FontAwesome 'fas fa-edit' icon for users who viewed their own profile which would redirect the user to the update profile page, allowing them to edit their profile
- Edited the social feed so that clicking on a post author's name, if it is not the user themselves, redirects the user to the author's profile page
- Changed the colour of the text for the usernames other than the current user to indicate that clicking on the text will redirect you to the user's profile page

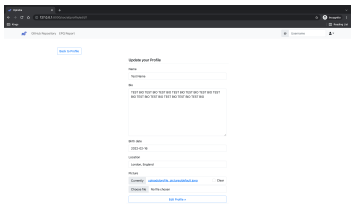
Examples of a profile page and profile edit page alongside the page which new users are redirected to upon creating an Ophidia account are visible in the figure below.



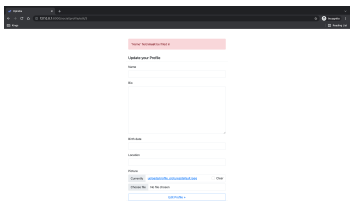
(a) Updated social feed



(b) Profile page



(c) Profile edit page



(d) New user profile edit page

Figure 20: Ophidia user profile functionality

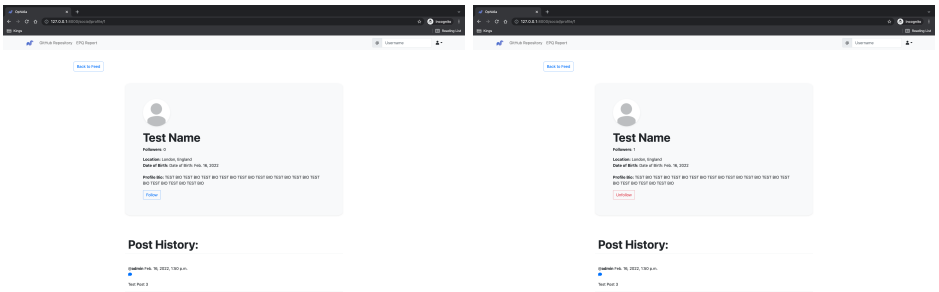
4.1.6 Follower System

One of the most integral parts of any social networking platform is the ability to follow users. To implement such a system with the Ophidia application, I had done the following:

- Created a Bootstrap primary outline 'Follow' button which appears under a profile page if the user viewing the profile is not the owner of the profile and is not currently following the user

- Created a Bootstrap danger outline 'Unfollow' button which appears under a profile page if the user viewing the profile is not the owner of the profile and is currently following the user
- Implemented a follower counter which appears in the user's profile page below the user's name

The figure below intends to demonstrate the follower functionality



(a) Before following (b) After following

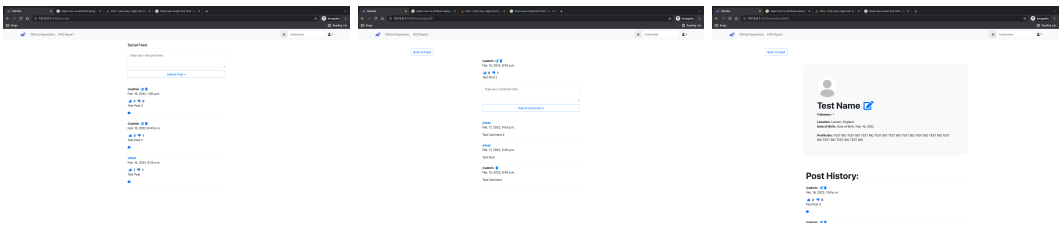
Figure 21: Ophidia following functionality

4.1.7 Rating Capabilities

Often on social media platforms, there is a rating system for posts, providing a quick insight to users about the quality of a post. To add a rating system to Ophidia, the following steps had to be taken:

- Added 'fas fa-thumbs-up' and 'fas fa-thumbs-down' FontAwesome icons next to posts which contributed to a like and dislike counter respectively
- Like and dislike counters were displayed next to their respective icons
- The maximum total tally of icons that a post could possibly have rose to five after this addition to the source code. To reduce crowding, the positioning of content and icons were shifted

The result of these changes can be seen in the figure below



(a) Updated social feed (b) Updated comment section (c) Updated profile page

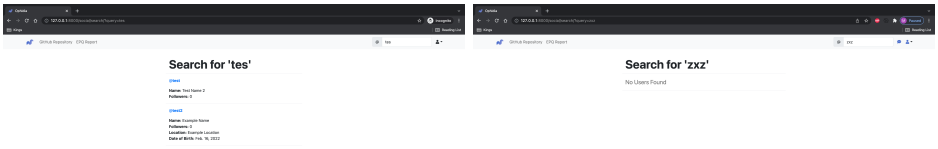
Figure 22: Ophidia rating system

4.1.8 Profile Searching System

There remained one functionality within the navigation bar which was left incomplete: the User Profile search bar. To develop a searching system, the following changes were made:

- Searches from the navigation bar redirect the user to a search query page for the specific input provided by the user
- Created a search query page which lists all possible users, their name and number of followers. It also shows their location and date of birth if it is given in their profile
- Incorrect searches showed users a 'No users found message'

The figure below attempts to present the affect of these changes



(a) Partial search (b) Incorrect search

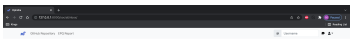
Figure 23: Ophidia user profile searching system

4.1.9 Direct Messaging Capabilities

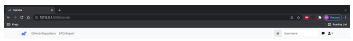
Within the literature review, I had created a list of features within Instagram, an existing Django social media application, which I thought were fundamental to any networking program. At this moment of the development process, there remained one feature yet to be emulated: the implementation of direct messaging functionality. To do so, I had:

- Added the FontAwesome icon 'fa fa-commenting' to the existing navigation bar which redirected users to their own respective inbox pages
- Updated the comment icon for posts 'fas fa-comment' to 'fas fa-comments' to reduce the similarity between the comments icon and the new direct messaging icon
- Created an inbox page within which users could view their existing message threads with users
- Added a button to the inbox page which redirected users to a page which allowed them to create messages threads with users
- Created a message thread creation page
- Created a message thread page which allowed users to send and receive messages
- Added the functionality to change message thread name, allowing for additional customisability

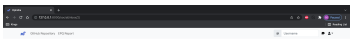
The changes to the application due to these modifications to the Ophidia source code can be seen in the figure below.



