CMPG323
Final Assessment
Project Scope

Table of Contents

Overview

A South African digital marketing company has expressed the need for a web application that serves as an online platform allowing users to store, view and share photos. Users should be able to create an account and login before utilising the platform. Once logged in, users should be able to view their own photos as well as the photos shared with them. The user should be able to create and manage the metadata recorded for each photo that the user has access to. The ability to share content with other registered users also needs to be built into the web application. Users should have the ability to download photos that they have access to. The web application should be designed as consistent, intuitive, easy-to-use and easy-to-navigate without needing to provide excessive user guide documentation. All information and media items that are stored should be stored securely and in accordance with POPI.

Functional Requirements

Functional requirements refer to the functionality that a system must have and how the functions should be performed. The functional requirements have been prioritised using the MoSCoW approach and are detailed below:

Must Have	
Upload, view and delete at least one photo	A user should be able to upload, view and delete (or hide) at least one photo at a time. User access and photo metadata should also be revoked (or deleted) when the photo is deleted. The supported file formats are: - .bmp - .ico - .jpeg - .jpg - .gif - .tiff - .png
Create, view, update and delete photo metadata	A user should be able to create, view, update and delete the metadata attached to their own photo (or photos). Metadata information to be captured for each photo should at least include: - Geolocation - Tags - Captured date - Captured by
User-based access control implementation	All data that is accessed through the web application should be subject to the user's access level. If a user does not have access to see the request data, appropriate error messages should be displayed
Should Have	
Manage shared photo metadata	A user should be able to create, view, update and delete the metadata attached to another user's photo (or photos). Metadata information to be captured for each photo should at least include: - Geolocation - Tags - Captured date - Captured by

Share one or more photos	A user should be able to share one or more photos with one or more different users. These users should already exist as users within the system, if they don't, an error message should appear.
Download one photo	A user should be able to download one singular photo from the web app which is then saved on the user's local computer.
Manage user access on at least one photo	A user should be able to allow, view, update and delete a user's access to at least one photo and the associated photo metadata.
Display photos	A user should be able to see their own photos as well as photos shared with them.
Could Have	
Create and manage albums (This criteria is a MUST HAVE for team projects)	A user should be able to create, view, update and delete albums which are used to group photos. These albums may be shared with other users that already exist on the system.
Search photo metadata (This criteria is a MUST HAVE for team projects)	A user should be able to search for photos through the metadata of the photos that the user has access to.

Non-Functional Requirements

Non-functional requirements refer to the aspects of a solution that have an impact on the quality attributes of a system (or platform). These non-functional requirements are deemed as supportive requirements to ensure that the functional requirements are implemented appropriately and according to good software practices. For the sake of this scope, the non-functional requirements include technical aspects that need to be considered, at all times, during the development and implementation of the project. The non-functional requirements have been prioritised using the MoSCoW approach and are detailed below:

Must Have	
Secure Data Storage	All data should be secured and should conform to POPI standards. No passwords should be visible in the database or in the code configuration.
Consistent Design	The web application front-end design should remain consistent across the user journey. There should be a consistent use of fonts, font sizes, margins and spaces, text position, animated effects, etc.
Design Pattern	The clear use of a design pattern should be present in the code structure. Pay close attention to polymorphism and object-orientated programming (OOP), including coding standards like SOLID, for example.
Architectural Pattern	The clear use of an architectural pattern should be presented by the code structure. Pay careful attention to the difference between an architectural and design pattern
Error Handling	Error handling methods should be used, along with logging, to ensure that the user does not receive any technical errors and so that the developer can identify the problem by investigating the logs.

Source Control	The audit log on the source control repository should clearly indicate that source control was used through the project. The audit logs should show multiple commits and pushes, based on the completion of functional requirements, along with a branching strategy.
Should Have	
Referential Integrity	When a photo is deleted, the photo metadata and user access should also be deleted in the database. Ensure that referential integrity is applied to the database.
Intuitive Design	The design should be user friendly; it should guide the user through the steps needed to functionally use the web application in a way that reduces the need to provide user guide documentation.
Intentional Design	The design should be targeting the functional intention of the content currently on screen. For example, do not show user access and photo metadata fields if the user hasn't uploaded the photo yet. The initial intentional focus is targeted at getting the user to upload the photo before proceeding to the next possible action, like user access and photo metadata, or even album association.
Responsive Design	The design should be implemented with the consideration of users who would want to use this web application from their mobile device. Mobile responsiveness would be an important aspect of implementing this capability.
Could Have	
Website Hosted on Cloud	The website should be accessible from any computer without having to compile and run the code on that computer. In order to achieve this, host the website on a cloud storage platform (like Azure, Google Cloud Platform, IBM, AWS, etc.). Avoid hosting the web application on a free Gearhost subscription as you are likely to run into storage constraints
File Storage Hosted on Cloud	The files should be stored on an online server or file storage mechanism that is secure. Files stored on a local computer and cannot be accessed from a different computer do not qualify as cloud storage. Do not use a database to store the images, platforms like Google Drive would suffice as long as the storage is secured. If the storage is not secure, these marks will be forfeited.
Database Hosted on Cloud	The database should be accessible, through the site, from any computer without having the database created on that computer. This could be achieved by hosting the database on a cloud platform (like Azure, Google Cloud Platform, IBM, AWS, Gearhost, etc.)
Inclusive Design	The design should be inclusive of all users in the target audience. For example, if a user were colour blind, they would not be able to infer the significance of coloured buttons if there weren't patterns or shading differences as well.
Unit Tests	Unit tests should be present to test at least 80% of all functionality. At least 90% of all unit tests should pass.

