**Introduction**

This document presents a comprehensive analysis of the development journey for Echo Gallery, a single-page web application (SPA) crafted for image sharing and social media functionalities, designed as a project for the Comp1004 module. It unfolds the narrative of the project's lifecycle, beginning with the initial concept and extending through to the strategic choices in software development lifecycle (SDLC) methodologies. Providing a detailed exploration of the project's inception, research foundation, design iterations, technological underpinnings, and the spectrum of challenges navigated, this report aims to deliver a view of the developmental achievements. Emphasising the nature of the project's progress, it highlights the agile Scrum framework's role in facilitating continuous planning, execution, and enhancement phases. This is further illustrated by detailed planning sprints and substantiated by the GitHub commit history, showcasing the dynamic evolution of the project.

**Software Development Life Cycle**

The 6 stages of the Agile SDLC are as follows:

* Plan
* Requirements Analysis
* Design
* Implementation
* Testing and Integration
* A diagram of software development cycle

  Description automatically generatedMaintenance

Agile SDLC Diagram (Pinheiro, 2018)

The SDLC is an industry standard, each stage of the SDLC is as important as the next, it provides a clear roadmap for the entire software development process, which, according to (Development, 2023) is essential for ensuring a well-orchestrated development process. Failure to follow all the stages and simply start product development without a pre-established guide, can lead to the project not being carried out properly also the final project can be plagued with errors. Consequently, leading it to not have the favouritism of the public and thus all the work done was forgotten and not worth it. Hence, the following of the SDLC stages is not just about adhering to best practices but ensuring that the end product stands the test of market viability and user satisfaction.

In the next parts of this report, I will explain about each step of the Software Development Life Cycle (SDLC), highlighting the indispensable value and specific contributions of every phase. This report will explore how the journey from a simple idea to a fully functioning software involves distinct steps that contribute to enhancing the software's quality and effectiveness. This thorough analysis aims to provide a clear understanding of how a methodical approach to the SDLC can lead to the creation of a robust, efficient, and successful software product.

**Project Vision**

* Talk about what I want to accomplish keeping it strict with laws (copy right law, GDPR, Data Security Requirements, Accessibility Requirements) etc
* SPA

**Background**

Social media is increasingly becoming more popular with the rise of technology through phones and ease of access. As Facebook has over 2.3 billion users and other social media sites being used by more than two-thirds of internet users and is still growing (Ortiz-Ospina, 2019). This shows that there is a growing demand for users to exchange everyday experiences, photographs, and establish connections with one other (University of Cumbria, 2019). Echo Gallery is designed to be a new social media platform offering a space for users to post and appreciate images shared by their peers. Additionally, it fosters a sense of community by enabling users to interact through comments and likes on shared content. Developing this project required an in-depth analysis of the Software Development Life Cycle (SDLC) methodologies and executing the Agile model successfully and consistently. This report delves into the specific phases of the Agile methodology that were employed and details their implementation throughout the course of the project development.

**Plan**

The plan stage is crucial to the SDLC as it sets the foundation for the entire project. During this phase, project goals are defined, and a high-level plan for the software project is established. It involves identifying the scope of the project, potential risks, resources required, and timelines. Effective planning ensures that the project team has a clear understanding of the project's objectives and a roadmap to achieve them. Skipping or poorly executing the plan stage can lead to misunderstandings, wasted resources, delayed timelines, and ultimately, project failure. Therefore, investing time and effort in thorough planning is crucial for the success of any software development project. This foundational stage paves the way for the next steps in the SDLC, ensuring that the project team is prepared to tackle the challenges ahead with a clear, strategic approach.

Below is a detailed Sprint plan I have created for my project; it is crafted with attention to ensure a phased and systematic development. This project places a strong emphasis on the user interface (UI), with an early focus on UI design underscoring our commitment to a user-centric methodology. The plan shows a logical progression in feature development, from basic layout to more complex functionalities like commenting, liking/disliking, and image uploading. I firmly believe that the allocation of time for soliciting user feedback and performing comprehensive final testing before fully integrating is crucial, this not only helps in identifying and fixing bugs but also ensures that the final product is polished and user-friendly. The final weeks are dedicated to reviewing the project and preparing for the presentation, allowing for reflection on what worked well and what challenges I encountered.