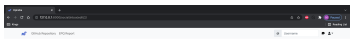
(a) Inbox page



(b) Updated social feed



(c) Message thread page



(d) Message thread name edit page

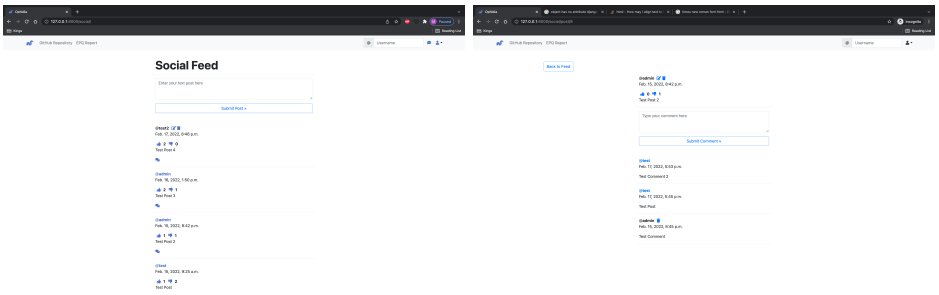
Figure 24: Screenshots presenting Ophidia's direct messaging functionality

4.1.10 Final Aesthetic Changes

Before commencing the testing phase of the project, there were a couple aesthetic changes that were made to the application. They were as follows:

- Changed the colour of FontAwesome icons from the Bootstrap primary button HTML hex code '#296efd' to the shade of blue used in the Ophidia logo '#526ccc' to add variety in colours used throughout the application
- Changed the title styles used in the social feed and inbox to match those used in the profile, search and message thread pages
- Slightly increased the width of the container for social feeds, comment sections and search pages from 'col-md-5' to 'col-md-6' to allow more space for content and to reduce crowding whilst still maintaining the intended mobile application aesthetic

The result of these changes can be seen in the figure below:



(a) Updated icon colours and title (b) Updated container width

Figure 25: Final aesthetic changes made to the Ophidia application before testing

5 System Testing

Following the conclusion of the programming phase, the resultant code had to be tested by examining and assessing the implemented features to verify that they function as intended and meet the outlined requirements established at the start of the project.

To ensure that all aspects of the application are tested, this chapter of the report will be split into the same subsections that were used during the programming phase.

For each individual system, it was important to create a checklist of all the features and desired outcomes for each respective system as to verify that I had thoroughly reviewed all aspects of the system.

5.1 Homepage Feature Test

The testing checklist for the user authentication system was as follows:

- All buttons work as intended
- Items within the homepage navigation bar successfully redirected users to the respective correct pages
- Background video plays automatically

Each testing run consists of the manual verification of each feature in the checklist. The testing run was successful at first attempt which indicated that no further changes had to be made to the source code of the Ophidia homepage.

5.2 User Authentication System Feature Test

The testing checklist for the user authentication system was as follows:

- Users are able to create an Ophidia account
- Users must require a strong password to create an Ophidia account
- Users are able to verify their Ophidia accounts via e-mail
- Users are able to log in to their Ophidia accounts

- Users with incorrect account information are unable to sign in
- Users are able to request for a link to change their password via e-mail
- Users are able to add multiple e-mails to their Ophidia accounts
- Users are able to remove an e-mail linked to their Ophidia accounts
- Users are able to choose a primary e-mail to use for communication
- Users are able to sign out
- Users are redirected to the social feed once logged in

The first testing run was not successful as the functionality of adding alternative emails to an Ophidia account was inhibited due to an unknown reason. This resulted in the ninth item within the checklist being unable to pass the test run and the tenth item could not be tested as the feature could only be accessed once there was more than one email associated with an account.

Before a second testing run could begin, I had to debug the issues highlighted by the first run. I began by assessing the differences in code between the original django-allauth repository and the source code for my application as the changes I made to the code had prevented users from adding alternative emails. In doing so, I was able to understand the origin of this error. While improving the aesthetics of buttons throughout the Ophidia site using Bootstrap, I had accidentally forgotten to add back the 'action_add' element to the button tag of the 'Add email' button while making those changes which had meant that the button was now not linked to any action. Reintroducing this element back into the code had resolved the issue. This change can be seen in the figure below.

```
<div class="row justify-content-center">
  <div class="col-md-6 col-sm-12">
    <form method="post" action="{% url 'account_email' %}" class="add_email">
      {% csrf_token %}
      {{ form|crispy }}
      <button class="btn btn-primary btn-block" name="action_add" type="submit">{% trans "Add E-mail" %}</button>
  </form>
  </div>
</div>
```

(a) Original django-allauth source code [42]

```
        <form method="post" action="{% url 'account_email' %}" class="add_email">
            {% csrf_token %}
            {{ form|crispy }}
            <div class = "d-grid gap-2">
              <button class="mt-3 btn btn-outline-primary primaryAction" type="submit">{% trans "Add E-mail" %}</button>
            </div>
        </form>
      {% endif %}
    </div>
</div>
```

(b) Ophidia source code before debugging

```
        <form method="post" action="{% url 'account_email' %}" class="add_email">
            {% csrf_token %}
            {{ form|crispy }}
            <div class = "d-grid gap-2">
              <button class="mt-3 btn btn-outline-primary primaryAction" type="submit" name = "action_add">{% trans "Add E-mail" %}</button>
            </div>
        </form>
      {% endif %}
    </div>
</div>
```

(c) Ophidia source code after debugging

Figure 26: Debugging the user authentication system

Once all the relevant changes were made, I performed a second testing run and all features functioned as intended and no further adjustments had to be made to the user authentication source code.

5.3 Social Feed and Posting Feature Test

The testing checklist to verify the functionality of the social feed and posting system was as follows:

- Users are redirected to the relevant pages when interacting with each item within the navigation bar
- Users are able to submit posts using the text form and submission button
- Users are able to view previously posted content below the submission form alongside the username of the user who submitted the post and the time of submission
- Posts are sorted so that the most recent appear first
- Users are able to see edit and delete icons underneath posts they have created but not under posts that they have not created
- Users are able to click the edit and delete icons which will redirect them to the relevant post edit and post delete pages
- Users are able to edit their posts through the post edit page
- Users are redirected to the post's comment page once they have edited their post
- Users are able to delete their posts through the post delete page
- Users are redirected to the social feed page once they have deleted their post
- Users are redirected back to the social feed once they press the 'Back to Post' button found within both the post edit and delete pages
- Users are redirected to the account log in page if they attempt to access any social feed links including post edit and delete pages without being logged in
- Users are returned a HTTP 403 Forbidden status code if accessing a post edit or delete URL of a post they have not created

All items on the testing checklist for the social feed and posting system passed on the first testing run and, therefore, no further changes had to be made to its source code.

