**Introduction**

This document offers an in-depth examination of Echo Gallery's creation, a Single-Page Web Application (SPA) for image sharing with social media features, developed as the project for the COMP1004 module. It charts the entire project lifecycle from conception through Software Development Life Cycle (SDLC) methods, detailing the project’s groundwork, design evolution, technology choices, and the obstacles overcome. The report highlights developmental successes and the agile Scrum framework’s key role in planning and refinement, with sprints and GitHub commits, demonstrating the project's growth.

**Software Development Life Cycle**

The 6 stages of the Agile SDLC are as followA diagram of software development cycle

Description automatically generateds:

Agile SDLC Diagram (Pinheiro, 2018)

The SDLC is a benchmark for the industry, with each stage being as crucial as the one before. It offers a structured path for software development, critical for a well-coordinated process, as highlighted by Development (2023). Skipping stages and diving into development without this framework risks improper execution and a final product riddled with errors. Adherence to the SDLC is not merely about following best practices; it's about guaranteeing that the product endures market pressures and fulfils user expectations.

In the next parts of this report, I will explain about each step of the Agile SDLC used, highlighting the indispensable value and specific contributions of every phase used. 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**

My vision is to develop a secure, engaging SPA that is compliant with legal standards, prioritising user needs and safety. The platform will support vibrant community interaction, where users can upload and share images freely while adhering to copyright laws, GDPR, data security protocols, and accessibility guidelines.

Compliance and User Safety:

* I'm dedicated to a secure experience, observing data protection laws, complying with GDPR (2018), and integrating privacy by design. The platform will maintain the highest security standards and respect intellectual property through educational tools and content moderation.

Performance:

* Utilising modern frontend technologies, the SPA will offer fast loading times and smooth transitions, essential for user engagement on various devices.

Data Security:

* Data security is paramount, incorporating advanced encryption, secure coding, and regular audits to safeguard against cyber threats and unauthorised access.

Accessibility:

* The platform will be accessible to all, complying with WCAG standards (W3C, 2018) to ensure it is easy for everyone to navigate, understand, and interact with.

Community and Interaction:

* Central to the platform, I aim to foster interaction through comments, reactions, and personalised galleries. Social interaction and custom content feeds will enhance the social environment.

This platform is not just a tool but a destination for creativity, cultural exchange, and community, promoting user rights, active participation, and accessibility for all.

**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 another (University of Cumbria, 2019). Echo Gallery enters this space as a fresh social media site focused on image sharing, promoting community interaction through likes and comments.

**Plan**

The planning stage is vital in the SDLC, laying the groundwork for the project. It outlines the project goals, delineates the scope, assesses potential risks, and allocates resources and timelines. Effective planning provides a clear set of objectives and a strategic approach to attain them. Neglecting this stage can result in wasted resources and project setbacks.

Below is a sprint plan that reflects a methodical approach to development, prioritising the user interface (UI) and progressing logically from simple layouts to advanced features like commenting and image uploads. Time is set aside for user feedback and thorough testing, ensuring the product is refined and meets user needs. The concluding weeks focus on project review and presentation preparation, providing a chance to reflect on successes and obstacles.

**Sprint Plan**

A green and white calendar

Description automatically generated

A screenshot of a computer program

Description automatically generated**Backlog (Actual Work Accomplished)**

**Requirements Analysis**

The Requirements Analysis stage is a crucial phase in the SDLC, as it establishes the project's foundation. It identifies and documents software functions and performance, guiding subsequent design and implementation. Effective analysis ensures proper project scoping, prevents feature creep, and minimises the risk of failing to meet user needs. This documentation provides clear guidelines for later SDLC stages, clarifying development objectives.

Below are diagrams illustrating my requirements analysis.

**User stories**

A white sheet with black text

Description automatically generated

**UML Diagram**

A diagram of a computer

Description automatically generatedA diagram of a computer

Description automatically generated

**Use Case**

A diagram of a person's work flow

Description automatically generated

**Use Case Description**

A screenshot of a computer program

Description automatically generated

A white sheet with black text

Description automatically generated

A white text with black text

Description automatically generated with medium confidence

A white sheet of paper with text

Description automatically generated

A screenshot of a sign up

Description automatically generated

A white sheet with black text

Description automatically generated

A sign out document with text

Description automatically generated

A white sheet with black text

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a navigation bar

Description automatically generated

A white text on a black background

Description automatically generated

**Design**

The design phase is vital in the SDLC, acting as a blueprint for the project. It transforms software specifications into a design plan, encompassing architectural design, component selection, UI design, and definitions of data structures, algorithms, and detailed software architecture. This early focus ensures scalability, maintainability, and compliance with requirements, identifying potential issues to streamline development. Effective design reduces development time and ensures the final product meets user expectations and quality standards.