|  |  |  |
| --- | --- | --- |
| **Week** | **From** | **Description** |
| **1** | 22 Jan | Analysis – research information/photo upload based websites (e.g. Pinterest, Instagram), create User Stories, UML Diagrams and Use Case to aid with development |
| **2** | 29 Jan | Design – Final design choices of UI. Incorporate insights from analysis phase into practise |
| **3** | 5 Feb | Sprint 1 – Pre-loaded images displayed in the UI and main body layout |
| **5** | 12 Feb | Sprint 2 – Develop navigation bar |
| **6** | 19 Feb | Sprint 3 – Upload image function and box |
| **9** | 26 Feb | Sprint 4 – Implement when user uploads photo with comment/information and able to add name and theme |
| **9** | 4 March | Sprint 5 – Implement when user uploads image it is displayed in main body with all details entered |
| **10** | 11 March | Sprint 6 – Create a function that when clicked, it shows more information about the image instead of just image and text, giving a collage feel. |
| **11** | 18 March | Sprint 7 – Add comment and like/dislike functionality |
| **12** | 25 March | Sprint 8 – Finalise any GUI development. Get feedback from users to note any bugs or improvements that can be made |
| **13** | 1 April | Sprint 9 – Final testing and resolve any bugs that may come up |
| **14** | 8 April | Review and finalise project report |
| **15** | 15 April | Final due diligence. Ensure everything is in order before the portfolio submission. |
| **Portfolio due Thursday 18th of April** | | |
| **16** | 22 April | Revise and make notes of what went well along the way and what was difficult for presentation |
| **17** | 29 April | Prepare for presentation |
| **18** | 6 May | Presentation |
| **Note:** | | Report will be updated when progress is made every week  Unit testing will be done throughout the project |

**Requirements Analysis**

The Requirements Analysis stage is fundamental in the SDLC as it sets the foundation for a successful project. This stage helps to identify and document what the software should do and how it should perform, guiding the design and implementation phases. Effective requirement analysis ensures that the project is correctly scoped from the start, avoiding feature creep, and reducing the risk of developing a product that fails to meet user needs. The documentation of requirements during this phase serves as a guideline for the subsequent stages of the SDLC, ensuring that development has a clear understanding of what needs to be built.

* Talk about what is below

**User stories**

|  |  |
| --- | --- |
| **User** | **Visitor** |
| As a user or visitor, I want to view images content. | |
| As a user or visitor, I want a navigation bar so that I can easily access different sections of the site. | |
| As a user, I want to log in. | As a visitor, I want to sign up so that I can create my own account, upload images, and interact with other users. |
| As a user, I want to securely sign out. |  |
| As a user, I want to comment on images. |  |
| As a user, I want to like or dislike images. |  |
| As a user, I want to upload images. |  |
| As a user, I want to edit my uploaded images. |  |
| As a user, I want to delete images that I have uploaded if I no longer want them to be displayed on the platform. |  |
| As a user, I want to delete my uploaded images. |  |

**UML diagram**

A diagram of a computer

Description automatically generated

A diagram of a computer

Description automatically generated

**Use Case**

A diagram of a person's work flow

Description automatically generated

**Use Case Description**

|  |  |
| --- | --- |
| **Name** | View Image |
| **Short Description** | Allows users to view details of an image, including a larger version, comments, and likes/dislikes. |
| **Precondition** | The image is publicly available, or the user has permission to view it.  The system is operational |
| **Post Condition** | User clicks on the arrow on the image, views the image and available interactions (comments, likes, dislikes etc). |
| **Error Situations** | Image fails to load due to a broken link or server error. |
| **System state in the event of an error** | The system displays an error message and does not show the image. |
| **Actors** | User, Visitor |
| **Triggers** | User selects an image to view from the gallery. |
| **Standard Process** | User clicks on an image thumbnail.  System retrieves and displays the full-size image along with uploader, upload date, comments, likes, and dislikes.  User can read comments and view like/dislike counts. |
| **Alternative Process** | 4’ If the image cannot be displayed, offer an error message and possibly a thumbnail or placeholder image. |

|  |  |
| --- | --- |
| **Name** | Upload Image |
| **Short Description** | Upload an image |
| **Precondition** | User is logged in  The system is operational |
| **Post Condition** | The image is displayed in the gallery with the provided details |
| **Error Situations** | The image file format is not supported.  The image file is too large.  There's a network or server error preventing upload. |
| **System state in the event of an error** | The system remains stable.  The image is not uploaded, and an error message is displayed. |
| **Actors** | User |
| **Triggers** | The user selects upload image |
| **Standard Process** | User clicks on the 'Upload Image' button.  User selects an image file to upload.  Image Preview is displayed.  User enters image details (title, description).  System validates the image file and details.  System uploads the image, updates the gallery, and confirms the upload to the user. |
| **Alternative Process** | 7’ If the image file is not in a supported format or exceeds size limits, the system prompts the user to select a different file.  8’ If there's a network or server error, the system offers the user the option to retry the upload. |