5.4 Comment Section Feature Test

The testing checklist for the comment section was as follows:

- Users are able to view and click a comment icon which will redirect them to the comment page for that specific post
- Users are redirected back to the social feed once they press the 'Back to Feed' button found within each comment page
- Users are able to submit comments using the text form and submission button
- Users are able to view previously posted comments below the submission form alongside the username of the user who submitted the comment and the time of submission

- Comments are sorted so that the most recent appear first
- Users are able to see a delete icon underneath comments they have created but not under comments that they have not created
- Users are redirected to a comment deletion page when they click on a delete icon
- Users are redirected back to the post's comment page once they press the 'Back to Post' button found within comment delete pages
- Users are able to delete their comments through the comment delete page
- Users are redirected to the post's comment page once they have deleted their comment
- Users are redirected to the account log in page if they attempt to access a post's comment page or comment delete page without being logged in
- Users are returned a HTTP 403 Forbidden status code if accessing a comment page or comment delete URL of a comment they have not created

All checklist items had passed the first testing run and, thus, no further modifications had to be made to the source code of the comment section.

5.5 User Profiles Feature Test

The testing checklist for testing the user profile functionality for the Ophidia application was as follows:

- Users are redirected to a profile update page after they have created an Ophidia account
- Users are required to enter their names within the profile update page or else they will be redirected back to the profile update page until they have done so
- Users are redirected to their profile page once they click 'Profile' within the drop-down menu within the navigation bar
- Users are redirected to the account log in page if they attempt to access a user's profile page or profile edit page without being logged in
- Users are returned a HTTP 403 Forbidden status code if accessing the URL of a profile edit page that is not their own
- Users are able to see an edit icon next to their names on their own profile but not other profiles
- Users are successfully redirected to an update profile page once they have clicked the edit icon
- Users are redirected to their profile page when they press the 'Back to Profile' button found on the update profile page
- Users are able successfully edit their name, location, date of birth, profile bio and profile image using the update profile page
- Users are redirected back to their profile page once they have edited their profile

- Users are able to the post history of the user whose profile page they are accessing, sorted so that the most recent post appears first

All checklist items had pass the first testing run and, thus, no edits were required to be made to the source code for the user profile system.

5.6 Follower System Feature Test

The testing checklist for testing the user profile functionality for the Ophidia application was as follows:

- Users are able to see a 'Follow' button which appears under the profile summary of a user if the user viewing the profile is not the owner of the profile and is not currently following the user
- Users are able to see a 'Unfollow' button which appears under the profile summary of a user if the user viewing the profile is not the owner of the profile and is currently following the user
- The follower counter shows the correct follower count

All checklist items had pass the first testing run and, thus, no modification were required to be made to the source code for the follower system.

5.7 Rating System Feature Test

The testing checklist for testing the user profile functionality for the Ophidia application was as follows:

- Users can press the like icon to increase the like tally by one
- Users cannot increase the like tally by more than one through one account
- Users can press the dislike icon to increase the like tally by one
- Users cannot increase the dislike tally by more than one through one account
- If a user likes a post after disliking the post, it will remove the dislike from the post and then will like the post
- If a user dislikes a post after liking the post, it will remove the like from the post and then will dislike the post

All checklist items had pass the first testing run and, thus, no modification were required to be made to the source code for the rating system.

5.8 Searching System Feature Test

The testing checklist for testing the user profile search system for the Ophidia application was as follows:

- Users are able to find the profiles of users by inputting the exact username of the desired profile

- Users are able to find the profiles of users by inputting the correct first characters of the username of the desired profile
- Users are able to view all possible users from their query or partial query alongside their name and number of followers. It also shows their location and date of birth if it is given in their profile
- Users are unable to enter an empty query
- Users are returned a 'No Users Found' message if they have provided an incorrect query

All checklist items had pass the first testing run and, therefore, edits were no required to be made to the source code for the searching system.

5.9 Direct Messaging Feature Test

The testing checklist for testing the user profile search system for the Ophidia application was as follows:

- Users are redirected to their inbox page after clicking the direct message icon visible on their navigation bar
- Users are able to see a list of all their active message threads in their inbox page
- Users are redirected to a message thread creation page when clicking the 'Start a new message thread' button found on their inbox page
- Users are unable to start a message thread unless they have written out the entire username of the desired chat participant spelt correctly
- Users are redirected to the relevant message thread page after correctly starting a message thread
- Users are able to send messages
- Users are able to receive messages
- Users are redirected to their inbox page once they have clicked the 'Back to Inbox' button found on message thread pages
- Users are able to rename their message threads
- Users are redirected to the account log in page if they attempt to access a user's message thread name edit page without being logged in
- Users are returned a HTTP 403 Forbidden status code if accessing the URL of a message thread name edit page that is not their own
- Users are redirected to the message thread page once they have clicked the 'Back to Thread' button found on the message thread name edit page

All items on the testing checklist passed on the first testing run and, therefore, no further changes had to be made to the source code allowing for direct messaging functionality.

6 System Deployment

With the conclusion of the programming and testing phases, it was now time to deploy Ophidia to allow users to access the application remotely. This is waives the requirement for users to install dependencies before accessing the application as this procedure is only necessary for local testers. It also links the application to a domain which can be readily access via any web browser, thus making the application much more accessible for users who are inexperienced with the command line interface.

6.1 Heroku

Taking inspiration from Devndra Ghimre's paper 'Comparative Study on Python Web Frameworks: Flask and Django' [15], I elected to use the Heroku cloud platform to host my application which allows free hosting for smaller-scale application. Heroku provides three deployment methods: the Heroku command line interface (CLI), GitHub and Container Registry. The two relevant options for this project were the Heroku CLI and GitHub since, up until this point, the source code for the application was up-to-date on a public GitHub repository.

I opted to use the Heroku CLI. This is because before the application could be deployed, I had to change the back-end code for the application to no longer send e-mails via the console since the console would no longer be visible while accessing the application remotely via a web browser. However, the modification required to send e-mail through an account involved implementing sensitive e-mail information within the source code of the application (see figure below), and, thus, this change could not be updated on the publicly available GitHub repository. Since the GitHub repository was no longer completely updated this left the only viable option of deployment to be through the Heroku CLI.

```
150  LOGIN_REDIRECT_URL = 'post-list'
151  ACCOUNT_EMAIL_REQUIRED = True
152  EMAIL_BACKEND = 'django.core.mail.backends.console.EmailBackend'
```

(a) Source code to send email via console

```
150  LOGIN_REDIRECT_URL = 'post-list'
151  ACCOUNT_EMAIL_REQUIRED = True
152  EMAIL_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'
153  EMAIL_HOST = 'smtp.gmail.com'
154  EMAIL_HOST_USER = [redacted]@gmail.com'
155  EMAIL_HOST_PASSWORD = '[redacted]'
156  EMAIL_PORT = 587
157  EMAIL_USE_TLS = True
```

(b) Source code for sending emails (sensitive information redacted)

Figure 27: Source code changes made to implement e-mail functionality within Ophidia

To deploy the application to Heroku, the following steps had to be taken:

- Initialised a git repository in a new directory
- Copied the Ophidia source code from the old GitHub directory to the new Heroku repository and made the relevant e-mail changes to the source code as mentioned above

- Installed and implemented the 'django-on-heroku' package within the Ophidia source coding which made the relevant changes to the settings file to ensure the application worked on the Heroku platform
- Added a 'requirements.txt' file to the root directory which informs Heroku about the dependencies of the application
- Added a 'Procfile' file to the root directory which specifies the commands that are executed by the commands on startup
- Within 'Procfile', added instructions to use Gunicorn, a Python web server gateway interface application server which will handle multiple running instances to Ophidia, verifying if they are healthy or restarting them if needed whilst distribution requests across these instances and communicating with the web server

Following the completion of these steps, I pushed the directory back into the main branch of the Heroku repository resulting in the application being available on the URL: http://ophidia.herokuapp.com

7 System Maintenance

Despite the thorough testing phase, issues may still have persisted within the Ophidia application as the testing may have not been as thorough as anticipated. Issues that remained hidden within my device during testing also may have presented themselves within the devices of others.