Error and Exception Handling

All errors and exceptions that occur are broken up into two groups, namely: application errors and business exceptions. All errors and exceptions should be appropriately logged so that they may be investigated, if necessary.

Application Errors

Application errors refer to any error that occur due to a technical issue. All application errors (including a 'page not found' error) should be handled in the backend with a user-friendly error displayed on the frontend. No unhandled technical errors or exceptions should be visible to the user. Unhandled technical exception messages that are presented to a user could pose a security risk.

Business Exceptions

Business exceptions refer to the known and anticipated errors that occur as a result of implemented business logic. The following business exceptions should be accommodated for using the actions as suggested below:

Exception	**Description**	**Action**
User not found	A user tries to share content with a user that does not exist (possibly due to a spelling mistake or because the user is not registered).	Display 'User not found, please ensure that the user is registered' message.
Access Required	A user tries to access content that they do not have access to.	Display 'Access Required' message and allow the user an option to request access.
Unsupported Format	A user tries to upload a file that is not an image.	Display 'Unsupported format, please upload the content in a different format' message.

Submission Details

The scope of this project has been issued as an **individual** assignment, *or can be completed as a team project of maximum 2 members (based on additional requirements in the software and documentation rubric)*. There is no dictated technology stack as long as the technology of choice allows for custom-built backend integration (into a database, for example). This would exclude tools like Wix and WordPress as viable technology stacks for this project.

Submission: Submit your relevant files on the CMPG323 eFundi assignment.

Deadline: 17h00 on 22 November 2021 (please note there are no alternative or late submission dates – if you miss this deadline you will forfeit the opportunity)

What to submit:

1) Provide the URL to your REPO (such as GitHub) where you made regular commits and your code can easily be accessed - copy the link into the textbox of the assignment.
2) Provide the URL to where your project has been hosted - copy the link into the textbox of the assignment.
3) Provide the URL to where your desktop presentation can be found on YouTube (for quicker and easier viewing) - copy the link into the textbox of the assignment. [NB! Your presentation should not be longer than 20 minutes - 1 mark will be deducted for each minute over this limit. You need to be concise and to the point, while still providing the necessary information. This is an expected skill in the industry and will be a good exercise for making sure that you only provide relevant information in a timely and professional manner]. If you are unable to upload your presentation to YouTube - you can add it as an attachment to the assignment - but make sure to run it through a program such as 'Handbrake' which reduces the size of the file. Your file should not be larger than 50MB.
4) Attach your zipped project file.
5) Attach your project documentation (outline and rubric available separately).
6) For team projects - complete the eFundi quiz for peer evaluation and contribution.

Marking Considerations

Please take not of the following considerations that will form part of the marking and moderation process:

- A rubric will be provided separately
- Your desktop video presentation should clearly explain how you addressed each rubric criteria in order – make sure that you demonstrate every possible detail around the functionality of the feature.
- Failure to access/ hear/ view your desktop presentation will result in 0
- Failure to upload any of the requirements for submission will result in 0
- Failure to complete this as an individual assignment will result in 0
- Failure to use a technology that allows for custom-built backend functionality will result in 0. Do not use Wix, WordPress or any other online website builder as a shortcut.

Marks being subtracted if any of the following occur:

- Passwords that are visible in database or in source control
- An insecure registration and login process
- Photos that are visible to users who do not have access
- Photos that are visible to users who have not logged in

Reading Materials

There are multiple aspects of the abovementioned scope that may be covered by

- Architectural Patterns:
  - https://www.dotnetcurry.com/patterns-practices/web-application-architecture
  - https://towardsdatascience.com/10-common-software-architectural-patterns-in-a-nutshell-a0b47a1e9013
- Design Patterns:
  - http://researchhubs.com/post/computing/web-application/design-pattern.html
  - https://hammockweb.com/blog_detail/Software-Design-Patterns-Web-Development.html
  - https://cubettech.com/resources/blog/introduction-to-repository-design-pattern/
- Good Practices for Displaying Error Messages:
  - https://uxdworld.com/2018/05/30/how-to-write-good-error-messages/
  - https://cxl.com/blog/error-messages/
- User Experience Best Practices:
  - https://729solutions.com/ux-ui-best-practices/
- User Management:
  - https://backendless.com/feature/user-management/
    - https://medium.com/bluecore-engineering/implementing-role-based-security-in-a-web-app-89b66d1410e4
    - https://selftaughtcoders.com/user-authentication-access-control-web-application/