|  |  |
| --- | --- |
| **Name** | Edit Image |
| **Short Description** | Allows a user to edit the details of an image they have previously uploaded |
| **Precondition** | User is logged in.  Must be owner of the image.  The system is operational |
| **Post Condition** | The image details are updated in the gallery. |
| **Error Situations** | User tries to edit an image they do not own.  Invalid input for image details.  Network or server error during the update process |
| **System state in the event of an error** | The system displays an error message and does not apply the changes. |
| **Actors** | User |
| **Triggers** | The user selects the 'Edit Image' option for one of their images. |
| **Standard Process** | User selects an image from their gallery and chooses the 'Edit' option.  User modifies the image details such as title, description, etc.  System validates the new input details.  System updates the image details and confirms the changes to the user. |
| **Alternative Process** | 5’ If the user does not own the image, the edit option is not available.  6’ If there is invalid input, the system prompts for correction before submission.  7’ If a network or server error occurs, the system provides an option to retry. |

|  |  |
| --- | --- |
| **Name** | Delete Image |
| **Short Description** | Allows a user to remove an image they have uploaded from the gallery. |
| **Precondition** | User is logged in.  Must be owner of the image.  The system is operational |
| **Post Condition** | The image is deleted from the gallery and is no longer accessible. |
| **Error Situations** | The system remains stable.  Network or server error during the deletion process. |
| **System state in the event of an error** | The system displays an error message and does not apply the changes.  The image is not deleted, and an error message is displayed. |
| **Actors** | User |
| **Triggers** | The user selects the 'Delete Image' option for one of their images. |
| **Standard Process** | User selects an image from their gallery and chooses the 'Delete' option.  The system asks the user to confirm the deletion.  Upon confirmation, the system deletes the image.  System updates the gallery and confirms the deletion to the user. |
| **Alternative Process** | 5’ If the user does not own the image, the delete option is not available.  6’ If a network or server error occurs, the system provides an option to retry the deletion. |

|  |  |
| --- | --- |
| **Name** | Sign Up |
| **Short Description** | Allows visitors to create a new user account in the system. |
| **Precondition** | Visitor is not already logged in. |
| **Post Condition** | A new user account is created, and the user is logged in. |
| **Error Situations** | Invalid input (e.g., non-valid email format).  Email already registered.  Server or database error during account creation. |
| **System state in the event of an error** | No new account is created, and the visitor is prompted to correct the input or try again later. |
| **Actors** | Visitor |
| **Triggers** | Visitor selects the 'Sign Up' option. |
| **Standard Process** | Visitor fills out the sign-up form with email, username, and password.  System validates the provided details.  System creates a new user account and logs the user in.  System confirms account creation to the user. |
| **Alternative Process** | 5’ If the email is already in use, prompt to try a different email.  6’ If the input is invalid, prompt to correct the details. |

|  |  |
| --- | --- |
| **Name** | Log In |
| **Short Description** | Allows users to access their account by providing credentials. |
| **Precondition** | User is registered and not currently logged in. |
| **Post Condition** | User is granted access to their account. |
| **Error Situations** | Incorrect credentials.  User account is locked or disabled.  Server or database error during authentication. |
| **System state in the event of an error** | Access is not granted, and the user is informed of the error. |
| **Actors** | User |
| **Triggers** | User selects 'Log In' and submits credentials. |
| **Standard Process** | User enters their email and password.  System validates the credentials.  System grants access and transitions to the user's profile or homepage. |
| **Alternative Process** | 4’ If credentials are incorrect, deny access and prompt to try again or reset password. |

|  |  |
| --- | --- |
| **Name** | Sign Out |
| **Short Description** | Allows users to securely exit their accounts. |
| **Precondition** | User is currently logged in. |
| **Post Condition** | User session is terminated, and the user is redirected to the homepage or sign-in page. |
| **Error Situations** | User's session does not close properly due to a server error.  There is a network error preventing the sign-out request from reaching the server. |
| **System state in the event of an error** | The system maintains the user’s session and prompts the error, advising to retry. |
| **Actors** | User |
| **Triggers** | User clicks the 'Sign Out' button. |
| **Standard Process** | The user clicks the 'Sign Out' button in the application interface.  The system processes the sign-out request and terminates the user's session.  The user is redirected to the homepage or sign-in page, and a message confirming successful sign-out is displayed. |
| **Alternative Process** | 4’ If the sign-out process fails, inform the user and provide the option to attempt to sign out again. |