7.1 Feedback Survey

To attempt to find these issues, I had intended to distribute Ophidia's URL amongst my school via the instant messaging platform Discord alongside a link which redirected users to a Google Form allowing them to report any bugs they may have come across.

7.1.1 Creation

I had intended for the form to be as simple as possible in order to encourage reporting. The consisted of only two questions which can be seen in Figure 28: asking for the respondent's name and requesting for details about any bugs. Since the user sample was contained within my school, requesting for the user's name within the form was enough for me to identify them and contact them regarding extra information on any errors they had reported.

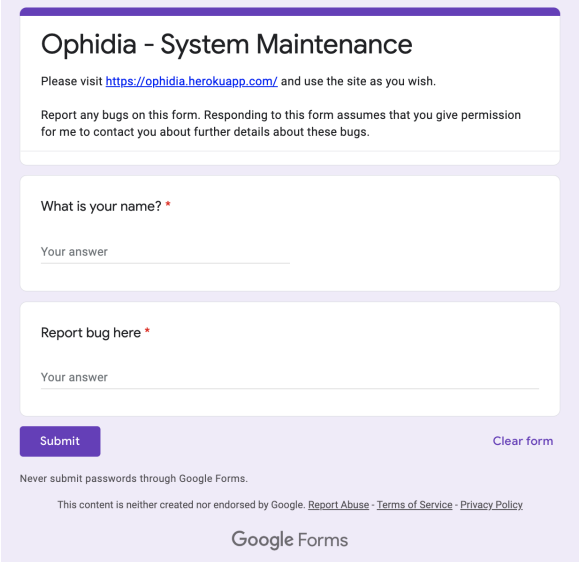


Figure 28: Bug reporting form

7.1.2 Results

The figure below shows the results from the form.

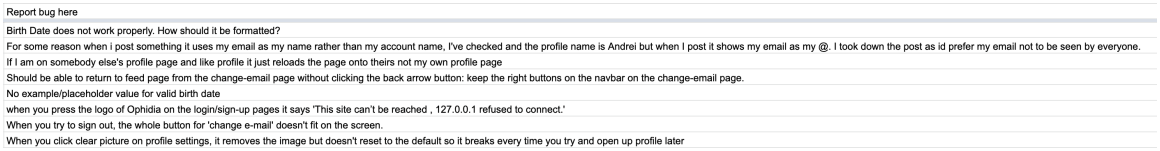


Figure 29: Bug reporting form results

The first step after collecting these responses was to rewrite these issues more formally using the help of the respective respondents. This resultant list of issues were as follows:

1. No information given regarding the formatting of the date
2. Email is used as the username and is publicly shown when posting
3. The application redirects to the same page when pressing the 'Profile' button on the navigation bar drop-down menu when on another user's profile page
4. 'Back to Feed' pages not found on the 'Change Email' page
5. No information given regarding the formatting of the date
6. Upon clicking the Ophidia icon on the navigation bar whilst on user authentication pages, the application redirects to '127.0.0.8' which does not exist
7. The items within the navigation bar do not fit the screen
8. Clearing the profile picture within the 'Profile Edit' causes the application to crash

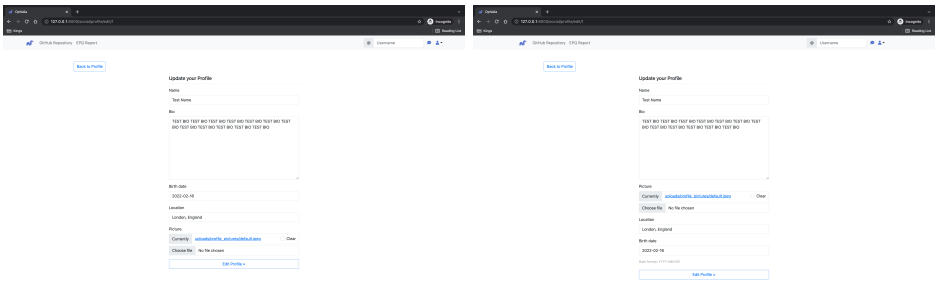
Before I was able to make changes to the application, I first had to filter out the responses that were attributed to user error and discard them. An example of this was the second response to the form, which I found out, after I had reached out to the respondent for extra information, was due to the user mistakenly inputting their e-mail address within the username quadrant of the sign up form. I was able to edit their username via my administrator Django super-user account which had successfully resolved the issue.

7.2 Subsequent Patches

Subsequently, I decided to implement seven patches within the application, each patch directly addressing each point within the form response. This required me to uncover the origin of each of the issues which I done so by requesting for the aid of the respective respondents.

To resolve entries one and five, I had to provide extra information regarding the formatting of the date within the 'Profile Edit' pages since it only accepted the unconventional date format YYYY-MM-DD. The change to the 'Profile Edit' page can be seen in Figure 30 (Overleaf).

To rectify the issue causing entry three, I had to review the source code of the navigation upon which I had realised that I had linked the profile button to 'user.profile.pk' when it should have been 'request.user.profile.pk'. This change can be seen in Figure 31 (Overleaf). The reason why this error had manage to go undetected through the testing process is because I had tested the function



(a) Profile edit form prior to patch (b) Profile edit form after patch

Figure 30: Modifications made to the application due to patch one

within the social feed and the error is only present when on another user's profile page. This change had to be made as it must be specified to the application that we are referring to the request user's profile page. If not done so, when on the social feed page, the programme assumes that we wish to be redirected to the logged in user's profile page and the site functions as intended. If on another user's profile page, however, the programme assumes that we wish to be redirected to the profile page of the user we are currently viewing, thus redirecting to the same page and causing the error.