**Architecture**

The development of this SPA for an image sharing social media platform is a complete project that brings together an array of technologies and services for good user experience. At its core, the platform features 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 SPA structure for smooth user navigation.

Below are diagrams illustrating my design phase.

**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**

A diagram of a company

Description automatically generated

**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 coding and software construction take place. Following detailed planning and design, this phase brings the project to life. Here, design documents and specifications are converted into source code, incorporating necessary components and libraries. Characterised by coding, unit testing, and integrating modules into a cohesive system, this stage marks the actual development of the software.

Below is review of the sprints carried out in my project. It details the struggles and achievements encountered throughout the development process.

**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 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, 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 – Developing the image upload functionality was crucial in my project, I implemented a Modal (W3Schools, 2019) that opens a user-friendly window for uploading images (Figure 7). The image upload function is within 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. Mastering CSS to ensure visual appeal was a significant part of this sprint's work.

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 – The sprint was smoother than anticipated. Enhancements to the modal from Sprint 3 involved adding fields for image name, author, description, and category (Figure 7), key for efficient content categorisation and retrieval. On upload, these details form a structured data object, sent to Firestore with asynchronous HTTP requests, ensuring metadata is searchable. These changes, requiring minor tweaks to the existing function due to robust groundwork, have improved the image gallery’s organisation and accessibility, significantly boosting functionality and user engagement.

Sprint 5 – In retrospect, a week was more than needed for this sprint, thanks to DOM and Firebase integration. Using DOM manipulation, I dynamically rendered category-specific images from Firestore, 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. Additionally, I implemented Firebase Authentication (Figure 10) to secure user sign-in and enable user-related features, like comments. This not only improved interactivity and security but also laid groundwork for advanced features, requiring users to sign in for full functionality. These developments were a leap forward for the project.

A screenshot of a computer

Description automatically generated

**Figure 10**

Sprint 6 – This sprint significantly boosted user engagement. I utilised JavaScript's template literals to generate interactive HTML elements like a down arrow, initiating an overlay modal that presents expanded image details, including the uploader's name and description (Figure 11). This method keeps users on the page by providing additional content through a floating overlay. The modal's dynamic HTML, sourced from Firebase, structures information in a user-centric way. This enhancement not only improved immediate access to image specifics but also set the stage for further interactive elements such as comments and reactions within the same modal context. The sprint was key to crafting an engaging and informative gallery experience.

A screen shot of a computer program

Description automatically generated

**Figure 11**

Sprint 7 – This sprint was intricate, focusing on user interaction through a new commenting feature on images. A dynamic comment system was embedded into the image context modal, with a textbox for user input and a posting function for interactivity (Figure 11). Additionally, I integrated like and dislike buttons with real-time counters, relying on Firestore's "comments" and "reactions" subcollections (Figure 12). Subcollection initialisation and features are triggered by user actions—likes, dislikes, or comment posts (Figure 13, Figure 14). Commenting involved saving user details and timestamps, with display functionality bringing comments into the modal (Figure 15). I initialised likes and dislikes at zero upon image upload to ensure accurate interaction tracking.

A screenshot of a computer

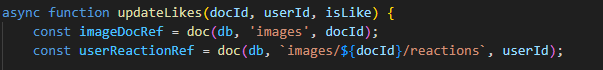
Description automatically generated

**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 – In this sprint I created edit and delete functions (Figure 16, 17) that are only accessible to the user that 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. 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 – The final stage was crucial; I set up a free domain (https://comp1004-azure.vercel.app/) for friends to test and upload images. A footer with a Google feedback form was added for anonymous suggestions and bug reports (Figure 20, Figure 21). Feedback highlighted a bug preventing 'unliking' or 'undisliking' images, which I quickly fixed with additional conditional logic. This refinement led to the completion of 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 a significant role in this project, providing a range of backend services that are efficient and easy to implement. Its real-time database ensures instant data sync, creating a responsive user experience. Firebase's authentication services streamline secure session management, while its storage options offer reliable file hosting. The platform's analytics offer critical insights for continuous improvement. Overall, Firebase has accelerated development, reduced backend upkeep, and allowed a greater focus on frontend development, aiding in the project's timely completion.