|  |  |
| --- | --- |
| **Name** | Comment on Image |
| **Short Description** | Allows users to add comments to images in the gallery. |
| **Precondition** | User must be logged in. |
| **Post Condition** | Comment is publicly visible on the image's context page. |
| **Error Situations** | Comment submission fails due to a network or server error.  Comment contains prohibited content, triggering moderation. |
| **System state in the event of an error** | Comment is not posted, and the user is notified of the failure. |
| **Actors** | User |
| **Triggers** | User submits a comment on an image's page by selecting ‘Post Comment’ or pressing enter. |
| **Standard Process** | User types a comment in the comment field.  System validates the comment.  System posts the comment below the image.  System displays the comment to all viewers of the image. |
| **Alternative Process** | 5’ If there's a network error, provide an option to retry.  6’ If the comment is above the word max, provide an error with how many they have used and what is the max |

|  |  |
| --- | --- |
| **Name** | Like/Dislike Image |
| **Short Description** | Allows users to express their opinion on an image with a like or dislike. |
| **Precondition** | User must be logged in. |
| **Post Condition** | Like or dislike count is updated, and user's preference is recorded. |
| **Error Situations** | Like/dislike action fails due to network or server error. |
| **System state in the event of an error** | The like or dislike is not registered, and the user is prompted to try again. |
| **Actors** | User |
| **Triggers** | User clicks the 'Like' or 'Dislike' button on an image. |
| **Standard Process** | User selects the 'Like' or 'Dislike' button.  System updates the like or dislike count for the image.  System records the user's action to prevent multiple votes. |
| **Alternative Process** | 4’ If there's a network error, provide an option to retry. |

|  |  |
| --- | --- |
| **Name** | Use Navigation Bar |
| **Short Description** | Allows users to navigate through different sections of the website using the navigation bar. |
| **Precondition** | The website is accessible, and the navigation bar is visible. |
| **Post Condition** | User is taken to the selected section of the website. |
| **Error Situations** | The selected section does not load due to a network or server error.  The user attempts to navigate to a restricted area without appropriate permissions. |
| **System state in the event of an error** | The system remains on the current section, and an error message is displayed. |
| **Actors** | User, Visitor |
| **Triggers** | User selects an option in the navigation bar. |
| **Standard Process** | User clicks on a section title in the navigation bar.  System loads and displays the selected section.  System updates the URL and browser history for possible future navigation. |
| **Alternative Process** | 4’ If a section fails to scroll, display an error, and provide a retry option. |

**Functional Requirements**

1. The platform shall allow users to create an account using an email address and a password.
2. The platform shall authenticate users via email and password upon logging in.
3. The platform shall enable users to upload images along with titles, author name and descriptions.
4. The platform shall allow users to log out.
5. Visitors without an account and users shall have the ability to view images.
6. The platform shall have a database to read from and write to.
7. The platform shall enforce privacy settings to ensure user-uploaded images are only editable or can be deleted by the uploader.
8. The platform shall ensure all user data is handled in compliance with GDPR.
9. The platform shall provide error messages and guidance for users when interactions fail, or inputs are invalid.
10. The platform shall include a navigation bar for easy access to different sections.
11. The platform shall display images in an organised gallery view.
12. All user interactions like sign-up, log in, image upload, and comments shall be logged for potential auditing.
13. The system shall be maintainable with clear documentation for future development and updates.
14. Users shall have the ability to like or dislike images.
15. The platform shall provide a comment feature on images for logged-in users.
16. The platform shall allow users to edit the details of images they have uploaded.
17. The platform shall permit users to delete their uploaded images.
18. The platform shall be responsive and functional on various devices and screen sizes.

Must: 1-9

Should: 10-13

Could: 14-18

Non-functional Requirements

1. The platform's user interface shall be intuitive and user-friendly, requiring minimal instruction for new users to navigate.
2. The platform shall ensure data security, especially user emails and authentication data.
3. The platform shall be developed with scalability in mind, able to accommodate an increasing number of users and images.

**Design**

The design phase is a crucial stage in the SDLC because it serves as a blueprint for the entire project, laying the groundwork for what will be built. This phase involves turning the software specifications into a design plan. It includes the architectural design, component selection, user interface design, and the definition of other crucial elements such as data structures, algorithms, and the detailed software architecture. By addressing these components early on, the design phase ensures that the software will be scalable, maintainable, and meet the specified requirements. Moreover, it helps in identifying potential problems early, reducing the risk of costly reworks and ensuring a smoother development process. Effective design is key to reducing development time, staying within budget, and ensuring the final product meets both customer expectations and quality standards.