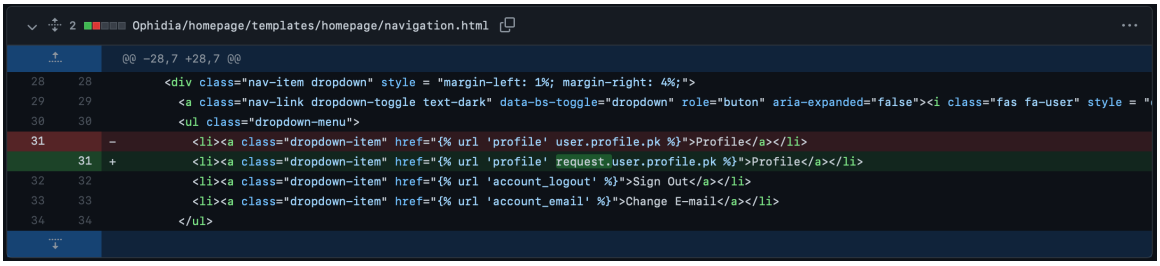
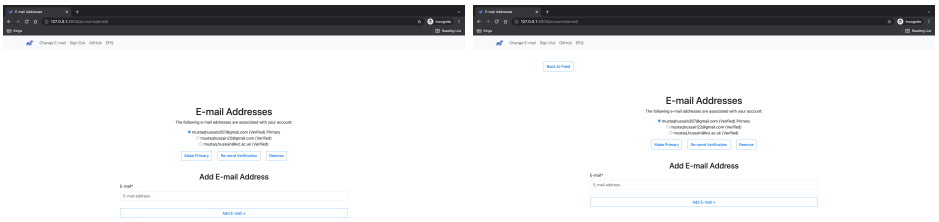


Figure 31: Changes made to the application source code due to Ophidia patch two

To resolve entry four, I had decided to add 'Back to Feed' buttons to the 'Change Email', 'Inbox' and 'Sign Out' pages. An example of this change to the site can be seen in the figure below.



(a) Change email page prior to patch (b) Change email page after patch

Figure 32: Modifications made to the application due to patch three

Entry six remained undiscovered during testing as I had forgotten that the navigation bar used within the User Authentication pages was developed separately from the navigation bar used throughout the application elsewhere. The intention for this was so that users who were not signed in were not able to access the searching bar and profile drop down menu found on the regular navigation bar. The functionality of this separate bar had not been tested and, thus, the error remained undetected. After reviewing the source code which can be found in the figure below, I had found that the issue originated from the referral link I used for the Ophidia icon. The changes to the source code implemented to rectify the issue can be visible in Figure 33.

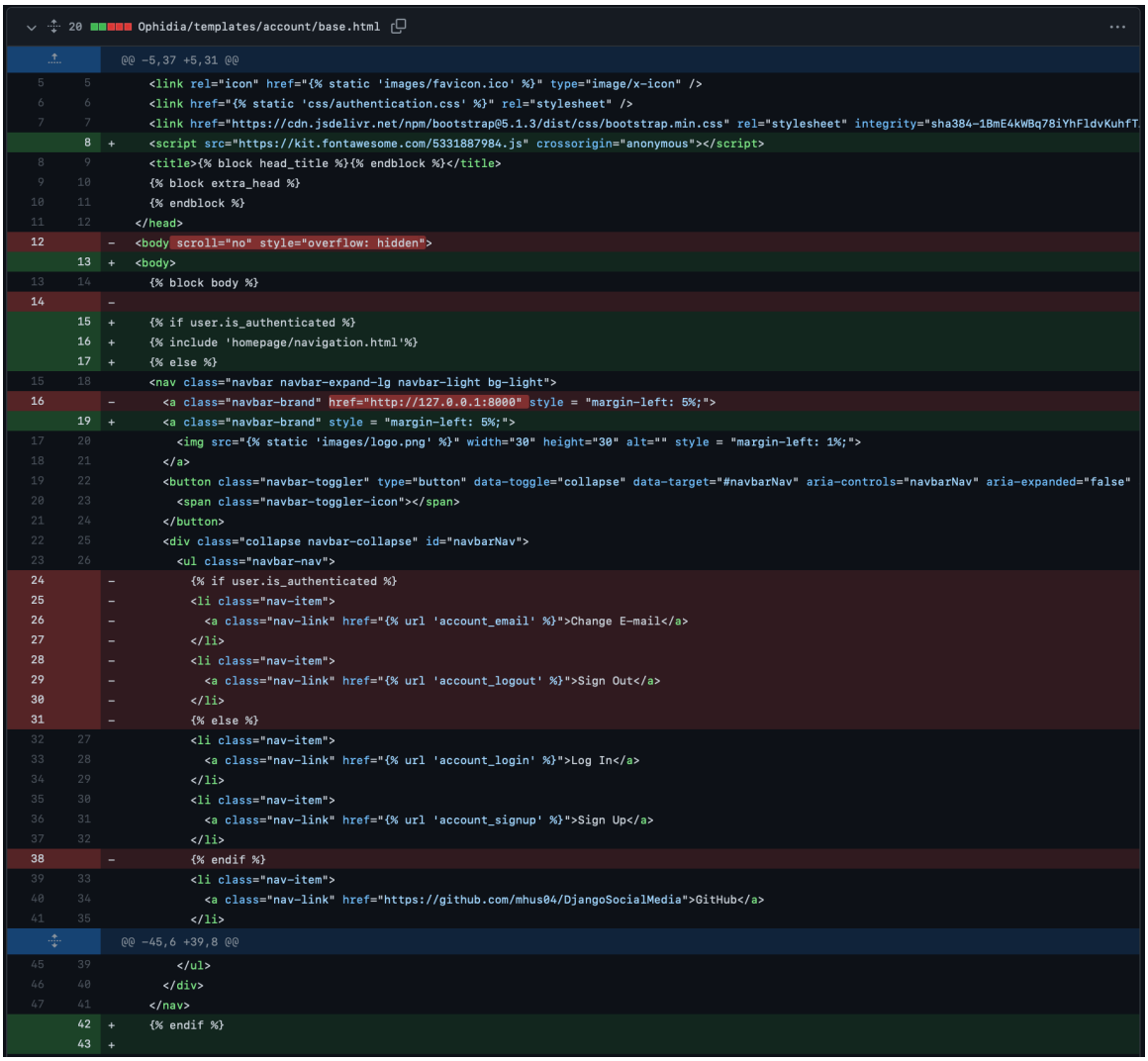
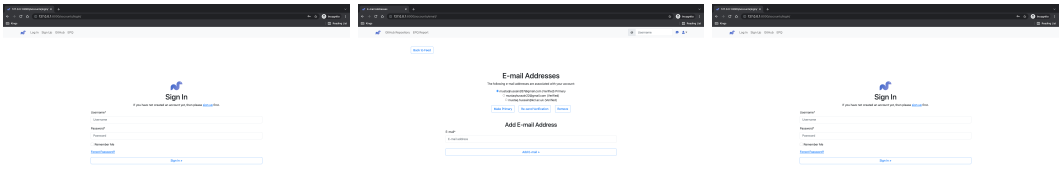