**Sprint Summary**

Throughout the implementation, I adhered to advanced software engineering principles, evolving the project from concept to a user-centric platform. The project evolved from a basic concept into a platform focused on user needs, highlighting the importance of adaptability and continuous user engagement. Agile methodologies steered the process, aligning with strategic objectives and enabling me to adeptly navigate challenges. The codebase, embodying engineering rigour with principles like DRY, YAGNI, KISS, and SOLID, culminated in a feature set that aligned seamlessly with project goals.

A white sheet of paper with black text

Description automatically generated**Testing**

**Issues and Constraints**

Sprint 8 posed challenges with image re-categorisation and deletion, emphasising the complexity of client-side rendering alongside database updates. Users encountered a bug when changing an image's category; the image would duplicate, appearing in both the old and new categories. This underscored the necessity of thorough testing and iterative development.

The issue persisted, demanding a nuanced understanding of DOM updates in reaction to data alterations. I revised the image rendering code, introducing a check to ensure an image only appeared in its assigned category. This was a learning curve, involving debugging and grasping the asynchronous nature of JavaScript and Firestore interactions. The solution entailed updating the image display logic to remove the image from the old category before re-rendering it in the new one (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**

Addressing the next challenge, I found that deleting images from Firestore didn't automatically remove their associated comments and reactions subcollections. This shed light on the importance of handling related data and the use of atomic transactions in database operations.

To solve this, I developed a deletion process that executed a batch operation, ensuring the simultaneous removal of the image document and its related comments and reactions (Figure 24). This approach was critical for thorough data removal, preventing orphaned records and preserving the integrity of the application's data structure.

A screen shot of a computer program

Description automatically generated

**Figure 24**

Managing the project as an SPA was both challenging and restrictive, necessitating a calculated approach to design and user experience. SPAs demand meticulous state management, dynamic content loading, and performance optimisation to ensure a smooth 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 its complexities, the SPA framework offers benefits like enhanced speed and seamless user experiences, as it refreshes only the needed content. It also complements contemporary web development practices, delivering an app-like experience users expect.

Adhering to SPA principles, I successfully incorporated all functionalities, ensuring the app remained responsive and user-friendly. Leveraging Firebase's real-time data updates and authentication services meshed well with the SPA structure. The outcome was a unified, efficient, and approachable application that fulfilled project goals while providing an engaging user experience.

**Reflection**

Reflecting on the project, the progression from idea to execution has been immensely fulfilling. The primary objective was to craft an SPA for image sharing with comprehensive features like image management and user interactions, supported by Firebase's robust backend capabilities, which have been instrumental in the application's success.

The achievement of a dynamic user interface stands out, allowing seamless user engagement. The SPA challenges were addressed through innovative methods like asynchronous JavaScript and Firebase's instantaneous database updates.

However, there are areas of potential growth and additional features that could be implemented:

* Comment Management: Allowing users to delete their comments will enable better control over their content, fostering a more secure and comfortable environment.
* Custom Usernames: Personalised usernames could create a more community-focused 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: Providing privacy settings would give users control over their content visibility.
* Bulk Image Uploads: A multiple image upload feature can streamline the sharing process, making the platform more user-friendly for those with multiple images to share.
* Expand Categories: Introduce more categories in the navigation bar, such as art, ‘most liked’ 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 an SPA. The platform’s interactivity highlighted the necessity of crafting a user-friendly interface and the importance of adept data management. The iterative development process has been enriched by integrating user feedback, providing valuable insights into desired future enhancements.

In conclusion, the project exemplifies the strengths of contemporary web development frameworks and the expansive potential of cloud services. This project not only served as a practical application of modern web development principles but also as an educational journey through the SDLC. It underlines the framework's critical role in project success and provides a blueprint for future endeavours. The lessons learnt from this experience will undoubtedly inform and enhance future projects.

**GitHub repo link:**

https://github.com/Alex-T-Draper/Comp1004/tree/main/Project

References:

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].

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].

GDPR (2018). *General data protection regulation (GDPR)*. [online] General Data Protection Regulation (GDPR). Available at: https://gdpr-info.eu/ [Accessed 10 Apr. 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].

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].

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].

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].

W3C (2018). *Web Content Accessibility Guidelines (WCAG) Overview*. [online] Web Accessibility Initiative (WAI). Available at: https://www.w3.org/WAI/standards-guidelines/wcag/ [Accessed 10 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].