* Architecture

The development of this Single Page Application for an image sharing social media platform is a complete project that brings together an array of technologies and services for seamless user experience. At its core, the platform features robust image management capabilities, enabling users to upload, view, and interact with visual content in a dynamic, intuitive interface.

It incorporates sophisticated user authentication mechanisms, courtesy of Firebase, ensuring secure access and personalised sessions. The application also taps into the cloud-based Firestore database for efficient, real-time storage and retrieval of images and user-generated metadata. The streamlined architecture not only facilitates image categorisation and community engagement through likes and comments but also maintains a Single Page Application (SPA) structure for smooth, uninterrupted user navigation.

* Package Diagram

A diagram of a software company

Description automatically generated

* Sequence Diagram

A diagram of a diagram

Description automatically generated

A diagram of a diagram

Description automatically generated with medium confidence

A diagram of a sign in a computer

Description automatically generated

**Sitemap**

This section provides an outline of how the application is designed. Whilst this is a single page application, the sitemap indicates how the user will navigate through the topics.

A diagram of a company

Description automatically generated

**Wireframes**

This section provides an illustration of the wireframes.

A screenshot of a website

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Implementation**

The Implementation stage is where the actual coding and construction of the software occur. After meticulous planning, design, and requirement analysis, the implementation phase brings the project to life. During this stage, I have translated the design documents and specifications into source code, integrating various components and libraries necessary to build the software. It's a phase characterised by actual development activities, including coding, testing individual units, and integrating different software modules into a cohesive system.

Below is review of the sprints carried out in my project. It details the struggles and achievements encountered throughout the development process, highlighting the iterative nature of building a dynamic and user-friendly web application.

**Review of Sprints**

Sprint 1 – This was a difficult sprint to start with. At first, I decided to have an “images” folder and manually have the sources of the images embedded into the HTML, however I decided against this. Instead, I used a database (Firebase, Firestore) which allowed me to store the images (Figure 1) into a database and dynamically displayed through JavaScript by looping through and searching for those categories within the database (Figure 2, 3, 4). This approach significantly streamlined the process for managing image-related data, such as likes, dislikes, comments, and categories in subsequent sprints, leveraging Firebase's data handling capabilities.

A screenshot of a computer

Description automatically generated

**Figure 1**

A black screen with text

Description automatically generated

**Figure 2**

A screen shot of a computer program

Description automatically generated

**Figure 3**

A screen shot of a computer code

Description automatically generated

**Figure 4**

Sprint 2 – Recognising the importance of user experience, I prioritised the development of a navigation bar so they can navigate the page effortlessly on where they want to go (Michaela, 2022). A well-designed navigation bar is essential for seamless website navigation, guiding users effortlessly to their desired content. After researching, I adapted a navigation template from (W3schools, 2019), customising it to fit the project's needs (Figure 5, 6). The resulting navigation bar was both comprehensive and professional, aligning with my objectives for this sprint.

A screen shot of a computer program

Description automatically generated

**Figure 5**

A screenshot of a computer

Description automatically generated

**Figure 6**

Sprint 3 – One of my most important functionalities in my project is the uploading of images, so giving myself time to develop this was of high importance. I used a Modal (W3Schools, 2019) that when pressed would open a window where the user could navigate through uploading their image (Figure 7). The image upload function is within in the DOM and retrieves all the data inputs from HTML. As stated previously, the use of Firestore database made things a lot easier, as I would use its capabilities for storing and retrieving data efficiently (Figure 8). By structuring the database to accommodate image metadata and relying on Firebase Storage for file handling, I created a user-friendly and responsive image upload system. This process has not only simplified the management of image data but also enhanced the overall user experience by providing immediate and clear feedback throughout the upload process. The accompanying JavaScript code (Figure 9) orchestrates the image upload logic, interfacing with Firebase to store image details and retrieve the necessary data. This sprint's challenge was mastering CSS to achieve the desired aesthetic, requiring extensive research and experimentation to implement effectively.

A screenshot of a upload page

Description automatically generated

**Figure 7**

A screenshot of a computer

Description automatically generated

**Figure 8**

A screen shot of a computer program

Description automatically generated

**Figure 9**

Sprint 4 – I thought this sprint would be a lot more difficult. Building on the modal implemented in Sprint 3, additional input fields were integrated to capture the image name, author, description, and category, each crucial for categorising and retrieving content effectively (Figure 7). When users upload their images, these details are encapsulated within a structured data object and sent to Firestore using asynchronous HTTP requests, ensuring that each image is accompanied by relevant, searchable metadata. This addition required minimal adjustments to the existing upload function, thanks to the robust infrastructure already in place. This efficient handling of image data allows for a more organised and accessible image gallery, enhancing both the functionality and user interaction with the platform.