Figure 33: Changes made to the application source code due to Ophidia patch four

I had used '127.0.0.8' which would have redirected users back to the landing page if the page was accessed locally. To resolve this issue for authenticated users, I decided to implement an 'if' selection statement within the programme which told the application to use the regular navigation

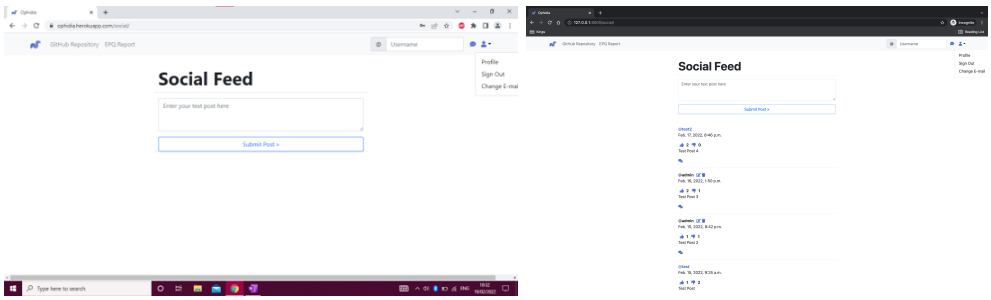
bar if the user is authenticated and, if they are not authenticated, to use the alternative navigation bar. To resolve this issue for users who were not logged in, I had removed the link from the icon altogether as it was redundant. The functionality of the landing page for non-authenticated users only consisted of allowing users sign in, sign up or access the GitHub repository and EPQ report. All these functions are found within the items of the navigation bar and, thus, the ability to return back to the landing page was an unnecessary function. The changes to the site can be seen in the figure below.



(a) Before patch (b) After patch: Signed in (c) After patch: Signed out

Figure 34: Modifications made to the application due to patch five

The issue causing entry seven had passed through testing as I was using a higher resolution display while testing compared to the respondent. This had meant that my web browser was able to successfully display all items within the navigation bar unlike this tester who was unable to fully view the drop-down menu items within the bar. The difference in content formatting for the different screen display can be seen in Figure 35.



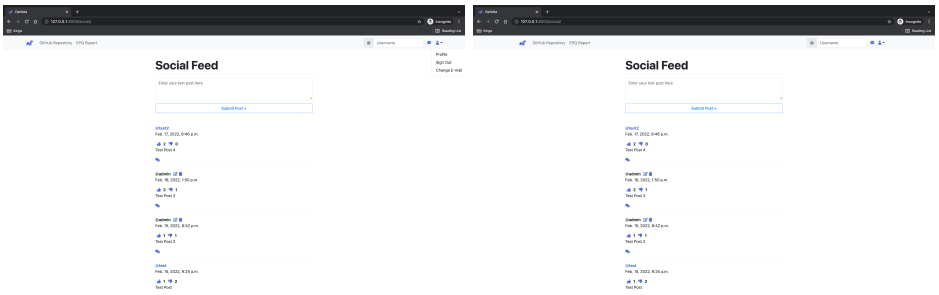
(a) Low resolution display (b) High resolution display

Figure 35: Difference in content formatting for the navigation bar on different screen display resolutions

To resolve this, I increased the left and right padding for the items within the navigation bar. This increased the distance between the edges of the display and the navigation bar elements, resulting in a design that was appropriate for a wider range of devices which can be seen in Figure 36 (Overleaf).

The seventh and last patch made to the Ophidia application was implemented to address the bug caused by clearing the profile picture of an account, causing the account to be locked and unable to access the rest of the site.

After analysing the source code of the 'Profile Edit' page, I was able to determine the origin of the issue. I had programmed the application to provide a default profile image to the user if they not uploaded their own but not for the default image to return after the custom image was removed.

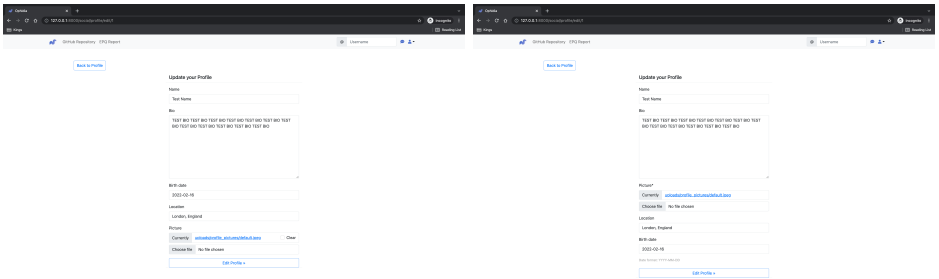


(a) Navigation bar before patch (b) Navigation bar after patch

Figure 36: Changes made to the application as a result of patch six

I had determined that removing the 'Clear image' feature, which had caused the error, did not significant effect the user experience, and, therefore, did not warrant the effort required to resolve the error in its functionality. This was also done as removing the ability to remove one's custom profile image encourages the user to use their own image, promoting customisability throughout the application.

The affect of the patch on the 'Edit profile' form can be seen in the figure below.



(a) Profile edit page before patch (b) Profile edit page after patch

Figure 37: Modifications made to the application due to patch seven

8 Personal Evaluation

With artefact development now concluded, I decided to perform an evaluation regarding the success of the project.

8.1 Artefact

As mentioned within the requirement analysis, the success of the artefact was dependent on the extent to which the artefact was able to fulfill its initial requirements. To reiterate, they were as follows:

1. The application must have a robust, secure user authentication system
2. The application must have concise and easy-to-use posting and commenting functionality
3. The application must allow users to edit or delete their posts
4. The application must allow users to like or dislike posts
5. The application must allow users to have and customise their own profile profile pages
6. The application must allow users to follow each other
7. The application must allow users to search for other users
8. The application must allow users to directly message other users
9. The application must have a user-friendly interface
10. The application must be remotely accessible via a domain

The Ophidia application had met all these requirements, however, I believe that certain elements within the requirements checklists had not been fulfilled to their fullest extent due to timing restrictions.

To improve requirement two, I could have allowed users to upload pictures within the direct messages and posts. As mentioned within the 'System Design' section of the report, I had decided to create the site such that the majority of its text was text-based due to the Heroku's ephemeral file storage system. Given more time, I could have avoided the limitations introduced by the file system by hosting the uploaded media from users on a third party cloud platform such as the free variant of Amazon S3 which provides 5 gigabytes of free storage space.

Regarding both requirements two and eight, the features could have been more 'easy-to-use' and been improved with more time. I knew that I was capable of implementing a notification system within the application which would notify users if new comment had appear on their existing or if there was an unread message within their inbox. However, upon analysing this feature, I had determined that this feature would be significantly time-consuming to implemented and, thus, was not incorporated.

Beyond the requirements of the artefact, I had personal objectives which I had originally planned to achieve as a consequence of undertaking a long-term independent project. Prior to the development of Ophidia, my web development experience was restricted to the front-end and I had intended for this project to contribute towards developing my proficiency within the field of server-side development. I believe that the project had successfully met this objective as I was able to develop and successfully deploy a fully functioning interactive website which was able to receive, store and

utilise user inputs. Evidence for a substantial understanding of the source code can be seen in my ability to identify and resolve all the faults within the application which were declared by users.

As mentioned within the 'Resources' section of the project, I had specifically utilised a Python web framework to create my application due to my previous experience with the language. This had meant that one of the primary objectives I had wished to obtain through the completion of this project was to further develop my current understanding of the Python programming language. Learning and implementing the Django web framework required knowledge of object-oriented programming, a topic that is not taught within the scope of the AS and GCSE Computer Science syllabus and, thus, I had limited prior competence in using the programming paradigm. The development of Ophidia had extensively utilised Python classes and objects which is proof of the successful development of my Python and general object-oriented programming capabilities as a result of undertaking this project.

If I were to carry out a similar project in the future, I would ensure to only utilise languages and frameworks which I had substantial experience in. As a consequence of my lack of knowledge in both the Django web framework and server-side development more generally, a substantial portion of the project timeline had to be dedicated to developing my skills to an extent which enabled me to produce a competent artefact. This resulted in a reduction of time dedicated to other integral phases of the project, such as development, and I was unable to implement as many features as I was capable of within the outlined time period.

8.2 Presentation

The final remaining task was to produce a presentation which would showcase Ophidia's development process. I opted to perform an analysis of the marking criteria that would be used to assess my performance in order to maximise the number of marks I was able to attain. This resulted in the creation of a checklist that covered each individual element of the mark scheme, mirroring the strategy used during the requirements analysis and testing phases of the project. A secondary purpose of creating such a list was to facilitate the presentation's evaluation process. Referring back to the list would allow me to easily analyse the extent to which the presentation was able to successfully address the points mentioned within the marking criteria. The checklist was as follows:

- The content within the presentation is clear, concise and easy to follow
- There is a clear and logical structure to the presentation
- Visual aids used are relevant and effective in supporting the presentation
- The presentation enables the presenter to engage the audience
- The presentation is well paced

During the delivery of my presentation, a major drawback had been emphasised. The audience would be unacquainted with the concepts of website development. Thus, to ensure that the that presentation would remain clear and easy to follow for the viewers, a significant proportion of slides would have to be dedicated in building familiarity with the technical terminology that was to be used. This was done by implementing a plethora of visual aids, allowing me to cover items 1 and 3 within the checklist simultaneously. However, in addressing these elements so thoroughly, it had induced a perverse effect. The number of slides to be used had risen to 22, which had meant that I was required to average 27 seconds per slide to remain within my allotted ten minute time period. This, not only, harmed my ability to express myself at a comfortable pace but also prevented me

from explaining the key concepts as thoroughly as I had originally intended to. The erratic pacing had unintentionally resulted in the presentation becoming less clear. It also harmed my ability to maintain a logical structure during the presentation. I had aimed to structure the presentation to imitate the linear and sequential properties of the Waterfall model, mirroring the structure of this EPQ report. This was to be done by linking each slide to each phase of the Waterfall process which can be seen within the highlighted region of the figure below.

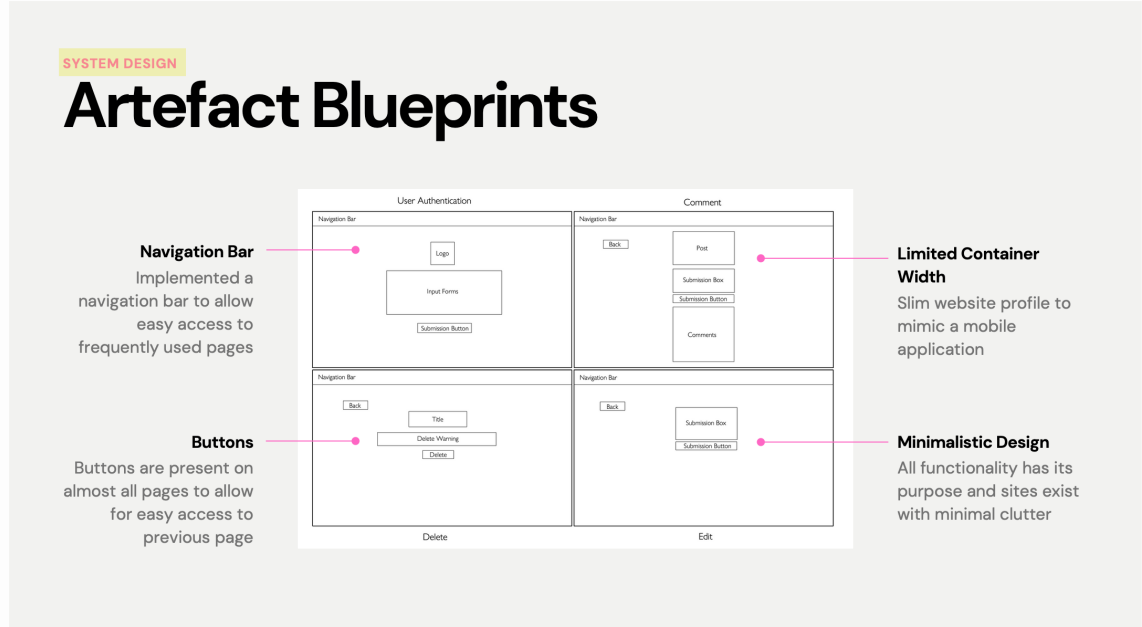


Figure 38: Slide from presentation

However, due to poor pacing, I had often forgotten to verbally make these links. Therefore, if I were to present again, I would reduce the amount of content within the presentation to thoroughly address only the fundamental portions of the development process, ensuring that I remain at a comfortable pace.