Sprint 5 – Giving myself a week for this sprint was unnecessary in hindsight, revealing the advantages of integrating the Document Object Model (DOM) directly with Firebase. By manipulating the DOM, I was able to dynamically generate content based on the data received from Firestore, which streamlined the process of rendering images for each category on the fly. This JavaScript-driven approach facilitated real-time updates to the UI without necessitating page reloads, ensuring a fluid user experience. With the categories array defined in the DOM (Figure 2), and the forEach loop invoking the displayImagesByCategory asynchronous function (Figure 3), the project's responsiveness was significantly improved. This meant that I was able to add the addition of Firebase Authentication (Figure 10). It allowed for a secure sign-in and sign-up process, which, in turn, became integral for subsequent functionalities such as commenting with user identification. Leveraging the DOM for dynamic content creation and Firebase for user management, was a strategic decision that substantially enhanced the interactivity and security of the application. This approach allowed me to add an extra layer of functionality within the project as the user now must be signed in to use most functions on the website. This was a great help with subsequent sprints.

A screenshot of a computer

Description automatically generated

**Figure 10**

Sprint 6 – Was a leap forward in enhancing user engagement with the application. I introduced an interactive element, a down arrow, using the template literal syntax in JavaScript to create a snippet of HTML. This down arrow serves as an intuitive trigger for a modal, like upload modal, that displays expanded details of an image, including the uploader's name and description. The modal, a floating overlay that provides additional information without navigating away from the page. By leveraging JavaScript's ability to dynamically inject HTML into the DOM, the image details retrieved from Firebase are presented in a structured and user-friendly format (Figure 11). This addition not only enriched the user experience by allowing immediate access to detailed image information but also laid the groundwork for more interactive features in upcoming sprints, such as comments and reactions, all within the modal's contextual environment. The successful implementation of this sprint was instrumental in creating an immersive and informative gallery experience.

A screen shot of a computer program

Description automatically generated

**Figure 11**

Sprint 7 – This sprint proved to be a complex yet essential part of the development process, focused on enhancing user interaction by enabling commenting on images. This functionality required meticulous attention to detail in appending a dynamic comment system to the existing image context modal. The addition of a textbox for input and a posting mechanism introduced an interactive layer to the application (Figure 11). Moreover, integrating like and dislike features with live counters presented its own set of challenges. These elements were not merely static; they were tied to intricate operations that updated user reactions in real time, supported by Firestore subcollection structure for "comments" and "reactions" (Figure 12). I created these subcollections when the user would use the functions (either by pressing like/dislike or pressing ‘Post Comment’ button, which activated them (Figure 13, 14). For commenting, I had to store the comment along with the username and timestamp of when it was made, then created another function which displayed the comments (Figure 15) into the modal using docID (for specific image) after retrieving from Firestore. Initialising likes and dislikes at zero when the user uploaded an image was crucial for maintaining an accurate and user-responsive counter system. The successful deployment of these features improved the real-time interaction capabilities of the project.

A screenshot of a computer

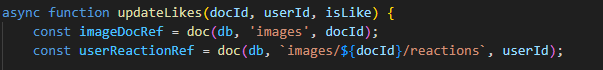
Description automatically generated

**Figure 12**

A screen shot of a computer program

Description automatically generated

**Figure 13**



**Figure 14**

A screen shot of a computer code

Description automatically generated

**Figure 15**

Sprint 8 – This sprint was the last sprint to add any functionality that I thought was missing from my project. In this sprint I created an edit and delete functions (Figure 16, 17) that would only appear if the user was the one who uploaded the image (Figure 18). I made it so that these buttons would only appear if the uploaded user email matched with the logged in user’s email address (Figure 19, 20). I thought these would be important as previously I was deleting images through Firestore, making it a lot easier to test my project. It also meant that users were able to edit the descriptions, categories, or uploader names of any images that they have uploaded, creating a more professional project. This sprint presented the most difficulties, which are noted in issues and constraints section of this report.

A screen shot of a computer program

Description automatically generated

**Figure 16**

A screen shot of a computer program

Description automatically generated

**Figure 17**

A screenshot of a computer

Description automatically generated

**Figure 18**

A screenshot of a video

Description automatically generated

**Figure 19**

A computer code on a black background

Description automatically generated

**Figure 20**

Sprint 9 – By this stage most of the coding and GUI development was complete. This final stage was very important, I created a free domain (<https://comp1004-azure.vercel.app/>) where I would send to friends so that they can test and upload their own images. I also added a footer where there would be a google feedback form so that I can gage anonymous feedback in what to improve or any bugs that they have found (Figure 20, 21). There was a note of a bug where if you liked or dislike an image you were not able to unlike or un-dislike it again, this was promptly fixed by incorporating another if statement that allowed them to do so. This then followed on by me finalising my report.