9 Bibliography

References

[1] Statista (2021). *Global social network penetration rate as of January 2021, by region.* https://www.statista.com/statistics/269615/social-network-penetration-by-region/. [Accessed 7th November 2021]

[2] Statista (2019). *Social network penetration worldwide from 2017 to 2025.* https://www.statista.com/statistics/260811/social-network-penetration-worldwide/. [Accessed 7th November 2021]

[3] Statista (2019). *Share of internet users worldwide who believe that social media platforms have had an impact on selected aspects of daily life as of February 2019.* https://www.statista.com/statistics/1015131/impact-of-social-media-on-daily-life-worldwide/. [Accessed 7th November 2021]

[4] Shilbey Telhami (2013). *The world through Arab eyes : Arab public opinion and the reshaping of the Middle East.* New York. Basic Books [Accessed 9th November 2021]

[5] The New York Times (2011). *Egypt Cuts Off Most Internet and Cell Service.* https://www.nytimes.com/2011/01/29/technology/internet/29cutoff.html [Accessed 9th November 2021]

[6] ProductPlan (2021). *What is the Software Development Lifecycle?.* https://www.productplan.com/learn/software-development-lifecycle/. [Accessed 9th December 2021]

[7] CloudDefence (2021). *System Development Life Cycle Guide.* https://www.clouddefense.ai/blog/system-development-life-cycle. [Accessed 14th December 2021]

[8] Virtasant (2020). *SDLC Methodologies: From Waterfall to Agile.* https://www.virtasant.com/blog/sdlc-methodologies. [Accessed 16th December 2021]

[9] TutorialsPoint (2021). *SDLC - Agile Model.* https://www.tutorialspoint.com/sdlc/sdlc_agile_model.htm. [Accessed 19th December 2021]

[10] Thandi Guilherme (2018). *I bet your Agile process is broken.* https://miro.medium.com/max/1400/1*d_-0VSR1TX636-s0H1FGAA.png. [Accessed 24th November 2021]

[11] W3Schools (2021). *SDLC Big Bang Model.* https://www.w3schools.in/sdlc-tutorial/big-bang-model/. [Accessed 4th December 2021]

[12] DistanceJob (2021). *What Are The Software Development Life Cycle (SDLC) Stages and Models.* https://distantjob.com/wp-content/uploads/2021/08/Big-Bang-SDLC-Model.jpg. [Accessed 25th December 2021]

[13] TutorialsPoint (2021). *SDLC - Waterfall Model.* https://www.tutorialspoint.com/sdlc/sdlc_waterfall_model.htm. [Accessed 27th December 2021]

[14] Software Testing Help (2022). *What Is SDLC Waterfall Model?*. https://www.softwaretestinghelp.com/wp-content/qa/uploads/2014/04/Waterfall-model-phases.jpg. [Accessed 1st February 2022]

[15] Ghimire D (2020). *Comparative study on Python web frameworks: Flask and Django*. Bachelor's Thesis. Metropolia Univesity of Applied Sciences; 2020. [Accessed 17th October 2021]

[16] Django Software Foundation (2021). *Django Documentation Release 3.2.9.dev*. [Accessed 16th October 2021]

[17] GitHub (2021). *Django GitHub Repository*. https://github.com/django/django [Accessed 17th October 2021]

[18] StackOverflow (2021). https://stackoverflow.com [Accessed 4th November 2021]

[19] StackExchange (2021). *All Sites - StackExchange*. https://stackoverflow.com [Accessed 4th November 2021]

[20] Instagram (2021). https://www.instagram.com [Accessed 4th November 2021]

[21] Statista (2021). *Most popular social networks worldwide as of July 2021, ranked by number of active users (in millions)*. https://www.statista.com/statistics/272014/global-social-networks-ranked-by-number-of-users/. [Accessed 4th November 2021]

[22] Django (2021). *Django Overview — Django*. https://www.djangoproject.com/start/overview/. [Accessed 4th November 2021]

[23] freeCodeCamp.org (2021). *Python Backend Web Development Course (with Django)*. [Video] https://www.youtube.com/watch?v=jBzwzrDvZ18&t=12323s [Accessed 18th October 2021].

[24] freeCodeCamp.org - YouTube (2021). *freeCodeCamp.org - About*. https://www.youtube.com/c/Freecodecamp/about

[25] GitHub (2021). *freeCodeCamp GitHub Repository*. https://github.com/freeCodeCamp/freeCodeCamp. [Accessed 18th October 2021]

[26] freeCodeCamp (2021). *About freeCodeCamp - Frequently Asked Questions*. https://www.freecodecamp.org/news/about. [Accessed 18th October 2021]

[27] Scott Chacon, Ben Straub (2014). *Pro Git 2nd ed. Edition*. Apress. [Accessed 4th November 2021]

[28] Serengeti Software Tech (2021). *Introduction to Git and Types of Version Control Systems*. https://serengetitech.com/tech/introduction-to-git-and-types-of-version-control-systems/. [Accessed 4th November 2021]

[29] GitLab (2021). *What is a centralized version control system?*. https://about.gitlab.com/topics/version-control/what-is-centralized-version-control-system/. [Accessed 4th November 2021]

[30] Eduonix (2018). *Learn The Three Different Types Of Version Control Systems*. https://blog.eduonix.com/software-development/learn-three-types-version-control-systems/. [Accessed 4th November 2021]

[31] Microsoft (2018). *Microsoft Acquires GitHub.* https://news.microsoft.com/announcement/microsoft-acquires-github/. [Accessed 6th November 2021]

[32] Software (2019). *Code Editor Overview: Why You Should Switch to VS Code.* https://www.software.com/src/why-you-should-switch-to-vs-code. [Accessed 6th November 2021]

[33] Microsoft (2021). *IntelliSense in Visual Studio Code.* https://code.visualstudio.com/docs/editor/intellisense. [Accessed 6th November 2021]

[34] Microsoft (2021). *Emmet in Visual Studio Code.* https://code.visualstudio.com/docs/editor/emmet. [Accessed 6th November 2021]

[35] Heroku (2021). *Active storage on Heroku.* https://devcenter.heroku.com/articles/active-storage-on-heroku. [Accessed 22nd February 2022]

[36] SnapChat (2011). *SnapChat Logo.* https://assets.stickpng.com/images/580b57fcd9996e24bc43c536.png. [Accessed 5th December 2021]

[37] Facebook (2004). *Facebook Logo.* https://upload.wikimedia.org/wikipedia/commons/thumb/0/05/Facebook_Logo_%282019%29.png/1024px-Facebook_Logo_%282019%29.png. [Accessed 5th December 2021]

[38] WhatsApp (2009). *WhatsApp Logo.* http://assets.stickpng.com/thumbs/5ae21cc526c97415d3213554.png [Accessed 5th December 2021]

[39] BluLeadz (2021). *Why Are Social Media Sites Blue? How Color Psychology Drives Engagement.* https://www.bluleadz.com/blog/why-are-social-media-sites-blue. [Accessed 7th November 2021]

[40] Color Psychology (2021). *Color Blue: Psychology, Meaning, Symbolism and more.* https://www.colorpsychology.org/blue/. [Accessed 7th November 2021]

[41] SimpleLogin (2019). *Why you shouldn't create your own authentication system.* https://simplelogin.io/blog/do-not-create-own-auth-system/. [Accessed 4th February 2022]

[42] GitHub - pennersr (2021). *django-allauth GitHub Repository.* https://github.com/pennersr/django-allauth. [Accessed 7th February 2022]