A screenshot of a chat

Description automatically generated

**Figure 20**

A screenshot of a test

Description automatically generated

**Figure 21**

**Firebase Implementation Overview**

Firebase has played an instrumental role in the success of this project by providing a comprehensive suite of backend services that are easy to use and require minimal setup. Its real-time database facilitates seamless data synchronisation across user interactions, ensuring immediate feedback and a dynamic user experience. Authentication services have streamlined the process of securely managing user sessions, while storage solutions have offered robust, scalable file hosting. Additionally, Firebase’s analytics tools have provided valuable insights into user behaviour, aiding in the iterative improvement of the application. Overall, Firebase’s integration has enabled rapid development, reduced the need for backend infrastructure management, and allowed for a focus on front-end development, accelerating the project’s production.

In summary during the implementation stage, I rigorously applied advanced software engineering principles, ensuring the delivery of a high-quality software product. Throughout the sprints, the project evolved from a basic concept to a comprehensive, user-centric platform, demonstrating the importance of adaptability, user feedback, and continuous improvement in software development. My approach was deeply rooted in the agile methodology, with each artifact and iteration carefully aligned with the proposed plans. Deviations from these plans were strategically considered and appropriately justified, showcasing my adaptability and decision-making skills in navigating unforeseen challenges. The codebase is a testament to my dedication to software engineering excellence, embodying key principles such as DRY (Don't Repeat Yourself) to avoid redundancy, YAGNI (You Aren't Gonna Need It) to prevent over-engineering, KISS (Keep it Simple Stupid) when designing UI and SOLID principles to ensure the software's maintainability and scalability. These practices were diligently applied, resulting in the development of functionalities that precisely matched the project's objectives. This careful and skilled implementation helps give evidence for comprehensive understanding and effective application of good software engineering practices in a complex project.

**Testing and Integration**

**Issues and Constraints**

In sprint 8 presented a unique set of challenges, particularly with the edit categorisation of images and deletion of images along with subcollections. It highlighted the intricacies of client-side rendering in response to database updates.

When users tried to change the category of an image. The image would appear twice: once in its new category and still in the old one. This situation relayed the importance of testing everything carefully and iterative development.

This bug was a persistent, and it necessitated a deeper understanding of how the DOM updates in response to data changes. I had to revisit the code for rendering images and implement a check to ensure that an image's presence was exclusive to its current category. The learning curve was steep; it involved debugging, testing various solutions, and understanding the asynchronous nature of JavaScript and how it interacts with Firestore's real-time data updates. Solving this issue required refining the image display logic to remove the image element from its previous category upon category update before re-rendering it under the new category (Figure 23). This fix not only resolved the issue but also served as a valuable lesson in the intricacies of web application state management.

A computer screen shot of text

Description automatically generated

**Figure 23**

The next issue was even though the images were being successfully removed from the Firestore database, the comments and reactions attached to them, which were stored in subcollections, weren't being deleted. This situation provided a valuable lesson in the effects of data relationships and the need for atomic transactions in database operations.

To tackle this, I had to enhance the deletion process to also get rid of related comments. I did this by refining the deletion process to include a batched operation or transaction that ensured both the image document and its linked comments and reactions were removed (Figure 24). This adjustment ensured a comprehensive cleanup of all related data, thereby preventing orphaned records and maintaining the robustness of the application's data architecture.

A screen shot of a computer program

Description automatically generated

**Figure 24**

Maintaining the project as a Single Page Application (SPA) presented both a challenge and a constraint that demanded a strategic approach to application design and user experience. SPAs require careful management of state, dynamic loading of content, and attention to performance to ensure a seamless and responsive interface.

One of the primary challenges was to ensure that the entire user experience happened within a single page, without the traditional page reloads associated with multi-page applications. This meant that all interactions, from image uploads, edits, category changes, to authentication flows, had to be handled dynamically. JavaScript played a pivotal role in this, providing the necessary tools to update the DOM in real time, reflecting changes immediately to the user without a full-page refresh.

Despite these challenges, a SPA architecture offers several benefits, such as a faster and smoother user experience, since only the necessary content is updated and re-rendered. Moreover, it aligns well with modern web development practices, offering a more app-like experience that users have come to expect.

By adhering to SPA principles, I managed to implement all required functionalities while keeping the application responsive and intuitive. The use of Firebase further facilitated this by providing real-time database updates and authentication services that integrate seamlessly into the SPA model. The end result was a cohesive, efficient, and user-friendly application that met the project's objectives while providing an engaging user experience.

**Reflection**

Reflecting on this project, it's evident that the journey from conception to implementation has been a deeply enriching experience. The project's primary goal was to create a Single Page Application (SPA) for image sharing, incorporating a range of functionalities from uploading and managing images to user authentication and interaction. Utilising Firebase as a backend service provided a robust and scalable solution for real-time data handling and storage, which has been instrumental in the application's success.

One of the project's most significant achievements is the dynamic user interface, allowing users to interact with the application seamlessly. The challenges of implementing a SPA were met with innovative solutions, such as asynchronous JavaScript for on-the-fly content updates and Firebase's real-time database for immediate feedback on user actions like likes and comments.

However, with every project, there are areas of potential growth and additional features that could be implemented. In future development, there are several functionalities I am eager to integrate to enhance the user experience and platform capabilities further:

* Comment Management: Allowing users to delete their comments will enable better control over their content, fostering a more secure and comfortable environment.
* Custom Usernames: Introducing usernames instead of relying solely on email addresses for identification could provide a more personalised and community-centric user experience.
* Like Transparency: Revealing who has liked an image can encourage more social interaction and connection among users, giving insights into common interests.
* Administrative Control: Implementing administrative rights to delete any image would be crucial for content moderation and maintaining community standards.
* Following Mechanism: Adding the ability to follow users can create a more engaging social platform, encouraging regular visits and interactions.
* Private Profiles: Private profiles would give users control over their privacy, deciding who can view their images and personal information.
* Bulk Image Uploads: Allowing users to upload multiple images simultaneously can streamline the sharing process, making the platform more user-friendly for those with several images to share.
* More Categories: Introduce more categories in the navigation bar, such as art, romantic, most liked, most disliked to enhance user experience.

These features would not only enhance user engagement but also contribute to a safer and more interactive community. They require careful consideration of user privacy and data security, ensuring compliance with regulations and ethical standards.

On a personal note, this project has been a learning curve, especially in terms of managing user experience and state within a SPA. The interactivity of the platform has underscored the importance of a well-designed user interface and the need for efficient data handling. Furthermore, integrating user feedback has been crucial in this iterative development process, as it gives an idea of what they would like to see in future development.

In conclusion, this project stands as a testament to the power of modern web development frameworks and the scalability of cloud services. The planned enhancements are aimed at creating a more comprehensive and immersive experience, evolving the application into a full-fledged social platform for image sharing.

Github repo link

https://github.com/Alex-T-Draper/Comp1004

References:

Ortiz-Ospina, E. (2019). *The Rise of Social Media*. [online] Our World in Data. Available at: https://ourworldindata.org/rise-of-social-media [Accessed 31 Mar. 2024].

Chaffey, D. (2024). *Global Social Media Research Summary 2024*. [online] Smart Insights. Available at: https://www.smartinsights.com/social-media-marketing/social-media-strategy/new-global-social-media-research/ [Accessed 31 Mar. 2024].

University of Cumbria (2019). *Why use social media? | MyCumbria*. [online] Cumbria.ac.uk. Available at: https://my.cumbria.ac.uk/Student-Life/it-media/Social-Media-Guidance/Why-use-social-media/ [Accessed 31 Mar. 2024].

Development, L.S. (2023). *Significance of Software Development Life Cycle (SDLC)*. [online] Medium. Available at: https://leeddev.medium.com/significance-of-software-development-life-cycle-sdlc-3617338d3883 [Accessed 31 Mar. 2024].

Pinheiro, J. (2018). *Software Development Life Cycle (SDLC) phases*. [online] Medium. Available at: https://medium.com/@jilvanpinheiro/software-development-life-cycle-sdlc-phases-40d46afbe384 [Accessed 31 Mar. 2024].

Michaela (2022). Why is website navigation so important? [online] blog.wurkhouse.com. Available at: https://blog.wurkhouse.com/importance-website-navigation-best-practices [Accessed 6 Apr. 2024].

W3schools (2019). *CSS Navigation Bar*. [online] W3schools.com. Available at: https://www.w3schools.com/css/css\_navbar.asp [Accessed 6 Apr. 2024].

W3Schools (2019). How To Make a Modal Box With CSS and JavaScript. [online] W3schools.com. Available at: https://www.w3schools.com/howto/howto\_css\_modals.asp [Accessed 6 Apr. 2024].